Future Generation Computer Systems 147 (2023) 275–291

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Analysis of the confirmation time in proof-of-work blockchains

Ivan Malakhov, Andrea Marin, Sabina Rossi *

Università Ca' Foscari Venezia, Italy

ARTICLE INFO

Article history:
Received 24 January 2022
Received in revised form 5 April 2023
Accepted 12 April 2023
Available online 19 May 2023

Keywords:
Blockchain
Proof of work
Queueing model

ABSTRACT

In blockchain networks driven by Proof of Work, clients spend a certain amount of cryptocurrency (called fees) to control the speed of confirmation of the transactions that they generate. In fact, transactions are confirmed according to a strong priority policy that favors those offering the highest fees. The problem of determining the optimal fee to offer to satisfy certain delay requirements is still widely open and, at the state of the art, mainly reactive methods based on historical data are available. In this work, we propose a queueing model based on the exact transient analysis of a $M/M^B/1$ system to address this problem. The model takes into account (*i*) the state of the Mempool (the backlog of pending work) when the transaction is generated, (*ii*) the current transaction arrival intensity and (*iii*) the distribution of the fees offered by other transactions to the miners. We apply the model to study the performance of the Bitcoin blockchain. Its parameterization is based on an extensive statistical analysis of the transaction characteristics. To this aim, we collected data from over 1.5 million of pending transactions observed in the Mempool of our Bitcoin node. The outcome of our analysis allows us to provide an algorithm to quickly compute the expected transaction confirmation time given the blockchain state, and to highlight new insights on the relations between the transaction fees and confirmation time in BTC blockchain.

1. Introduction

In recent years, the economic system that allows blockchain distributed ledgers to operate has attracted a lot of attention. In particular, the fees offered by the users to pay for the services provided by the system have been recognized as a pivotal aspect of this technology [1–3]. To make this topic even more important, we must consider that in few months an important blockchain like Bitcoin will rely just on users' fees to support the energy and hardware costs faced by the miners, i.e., those users that invest a huge amount of resources[1] to support the blockchain operations.

This mechanism causes dynamics worthy of scientific investigation, such as those highlighted in [4–7].

Blockchains are distributed ledgers based on peer-to-peer (P2P) consensus protocols that are becoming widely popular nowadays. Beside their most well-known applications for cryptocurrency trading, blockchains are the enabling technology for many other applications that require the permanent and immutable storage of data, or even the execution of publicly known programs called *smart contracts*.

In these systems, data are organized in transactions, and transactions are stored into blocks to form a linked list called blockchain. Once a transaction is included in a block, it is possible to quantify the energy costs for changing its content or delete it. For popular blockchains, such as that underlying Bitcoin, this cost becomes quickly prohibitive even for recently confirmed transactions.

A crucial phase of this process is the transaction validation, i.e., the P2P community must find an agreement on its validity according to some rules. For example, in blockchains based on cryptocurrency trading, the rules for validation include the verification of the identity of the user(s) and a check to avoid the double spending of the same amount of cryptocurrency.

Several protocols have been devised to reach consensus, possibly inspired by the Byzantine fault tolerance problem. In this paper, we consider the original and mostly applied consensus protocol: the *Proof of Work* (PoW). PoW is applied in Bitcoin blockchain [8] as well as in Bitcoin cash, Ethereum[2] and many other ledgers [9]. Henceforth, we focus on the behavior of the Bitcoin (BTC) system since it is the most used blockchain and its protocols have inspired many other blockchain systems.

The set of users that verify the transactions, consolidate the blocks and store a copy of the blockchain are called *miners*. When miners receive a new transaction, they store it in a special buffer

* Corresponding author.
E-mail addresses: ivan.malakhov@unive.it (I. Malakhov), marin@unive.it (A. Marin), sabina.rossi@unive.it (S. Rossi).

[1] According to Cambridge Centre for Alternative Finance, the estimated annual energy consumption of Bitcoin blockchain is 145.63 TWh, i.e., roughly the annual electrical energy consumption of a country like Sweden. Source: https://cbeci.org.

[2] https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/

https://doi.org/10.1016/j.future.2023.04.016
0167-739X/

for pending transactions that is usually called *Mempool*. While each miner maintains a private Mempool, their states are rather similar as the propagation delays in the network are significantly small compared to the average time between the blocks [10]. In fact, there are several online services that show the state of the Mempool as if it was a common queue.[3,4] We further discuss differences of the Mempool states in Remark 1. A transaction that leaves the Mempool and is included in a block is said to be *confirmed*.

Each block of the chain contains a subset of the transactions present in the Mempool at the moment of its consolidation and the maximum amount of transactions that fit in a block is given by some invariant properties of the blockchain. For example, in BTC, every block can be at most 1 MB large, which translates into an average of 2300 transactions per block.

As far as this paper is concerned, the crucial aspect of the mining process is the way in which the transactions are selected from the Mempool by the miners. Indeed, transactions offer a fee for their confirmation that is publicly known, and miners select from the Mempool the transactions that are more profitable, i.e., those that offer the highest fee per byte.

Since in BTC the maximum block size is fixed and the generation of blocks is designed in such a way that one block is consolidated every 10 min on average, it is clear that, especially on heavy load conditions, the offered fee becomes determinant for delay-intolerant operations. For example, the high volatility of BTC price requires payments to take place (i.e., to be added to a block) within a few blocks after their request. The mining process is depicted in Fig. 1.

This paper proposes a queueing model to answer the following questions: *given the state of the Mempool and the intensity of the workload, what is the expected number of blocks that a transaction offering a certain fee should wait for its confirmation?* It is worth to notice that the state of the Mempool (including the distribution of the fee offered by the transactions therein) and the intensity of the arrival process are publicly available information that may be obtained either by running a BTC miner node, or by using one of the many free online services.[5]

1.1. Contribution

This paper starts from the observation that a transaction x whose fee per byte ratio is f experiences a waiting time formed by the sum of two delays:

- The system first confirms all the transactions present in the Mempool at its arrival epoch whose fee per byte is greater or equal to f;
- Moreover, other transactions arrive after x but before its confirmation, and if their fee per byte is higher than f they will be confirmed before x.

The confirmation of the transactions takes place in batches, i.e., the newly generated block contains all the transactions that it can fit. Therefore, the whole process can be seen as $M/M^B/1$ queueing process [11,12], where, according to Kendall's notation [13], M denotes that both the transaction inter-arrival times and the inter-block generation times are independent and exponentially distributed, B stands for the batch size that simply represents the number of transactions that a block can fit, and finally 1 denotes that the system consolidates one block at a time.

[3] https://mempool.space
[4] https://mempool.observer
[5] http://www.blockchain.com

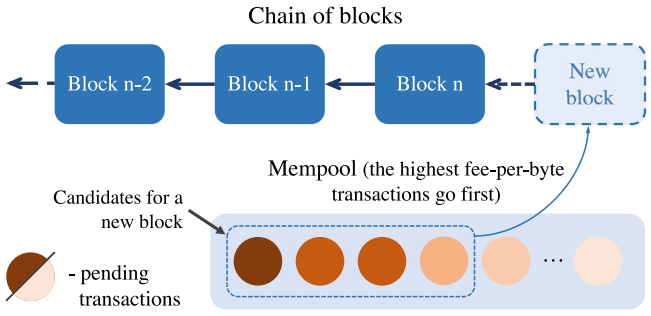


Fig. 1. Sketch of the mining process in BTC. Darker colors represent higher fee-per-byte ratios.

We will support the assumptions on the exponential service time and Poisson arrival process in Section 3.

Given a tagged transaction x offering f, its consolidation delay, i.e., its residence time in the Mempool measured in number of consolidated blocks, corresponds to the time required by the $M/M^B/1$ queue starting from the *initial state* to reach the *empty state*. The former corresponds to the number of transactions $Y+1$ (including the tagged transaction) whose fees are higher than f found in the Mempool at its arrival epoch. Consequently, the latter refers to the circumstance when there will be no more transactions that offer higher fee $f' : f' > f$ in the Mempool. The system as seen by x is subject to an arrival process filtered to take into account only the transactions that are more valuable than f.

We provide the transient solution of such a system based on the technique of generating functions. Theorem 1 gives an iterative method for the exact computation of the expected transaction's confirmation time given the root of a certain polynomial that can be easily obtained with a numerical procedure. Although in this paper we focus on the first moment of the expected confirmation time, the analysis that we propose allows also for an approximate computation of further moments where the approximation error is bounded thanks to the theory of residuals in power series.

Finally, we provide an extensive set of experiments with the aim of studying the impact of the Mempool state and the system's load factor on the choice of the fee to offer in order to satisfy certain delay requirements on the transaction confirmation.

We believe that the results proposed in this work are of high importance for every transaction issuer. Clearly, to optimize the costs it is crucial for them to know the minimum fee to pay in order to have their transactions confirmed within a certain desirable time, as in case, for example, of speculative exchanges of the cryptocurrency. Conversely, one may also be keen to know how long the confirmation delay would be if a certain fee for the transaction is set.

1.2. Structure of the paper

The paper is structured as follows. First, Section 2 describes the related work on this subject. In Section 3, we describe our research problem and review the PoW systems. Moreover, we show some data analysis of BTC blockchain to motivate the introduction of our queueing model. This model is presented and solved in Section 4. In Section 5, we use the results of Section 4 to study the expected confirmation time in BTC blockchain. What is more, we validate our model both with trace driven and stochastic simulations. Finally, Section 6 concludes the paper.

2. Related work

In general, the quantitative analysis of blockchain systems has drawn a lot of attention from the scientific community (see, e.g., [14–16] and the references therein).

In particular, the estimation of transactions' confirmation time with queueing theory has been explored in some very recent works [17–21]. In this section, we focus on those works whose aim is that of studying the confirmation delay as function of the offered fee. In this context, the advantage of a queueing theoretical model with respect to other approaches based on prior statistics is that the former reacts quicker to changes of the arrival process. In fact, predictive models based on historical data recommend increasing the fee to achieve a certain target expected consolidation time once they record that the previous fee does fit the requirements. Fig. 3(a) shows the intensity of the arrival process and the expected fee per byte of the transactions in BTC during five days. We can observe a delay of approximately three hours between the reaction of the current predictive model and the change in the system's workload.

The major difference between our contribution and those described in [17–19,21] is that we consider a transient analysis instead of a steady-state one. This has several consequences. The first is that our model takes into account the state of the Mempool at the moment in which the transaction is generated. As we will observe in Section 5, this has an important impact on the confirmation delay and is actually information available to the users that should be used. The second difference is that the priority queue analysis provided in [18,19,21] requires one to cluster the transactions into few classes based on the offered fee, while we can handle continuous distributions (e.g., obtained by fitting of real data) of offered fees per byte.

The works [18,19] differ from [21] and ours for the consolidation policy. Indeed, the former two assume that once the miner chooses the transactions to add to the next block, this will not be changed. Conversely, [21] and our work considers the fact that miners update their choices upon the arrival of more profitable transactions. If this happens, the cheapest transaction is removed from the candidate block and is replaced by the newly, more profitable, arrived one. Since the mining process is memoryless and the PoW is only marginally affected by a change in the selection of the set of transactions to consolidate, this policy is closer to what happens in real systems.

In [22], the authors propose a queueing model at the base of a classifier for the transactions. The work presents interesting measurements that show an important insight of the BTC blockchain, especially regarding the characteristic of dropped transactions. However, the impact of the offered fee per byte of the confirmation delay is not considered by the model (although it is experimentally measured for the dropped transactions).

In [17], the authors propose an iterative solution for the stationary distribution of the embedded Markov chain of a $G/G^B/1$ queue and validates the analysis with measurements collected from the Ethereum blockchain. The conditional confirmation delay from a single transaction perspective is not considered, although the model is well designed for the overall analysis of the system, e.g., to estimate the expected size of the Mempool in steady-state.

Another important related work is [23]. This contribution shares with [18,19,21] the stationary analysis of the queueing model and the introduction of the customer priority classes. However, its aim is that of proposing a game theoretical framework in which the dynamic of the fees are studied in relation with the economical interests of the miners.

In [24,25], the authors propose to use the process named Cramer–Lundberg to evaluate the confirmation time of transactions. Similarly to our contribution, the authors take into account the initial state of the Mempool and assume a homogeneous Poisson process for the block generation counting. In order to overcome the computational complexity for the solution of the process in heavy load, they introduce a diffusion approximation with shifted initial point in order to avoid a premature hitting of the absorbing state. With respect to this work, our model maintains the stochastic nature of the transaction arrival process (while the Cramer–Lundberg model requires a constant arrival flow) and is solved with an exact method unveiling some new results for the $M/M^B/1$ queueing systems.

For what concerns the queueing theoretical results, several works have studied the single server queue with batch departures (see, e.g., [26]) but to the best of our knowledge, this is the first time that the exact solution of the expected time (in number of completed services) to the absorption in the 0 state of a $M/M^B/1$ queue starting from an arbitrary state is presented. In [27], the author performs a discretization similar to ours to study a queue with batch service. In this case, the batch is formed immediately *after* the completion of a service, i.e., similarly to [18,19], while the system under study requires to form the batch immediately before the service with all the available transactions in the Mempool. The behavior of the queue becomes quite different, and no algorithm similar to that of Theorem 1 is given.

3. Background and motivation

This section consists of two parts: a review of the PoW consensus algorithm with particular attention to BTC blockchain and the description of the problem we aim to address.

3.1. Consolidation of a block with PoW

The consolidation of a block is done in two phases:

1. The miner selects and validates the transactions to include in their candidate block.
2. The miner competes with the others to solve the puzzle required by the PoW.

PoW is the consensus mechanism used by most of the blockchains to ensure the security properties of the system without the need of a central, trusted authority.

Informally, we can say that the computational effort required for mining blocks is the key factor that allows the community to trust the security of the blockchain: the bigger the total computational power of all miners, the more secure the system.

In practice, all miners are required to verify the validity of the transactions they would like to have in their new block. In this phase, each miner verifies the transaction signatures to assess their authenticity. Moreover, before including the transactions in the new block, each miner verifies that there are no double spending issues. Notice that, at the announcement of a new block, these checks are performed also on the transactions contained in it, thus reaching a distributed consensus. After doing so, miners have to consolidate the block by *solving a puzzle*. Informally, in BTC and many other blockchains, this means finding a certain *nonce* such that the binary version of the hash of the block including that nonce begins with a certain number of zeros. Since it is assumed that each hash is equiprobable given a certain block, and given that the nonce is a 32 bits integer, we can safely assume that the solution of the puzzle is a memoryless process. This implies that the number of hashes computed between two successive blocks is geometrically distributed, and hence the time is independent and approximately exponentially distributed.

In BTC blockchain, the difficulty of the puzzle is dynamically set in such a way that blocks are generated on average every 10 min. Thus, it can be safely assumed that the distance between

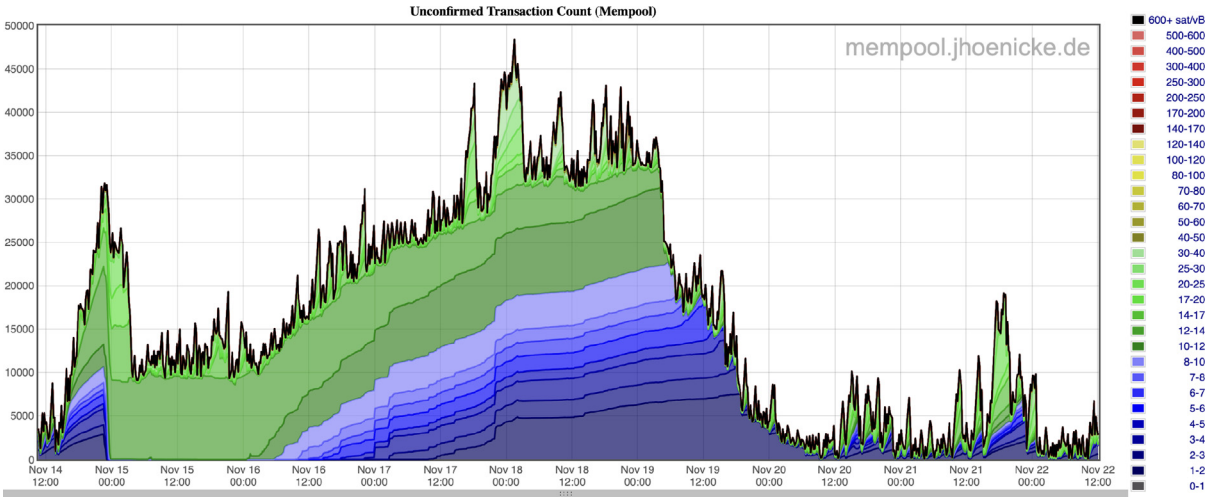


Fig. 2. Mempool dynamics from November 14th to November 22nd, 2022.

consecutive blocks is independent and exponentially distributed with a mean 10 min. This is the first invariant property of BTC blockchain.

The second invariant property that characterizes most PoW blockchains is the maximum block size. For example, for BTC this is 1 MB. The estimation of the expected number of transactions that can fit in a block must be carefully done taking into account some protocol characteristics, e.g., Segregation Witness (or *SegWit*) in BTC.[6] Generally speaking, the implementation of the SegWit transactions allows the separation of the transactions' signatures from their other data. Hence, a block completely filled with the SegWit transactions carries the same information of a 1.7MB block without SegWit (see Remark 3 for more details).

In addition, it is known that, given the average transaction size, a block can contain approximately 2300 items that corresponds to a maximum throughput of $2300/(10 \times 60) \simeq 3.8$ transactions per second, that is expected to increase as more transactions will adopt the SegWit standard.

3.2. Auction and transaction confirmation time

When a new transaction is seen by the miners, it is included into a local queue called Mempool. The pending transactions in the Mempool of each miner are still not effective since only those appearing in the consolidated blocks (i.e., the *confirmed* ones) can be universally considered immutable, or at least the energy cost for their change can be estimated. The time between the arrival epoch and the inclusion in a block is called *transaction confirmation time*.

Users can control the transaction confirmation time by offering a fee that will be cashed by the miners at its consolidation. Since miners aim to maximize their profit, they tend to choose the most profitable transactions from the Mempool to be included in the block. Because of the possible different transaction sizes, they use the fee per byte ratio (sometimes called *fee density*) as a metric to assign priority to the transactions. Thanks to the memoryless property of the PoW, highly profitable transactions are immediately included in a new block by evicting the less profitable ones that stay in the Mempool.

It is not surprising that when the system is close to saturation, the expected confirmation delay grows and the auction on the offered fees becomes more expensive for the users as shown by Figs. 3(c) and 3(d).

In the BTC blockchain, the cryptocurrency is the Bitcoin whose value in FIAT is very volatile. Therefore, coherently with the online community best practices, we will resort to the Satoshi (sat) to specify the fees, keeping in mind that $1BTC = 10^8 sat$.

Remark 1 (*Do All the Miners See the Same Mempool?*)**.** Blockchain is a peer-to-peer network and information propagates thanks to a controlled flooding mechanism. A transaction is firstly accounted by a miner and then it is broadcasted to the others. Technically speaking, there is the possibility that the Mempools seen by the miners are not exactly the same. The Bitcoin Network Monitor[7] [10] shows that within 16 s at least 90% of the miners are ready to announce a newly generated transaction (so they have surely received it before) and the block propagation delay is within 2 s. Thus, these delays are reasonably small to support our assumption coherently with other works in this field [18,19,21,23,24]. Another aspect that we should bare in mind is that the protocol does not specify which transactions a miner has to select from the Mempool. However, the fact that the most widely used software for mining applies the greedy approach on the selection of the most profitable transactions supports our assumption.

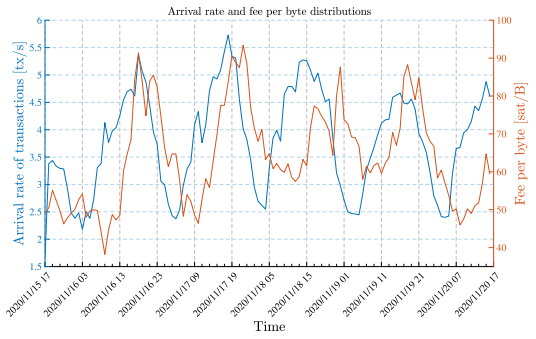
Remark 2 (*Network Neutrality in Bitcoin*)**.** Recent research [28,29] has found that mining pools may show bias towards selecting transactions from their own pool when creating new blocks, which could potentially compromise the neutrality of the process. While the protocol allows miners the freedom to choose which transactions to include, this behavior raises concerns about fairness. However, despite this phenomenon, the correlation between offered transaction fees and expected confirmation time remains consistent and predictable, as reported by various Bitcoin mempool monitors. We show one[8] in Fig. 2. The greedy behavior of miners is evident since at each block consolidation, the most valuable transactions are removed with strict priority.

Remark 3 (*Transaction Sizes and Segwit*)**.** Since 2015, the SegWit standard has become more and more used in BTC blockchain. Nowadays, the vast majority of miners accept the SegWit transactions as witnessed by the fact that basically all the recent non-empty blocks contain at least one SegWit transaction. The

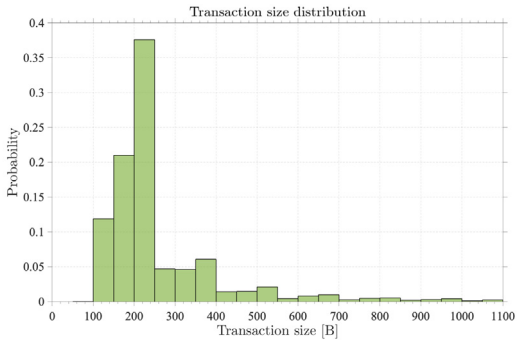
6 Bitcoin Improvement Proposal 141: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.

7 https://www.dsn.kastel.kit.edu/bitcoin/

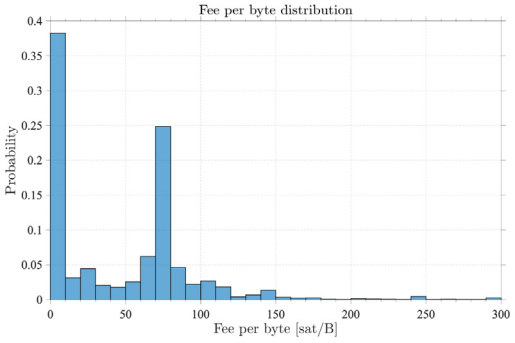
8 https://jochen-hoenicke.de/queue



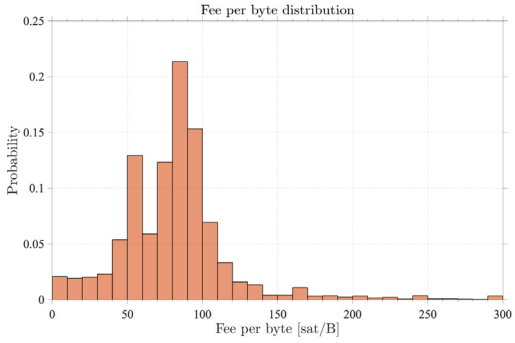
(a) Arrival rate and fee per byte as functions of the time.



(b) Empirical probability density function of transaction size.



(c) Empirical probability density function of fee per byte in moderate workload conditions.



(d) Empirical probability density function of fee per byte in heavy workload conditions.

Fig. 3. Data retrieved from the BTC blockchain analysis.

idea behind this standard is that part of the transaction data can be stored in a parallel chain and hence the size of 1 MB per block becomes less restrictive. At the moment, the percentage of SegWit transactions is approximately 50%. In determining the distribution of the transaction size, we considered only the size of the data to be stored in the block subject to 1 MB limitation, because this is what determines the maximum throughput of the blockchain. Coherently, when the miners compute the fee-per-byte ratio they use the effective space occupied in the block rather than the total transaction size.

It is expected that the percentage of SegWit transactions will increase in the next months given the trend of the fees observed in the last few months.

3.3. Problem statement and motivation

Technically, there is no limit to the fee that blockchain users can offer for their transactions. However, there exists a natural tension between the need of reducing the operating costs and the confirmation delay that a user accepts to wait. Indeed, if a transaction includes a payment (obviously in cryptocurrency), the users expect a short confirmation delay because the high fluctuations of the cryptocurrency value may affect the economical conditions of the deal. This is even more evident if the transactions are associated with financial speculation on trading cryptocurrencies. The problem is enhanced by the fact that the offered fee cannot be changed once the transaction is in the Mempool. However, in some other cases, the transaction will store in the blockchain some delay tolerant data and hence the fee offered can be drastically smaller than that needed in the previous case.

As a consequence, users need a method to tackle the trade-off between the cost of processing the transaction and its confirmation delay.

Nowadays, this problem is only partially covered by built-in methods. Current methods of optimal fee determination include Monte Carlo simulations as well as history-based approaches, e.g., 'estimatesmartfee' function in the Bitcoin core methods.[9]

Fig. 3(a) shows the arrival process intensity at the BTC Mempool and the expected offered fees between *2020/11/15* to *2020/11/20*. The plots are based on the statistics collected on over *1.5* million transactions seen at our BTC node.

We observe that the arrival process (blue line) in BTC blockchain is subject to high fluctuations. Moreover, the fee-per-byte ratios of transactions (orange line) tends to reflect the behavior of the arrival process with a delay of approximately 3 h. This is due to the reactive nature of the current fee estimation algorithms based on the past statistics to predict the best fee.

In contrast, the prediction queueing model that we propose is proactive, and reacts as soon as the occupancy of the Mempool or the arrival rate grows.

Fig. 3(b) shows the distribution of sizes of pending transactions in the observed period of time. Most of the size values are located between 100 and 250 bytes which is about 70% of all pending transactions in the Mempool. The probability of finding the transaction with greater size drops dramatically for sizes larger than 400 bytes.

Figs. 3(c) and 3(d) show empirical probability density function of fee-per-byte ratios for two periods of time with moderate and heavy workload conditions respectively. The plots support the intuition that, when the load is moderate, there is a lower competition for accessing the new blocks, hence the fee-per-byte ratio tends to be as small as possible. Indeed, in moderate load, almost 40% of the transactions offer a fee-per-byte just above 0 sat/B.

Conversely, in heavy load conditions (see Fig. 3(d)), users offer higher fees to solicit miners to select their transactions for inclusion in the next blocks. The majority of the transactions

[9] https://bitcoin.org/en/download

have fee-per-byte values between 50 and 100 sat/B, with a peak around 85 sat/B.

This paper proposes a queueing model that, given the traffic intensity, the distribution of the fee per byte and the state of the Mempool, predicts the statistics of the number of blocks required to confirm a transaction offering a certain fee. Although we will mainly focus on the estimation of the expected number of blocks for transaction confirmation, the method can be extended to address the estimation of successive moments. The first moment can be derived with a finite number of operations given the root of a certain polynomial, while successive moments require the truncation of a power series and hence can be used to obtain approximate results.

Remark 4 (*How to Measure the Confirmation Delay?*)**.** In this work, we measure the confirmation delay in number of blocks rather than in seconds. This is coherent with the needs of the blockchain users as witnessed by the active services of fee prediction. For example, the reactive service implemented in the main software for BTC usage, Bitcoin Core,[10] is used by the wide majority of users and implements the smartfeeprediction service based on historical data. This service returns the expected number of blocks for confirmation given a certain fee. Analogously, external private services offer predictions with other methods (e.g., by using Monte Carlo simulation) but always expressing the confirmation delay in number of blocks. This is explained by the fact that the meaningful events in the blockchain are those associated with the transactions in the block. For example, a transaction offering a very high fee per byte is almost sure to enter in the next available block but is still subject to the uncertainty of when that block will be mined. Still, it will overtake the other transactions in its confirmation and this is what is crucial for the system.

4. The queueing model and its solution

In this section, we first assume that a transaction arrives at the system offering the lowest possible fee, i.e., it will be included in a block only when all the other transactions in the Mempool at its arrival epoch and those that will arrive during its waiting time are confirmed.

After providing the model description and assumptions, we give a general solution based on generating function method. This consists in four phases:

1. Discretization of the continuous time Markov chain into a discrete time one. Intuitively, this corresponds to the observation of the states of the system immediately after each block consolidation. This is done in Section 4.1.
2. Derivations of the equations describing the system dynamics, i.e., the expected number of blocks to the confirmation given the initial Mempool occupancy. This is presented in Section 4.2.
3. Solution of the infinite set of equations derived in point 2 by resorting to the generating function method. This allows us to analytically derive the average performance indices as carried out in Section 4.3 where a numerical procedure for the computation of the expected confirmation time of the transactions is presented.
4. Finally, in Section 4.4, we extend our results to transactions offering an arbitrary fee.

10 http://bitcoin.org

4.1. Model description and notation

Transactions arrive at the Mempool according to a stationary Poisson process with intensity λ. The generation of blocks occur at the random times $t_0, t_1, \ldots$ and we have:

$$Pr\{t_{n+1} - t_n \leq x\} = 1 - e^{-\mu x}, \quad \forall n \geq 0, \tag{1}$$

i.e., the time between two consecutive block consolidations is exponentially distributed with rate μ, e.g., for BTC $\mu = 6$ blocks per hour. Each block contains at most B transactions and consumes all the possible transactions in the Mempool, i.e., it is generated even if it is not completely full.

For the moment, we assume that all the transactions offer the same fee with the exception of a tagged transaction that offers less than all the others, i.e., it will be processed only when there is not any other transaction to be included in the block. The order of service of the non-tagged transactions is irrelevant.

The service policy adds the transactions to the next batch as soon as they arrive, if some space is available. In other words, we can imagine that the system first draws the next block consolidation time and then selects from the Mempool B transactions (if available) to serve that may include those arrived between the previous and the current consolidations.

Let $\eta(t)$ be the number of transactions in the Mempool at time t, with $\eta(0) = Y$, $Y \geq 1$ be its occupancy at the tagged transaction arrival, including the transaction itself.

In order to work in a discrete time setting, let:

$$\eta_n \triangleq \eta(t_n),$$

i.e., η_n is the number of transactions in the Mempool immediately after the consolidation of the nth block after the tagged transaction arrival. So, our time slot begins immediately after a new block generation and finishes immediately with the consolidation of the next one. From a queueing theory perspective, we are taking an arrival-before-service approach for the discretization of the system's time (see, e.g., [11]), thus we have $\eta_0 = Y$.

The collection of random variables $\{\eta_n : n \geq 0\}$ is a discrete time Markov chain (DTMC) since it trivially satisfies the Markov property [30]. Define the probability that the Mempool will be empty after n steps, given the initial state Y as:

$$P_Y^n \triangleq Pr\{\text{State 0is reached for the first time in exactly } n \text{ transitions } | \eta_0 = Y\}.$$

We observe that the distribution a_j of the number of arrivals between the consolidation of two consecutive blocks is given by:

$$a_j = \mu \int_0^\infty \frac{(\lambda t)^j}{j!} e^{-(\lambda+\mu)t} dt = \frac{\mu}{\lambda+\mu} \left(\frac{\lambda}{\lambda+\mu}\right)^j,$$

i.e., a_j, as expected by the memoryless property of the service and arrival process, forms a geometric distribution with $\alpha \triangleq \lambda/(\lambda+\mu)$ and $\beta \triangleq 1-\alpha$. Henceforth, we rewrite a_j as:

$$a_j = \beta\alpha^j.$$

Notice that the probability of receiving strictly less than j transactions in a time slot is:

$$1 - \sum_{k=j}^{\infty} a_k = 1 - \alpha^j.$$

4.2. Solution of the model

The main result of this section is Theorem 1 which gives the expected confirmation time as function of the model parameters. Its proof is based on a set of lemmata, the most important of

which are presented in this section. The detailed proof of the theorem is reported in Appendix D.

First, we consider the case $n = 1$. We may easily write P_j^1 as:

$$P_j^1 = \begin{cases} 1 - \alpha^{B-j+1} & \text{if } j \le B \\ 0 & \text{if } j > B \end{cases}. \tag{2}$$

Let us consider the case $n > 1$. The first step analysis of the DTMC allows us to write the following equations:

$$P_j^n = \sum_{k=\max(1,j-B)}^{\infty} P_k^{n-1} a_{k-j+B} = \sum_{k=\max(1,j-B)}^{\infty} P_k^{n-1} \beta \alpha^{k-j+B}. \tag{3}$$

For $n = 2$, we can easily derive P_j^2. In fact, using Eq. (3), for $j \le B$, we have:

$$P_j^2 = \sum_{k=1}^{\infty} P_k^1 \beta \alpha^{k-j+B} = \sum_{k=1}^{B} P_k^1 \beta \alpha^{k-j+B} = \alpha^{B+1-j}(1 - \alpha^B(B + 1 - \alpha B)).$$

For $B + 1 \le j \le 2B$, we obtain similarly:

$$P_j^2 = 1 - \alpha^{2B-j+1}(1 + (1 - \alpha)(1 + 2B - j)).$$

Clearly, for $j > 2B$, $P_j^2 = 0$.

In general, we rewrite Eq. (3) for $n \ge 2$ as stated by the following lemma.

Lemma 1. *For $n \ge 2$, the system of Eq. (3) can be rewritten as:*

$$\begin{cases} \alpha P_j^n = P_{j-1}^n & 2 \le j \le B + 1 \\ \alpha P_j^n = P_{j-1}^n - \beta P_{j-B-1}^{n-1} & j > B + 1 \end{cases} \tag{4}$$

Proof. Let us consider $2 \le j \le B$. Using Eq. (2), we obtain:

$$P_j^n = \sum_{k=1}^{\infty} P_k^{n-1} \beta \alpha^{k-j+B} = \frac{1}{\alpha} P_{j-1}^n.$$

Similarly, we have:

$$P_{B+1}^n = \sum_{k=1}^{\infty} P_k^{n-1} \beta \alpha^{k-B-1+B} = \frac{1}{\alpha} P_B^n.$$

For $j > B + 1$, we have:

$$P_j^n = \sum_{k=j-B}^{\infty} P_k^{n-1} \beta \alpha^{k-j+B} = \sum_{k=j-B-1}^{\infty} P_k^{n-1} \beta \alpha^{k-j+B} - \frac{\beta}{\alpha} P_{j-B-1}^{n-1} = \frac{1}{\alpha} P_{j-1}^n - \frac{\beta}{\alpha} P_{j-B-1}^{n-1}. \quad \square$$

Under stability condition $\lambda < B\mu$, i.e., $\alpha < B/(B+1)$, the states of the process are all positive recurrent, i.e., starting from any state j we reach state 0 with probability 1 in a finite expected time. Thus, P_j^n, given j, is a probability distribution and we can introduce its probability generating function:

$$P_j(w) \triangleq \sum_{n=1}^{\infty} P_j^n w^n,$$

where $w \in \mathbb{C}$ and $|w| \le 1$. We can multiply each equation for P_j^n of System (4) by w^n and summing up, we obtain for $2 \le j \le B+1$:

$$\alpha \left(P_j(w) - P_j^1 w\right) = P_{j-1}(w) - P_{j-1}^1 w. \tag{5}$$

For $j > B + 1$, we have:

$$\alpha \left(P_j(w) - P_j^1 w\right) = P_{j-1}(w) - P_{j-1}^1 w - \beta w P_{j-B-1}(w). \tag{6}$$

Let us introduce the following generating function [11] for $z \in \mathbb{C}$ and $|z| < 1$:

$$P(z, w) \triangleq \sum_{j=1}^{\infty} P_j(w) z^j,$$

and we sum Eqs. (5) and (6) multiplied by z^j for $j \ge 2$. Thus, we have:

$$\sum_{j=2}^{\infty} \alpha(P_j(w) - P_j^1 w) z^j = \sum_{j=2}^{\infty} (P_{j-1}(w) - P_{j-1}^1 w) z^j - \beta w \sum_{j=B+2}^{\infty} P_{j-B-1}(w) z^j$$

This can be conveniently rewritten as:

$$\alpha P(z, w) - \alpha z P_1(w) - \alpha w \sum_{j=1}^{\infty} P_j^1 z^j + \alpha w z P_1^1 = z P(z, w) - wz \sum_{j=1}^{\infty} P_j^1 z^j - \beta w z^{B+1} P(z, w). \tag{7}$$

Now, observe that:

$$\sum_{j=1}^{\infty} P_j^1 z^j = \sum_{j=1}^{B} (1 - \alpha^{B-j+1}) z^j = \frac{z\left(\alpha - \alpha^{B+1}(1 - z) - z + (1 - \alpha) z^{B+1}\right)}{(1 - z)(\alpha - z)} \triangleq h(z).$$

We can simplify Eq. (7) as:

$$P(z, w)(\alpha - z + \beta w z^{B+1}) = (\alpha w - wz) h(z) + \alpha z P_1(w) - \alpha w z P_1^1,$$

and obtain the expression for $P(z, w)$:

$$P(z, w) = \frac{w(\alpha - z) h(z) + \alpha z P_1(w) - \alpha w z P_1^1}{\alpha - z + \beta w z^{B+1}}, \tag{8}$$

that depends on the unknown function $P_1(w)$.

Lemma 2. *The denominator of the right-hand side of Eq. (8) has only one zero ξ (that depends on w) in the open unitary disk if the stability condition $\alpha < B/(B + 1)$ holds, $|w| \le 1$ and $\alpha \ne j/(j + 1)$ for $j = 1, \ldots, B$.*

The proof is given in Appendix A.

It is worth of notice that the Lemma does not follow from an immediate application of Rouché theorem as it would be in the domain $|w| < 1$ and $|z| \le 1$. Lemma 2 requires us to avoid some values of α. In practice, this is not a problem given the continuous nature of α, and we will deal with them by resorting to a continuity argument on the performance indices.

Since, by definition, $P(z, w)$, converges for all the values $|w| \le 1$ and $|z| < 1$, ξ must also be a zero of the numerator of Eq. (8). Thus, we can express $P_1(w)$ as:

$$P_1(w) = \frac{\alpha w \xi P_1^1 - w(\alpha - \xi) h(\xi)}{\alpha \xi}.$$

Let us introduce an auxiliary function:

$$f(\xi) \triangleq \frac{\alpha w \xi P_1^1 - w(\alpha - \xi) h(\xi)}{\alpha \xi}, \tag{9}$$

where $w = (\xi - \alpha)/(\beta \xi^{B+1})$. By Lagrange's theorem [31], we can rewrite $f(\xi)$ as:

$$f(\xi) = f(\alpha) + \sum_{t=1}^{\infty} \frac{(\beta w)^t}{t!} \left[\frac{\partial^{t-1}}{\partial x^{t-1}} \left(f'(x) x^{t(B+1)} \right) \right]_{x=\alpha}. \tag{10}$$

We can easily compute $f(\alpha)$ by substitution using Definition (9) that gives 0. We also have:

$$f'(x) = \frac{-\beta x^{B+1} + Bx^2 + (\beta - B(\alpha+1))x + \alpha B}{x^{B+1}\alpha(1-x)^2}.$$

We may conveniently rewrite $f'(x)x^{t(B+1)}$ as follows:

$$f'(x)x^{t(B+1)} = -\frac{\beta}{\alpha}\frac{x^{t(B+1)}}{(1-x)^2} + \frac{B}{\alpha}\frac{x^{(t-1)(B+1)+2}}{(1-x)^2} + \frac{\beta - B(\alpha+1)}{\alpha}\frac{x^{(t-1)(B+1)+1}}{(1-x)^2} + B\frac{x^{(t-1)(B+1)}}{(1-x)^2}. \quad (11)$$

Lemma 3. *Let:*

$$g(t) \triangleq \frac{\partial^{t-1}}{\partial x^{t-1}}\left(f'(x)x^{t(B+1)}\right)\Big|_{x=\alpha},$$

then, we have:

$$g(1) = \frac{1-\alpha^B}{1-\alpha},$$

and, for $t \geq 2$:

$$g(t) = \alpha^{B(t-1)}\left(B\alpha - \beta B^2 + \beta(1+B)^2 t\right)\frac{[(t-1)(B+1)]!}{[B(t-1)]!} - \alpha^{B(t-1)-1}\beta(\alpha-\beta B)\frac{[B(t-1)+t+1]!}{[B(t-1)]!(t+1)}\,{}_2F_1\left[\begin{matrix}1 & -B(t-1)\\ t+2\end{matrix}; -\frac{\beta}{\alpha}\right] - \alpha^{Bt-2}\beta B\frac{[t(B+1)]!}{(Bt)!(t+1)}\,{}_2F_1\left[\begin{matrix}2 & 1-Bt\\ t+2\end{matrix}; -\frac{\beta}{\alpha}\right], \quad (12)$$

where ${}_2F_1$ is the Gaussian hypergeometric function.

Although we do not provide the proof of this lemma, it can be easily derived by applying the algebraic properties of Gaussian hypergeometric functions.

By Lemma 3, we can write:

$$P_1(w) = (1-\alpha^B)w + \sum_{t=2}^{\infty}\frac{\beta^t w^t}{t!}g(t). \quad (13)$$

By taking the derivative of both sides of Eq. (13) evaluated in $w = 1$, we obtain the factorial moments of the distribution of the number of batches that have to be served in order to reach the absorption starting from state 1 (see, e.g., [11]). In general, the nth factorial moment of the distribution of the number of consolidations required to serve the tagged transaction when the queue contains $Y-1$ jobs at the arrival time ($Y \geq 1$) is:

$$M_n^Y = \frac{1}{Y!}\frac{\partial^Y}{\partial z^Y}\left(\frac{\partial^n P(z,w)}{\partial w^n}\Big|_{w=1}\right)\Big|_{z=0}.$$

However, since we do not have a closed form expression for $P_1(w)$, this expression should be considered to obtain an approximation of the factorial moments because Series (13) needs to be truncated. In Section 4.3, we show that the first moment can be obtained in an exact way thanks to a different approach to the computation of $P_1'(1)$ whose only numerical step consists in the computation of the real root of a polynomial inside the unit disk.

4.3. Numerical solution for the mean confirmation time

In this section, we derive the expression for the expected number of blocks that have to be consolidated before the tagged transaction is served.

In order to obtain M_1^Y, we are interested in the derivation of $P_1'(1)$, i.e., the expected number of steps to reach the absorbing state when the initial state is 1. Eq. (13) leads to the following expression:

$$P_1'(1) = 1 - \alpha^B + \sum_{t=2}^{\infty}\frac{\beta^t}{(t-1)!}g(t),$$

that unfortunately does not admit a known closed-form expression. However, $P_1'(1)$ can be derived in an alternative way that is more computationally efficient. Indeed, $P_1'(1)$ corresponds to the expected number of batches served during a busy period of the $M/M^B/1$ queueing system. The stationary distribution of this queueing system is well-known [21,32] and has a geometric distribution:

$$\pi_i = (1-\dot\rho)\dot\rho^i,$$

for each state $i \geq 0$, where $\dot\rho$ is the root inside the unit disk (which is known to be unique, real and positive in stability) of the polynomial:

$$\mu\rho^{B+1} - (\lambda+\mu)\rho + \lambda.$$

Notice that $\rho = 1$ is a root of the polynomial and there exists only one real root in $[0, 1)$. Therefore, $\dot\rho$ can be efficiently numerically derived thanks, e.g., to the bisection method.

The expected duration of the busy period can hence be obtained by observing that in steady-state π_0 represents the ratio between the expected idle period lengths and the sum of the expected idle and busy period lengths, thus obtaining:

$$P_1'(1) = \left(\frac{\dot\rho}{1-\dot\rho}\right)\frac{1}{\lambda} = \left(\frac{\dot\rho}{1-\dot\rho}\right)\frac{1-\alpha}{\alpha}\frac{1}{\mu}.$$

Indeed, the expected number of consolidated blocks during an idle/busy period must be proportional to their length. To easily see this, consider the epoch of beginning of an idle period (a renewal instant). By the memoryless property of the timers involved, the expected number of consolidated blocks during the idle period is μ/λ. Therefore, for a busy period of expected length $1/b$, the expected number of blocks must be μ/b since the process of block consolidation is a homogeneous Poisson process with rate μ.

We are now in position to state the main theorem that allows us to study this queueing system (see Theorem 1 below). Further factorial moments M_k^Y, for $k > 1$, may be derived in a similar way, although they will depend on $P_1^{(k)}(1)$, i.e., on the knowledge of the factorial moments greater than 1 for the busy period of the $M/M^B/1$ queueing system or alternatively, they may be approximated with a controlled truncation of the derivatives of the Series (13).

Theorem 1. *Let M_1^Y be the expected number of steps to reach the absorbing state when the queue satisfies the stability condition starting from state Y. Then, the following recursive scheme can be used to derive M_1^Y:*

$$\begin{cases} M_1^1 = P_1'(1) \\ M_1^{Y+1} = M_1^Y + \frac{T_{Y-1}}{\alpha^{Y-1}}\left(M_1^1 + \frac{\beta}{\alpha}\right) - \frac{T_Y}{\alpha^Y}M_1^1, \end{cases} \quad (14)$$

where:

$$T_Y \triangleq \sum_{c=0}^{\lfloor\frac{Y}{B+1}\rfloor}(-1)^{c+1}\binom{Y-Bc}{c}\alpha^{Bc}\beta^c. \quad (15)$$

The proof is given in Appendix D.

Remark 5 (*Numerical Stability of the Algorithm*)**.** In this remark, we discuss the numerical stability and complexity of the recursive algorithm that uses the equations of Theorem 1 to compute the average time to absorption.

We first notice that the binomial coefficient in Eq. (15) can be computed rather easily since index c ranges between 0 and $\lfloor Y/(B+1) \rfloor$. Thus, in practice, the lower index is much smaller than the upper one and when it grows, the upper one decreases quickly. Consequently, in our experiments we did not need to resort to Stirling's approximation.

Conversely, some problems of numerical instability may be caused by the subtraction present in Eq. (14). In fact, for low values of α, that translates in very low workload intensity, the recursive scheme becomes numerically unstable. This reflects the fact that for very low α, we have $\lambda \to 0$ and hence M_1^Y becomes a step function. In our experiments, we have not observed these problems for load factors of the system higher than 0.2. On the other hand, when the arrival process is very low, the prediction of the expected confirmation time may be simply approximated by dividing Y by the block size and then by taking the upper integer.

Finally, the computational complexity of the recursive scheme is bounded by $\mathcal{O}(Y^2)$.

Summarizing, Theorem 1 allows us to compute the expected number of block consolidations that take place until the transaction with the lowest fee density is included in a new block considering that there are exactly Y transactions in the Mempool at the time of its arrival.

The following subsection will describe the scenario when the tagged transaction has a fee-per-byte ratio other than the lowest one.

4.4. Extension of the model to transactions with arbitrary fee

So far, we have assumed that the tagged transaction offers the lowest fee per byte in the system. Now, we remove this assumption. Assume that f is the fee offered by the tagged transaction, and $F_1, F_2, \ldots$ are the continuous i.i.d. random variables associated with the fees per byte offered by all the other transactions. F_i are independent of the arrival times, therefore the arrival process formed by filtering out the transactions with fee lower than f, i.e., with probability $Pr\{F_i < f\}$, is still a Poisson process with intensity:

$$\lambda_f = \lambda \overline{F}(f),$$

where $\overline{F} = 1 - F(f)$ is the *complementary cumulative density function* (CCDF) of F_i. Now, let us consider the occupancy of Mempool. If we assume that there are Y pending transactions at the tagged transaction arrival, then we can count how many of these transactions offer a fee per byte higher than f. Let us call this number Y_f. Notice that the fee offered by the transactions in the Mempool is publicly known, thus Y_f is deterministic.

A transaction offering a fee per byte f, on average, has to wait a number of blocks that is given by $M_1^{Y_f}$ in a queueing system where the batch size is still B and the arrival rate is $\lambda\overline{F}(f)$. We will call $\lambda\overline{F}(f)$ and $\rho\overline{F}(f)$ the perceived arrival rate and load factor, respectively, where $\rho = \lambda/(B\mu)$.

If F is approximated by a discrete distribution, we have to deal with ties. In case of ties, transactions are usually served according to their arrival order, thus in determining Y_f, we must count all the transactions with a fee per byte higher or equal to f, while in determining the arrival intensity we count only the transactions with strictly higher values.

In the next section, we use the data presented in Figs. 3(c) and 3(d) to calculate the corresponding CCDFs that allow us to parameterize the model and carry out the numerical evaluation. Intuitively, by increasing the offered fee per byte f, the users obtain two major benefits: *(i) the reduction of the perceived occupancy in the Mempool* and *(ii) the reduction of the intensity of the perceived arrival process.*

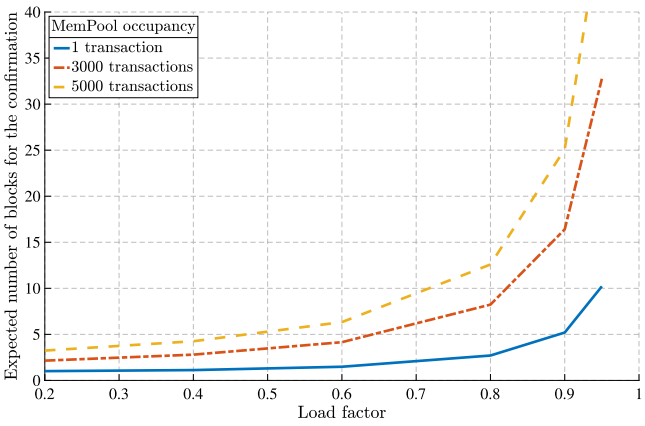


Fig. 4. Expected number of blocks for the confirmation with different number of transactions in the MemPool as function of load factor.

5. Numerical evaluation

In this section, we show some numerical experiments with our model and comment on the insights that they reveal on the system under study. Moreover, we resort to Monte Carlo simulation to test the robustness of the most important assumptions, and to a trace-driven simulation to compare the model predictions with real data.

5.1. Impact of the perceived load factor on the expected confirmation time

The perceived load factor depends on the fee offered by a transaction. In this set of experiments, we study its impact on the expected consolidation delay.

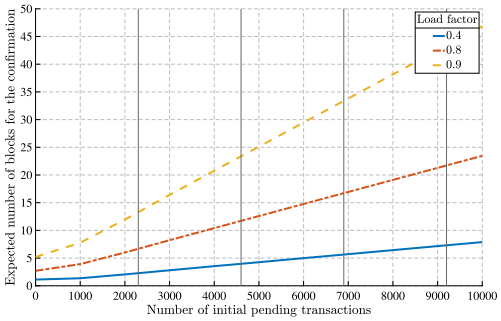
Fig. 4 shows the impact of the perceived load factor on the expected confirmation delay for different initial Mempool sizes. The figure reveals several insights about the system that we are studying. The first is that the initial Mempool size is very important to determine the average confirmation time, especially in heavy load conditions. This is due to the fact that, in order to serve the backlog found at the tagged transaction arrival time t_0, the Mempool accumulates the transactions arriving after t_0 but before the tagged transaction's confirmation instant. This creates an unfavorable working condition that moves the expected delay from 10 to 32 block consolidations in the cases of 1 or 3000 transactions in the Mempool, under a load factor of 0.95.

This supports the idea that, in heavy load, the knowledge of the initial Mempool state is crucial for an accurate prediction of the expected confirmation delay, and this is a novelty of our queueing model with respect to the state of the art.

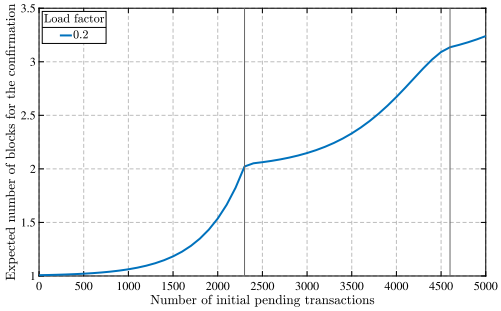
Another important observation is that expected confirmation time tends to grow to infinity as the load factor approaches 1. This is explained by the fact that when $\rho \geq 1$, the Markov chain underlying the model is not positive recurrent, therefore, starting from any initial state, there is a positive probability of having an infinite consolidation delay. It is possible to study the same model to determine the probability of confirmation in case of $\rho \geq 1$, but this is out of the scope of this work.

5.2. Impact of the initial Mempool state on the expected confirmation time

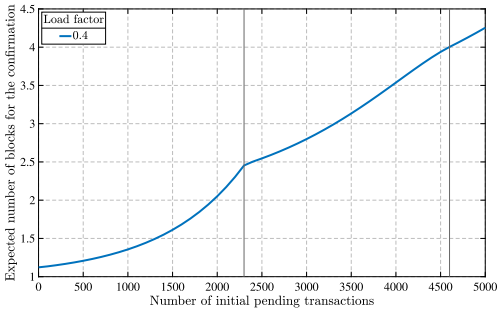
In the previous experiment, we have already discussed the importance of the Mempool state at the tagged transaction arrival. This observation is even more evident thanks to the plots



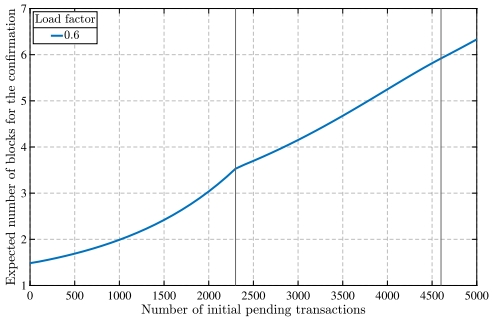
(a) Expected number of blocks for the confirmation as a function of number of initially pending transactions for different load factors.



(b) Expected number of blocks for the confirmation as a function of number of initially pending transactions for $\rho = 0.2$.

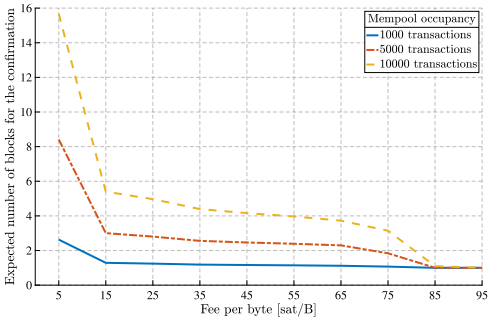


(c) Expected number of blocks for the confirmation as a function of number of initially pending transactions for $\rho = 0.4$.

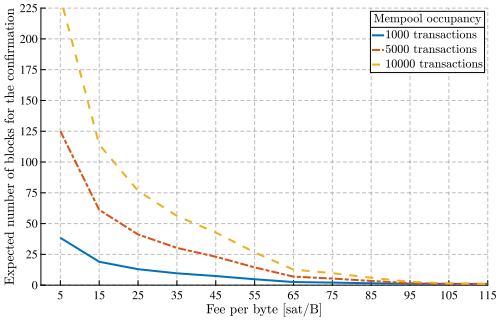


(d) Expected number of blocks for the confirmation as a function of number of initially pending transactions for $\rho = 0.6$.

Fig. 5. Impact of the initial Mempool size on the expected confirmation delay.



(a) Expected number of blocks for the confirmation as a function of fee per byte in moderate workload conditions.



(b) Expected number of blocks for the confirmation as a function of fee per byte in heavy workload conditions.

Fig. 6. Expected confirmation delay for different transaction fees per byte.

of Fig. 5(a). It is interesting to observe that the function describing the expected confirmation delay given the initial number of pending transactions has abrupt changes in its growth for values corresponding to integer multiples of the block size. This is clearly shown by Figs. 5(b)–5(d). This characteristic is less evident with higher load factors, and, if we assume the limiting case $\lambda \to 0$, this function becomes a step function with unit increase at B, $2B$, $3B$ and so on.

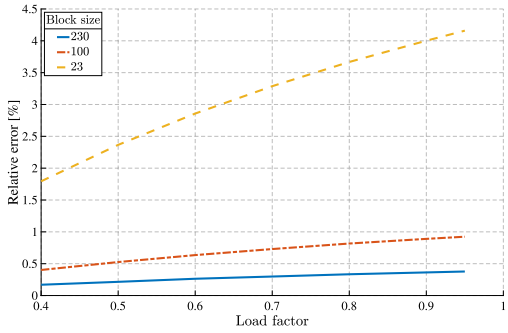
5.3. Impact of the transaction fee on the expected confirmation time

From a practical point of view, the main goal of the model is that of supporting the decision on which fee to offer to confirm a transaction with a given expected delay. Figs. 6(a) and 6(b) show the expected confirmation delay as function of the offered fee for the tagged transaction, in moderate and heavy load. For the plot of Fig. 6(a), we have used the fee distribution of Fig. 3(c), while for that of Fig. 6(b) we have used the distribution of Fig. 3(d). We should bear in mind that when we increase the offered fee we reduce both the perceived arrival rate and the occupancy of the Mempool. This explains the fast decrease of the expected confirmation delay shown in the plots. In case of moderate load, with 85 sat/B we reach an expected confirmation delay close to 1 block, while higher values are required in case of heavy load.

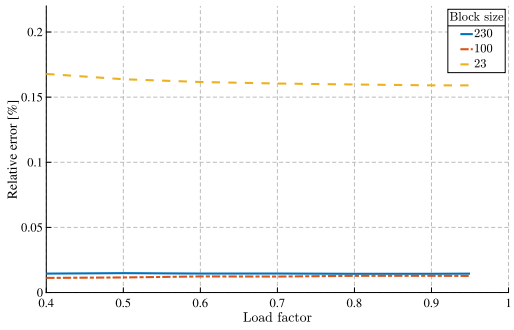
This observation is coherent with the distribution of the offered fees that we measured. In fact, in moderate load, 85 sat/B are sufficient to be confirmed quickly, while for heavy load one needs to almost double this offer.

5.4. Validation of the model

In this subsection, we present some experiments aimed at showing the robustness of the model. We answer three questions: is it accurate to estimate the expected confirmation delay by considering a blockchain with smaller block size but same load factor? Does the replacement of a random block size with its



(a) Relative error on the computation of M_1^1 obtained by re-scaling the block size.



(b) Relative error on the computation of M_1^{20000} obtained by re-scaling the block size.

Fig. 7. Analysis of the relative error obtained by re-scaling.

Table 1
Comparison of M_1^1 obtained analytically with fixed block size and simulation estimates with random block size. Confidence intervals are at 98% based on 15 independent experiments.

ρ	Model	Simulation	Relative error %
0.4	1.1205	$1.1234 \pm 3 \cdot 10^{-4}$	0.26
0.6	1.4803	$1.4813 \pm 3 \cdot 10^{-4}$	0.073
0.8	2.6937	$2.6728 \pm 3 \cdot 10^{-4}$	0.78
0.9	5.1808	$5.0341 \pm 1.5 \cdot 10^{-2}$	2.91

mean have a strong impact on the expected consolidation delay? Are the assumptions on the transaction arrival process robust with respect to the real system?

In order to answer these questions, we resort to Monte Carlo simulations whose data (e.g., transaction fees, arrival stream, transaction sizes) are retrieved from the real BTC network. Unfortunately, real measurements on the confirmation delay conditioned to the state of the Mempool, distribution of the offered fees and arrival rate are not possible because most of these information are not present in the blockchain logs.

5.4.1. Re-scaling of the block size

Currently, the BTC blockchain block can host on average approximately 2300 transactions. Other blockchain networks, as that of Bitcoin cash, allow for bigger blocks and hence there is space for more transactions.

If we imagine to cluster the transactions with approximately the same fee per byte into macro-transactions, we simplify the model by considering smaller blocks as well as a statistically smaller population in the Mempool. However, we can maintain an identical load factor. Clearly, the modified system is an approximation of the original one, nevertheless it is easier to study.

In Fig. 7, we show the relative error measured by the computation of M_1^1 and M_1^{20000} for the re-scaled block sizes B. We can see that the tenfold size downscale from $B = 2300$ to $B = 230$ produces practically indistinguishable results from the original model, while the relative error is more evident for smaller block sizes.

We conclude that, in the case of the BTC system, in order to speed up the computations for practical purposes, the block size can be safely re-scaled by a factor of 10 without losing significant precision in the obtained performance index.

5.4.2. Robustness of the deterministic maximum block size assumption

In the model of Section 4, we assume that B is fixed. As shown in Fig. 3(b), the transaction size may vary and hence the maximum capacity of a block is, in practice, a random variable. In this experiment, we aim at assessing the error in the computation of M_1^1 that we commit using our simplifying assumption, then from M_1^1 we can derive all the other M_1^Y. Thus, for this experiment, for each transaction, we sample its size from the empirical distribution shown in Fig. 3(b) and proceed by selecting from the Mempool the largest amount of transactions (ordered by offered fee per Byte in descending order) until we reach the maximum block size of 1 MB. In this process, we take into account technicalities such as Segwit transactions, i.e., the possibility of transactions to store part of their information outside the block (and hence allowing a larger number of elements in the block).

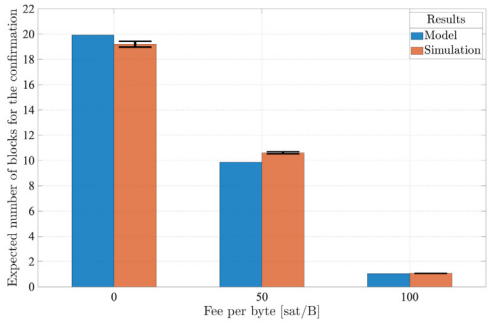
The model results are compared with estimates obtained with Monte Carlo simulation (see Table 1). The simulations consist of 15 independent experiments each of which consists of 10^6 independent runs.

Indeed, the relative error remains below 3% and is more evident in heavy load. What is more, the accuracy is essentially maintained even for larger values of Y. As we consider an initial Mempool size of 20,000 transactions and a load factor of $\rho = 0.9$, the model predicts that expected consolidation delay is 91.6 blocks while the Monte Carlo simulation using transaction sizes taken from real data estimates 87.08 with an error of 5.1%. Notice that the stochastic simulation with fixed block size gives 91.44, fully confirming the model prediction.

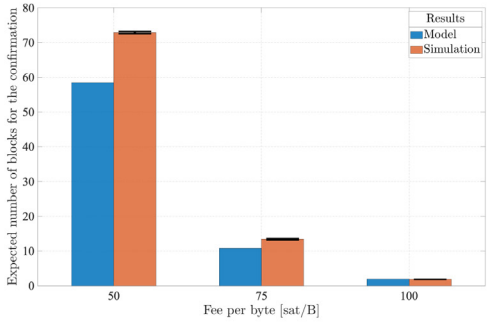
A relative error of 5.1% on 87.08 blocks is widely tolerable, since 87 blocks require 14 h on average to be consolidated, and the introduced noise, e.g., by the fluctuation of the arrival rate, surely has a higher impact. Thus, we conclude that considering the queueing model with random sized batches is clearly an intriguing mathematical problem, yet it does not significantly improve the applicability of the results.

5.4.3. Comparison with trace-driven simulation

So far, we have assumed that the arrival process is a time-homogeneous Poisson process with intensity λ. In this experiment, we evaluate the accuracy of the model prediction by resorting to trace-driven Monte Carlo simulation. To this aim, we have collected the arrival timestamps of the transactions for 5 days in the BTC mining node. Starting from a certain t_0 in the collection, we estimate the arrival rate measured in the interval $[t_0-7200, t_0)$, where time units are seconds. The generation times of blocks and the number of transactions that they contain are exponentially distributed with mean 600s and obtained from the distribution of Fig. 3(b), respectively. Offered fees per byte are sampled from the distributions depicted by Fig. 3(d) since we are interested in studying the system in heavy load. The initial number of transactions in the Mempool is determined according to the offered fee: if with the lowest fee we run an experiment with Y_0 transactions in the Mempool, we assume that $Y_f = \overline{F}(f)Y_0$. Notice that the trace-driven arrival stream is the same



(a) Expected number of blocks for the confirmation as function of fee per byte for model and simulation results with $Y = 6,000$ and $\lambda = 3.21$ transactions per second.



(b) Expected number of blocks for the confirmation as function of fee per byte for model and simulation results with $Y = 12,000$ and $\lambda = 4.02$ transactions per second.

Fig. 8. Comparison between model results and trace-driven simulation estimates.

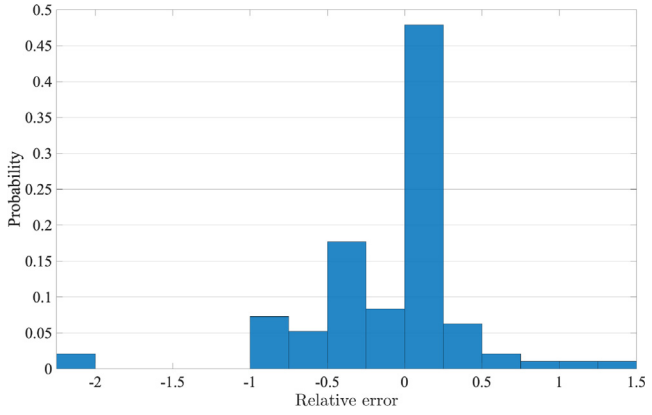


Fig. 9. Empirical probability density function of relative error of actual and predicted confirmation delays.

for all the experiments. This is due to the difficulty of finding a sufficiently large set of initial times in our time series that recreate the same initial conditions.

The first experiment is done by measuring $\lambda = 3.21$ transactions per second, i.e., $\rho \simeq 0.85$. In this case, transactions offering 0 sat/B[11] are almost sure to be confirmed. We assume that the Mempool contains 6000 transactions. Fig. 8(a) shows the model predictions and the simulation estimates assuming that fees of 0, 50 or 100 sat/B are offered. We can see that, in these cases, there is an excellent agreement among the data.

In order to stress our model, we consider a situation in which the measured arrival rate is $\lambda = 4.02$ tx/s and we assume the Mempool to contain 12,000 pending transactions. It is worth of notice that the cheapest transactions may be not confirmed (the protocol implementations usually evict cheapest transactions when the Mempool size exceeds a certain threshold or when they stay in the queue longer than 3 days) since $\rho > 1$. Thus, we consider a minimum fee of 50 sat/B.

Fig. 8(b) shows the comparison between the model predictions and the simulation estimates. We can see that, in this case, the precision is lower. This can be explained by several factors. First, the working condition of the queue is closer to saturation (with 50 sat/B) than what we had in the first experiment, and small variations in the arrival rate can have stronger impacts on the confirmation time. The second reason is connected to the fact that, by investigating the time series of the arrivals, there is the moment of extremely heavy load (5 tx/s) after our starting point which is reached for fees of 50 and 75 sat/B due to the large backlog of transactions in the Mempool. Given the unfavorable conditions, the model manages to maintain a relative error below 20%. Moreover, we believe that the accuracy can be further increased with appropriate techniques of workload predictions.

5.5. Validation of the prototype of the confirmation time estimator

One application of our results consists in the development of a confirmation time predictor. This service monitors the blockchain status (Mempool occupancy, arrival rate and fee distribution) and uses Theorem 1 to predict the average number of blocks required to confirm a transaction given its fee.

In this section, we carry out an experiment aimed at assessing the accuracy of the model forecast. For this purpose, we collected data about pending transactions occurring in the Mempool of our installed node for 96 h. Further, we randomly picked one transaction for each hour in our dataset with offered fee-per-byte ratios between 10 and 20 sat/B. This interval includes the majority of all transactions. Finally, for every transaction from the sample, we first calculated the predicted confirmation time measured in blocks and compared it with the actual number of blocks after which the transaction appeared in the block ledger. Among the selected 96 transactions only 2 were not confirmed while all the other 94 appeared in a block.

Recall that the model provides the prediction of the expected number of blocks for the confirmation. Thus, it is normal to observe a certain discrepancy between real and analytical data for a few transactions. This is can be seen in Fig. 9. On the x-axis we report the relative error computed as the difference between the value measured in the real system and the corresponding analytical prediction, divided by the first. The y-axis reports the percentage of transactions that fall in a certain interval of relative error. It is interesting to note that almost 50% of transactions have a prediction with negligible relative error. Overall, the mean of the relative errors for all the considered transactions is very small and equals to -0.13. Finally, we notice that the histogram resembles the bell shape and this supports the applicability of our approach in real world scenarios.

6. Conclusion

In this paper, we propose a transient analysis of a $M/M^B/1$ queueing model that allows the definition of a new method for estimating the expected transaction confirmation time in blockchain based on PoW. The model uses 3 key parameters: the observed state of the Mempool, the current arrival intensity and the distribution of the offered fees. With respect to the queueing models proposed at the state of the art, we take into account the initial state of the Mempool and the numerical experiments have

[11] BTC miners usually avoid transactions whose fee is 0 to prevent flooding attacks, therefore the actual fee is just above 0.

shown that this has a strong impact on the estimations. In fact, the stationary analysis proposed in [18,19,21] cannot depend on the initial state of the queue, but this means that they ignore an important piece of information that is in practice available to the users.

Although the model was studied on the Bitcoin network, it can be applied for any kind of PoW-driven blockchains where transactions are confirmed according to an auction on the fees.

From the application point of view, the proposed algorithm is generally much faster than those based on Monte Carlo simulations. In fact, the simulation of the $M/M^B/1$ model, with large B and in heavy load can exhibit a quite slow convergence.

Notice that, since the confirmation time is monotonic non increasing with the offered fee, determining the optimal offered fee for a given expected confirmation time requires only a few computations (e.g., thanks to the bisection method).

The results of the models have been compared with trace-driven simulations under heavy and very heavy workloads. The accuracy is generally very good, although it may deteriorate for long-term predictions in very heavy load since small errors in the estimations of the arrival intensity may cause important changes in the validation delay.

In order to improve the accuracy of the model, it seems promising to try to predict the arrival rate during the consolidation delay. This and the study of the probability of consolidation when the load factor $\rho \geq 1$ will be subjects of future work.

Finally, we would also like to apply the model for the analysis of general scheduling disciplines in which customers are ordered on the base of some strong priority rule. In these cases, it will become important to derive the expression of the waiting time in the continuous model that may be non-trivial in general.

CRediT authorship contribution statement

Ivan Malakhov: Data curation, Software, Validation, Writing – review & editing. **Andrea Marin:** Supervision, Conceptualization, Methodology, Writing – original draft. **Sabina Rossi:** Supervision, Formal analysis, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Andrea Marin and Sabina Rossi reports financial support was provided by Government of Italy Ministry of Education University and Research.

Data availability

Public data from the blockchain has been used.

Acknowledgments

This work is partially supported by the Project NiRvAna: Noninterference and Reversibility Analysis in Private Blockchains (20202FCJMH) funded by MUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2020 , and by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

Appendix A. Proof of Lemma 2

We consider three cases: $|w| < 1$, $w = 1$ and, $|w| = 1 \wedge w \neq 1$.

Case $|w| < 1$. We apply Rouché's theorem to equation $z = \alpha + \beta w z^{B+1}$. Notice that z^{B+1}, with $|z| \leq 1$, is analytic in the closed unitary disk and that:

$$|\beta w z^{B+1}| < |z - \alpha|$$

on the disk perimeter. Thus, the equation has a unique root ξ is in the open unitary disk.

Case $w = 1$. In this case, we need to prove that the polynomial $\alpha - z + (1-\alpha)z^{B+1}$ has exactly one root inside the unit disk. We resort to Theorem [33, Thm 2.1] reported below.

Theorem 2.1 ([33]). *Let $a > b > 0$ be real numbers and let $n > m > 0$ be integers. Then, the number of zeros of $bz^n - az^m + a - b$ strictly inside the unit disk is $m - gcd(m, n)$ if $a/b \geq n/m$ and m otherwise.*

In our case, we have $n = B+1$, $m = 1$, $a = 1$ and $b = (1-\alpha)$, hence the conditions of the theorem are trivially satisfied. Notice that $\alpha < B/B+1$ is equivalent to $(1-\alpha)^{-1} < B+1$, which implies are only $m = 1$ roots in the unit disk as required.

Now, we consider the case in which $|w| = 1$, but $w \neq 1$. We restrict our analysis to the case $B > 2$, since $B = 1$ and $B = 2$ can be easily considered given the relatively simple closed-form expressions of the roots of the corresponding polynomials of second and third degree.

Given a polynomial $f(z)$, let us define $f^*(z)$ as follows:

$$f^*(z) = z^n f(1/\bar{z}),$$

where n is the degree of f and $\bar{z}$ denotes the conjugate of z. Let $f_0(z) = \alpha - z + (1-\alpha)e^{i\theta}z^{B+1}$, then we define the following sequence of polynomial as in [34, Ch. X]:

$$f_{j+1} = \bar{a}_0^{(j)} f_j(z) - a_{n-j}^{(j)} f_j^*(z),$$

for $j = 0, \ldots, B$, where $a_k^{(j)}$ denotes the coefficient of the term z^k in f_j. Notice that the degree of f_{j+1} is always strictly lower than that of f_j and that, in our case, $a_k^{(j)} \in \mathbb{R}$ for $j = 0, \ldots, B+1$, hence we can ignore the conjugate on $a_0^{(j)}$.

Let $P_k = a_0^{(1)} a_0^{(2)} \cdots a_0^{(k)}$, with $k = 1, \ldots, B+1$, then the number of roots inside the unit disk is given by the number of negative elements in the collection $P_1, \ldots, P_{B+1}$ [34, Thm. 42,1].

In order to conclude the proof of Lemma 2, we need Lemmas 4 and 5 reported below and proved in Appendices B and C, respectively.

Lemma 4. *Let $\Psi_{j,\alpha} \triangleq (j+1)\alpha - j$ for $j = 1, 2, \ldots$. Then, we have:*

$$a_0^{(1)} = \Psi_{1,\alpha}\,,\quad a_1^{(1)} = -\alpha\,,\quad a_B^{(1)} = (1-\alpha)e^{i\theta}$$

and, for $2 \leq j < B$:

$$a_0^{(j)} = \alpha^{2^{j-2}} \Psi_{j,\alpha} \prod_{k=1}^{j-2} \Psi_{k,\alpha}^{2^{j-2-k}}\,, \tag{A.1}$$

$$a_1^{(j)} = -\alpha^{2^{j-2}} \Psi_{j-1,\alpha} \prod_{k=1}^{j-2} \Psi_{k,\alpha}^{2^{j-2-k}}\,,\quad a_{B+1-j}^{(j)} = \alpha^{2^{j-2}}(1-\alpha) \prod_{k=1}^{j-2} \Psi_{k,\alpha}^{2^{j-2-k}} e^{i\theta}$$

where all the unspecified coefficients are set to 0.

The proof of Lemma 4 is given in Appendix B.

Observe that $f_{B+1}(z)$ is a constant, and the following lemma provides its value.

Lemma 5. *For $j = B$ and $j = B+1$ we have the following coefficients:*

$$a_0^{(B)} = \alpha^{2^{B-2}} \Psi_{B,\alpha} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-2-k}}\,, \tag{A.2}$$

$$a_1^{(B)} = \alpha^{2^{B-2}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-2-k}} \left((1-\alpha)e^{i\theta} - \Psi_{B-1,\alpha}\right) \tag{A.3}$$

$$a_0^{(B+1)} = 2\alpha^{2^{B-1}}(1-\alpha)\Psi_{B-1,\alpha}(\cos\theta - 1)\prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}}. \tag{A.4}$$

The proof of Lemma 5 is given in Appendix C.

To conclude the proof of Lemma 2, we resort to Marden's theorem [34, Thm 42,1]. Consider the following sequence of products:

$$P_j = \prod_{k=1}^{j} a_0^{(j)}, \quad j = 1, \ldots, B+1.$$

If they are all different from zeros, the number of roots of $f_0(z)$ inside the unit circle is equal to the number of negative elements in the collection P_1, ..., P_{B+1}. Assume now that $\alpha < B/(B+1)$, with $\alpha \neq j/(j+1)$ for all $j = 1, \ldots, B-1$. We begin by noticing that all the coefficients are not null, in particular $a_0^{(B+1)}$ is not null because we are considering $w \neq 1$, and hence $\theta \neq 2k\pi$ for all $k \in \mathbb{Z}$. In order to have a unique root of the polynomial inside the unit disk, we must have that if P_{j^*} is the first negative element of the sequence, then either $j^* = B+1$ or $a_0^{(j^*+1)} < 0$ and $a_0^{(h)} > 0$ for all $h > j^*+1$. Assume $\alpha < B/(B+1)$ and let us write:

$$\left(0, \frac{B}{B+1}\right) \setminus \left\{\frac{j}{j+n}, j = 1, 2, \ldots, B-1\right\} = \left(0, \frac{1}{2}\right) \cup \cdots \left(\frac{j}{j+1}, \frac{j+1}{j+2}\right) \cdots \cup \left(\frac{B-1}{B}, \frac{B}{B+1}\right)$$

Let I_j be the interval $((j-1)/j, j/(j+1))$ for $1 \leq j \leq B$ and suppose $\alpha \in I_j$. Notice that all $\Psi_{k,\alpha} > 0$ if $k < j$, while the remaining ones are negative. Therefore, $P_1, P_2, \ldots, P_{j-1}$ are positive and $j^* = j$ is the smallest index in the sequence such that $P_{j^*} < 0$. Suppose $j < B$ observe that $a_0^{(j+1)} < 0$ since in Eq. (A.1) we compute the product of Ψ_{j+1}, α which is negative and all the previous terms excluding $\Psi_{j,\alpha}$ while are positive by assumption. Thus $P_{j+1} < 0$. All the remaining elements of the sequence, i.e., for $k > +1$ are positive because they contain the product of two negative Ψs. In fact (A.4) because $\cos\theta - 1 < 0$, $\Psi_{B-1,\alpha} < 0$ and all the remaining factors in the product are raised at a positive power of 2. Now, assume that $j = B$ and observe that $a_0^{(B)}$ is negative by Eq. (A.2). Also $a_0^{(B+1)}$ is negative and thus $P_B < 0$ and $P_{B+1} > 0$. This concludes the proof of Lemma 2. □

Appendix B. Proof of Lemma 4

For $j = 1$, we have:

$a_0^{(1)} = \alpha^2 - (1-\alpha)^2 = 2\alpha - 1$,

$a_1^{(1)} = \alpha(-1) = -\alpha$, and $\alpha_B^{(1)} = -(-1(1-\alpha)e^{i\theta}) = (1-\alpha)e^{i\theta}$. For $j \geq 2$, we proceed by induction taking the case $j = 2$ as base. Indeed, we have:

$$a_0^{(2)} = [a_0^{(1)}]^2 - a_B^{(1)}\overline{a}_B^{(1)} = \alpha(3\alpha - 2) = \alpha\Psi_{2,\alpha},$$

as required. The expressions of the two remaining coefficients follow from their definitions in a similar way.

Consider now $j > 2$ and let us determine $a_0^{(j+1)}$ using the inductive hypothesis. We have:

$$a_0^{(j+1)} = [a_0^{(j)}]^2 - a_{N-j+1}^{(j)}\overline{a}_{B-j+1}^{(j)} = \left[\alpha^{2^{j-2}}\Psi_{j,\alpha}\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-2-k}}\right]^2 - \left[\alpha^{2^{j-2}}(1-\alpha)\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-2-k}}\right]^2$$

$$= \left[\alpha^{2^{j-2}}\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-2-k}}(\Psi_{j,\alpha} - 1 + \alpha)\right] \cdot \left[\alpha^{2^{j-2}}\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-2-k}}(\Psi_{j,\alpha} + 1 - \alpha)\right].$$

Notice that $\Psi_{j,\alpha} - 1 + \alpha = \Psi_{j+1,\alpha}$ and that $\Psi_{j,\alpha} - 1 + \alpha = \Psi_{j-1,\alpha}$, hence we can write:

$$a_0^{(j+1)} = \alpha^{2^{j-1}}\Psi_{j+1,\alpha}\Psi_{j-1,\alpha}\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-1-k}} = \alpha^{2^{j-1}}\Psi_{j+1,\alpha}\prod_{k=1}^{j-1}\Psi_{k,\alpha}^{2^{j-1-k}},$$

as required.

Let us consider $a_1^{(j+1)}$ whose computation is:

$$a_1^{(j+1)} = a_0^{(j)}a_1^{(j)} = \left[\alpha^{2^{j-2}}\Psi_{j,\alpha}\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-2-k}}\right] \cdot \left[-\alpha^{2^{j-2}}\Psi_{j-1,\alpha}\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-2-k}}\right]$$

$$= -\alpha^{2^{j-1}}\Psi_{j,\alpha}\Psi_{j-1,\alpha}\prod_{k=1}^{j-2}\Psi_{k-1,\alpha}^{2^{j-1-k}} = -\alpha^{2^{j-1}}\Psi_{j,\alpha}\prod_{k=1}^{j-1}\Psi_{k-1,\alpha}^{2^{j-1-k}},$$

as required. Finally, we have:

$$a_{B-j}^{(j+1)} = -a_{B+1-j}^{(j)}\overline{a}_1^{(j)}$$

$$= -\left[\alpha^{2^{j-2}}(1-\alpha)\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-2-k}}e^{i\theta}\right] \cdot \left[-\alpha^{2^{j-2}}\Psi_{j-1,\alpha}\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-2-k}}\right]$$

$$= \alpha^{2^{j-1}}(1-\alpha)\Psi_{j-1,\alpha}\prod_{k=1}^{j-2}\Psi_{k,\alpha}^{2^{j-1-k}}e^{i\theta}$$

$$= \alpha^{2^{j-1}}(1-\alpha)\prod_{k=1}^{j-1}\Psi_{k,\alpha}^{2^{j-1-k}}e^{i\theta},$$

as required. It is easy to notice that all the remaining coefficients are zeros by construction. □

Appendix C. Proof of Lemma 5

The derivation of Eqs. (A.2) and (A.4) follows the same lines of Lemma 4.

For Eq. (A.3), we have:

$$a_0^{(B+1)} = [a_0^{(B)}]^2 - [a_1^{(B)}\overline{a}_1^{(B)}]$$

$$= \alpha^{2^{B-1}}\Psi_{B,\alpha}^2\prod_{k=1}^{B-2}\Psi_{k,\alpha}^{2^{B-1-k}} - \left(\alpha^{2^{B-2}}\prod_{k=1}^{B-2}\Psi_{k,\alpha}^{2^{B-2-k}}\left((1-\alpha)\cos\theta - \Psi_{B-1,\alpha} + i(1-\alpha)\sin\theta\right)\right) \cdot \left(\alpha^{2^{B-2}}\prod_{k=1}^{B-2}\Psi_{k,\alpha}^{2^{B-2-k}}\left((1-\alpha)\cos\theta - \Psi_{B-1,\alpha} - i(1-\alpha)\sin\theta\right)\right)$$

$$= \alpha^{2^{B-1}}\Psi_{B,\alpha}^2\prod_{k=1}^{B-2}\Psi_{k,\alpha}^{2^{B-1-k}} - \alpha^{2^{B-1}}\prod_{k=1}^{B-2}\Psi_{k,\alpha}^{2^{B-1-k}} \cdot \left((1-\alpha)^2\cos^2\theta + \Psi_{B-1,\alpha}^2 - 2\Psi_{B-1,\alpha}(1-\alpha)\cos\theta + (1-\alpha)^2\sin^2\theta\right).$$

By collecting the common factors, we simplify the expression to:

$$\alpha^{2^{B-1}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}} \left(\Psi_{B,\alpha}^2 - (1-\alpha)^2 - \Psi_{B-1,\alpha}^2 + 2\Psi_{B-1,\alpha} \cdot (1-\alpha)\cos\theta\right) = \alpha^{2^{B-1}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}} \cdot \left(2(1-\alpha)^2 B - 1 + \alpha^2 - (1+\alpha^2-2\alpha) + 2\Psi_{B-1,\alpha}(1-\alpha)\cos\theta\right)$$
$$= 2\alpha^{2^{B-1}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}} \left((1-\alpha)^2 B - 1 + \alpha + \Psi_{B-1,\alpha}(1-\alpha)\cos\theta\right)$$
$$= 2\alpha^{2^{B-1}} (1-\alpha)\Psi_{B-1,\alpha}(\cos\theta - 1) \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}},$$

as required. □

Appendix D. Proof of Theorem 1

Let us consider $P(z, w)$ as defined by Eq. (8). Simple algebraic computations show that:

$$M_1(z) \triangleq \left.\frac{\partial P(z,w)}{\partial w}\right|_{w=1} = \beta \frac{z^2}{(1-z)(z-\alpha-\beta z^{B+1})} - \alpha M_1^1 \frac{z}{z-\alpha-\beta z^{B+1}}, \tag{D.1}$$

where $M_1^1 = P_1'(1)$ is the expected consolidation time given that the system contains one transaction. Successive derivatives of $M_1(z)$ evaluated in $z = 0$ give the expected consolidation times conditioned to the number of transactions present in the queue at the arrival epoch (including that just arrived):

$$M_1^Y = \frac{1}{Y!} \left.\frac{\partial^Y M_1(z)}{\partial z^Y}\right|_{z=0}.$$

Let us consider function:

$$g_1(z) \triangleq \frac{z^2}{(1-z)(z-\alpha-\beta z^{B+1})}.$$

We can show that, for $Y \geq 2$:

$$\frac{\partial^Y g_1(z)}{\partial z^Y} = \sum_{k_1=0}^{Y} \sum_{k_2=0}^{Y} \frac{(-1)^{k_1}}{(1-z)^{k+1}} z^{2-k_2} \frac{\partial^{Y-k_1-k_2}}{\partial z^{Y-k_1-k_2}} \frac{1}{z-\alpha-\beta z^{B+1}} \cdot (-1)^{k_1} k_1! (\delta_{k_2=0} + 2\delta_{k_2=1} + 2\delta_{k_2=2}) \binom{Y}{k_1, k_2, Y-k_1-k_2}, \tag{D.2}$$

where the multinomial coefficient with negative entries is assumed to be 0. Since we evaluate this derivative in 0, we have that the only non-zero term of the inner sum is $k_2 = 2$, thus:

$$\left.\frac{\partial^Y g_1(z)}{\partial z^Y}\right|_{z=0} = \sum_{k_1=0}^{Y} \left.\frac{\partial^{Y-k_1-2}}{\partial z^{Y-k_1-2}} \frac{1}{z-\alpha-\beta z^{B+1}}\right|_{z=0} \cdot 2k_1! \cdot \frac{Y!}{k_1!2!(Y-k_1-2)!} = \sum_{k_1=0}^{Y-2} \frac{Y!}{(Y-k_1-2)!} \left.\frac{\partial^{Y-k_1-2}}{\partial z^{Y-k_1-2}} \frac{1}{z-\alpha-\beta z^{B+1}}\right|_{z=0},$$

In order to conclude the proof of Theorem 1, we use Lemma 6 and Lemma 7 reported below and proved in Appendices E and F, respectively.

The following lemma allows us to compute the nth order derivative that appears in Eq. (D.2) evaluated for $z = 0$:

Lemma 6. *The following relation holds:*

$$\left.\frac{\partial^k}{\partial z^k} \frac{1}{z-\alpha-\beta z^{B+1}}\right|_{z=0} = \frac{k!}{\alpha^{k+1}} \sum_{\ell=0}^{k} (-1)^{\ell+k+1} \binom{k+1}{\ell+1} \cdot \sum_{c=\lceil \frac{k-\ell}{B} \rceil}^{\lfloor \frac{k}{B+1} \rfloor} \binom{\ell}{k-(B+1)c, \ell-k+Bc, c} (-\alpha)^{Bc} (-\beta)^c, \tag{D.3}$$

The proof of Lemma 6 is given in Appendix E.

The following Lemma is used to simplify Eq. (D.3) and then derive a recursive expression for the computation of the first moment.

Lemma 7. *Eq. (D.3) can be rewritten as:*

$$\left.\frac{\partial^k}{\partial z^k} \frac{1}{z-\alpha-\beta z^{B+1}}\right|_{z=0} = \frac{k!}{\alpha^{k+1}} \left(\sum_{c=0}^{\lfloor \frac{k}{B+1} \rfloor} (-1)^{c+1} \binom{k-Bc}{c} \alpha^{Bc} \beta^c \right). \tag{D.4}$$

The proof of Lemma 7 is given in Appendix F.

Let us consider again the computation of the Yth order derivatives of $g_1(z)$ evaluated in 0. We can write:

$$\left.\frac{\partial^Y g_1(z)}{\partial z^Y}\right|_{z=0} = \sum_{k_1=0}^{Y-2} \frac{Y!}{\alpha^{Y-k_1-1}} \sum_{c=0}^{\lfloor \frac{Y-k_1}{B+1} \rfloor} (-1)^{c+1} \binom{Y-k_1-Bc}{c} \alpha^{Bc} \beta^c,$$

and by rearranging the sum indices:

$$\left.\frac{\partial^Y g_1(z)}{\partial z^Y}\right|_{z=0} = Y! \sum_{k_1=0}^{Y-2} \frac{1}{\alpha^{k_1+1}} \sum_{c=0}^{\lfloor \frac{k_1}{B+1} \rfloor} (-1)^{c+1} \binom{k_1-Bc}{c} \alpha^{Bc} \beta^c.$$

Now, we consider the second part of Eq. (D.1):

$$g_2(z) \triangleq \frac{z}{z-\alpha-\beta z^{B+1}}.$$

In this case, we have:

$$\frac{\partial^Y g_2(z)}{\partial z^Y} = Y \frac{\partial^{Y-1}}{\partial z^{Y-1}} \frac{1}{z-\alpha-\beta z^{B+1}} + z \frac{\partial^Y}{\partial z^Y} \frac{1}{z-\alpha-\beta z^{B+1}}. \tag{D.5}$$

Since we need to evaluate the derivative in $z = 0$, we obtain:

$$\left.\frac{\partial^Y g_2(z)}{\partial z^Y}\right|_{z=0} = Y \frac{(Y-1)!}{\alpha^Y} \sum_{c=0}^{\lfloor \frac{Y-1}{B+1} \rfloor} (-1)^{c+1} \binom{Y-1-Bc}{c} \alpha^{Bc} \beta^c = \frac{Y!}{\alpha^Y} \sum_{c=0}^{\lfloor \frac{Y-1}{B+1} \rfloor} (-1)^{c+1} \binom{Y-1-Bc}{c} \alpha^{Bc} \beta^c.$$

In conclusion, for $Y \geq 2$:

$$M_1^Y = \beta \sum_{k_1=0}^{Y-2} \frac{1}{\alpha^{k_1+1}} \sum_{c=0}^{\lfloor \frac{k_1}{B+1} \rfloor} (-1)^{c+1} \binom{k_1-Bc}{c} \alpha^{Bc} \beta^c - \frac{M_1^1}{\alpha^{Y-1}} \sum_{c=0}^{\lfloor \frac{Y-1}{B+1} \rfloor} (-1)^{c+1} \binom{Y-1-Bc}{c} \alpha^{Bc} \beta^c. \tag{D.6}$$

Let us call:

$$T_Y \triangleq \sum_{c=0}^{\lfloor \frac{Y}{B+1} \rfloor} (-1)^{c+1} \binom{Y-Bc}{c} \alpha^{Bc} \beta^c,$$

then we can write for $Y \geq 1$:

$$M_1^{Y+1} = M_1^Y + \frac{M_1^1}{\alpha^{Y-1}} T_{Y-1} - \frac{M_1^1}{\alpha^j} T_Y + \frac{\beta}{\alpha^Y} T_{Y-1} = M_1^Y + \frac{T_{Y-1}}{\alpha^{Y-1}} \left(M_1^1 + \frac{\beta}{\alpha} \right) - \frac{T_Y}{\alpha^Y} M_1^1 ,$$

as required by Theorem 1. □

Appendix E. Proof of Lemma 6

We use the following identity for the derivative [35]:

$$\frac{\partial^k}{\partial z^k} \frac{1}{f(z)} = \sum_{\ell=0}^{k} (-1)^\ell \binom{k+1}{\ell+1} \frac{1}{[f(z)]^{\ell+1}} \frac{\partial^k}{z^k} [f(z)]^\ell , \tag{E.1}$$

where, in our case, $f(z) = z - \alpha - \beta z^{B+1}$. The expansion of the power of this trinomial can be obtained as follows:

$$(z - \alpha - \beta z^{B+1})^\ell = \sum_{\substack{a,b,c \\ a+b+c=\ell}} \binom{\ell}{a, b, c} z^a (-\alpha)^b (-\beta)^c z^{a+(B+1)c},$$

where a, b, c are non-negative indices. The nth order derivative of this expression is:

$$\frac{\partial^k (z - \alpha - \beta z^{B+1})^\ell}{\partial z^k} = \sum_{\substack{a,b,c \\ a+b+c=\ell}} \binom{\ell}{a, b, c} \frac{(a + (B+1)c)!}{(a + (B+1)c - k)!} \cdot z^a (-\alpha)^b (-\beta)^c z^{a+(B+1)c-k} \delta_{k \leq a+(B+1)c} .$$

Since we are interested in evaluating this expression only for $z = 0$, we have that the only non-zero terms of the sum are those such that $a + (B+1)c = k$, i.e., $a = k - (B+1)c$. We can write:

$$\left. \frac{\partial^k (z - \alpha - \beta z^{B+1})^\ell}{\partial z^k} \right|_{z=0} = \sum_{c=\lceil \frac{k-\ell}{B} \rceil}^{\lfloor \frac{k}{B+1} \rfloor} \binom{\ell}{k - (B+1)c, \ell - k + Bc, c} \cdot (-\alpha)^{\ell - k + Bc} (-\beta)^c k! .$$

Thus, Eq. (E.1), evaluated for $z = 0$, becomes:

$$\sum_{\ell=0}^{k} (-1)^\ell \binom{k+1}{\ell+1} \frac{1}{(-\alpha)^{\ell+1}} \sum_{c=\lceil \frac{k-j}{B} \rceil}^{\lfloor \frac{k}{B+1} \rfloor} \binom{\ell}{k - (B+1)c, \ell - k + Bc, c} \cdot (-\alpha)^{\ell - k + Bc} (-\beta)^c k! ,$$

This can be rewritten as in Eq. (D.3) as required. □

Appendix F. Proof of Lemma 7

By Lemma 6, we have:

$$\left. \frac{\partial^k}{\partial z^k} \frac{1}{z - \alpha - \beta z^{B+1}} \right|_{z=0} = \frac{k!}{\alpha^{k+1}} \sum_{\ell=0}^{k} (-1)^{\ell+k+1} \binom{k+1}{\ell+1} \cdot \sum_{c=\lceil \frac{k-\ell}{B} \rceil}^{\lfloor \frac{k}{B+1} \rfloor} \binom{\ell}{k - (B+1)c, \ell - k + Bc, c} (-\alpha)^{Bc} (-\beta)^c$$

$$= \frac{k!}{\alpha^{k+1}} \sum_{c=0}^{\lfloor \frac{k}{B+1} \rfloor} (-\alpha)^{Bc} (-\beta)^c \sum_{\ell=k-Bc}^{k} (-1)^{\ell+k+1} \binom{k+1}{\ell+1} \cdot \binom{\ell}{k - (B+1)c, \ell - k + Bc, c}$$

$$= \frac{k!}{\alpha^{k+1}} \left(-1 + \sum_{c=1}^{\lfloor \frac{k}{B+1} \rfloor} (-1)^{c+1} (\alpha)^{Bc} (\beta)^c \sum_{\ell=k-Bc}^{k} (-1)^{\ell+k+Bc} \binom{k+1}{\ell+1} \cdot \binom{\ell}{k - (B+1)c, \ell - k + Bc, c} \right).$$

We show now that, for $1 \leq c \leq \lfloor \frac{k}{B+1} \rfloor$,

$$\sum_{\ell=k-Bc}^{k} (-1)^{\ell+k+Bc} \binom{k+1}{\ell+1} \binom{\ell}{k - (B+1)c, \ell - k + Bc, c} = \binom{k - Bc}{c}.$$

Indeed,

$$\sum_{\ell=k-Bc}^{k} (-1)^{\ell+k+Bc} \binom{k+1}{\ell+1} \binom{\ell}{k - (B+1)c, \ell - k + Bc, c}$$

$$= \sum_{h=0}^{Bc} (-1)^{h+2k} \binom{k+1}{h + k - Bc + 1} \binom{h + k - Bc}{k - (B+1)c, h, c}$$

$$= \sum_{h=0}^{Bc} (-1)^h \binom{k+1}{h + k - Bc + 1} \binom{h + k - Bc}{k - (B+1)c, h, c}$$

$$= \binom{k - Bc}{c} \binom{k}{Bc} (k+1) \sum_{h=0}^{Bc} (-1)^h \frac{1}{(h + k - Bc + 1)} \binom{Bc}{h} .$$

Now, by applying the following identity [35]:

$$\sum_{h=0}^{a} (-1)^h \frac{1}{(h+x)} \binom{a}{h} = \frac{a!(x-1)!}{(a+x)!},$$

we obtain:

$$\binom{k - Bc}{c} \binom{k}{Bc} (k+1) \sum_{h=0}^{Bc} (-1)^h \frac{1}{(h + k - Bc + 1)} \binom{Bc}{h}$$

$$= \binom{k - Bc}{c} \binom{k}{Bc} (k+1) \frac{(Bc)!(k - Bc)!}{(k+1)!} = \binom{k - Bc}{c} .$$

This concludes the proof of Lemma 7. □

References

[1] D. Easley, M. O'Hara, S. Basu, From mining to markets: The evolution of Bitcoin transaction fees, J. Financial Econ. 134 (1) (2019) 91–109.

[2] G. Huberman, J. Leshno, C. Moallemi, Monopoly without a monopolist: An economic analysis of the Bitcoin payment system, Rev. Econom. Stud. (2021) 1–30.

[3] H. Qiu, T. Li, Auction method to prevent bid-rigging strategies in mobile blockchain edge computing resource allocation, Future Gener. Comput. Syst. 128 (2022) 1–15.

[4] M. Alharby, A. van Moorsel, The impact of profit uncertainty on miner decisions in blockchain systems, in: Electronic Notes on Theoretical Computer Science. Vol. 340, 2018, pp. 151–167.

[5] S.M. Werner, P.J. Pritz, W.J. Knottenbelt, Uncle traps: Harvesting rewards in a queue-based ethereum mining pool, in: Proc. of EAI Int. Conf. on Perf. Eval. Methodologies and Tools, VALUETOOLS, 2019, pp. 127–134.

[6] M. Alharby, R. Castagna Lunardi, A. Aldweesh, A. van Moorsel, Data-driven model-based analysis of the ethereum verifier's dilemma, in: IEEE/IFIP Int. Conf. on Dependable Systems and Networks, DSN, IEEE, 2020, pp. 209–220.

[7] D. Smuseva, I. Malakhov, A. Marin, A. van Moorsel, S. Rossi, Verifier's dilemma in ethereum blockchain: A quantitative analysis, in: Quantitative Evaluation of Systems: 19th International Conference, QEST 2022, Warsaw, Poland, September 12–16, 2022, Proceedings, Springer, 2022, pp. 317–336.

[8] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2009, URL http://www.bitcoin.org/bitcoin.pdf.

[9] I. Malakhov, A. Marin, S. Rossi, D. Smuseva, On the use of proof-of-work in permissioned blockchains: Security and fairness, IEEE Access 10 (2022) 1305–1316.

[10] M. Grundmann, H. Amberg, M. Baumstark, H. Hartenstein, Short paper: What peer announcements tell us about the size of the bitcoin P2P network, in: I. Eyal, J. Garay (Eds.), Financial Cryptography and Data Security, 2022, pp. 694–704.

[11] L. Kleinrock, Queueing Systems Volume 1: Theory, Wiley, 1975.
[12] P.G. Harrison, A. Marin, Product-forms in multi-way synchronizations, Comput. J. 57 (11) (2014) 1693–1710.
[13] D.G. Kendall, Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chains, Ann. Math. Stat. 24 (3) (1953) 338–354.
[14] A. Gopalan, A. Sankararaman, A. Walid, S. Vishwanath, Stability and scalability of blockchain systems, Proc. ACM Meas. Anal. Comput. Syst. 4 (2) (2020) art. n. 35.
[15] M.H. Nasir, J. Arshad, M.M. Khan, M. Fatima, K. Salah, R. Jayaraman, Scalable blockchains - A systematic review, Future Gener. Comput. Syst. 126 (2022) 136–162.
[16] K.M. Khan, J. Arshad, M.M. Khan, Investigating performance constraints for blockchain based secure e-voting system, Future Gener. Comput. Syst. 105 (2020) 13–26.
[17] S. Geissler, T. Prantl, S. Lange, F. Wamser, T. Hossfeld, Discrete-time analysis of the blockchain distributed ledger technology, in: Proc. of the 31st International Teletraffic Congress, ITC, 2019, pp. 130–137.
[18] S. Kasahara, J. Kawahara, Effect of Bitcoin fee on transaction-confirmation process, J. Ind. Manag. Opt. 15 (1) (2019) 365–386.
[19] Y. Kawase, S. Kasahara, Priority queueing analysis of transaction-confirmation time for Bitcoin, J. Ind. Manag. Opt. 16 (3) (2020) 1077–1098.
[20] B. Fralix, On classes of bitcoin-inspired infinite-server queueing systems, Queueing Syst. 95 (2020) 29–52.
[21] S. Balsamo, A. Marin, I. Mitrani, N. Rebagliati, Prediction of the consolidation delay in blockchain-based applications, in: Proc. of Int. Conf. on Performance Engineering, ICPE, 2021, pp. 81–92.
[22] S. Ricci, E. Ferreira, D.S. Menasche, A. Ziviani, J.E. Souza, A.B. Vieira, Learning blockchain delays: A queueing theory approach, ACM Perf. Eval. Rev. 46 (3) (2019) 122–125.
[23] J. Li, Y. Yuan, F.-Y. Wang, Analyzing Bitcoin transaction fees using a queueing game model, Electr. Commerce Res. (2020) 1–21.
[24] R. Gunlach, M. Gijsbers, D. Koops, J. Resing, Predicting confirmation times of Bitcoin transactions, ACM Perf. Eval. Rev. 48 (4) (2021) 16–19.
[25] I. Stoepker, R. Gundlach, S. Kapodistria, Robustness analysis of Bitcoin confirmation times, SIGMETRICS Perf. Eval. Rev. 48 (4) (2021) 20–23.
[26] J.W. Cohen, The Single Server Queue, revised ed., in: North-Holland Series in Applied Mathematics and Mechanics, vol. 8, North-Holland, 1979.
[27] M.F. Neuts, The busy period of a Queue with Batch Service, Oper. Res. 13 (5) (1965) 815–819.
[28] J. Messias, M. Alzayat, B. Chandrasekaran, K.P. Gummadi, On Blockchain Commit Times: An analysis of how miners choose Bitcoin transactions, in: The Second International Workshop on Smart Data for Blockchain and Distributed Ledger, SDBD2020, 2020, pp. 1–9.
[29] J. Messias, M. Alzayat, B. Chandrasekaran, K.P. Gummadi, P. Loiseau, A. Mislove, Selfish & opaque transaction ordering in the Bitcoin blockchain: the case for chain neutrality, in: Proceedings of the 21st ACM Internet Measurement Conference, 2021, pp. 320–335.
[30] S. Ross, Stochastic Processes, second ed., in: Wiley Series in Probability and Statistics, Wiley, 2008.
[31] W.E. T., G.N. Watson, A Course of Modern Analysis, Cambridge at the University Press, 1920.
[32] N. Bailey, On queueing processes with bulk service, J. R. Stat. Soc. Ser. B 16 (1) (1954) 80–87.
[33] K. Dilcher, J.D. Nulton, K.B. Stolarsky, The zeros of a certain family of trinomials, Glasg. Math. J. 34 (1992) 55–74.
[34] M. Marden, Geometry of Polynomials, in: Mathematical Surveys and Monographs, vol. 3, AMS, 1966.
[35] H. Gould, Combinatorial Identities: A Standardized Set of Tables Listing 500 Binomial Coefficient Summations, Gould, 1972.

Ivan Malakhov received a B.S. degree in information technology, information and communication from Higher School of Economics, Moscow, Russia in 2017 and an M.S. degree in computer system networking and telecommunications, information and communication from the same university in 2019. He also engaged in part of his M.S. degree in computer science at University Ca' Foscari of Venice, Italy in 2019, where he is currently pursuing a Ph.D. degree in computer science. His research interests include the quantitative analysis of public and private blockchains.

Andrea Marin is an associate professor of computer science at University Ca' Foscari of Venice. He received his Ph.D. degree in computer science in 2007 from the same university. His main research interests include the stochastic modeling of computer and communication systems for performance and reliability analysis, queueing theory, and models with product-form solutions. He has contributed to developing a probabilistic calculus for the formal analysis of wireless ad hoc networks. Since 2021, he has been a senior member of the IEEE.

Sabina Rossi received her Ph.D. in computational mathematics and informatics from the University of Padova in 1994. She is an associate professor of computer science at the University Ca' Foscari of Venice. She has been a visiting professor at Universitè Paris 7 (2007) and research fellow at the Universitè Catholique de Louvain-la-Neuve, Belgium (1997). Her current research focuses on the development of formal tools for analysis and verification based on process algebraic techniques and, specifically, on stochastic process algebras.