Analysis of the transaction confirmation process and fairness in Proof-of-Work blockchains.

Ivan Malakhov

March 15, 2023

Abstract

In the last few years, blockchains have gained a great importance as a technology to implement distributed ledgers. In this thesis, we study blockchains based on the most important consensus mechanism, namely the Proof-of-Work (PoW). In these blockchains, there is a tension between users' operating costs, i.e., the fees paid to add a transaction to the ledger, and the quality of service obtained, namely the transaction confirmation time and the transaction confirmation probability.

To study the trade-off between operating costs and reliability or confirmation time, we introduce a set of stochastic models. Technically speaking, the models that allow the computation of the reliability and performance from the ledger point view will be in steady-state, i.e., they are able to evaluate the desired indices in a sufficiently long interval of time. Among these models, we also study the fairness of PoW in permissioned blockchain.

Another class of models that we introduce are the transient ones that answer the same questions from the user perspective, i.e., they can be used by the end users to determine the optimal costs to obtain a certain Quality of Service in terms of delays and reliability.

Contents

Chapter 1

Introduction

Blockchains have been attracting more and more attention from the academic and industrial communities from the perspectives of finance, logistics, security and many others [1, 8, 50]. More recently, the quantitative analysis of the blockchains has also emerged as highly valuable research challenge [28,37,48,62].

The blockchain often refers to the decentralized and distributed peer-to-peer network that clusters its data into blocks in such a way that the data immutability is guaranteed by design. Moreover, the blockchain can stand for the chain of blocks, i.e., linked objects containing the data in a form of transactions or simply the technology underlying the network.

The smallest data unit in blockchain is called *transaction* and generally consists of information about a sender, a recipient and most importantly data that should be transferred. Further, a set of transactions forms a block, that in Bitcoin (BTC), has a maximum size of 1 MB[1] and is consolidated, on average, every 10 minutes. Bitcoin applies difficulty parameter to the new blocks to maintain network consolidation speed at the same level considering fluctuations of the network computational power mainly due to the continuous changes in number of active blockchain users. This implies that the maximum *throughput* of such system is fixed.

In this section, to simplify the exposition, we assume that all transactions have the same size. So transactions are proposed by the end users and are stored by the *miners* (who use transactions to form their blocks) in a priority queue known as Memory Pool or *Mempool*. Each transaction contains a tip for miners called "fee" that can be greater or equal to zero. Intuitively, to maximize their profits miners should form the blocks with the transactions offering the highest fees. When a transaction is included in a block, we say that it is *confirmed*, i.e., it is permanently stored in the system. The transaction residence time in the Mempool is called *confirmation delay*. This delay is crucial in determining the Quality-of-Service (QoS) of applications based on blockchains.

[1] 1 MB stands for the maximum size of the block full of non-SegWit transactions. Section.. gives more thorough description for this type of transactions.

All blockchain data are stored in a form of the distributed ledger that serves substantially three main roles: (i) verify the information or procedures that end users want to store according to a set of network rules, namely *blockchain protocol*, (ii) guarantee the immutability of the stored information and (iii) make the information or procedure publicly available without loss of confidentiality. Indeed, there exist hundreds if not thousands of blockchain protocols nowadays. However, in our study, we appeal to the very first as well as highly popular one, namely Bitcoin blockchain driven by *Proof-of-Work* (PoW) algorithm. PoW has been introduced by the seminal paper [61] by the pseudonym Satoshi Nakamoto with the aim of creating a distributed ledger for economical transactions based on the Bitcoin cryptocurrency (BTC). What is more, together with Ethereum, Bitcoin is known to be the biggest and well-known blockchain protocol worldwide.

This thesis studies the impact of the most popular family of consensus protocols (those based on PoW) on the performance, reliability and fairness of the blockchain system. While performance and reliability will be addressed from the end-user perspective and from the system perspective, the fairness is a property that concerns miners.

1.1 Description of the problem

1.1.1 Performance and Reliability analysis of the Confirmation process of transactions

In order to understand the quantitative dynamics of the transaction conformations, we need to review the procedure implemented by miners to secure the blockchain system. Each miner selects from the Mempool a set of transactions to fill a block, then they check their integrity (e.g., when there is a transfer of cryptocurrency it verifies that there is not double spending) and finally it works on a computational problem that requires a large amount of energy in order to be solved. This latter step is called PoW. The miner that announces the solution of his/her computational problem first is entitled to add his/her new block to the blockchain after the other peers have verified the correctness of the solution. For the successful block consolidation a miner receives a reward consisting of the sum of the fees of the transactions in the block and a standard unconditional reward[2].

It might be the case where two different blocks were mined at almost the same time thus provoking fork occurrence, that is the situation when the network sees multiple valid chains with the same height. However, such forks have temporary nature are known to be solved once one of the forks become longer than others in PoW environment.

From the end user's point of view, an interesting trade-off arises: on the one hand, they wish to offer the lowest possible fee to reduce the running costs

[2]According to the BTC developers the latter is going to be removed from the protocol in 2023.

of their activities, on the other they may have some requirements on the QoS, e.g., the need to confirm the transactions within the certain time interval. For example, the transactions may be associated with a trading speculation and hence must be confirmed as soon as possible, or may be a bid for a certain auction with a deadline.

Furthermore, in some cases an end user is not interested to know the confirmation time of his/her transaction but he/she seeks to know probability whether the transaction will be eventually confirmed or not (e.g., in the scenario of storing data in blockchain from the IoT networks). Also this probability depends on the offered fees since miners tend to drop the cheapest transactions when the traffic is heavy.

Blockchain systems can be studied as distributed systems by means of formal methods (e.g., [22, 25]). Queueing theory allows us to study the relation between the holding time in the Mempool, the confirmation probability, the arrival intensity of the transactions and the fees offered by users. Clearly, when the holding times increase, the transaction fees that a miner needs to offer to meet certain quality of service goals also increase. However, while for high fee transactions consolidated in a few blocks, it is safe to assume that the arrival intensity is time homogeneous, for transactions offering low fees this is unrealistic. For this reason, it is necessary to have a framework capable of predicting the fluctuations of the arrival process intensity for long-term analyses. We propose a solution based on Meta-Prophet to evaluate its accuracy.

1.1.2 Fairness in blockchain networks

So far, we assumed Bitcoin network driven by PoW, where participant are, in general, unknown and anybody can join the network; such networks are called *public*. However, there is another type of networks where users' identities are generally known and only a certain authorised set of entities can be present, i.e., *private* blockchains. In the latter case, we observe two types of trust relations:

- Internal: trust among miners of one network
- External: trust between end users and miners.

Traditional PoW is generally unable to handle the internal trust problem. In fact, some miners could try to gain sufficient computational power in order to obtain an extra advantage and, in the worst case, to obtain control on the network. Recall that, in a private blockchain the available computational power is much lower than in a public one like Bitcoin. Thus, they would then be able to modify the entire blockchain history. Conversely, blockchain solutions based on voting (e.g., *Proof-of-Stake*) are effective in ensuring internal trust as they rely on votes of the set of miners and not how much computational resources belong to a certain miner.

Regarding the external trust problem, we have the opposite situation. In fact, PoW guarantees the expected cost of modifying a confirmed transaction since the hash power (HP), that is the number of hashes generated over time,

used to ensure the immutability of data is publicly known and any change comes at the cost of spent hash power *by design*. Voting solutions are unable to provide the same guarantees. Whenever majority of miners agree on the modification of an arbitrary block or transaction, they can implement it out at basically no cost; hence, the end users must trust the consortium as a reliable entity.

Once PoW is used in permissioned blockchains (another term for private networks), we consider both the internal and external trust problems. However, the operating costs should be evenly shared among the participants, i.e., *each miner should invest approximately the same amount of HP to secure the blockchain*. This is the *fairness* property that we are interested to study. In general, it is difficult to achieve the fairness in such settings because of the intrinsic randomness of the block generation process and the interests of miners of reducing the energy used to solve the PoW.

To address this problem we propose an algorithm that makes improbable that consecutive blocks are mined by the same miners. This makes attacks to the blockchain more difficult to realize. Indeed, while in public blockchains the huge amount of HP makes the 50% attack (i.e., a group of miners controlling more than a half of the entire computational power) very unlikely, in permissioned blockchain this cannot be excluded. For example, the miners can temporarily hire a massive computational power to modify some transactions stored in the blockchain, but this would require the consolidation of consecutive blocks that our protocol does not generally allow. While we mainly study the network behavior with a single corrupted miner we also devote some attention to the possibility of collusion of several miners with the aim of attacking the blockchain.

1.2 Contributions

In our work, we study two challenging questions in the area of PoW-driven blockchains, that are (I) estimation of the optimal transaction confirmation time and confirmation probability and additionally (II) implementation of fairness and internal trust. While the major contribution is focused on the former problem considering public-based settings the latter regards research in private-based PoW networks and is of complementary contribution.

I. We introduce analytical models that study the transaction confirmation process considering two different perspectives:

- System-driven perspective. In this case, we propose a queueing model to answer the following question: *in a stationary condition, what is the long-term probability that transactions offering a certain fee are confirmed? And if they are confirmed, what is their holding time in the Mempool?*

- User-driven perspective. We examine the queueing model to answer the following question: *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?* As a corner case,

we introduce a model to derive the transaction dropping probability for low fee transactions.

To the best of our knowledge, the models that we propose or their solution techniques are novel in the literature.

Since the intensity of the arrival process can be considered homogeneous only for relatively short time horizons we additionally support the latter with complementary research that is dedicated to prediction of the transaction arrival rate using two different prediction models.

II. We propose an extended PoW algorithm for permissioned networks, which is based on the use of a sliding window. The main idea is that each miner maintains a control window of size N that stores the information about the consolidators of the latest N blocks in the blockchain. The rule is that a miner m can be present in the window at most once. When a node receives a block from miner m and m is not present in the window, then it verifies the transactions and the block hash; and if these are correct, it accepts the new block, otherwise, the block is rejected.

We study the security of this protocol with respect to the two major security threats of PoW: the 50% attack and the greedy miner attack. Moreover, we provide a quantitative Markovian model of the system to study its fairness intended to reduce the gaps among the available HP of the miners and determine the optimal configuration according to the design needs.

1.3 Published papers

In this section, we list all the papers we were working on during the period of the PhD study from 2019 to 2023 in chronological order. At the time of writing this thesis, two of them are under submission with expected publishing period in early 2023.

1. Malakhov I., Marin A., Rossi S., Smuseva D., *Fair Work Distribution on Permissioned Blockchains: a Mobile Window Based Approach.* IEEE International Conference on Blockchain, Blockchain 2020 (see Chapter 4)
2. Malakhov I., Gaetan C., Marin A., Rossi S., *Workload Prediction in BTC Blockchain and Application to the Confirmation Time Estimation.* Performance Engineering and Stochastic Modeling - 17th European Workshop EPEW 2021 (see Chapter 7)
3. Malakhov I., Marin A., Rossi S., Smuseva D., *On the Use of Proof-of-Work in Permissioned Blockchains: Security and Fairness.* IEEE Access 2022 (see Chapter 4)
4. Balsamo S., Malakhov I., Marin A., Mitrani I., *Transaction confirmation in proof-of-work blockchains: auctions, delays and droppings.* 20th Mediterranean Communication and Computer Networking Conference, MedComNet 2022 (see Chapter 5)

5. Smuseva D., Malakhov I., Marin A., van Moorsel A., Rossi S., *Verifier's Dilemma in Ethereum Blockchain: A Quantitative Analysis*. Quantitative Evaluation of Systems - 19th International Conference, QEST 2022

6. Malakhov I., Marin A., Rossi S., *Analysis of the Confirmation Time in Proof-of-Work Blockchains*. Submitted to Future Generation Computer Systems Volume 139 2023 (see Chapter 6)

7. Malakhov I., Marin A., Menasché D.S., Rossi S., *Confirmed or Dropped? Reliability Analysis of Transactions in Blockchains*. Submitted to Proceedings of the ACM Web Conference 2023 (see Chapter 8)

In all papers, with the exception the forth, Ivan Malakhov is the corresponding author and made a significant contribution in model definition and accuracy assessment as well as in the implementation tasks and data analysis. In the paper addressing the problem of Verifier's Dilemma, he contributed the background part of the work and discussion of the results.

1.4 Structure of the thesis

The thesis is structured as follows, Chapter 1-3 provide the motivation and background of the thesis while Chapter 4-8 show our contributions. In particular, Chapter 2 provides general knowledge about blockchain networks with the reference to works of other authors. In Chapter 3 we describe the modeling techniques that are used to model and analyse the network behavior. Next, Chapter 4 introduces the sliding window approach to study fairness and security aspects of the private blockchain network. In Chapter 5 we propose a model to study the confirmation process from the system perspective that provides the foundation for the results in the following chapters. Next, in Chapter 6 we introduce the analytical model for determining optimal fee for certain confirmation delay that is in favor of end users. Chapter 7 complements the previous chapter by assessing the prediction models for forecasting of persistent arrival rate of transactions. In Chapter 8 we study the corner case of the confirmation process in which we focus on estimation of the probability of eventual confirmation for transactions with relatively low fees. Finally, Chapter 9 concludes the thesis.

Chapter 2

Background on blockchain and state of the art

2.1 Introduction

The section provides general information about major aspects of blockchain networks and especially PoW-driven ones. Furthermore, we discuss the research done in the areas of our interests to compare it with the results we obtained.

2.1.1 Types of blockchains

This section provides information about different kinds of blockchains as classified by the academic community.

Usually, blockchain networks are divided according to their type of access:

- Public blockchains. They are associated with open access policy and generally high anonymity among in- and out-side of the network. The most well-known examples of this class can be Bitcoin[1] and Ethereum[2].

- Private blockchains. They are characterized with limited access policy where information about all the participants is known. Such networks can be met in industry where one or several companies decide to maintain a secure ledger. For instance, Ethereum Enterprise[3] and R3 Corda[4].

Another way to distinguish one blockchain from another can be by type of consensus mechanism. There is a great variety of the consensus tools that can be applied to the networks (e.g., Proof-of-Work, Proof-of-Stake, Proof-of-Space,

[1] `https://bitcoin.org`
[2] `https://ethereum.org`
[3] `https://ethereum.org/en/enterprise/private-ethereum`
[4] `https://r3.com/products/corda`

Proof-of-Authority and others [76]). However, at the moment, the most common ones remain *Proof-of-Work* and *Proof-of-Stake* (PoS). The main difference between these two is in a way they reach the consensus.

In the PoW blockchain, every change of the global network state is computationally predetermined, i.e., to consolidate a new block the participant has to solve the crypto puzzle in order to comply with the norms of a valid hash for his/her block. Thus, modifications of the data in the past is linked to high computational demand which is known. The system is considered safe until one or a coordinated group of miners obtains sufficient amount of computational power as in case of PoW at least 50% of the whole network power.

On the other hand, PoS is known to have block confirmation once the majority (two thirds) of the committee votes for this. In fact, PoS was made after PoW in order to solve the problem of heavy computational work usage. Nevertheless, it naturally implies that such systems are more vulnerable to the network attacks as there is literally no cost to create a new block as the newly formed committee can agree to change the past data and thereby corrupt the network.

Generally speaking, in Proof-of-Stake networks participants become validators (analogously to miners in PoW) if they decide to deposit certain amount of cryptocurrency to the network. Then they are committed to follow the protocol by validating new blocks and proposing their own as they can be punished for not doing so. For instance, in Ethereum, the second after Bitcoin largest blockchain, the validators are entitled to stake 32 ETH to activate block validation functionality.[5]

Since in following chapters we rely on PoW-driven blockchains it is crucial to provide more thorough information of such mechanism. The following section describes the functional details of the blockchains with PoW consensus mechanism.

2.1.2 PoW-driven blockchains

In this section, we review the salient aspects of Proof-of-Work blockchains that the thesis considers.

Blockchain networks driven by the Proof-of-Work consensus mechanism are the most well-known and actively studied class of blockchains. The idea of PoW was originally introduced in the notorious work of Satoshi Nakomoto [61] in 2008. One may treat PoW as an ancestor technology of every other blockchain consensus approach existing nowadays.

In order to describe the algorithm behind the PoW blockchain depicted in Figure 2.1 let us first introduce a few general concepts that it based on. Firstly, blockchain networks are decentralized and distributed. Data are stored in such a way that it is essentially impossible to modify them once they are confirmed, or, more precisely, it is possible to estimate the cost of any modification that turns out to be extremely high. Data are organized in transactions and stored in

[5] `https://ethereum.org/en/staking`

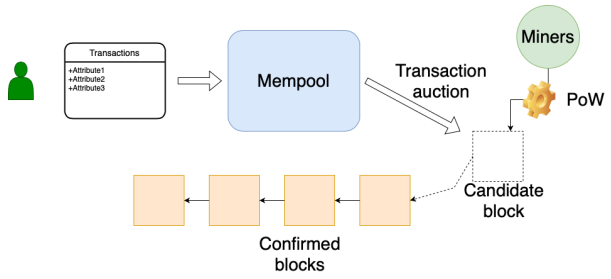


Figure 2.1: Sketch of PoW with transaction auction.

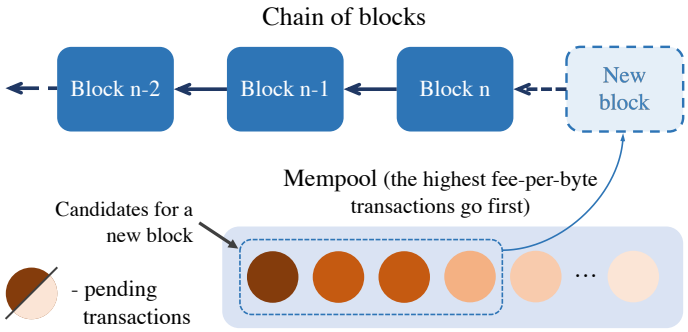


Figure 2.2: Sketch of the mining process in BTC. Darker colors represent higher fee-per-byte ratios.

a list of blocks that contain links to a unique predecessor. An attempt to break the integrity of any block requires to regenerate all the subsequent blocks: PoW is a mechanism designed to make the consolidation of blocks very expensive in terms of energy consumption. Currently, the mining in Bitcoin demands 0.56% of worldwide electrical power production and this value is in continuous growth[6].

The process of consolidating a new block is called *mining* and users who perform the mining are called *miners*. The mining process is depicted in Figure 2.2.

Mining a block basically consists of two phases:

1. Select among the transactions asking for confirmations those to be added to the new block and validate them. If there are no smart-contracts, the validation is in general a quick step that verifies if the transaction is valid (e.g., signatures, no double spending etc.). The pending transactions are stored in Mempool.

2. Solve a computationally hard problem. Usually, this consists in finding a certain number (called *nonce*) to be included in the block such that the

[6] According to the Cambridge Centre for Alternative Finance `https://ccaf.io`

hash of the entire block satisfies certain properties (e.g., it begins with a certain number of zeros). Since the block contains also the hash of the previous block, this guarantees the practical immutability of stored data in large blockchains.

Since it is assumed that each hash is equiprobable given a certain block, and given that the nonce is a 32-bit 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.

There is a clear incentive mechanism to support miners' work: for every mined block, miners earn a fixed reward in cryptocurrency and an additional reward for every included transaction, e.g., this is true for Bitcoin. The former is generated by the system itself, the latter is paid as transaction fees by users.

Every user can offer a fee for his/her transaction based on the urgency of its confirmation. Miners select the transactions to be included in the next block by choosing the most profitable ones, i.e., those offering the highest fee per Byte. This creates an interesting dynamics where the arrival order is not important to determine the confirmation time as in a First-Come-First-Served discipline (FCFS) while the offered fee is crucial.

The last aspect that is necessary to review is that most blockchains have a maximum throughput that is determined by two invariant properties: the maximum block size and the average delay between two consecutive blocks.

Although the former property is easy to be ensured, 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., Segregated Witness (*SegWit*) in Bitcoin[7]. Currently, 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 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 80%[8] and it is expected to be increased even more as overall trend remains. 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 calculate the fee-per-byte ratio they use the effective space occupied in the block rather than the total transaction size. Generally speaking, the implementation of the SegWit transactions allows the separation of the transactions' signatures from their other data. Hence, a 1 MB block completely filled with the SegWit transactions carries the same information of a 1.7 MB block without them.

Additionally, the delay between two consecutive blocks executions is guaranteed because the system sets a relative difficulty of the computational problem

[7]Bitcoin Improvement Proposal 141: `https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki`

[8]`https://charts.woobull.com/bitcoin-segwit-adoption`

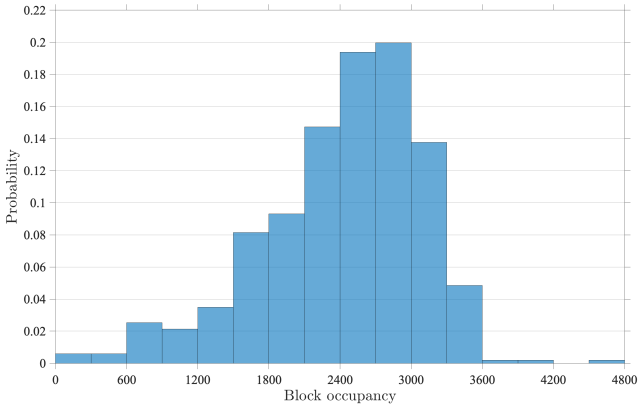


Figure 2.3: Empirical probability density function of block occupancy in Bitcoin network.

that is proportional to the computational power provided by all miners. For example, in Bitcoin, an average block occupancy varies and can be approximately $2,000 - 2,500$ transactions and on average block is generated every 10 minutes. This implies that the maximum throughput can be estimated in about 4 tx/s that is expected to grow as more transactions will adopt the SegWit standard.

Clearly, in periods of heavy traffic, the auction among the transactions becomes more expensive for the users and miners' revenues are higher. Conversely, one can experience a short confirmation time even for low transaction fees if the system load is low. Transactions that stay in the Mempool more than 48 or 72 hours are evicted by miners. This policy is necessary to ensure the stability of the Mempool occupancy in periods of heavy load. Moreover, eviction also occurs when the Memepool size exceeds 300 MB[9].

2.1.3 Fixed size and fixed capacity blocks

In general, transactions have various sizes. Thus, the number of transactions that a block with the fixed maximum size can fit is not predetermined and is random. Figure 2.3 shows the empirical probability density function of block occupancy. The data were retrieved from the node monitoring the Bitcoin blockchain at the University Ca' Foscari and represent the information about 516 full blocks.

It is clear that the distribution quite well fits the bell shape where approximately 40% of blocks fall within an interval between 2400 and 3000 transactions.

Notice that, when we talk about fixed size we mean, more precisely, fixed maximum size. Indeed, some blocks may be only partially filled because, at its consolidation epoch, there were not enough transactions in the Mempool.

[9] `https://bitcoin.org/en/bitcoin-core/`

Alternatively, we say that blocks have fixed capacity if the protocol establishes a maximum number of transactions per block rather than a maximum size in Bytes. Although we are unaware of any blockchain operating this way, most of the models work under the assumption of fixed capacity rather than fixed size. Thus, we think it is important to investigate the soundness of this assumption.

2.1.4 Auction and transaction confirmation time

When a new transaction is seen by the miners, it is included into Mempool. All the pending transactions 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* or *confirmation delay*.

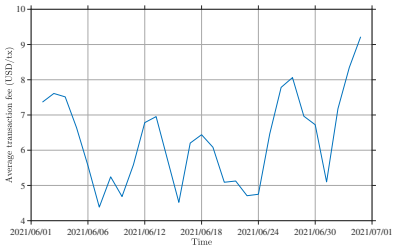
The confirmation delay is usually measured in number of blocks rather than in seconds. This is coherent with the needs of the blockchain users as witnessed by the active services of fee prediction. For example, the reactive service implemented in the main software for BTC usage, Bitcoin Core[10], is used by the wide majority of users and implements the "estimatesmartfee" service based on the user's 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 certain to enter the next available block but it 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.

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 maximise 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.

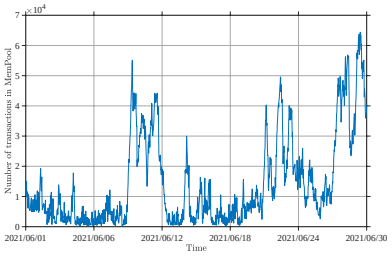
2.1.5 Predicting the minimum fee for QoS

Recall that in the BTC blockchain, miners are rewarded in two ways: i) for each confirmed block, the miner who created it receives a certain amount of cryptocurrency and ii) for each transaction included in the block, the same miner receives the fee offered by the user who created that transaction.

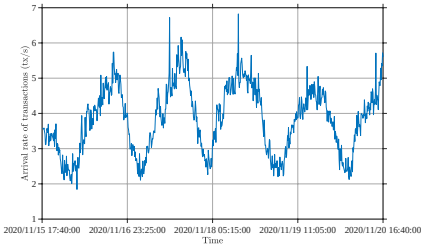
[10] `https://bitcoin.org/en/download`



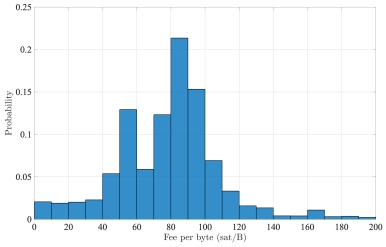
(a) Transaction fee in USD in the Bitcoin network. The data are retrieved from `http://www.blockchain.com`



(b) Memory Pool size in transactions in the Bitcoin network. The data are retrieved from `http://www.blockchain.com`



(c) Arrival rate of transactions as a function of time with step of 10 minutes. The data are retrieved from the installed node.



(d) Empirical probability density function of fee per byte in heavy workload conditions. The data are retrieved from the installed node.

Figure 2.4: Blockchain network indicators.

While the former reward is going to be dismissed in the next years, the latter plays a crucial role in understanding the QoS of applications that use BTC blockchain. Indeed, miners aim at maximising their profit and thus choose to include in the block the transactions with the highest fee.

Transaction fees are known to be subject to high fluctuations as shown by Figure 2.4a. We may notice that the average fee for a transaction can vary from around 4.5 to 9 USD in a month. How to decide which fee to offer to have an expected confirmation delay?

It is important to understand that the answer to this question depends on several state variables of the blockchain. First, we should consider the Mempool occupancy (usually called improperly Mempool size), i.e., the backlog of the transactions that are waiting to be confirmed. Figure 2.4b shows the trace of the Mempool occupancy in June of 2021. The are several bursts that clearly affect the decision on the fee to be offered.

However, the most important factor is the transaction arrival process. Recall that all the transactions arriving after a tagged transaction t offering a fee per Byte f will overtake t if they offer more than f. Fee per Byte is commonly used to compare the cost of transactions because these may have different sizes. Since the block size is fixed and the intergeneration time of blocks is on average 10 minutes the competition among the transactions gets tougher when the traffic is higher. Figure 2.4d shows the distribution of the fee per Byte offered under heavy-load conditions as measured by our monitor.

Summarising, the confirmation time of a transaction t arriving at time τ depends on the following aspects:

- The arrival rate of the transactions after τ and before the confirmation of t, limited to those whose fees are higher than the fee offered by t
- The state of the Mempool at time τ
- The distribution of fees offered by the other users.

Although the confirmation delay is measured in number of blocks rather than in seconds, it is well-known that the time between consecutive block consolidations is approximately exponential with mean of 600 s [17].

Figure 2.4c shows the intensity of the arrival process in a period of time. This is subject to high variability and exhibits a clear seasonality. Therefore, in any procedure aimed at predicting the expected confirmation time of transactions, we must implicitly or explicitly deal with the prediction of the arrival intensity at the moment in which the transactions is sent to the ledger. Chapter 7 provides a study of prediction models in order determine such intensity.

2.1.6 Transaction dropping policy

First, recall that the difficulty of the PoW puzzle is dynamically set in such a way that the average block generation delay remains the same such as 600 seconds in Bitcoin. This property, joint with the maximum block size (1 MB in Bitcoin), imposes the maximum theoretical throughput for the blockchain.

The Mempool occupancy can grow significantly considering the intrinsic randomness in the block consolidation process and the fact that for a long period of time we may observe an intensity of the arrival process significantly higher than the maximum throughput. To avoid an excessive resource consumption at miners' servers, the implementation of the protocol usually puts some limits on Mempool size. For the standard implementation of Bitcoin protocol this is 300 MB. Moreover, always in Bitcoin, transactions that reside in the Mempool for more than 2 weeks are dropped. We assess the first mechanism of dropping, since the latter regards mostly transactions that do not pay any fee and hence may not be included in blocks even if some space is available.

In addition, whenever it is needed to evict a transaction from the Mempool, the one with the lowest fee per Byte is chosen and, in case of tie, the oldest one.

2.2 Literature review

This section demonstrates the progress made by the scientific community compared to our works in the areas we are interested in. All the related work is divided on subsections with respect to the subject of review.

2.2.1 Fairness in PoW blockchains

Since the pioneering work by Nakamoto [61], many research efforts have been devoted to the analysis of the security and performance of blockchain systems. For example, in [17], the authors study how data broadcasting delays may favour certain types of attacks and unnaturally increase the number of forks. The importance of the contribution also relies on the network model and its configuration, which helps the understanding of blockchain fork occurrences.

Many works study the pros and cons of several consensus mechanisms. The *Byzantine fault tolerance* (BFT) based on voting[11] or hybrid consensus mechanisms are proposed in [76], [44], [9], [49], [66], and [20]. In some cases, BFT is also suggested for public blockchains. The main advantage of adopting a non-PoW consensus is the reduction of the power consumption and the greater transaction throughput. However, these systems cannot be considered to be as quantitatively secure as PoW-based ones, where the end user can calculate the cost of a modification of a consolidated transaction.

Particularly, the authors [20] propose a "new generation" hybrid Bitcoin protocol where the process is separated on *leader election* and *transaction serialization* applying better scaling and throughput with a certain level of fairness. Although it helps to outperform the classical Bitcoin protocol, this protocol remains vulnerable to the double spending and selfish miner attacks when the fraction of the Byzantine nodes reaches at least 25%.

[11] The algorithms in which the block confirmation happens once the majority of authorized miners (called *committee*) validates a new block. Hereafter, we refer to this class of algorithms by simply using the acronym BFT

In addition, one of the HyperLedger projects, Sawtooth, started supporting practical BFT (PBFT) as a consensus approach in addition to the initially utilised *Proof-of-Elapsed-time* mechanism. However, it is worth noting that the current versions of Sawtooth with PBFT are still recent implementations and have some limitations, such as the full peering requirement and lack of open network enrolment [64].

A comparison between PoW and BFT consolidation policies for permissioned blockchains was conducted by Vukolić in [75] and [74]. He analyses the issues of applying such consensus approaches predominantly in the permissioned settings. Finally, the author confirms that although BFT-based consensus mechanisms have the advantages of higher transaction rates and low energy consumption, PoW blockchain networks yield unique security features that we discuss in this paper.

Fairness is another property that attracts much attention from the community. In public networks, fairness is defined as the property that allows a miner to consolidate a fraction of blocks that is proportional to his fraction of HP in the long term. Note that this differs from our notion of fairness since we consider permissioned blockchain systems in which rewards for block mining are usually absent.

For public systems, [65] proposes a PoW-based protocol, namely *Fruitchain*. The authors prove that the protocol also achieves quantitatively predictable fairness in the presence of greedy miners. Note that unlike our proposal, Fruitchain aims to obtain a network in which miners receive rewards that are proportional to the invested HP. In our case, we want to avoid the possibility that a miner controls a network by exposing a huge HP. In our sense, fairness means regulating the used HP of the miners so that they tend to consolidate the same number of blocks in the long run.

In [30], the authors study the fairness properties of a blockchain and describe the behavior of two honest miners experiencing different propagation delays. The main result is that the propagation delay, as well as the HP, impacts the network fairness. Moreover, in the case of two miners with the same HP, there is an advantage for the miner who belongs to the stronger cluster of miners. They demonstrate that as network latency increases, the protocol remains stable.

Fairness is addressed also in [47]. The authors introduce a new blockchain protocol called *DECOR+HOP*. It provides fairness among miners by distributing the block generation rewards among all the miners that originate the same forks. In this way, the overall fairness of the network is improved, and the expected number of forks is reduced.

Fairness in permissioned blockchains implementing Proof-of-Stake consolidation was investigated in [4]. In this context, fairness is defined in terms of distribution in the selection mechanism (*forming a committee*) and reward mechanism (*sharing goods*). The authors examine the fairness in synchronous systems and prove that it is the optimal solution.

With respect to our work, the latter two papers consider a completely different consolidation mechanism, i.e., PoS, while we rely on PoW.

2.2.2 Transaction confirmation process

System perspective

The model evaluation of the blockchain networks gains rather high interest among scientists. Some authors [19, 37, 48] assess the effect of transaction fees by applying game-theoretic models. Particularly, [48] demonstrate a game-theoretic framework where the dynamic of the fees are studied in relation with the economical interests of the miners. With respect to these contributions, our modelling efforts have different goals. Indeed, the authors of [19,37,48,52] aims at studying the impact of the design choices of the blockchain on the dynamics of the fees, while in our case, we want to study the delays and dropping probabilities experienced by an application using an existing blockchain.

Other authors aim to estimate the confirmation delay of transactions using queueing theory [7,24,26,39,41]. With respect to these papers, our work makes the following contributions: (i) the block formation mechanism is more realistic than [39, 41] since new transactions with high fee per Byte are immediately included in the candidate block by the miners; (ii) we consider the case of blockchains with fixed block size rather than fixed capacity; (iii) we consider the problem of transaction blocking that is ignored by all the mentioned papers with the exception of [7]. With respect to this, we propose a new way to derive the dropping probability that is more accurate and simple.

In [51], the authors analyse behavior of the arrival rate of transactions in Bitcoin network using different prediction models and use the outcomes in order to estimate the transaction waiting time measured in number of blocks.

In [31], the authors apply the Cramér-Lundberg process for assessment of the transaction confirmation time. The chosen model is known to require a constant arrival flow while the paper's model maintains the stochasticity of the arrival process.

User perspective

In general, the quantitative analysis of blockchain systems has drawn a lot of attention from the scientific community (see, e.g., [28,43,62] and the references therein).

In particular, the estimation of transactions' confirmation time with queueing theory has been explored in some very recent works [6,24,26,39,41]. 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.

The major difference between our contribution and those described in [6, 26,39,41] 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 Chapter 6, 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 [6, 39, 41] 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 [39, 41] differ from [6] 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, [6] 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 [68], 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 [26], the authors propose an iterative solution for the stationary distribution of the embedded Markov chain of a $G/G^B/1$ queue (see Chapter 3 for more details about queueing models) 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 [48]. This contribution shares with [6, 39, 41] 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 [31, 72], the authors propose to use the process named Cramér-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 Cramér-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 stud-

ied the single server queue with batch departures (see, e.g., [16]) 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 [63], 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 [39, 41], 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 8 is given.

Eviction of Mempool transactions

Most of the modeling efforts in [7,39,41,68] have been devoted to the prediction of transaction confirmation times given a certain offered fee per Byte, while reliability analysis has remained an open problem.

In the field of performance analysis of blockchains, queueing models have been widely applied. In [39, 41], the authors investigate the queueing process underlying the Mempool with attention to the relations between the fee per Byte offered by a transaction and its expected confirmation time. The resulting model is queue with a scheduling with strict priorities based of the outcomes of the auction run by miners. In [7], the authors refine the model by considering a more accurate block creation policy. All these papers show a good agreement with the model predictions and the outcomes. The contribution in [68] combines machine learning and queuing theory to study the confirmation process of transactions. Specifically, this allows them to study delays in transaction confirmation in blockchain.

A game theoretical framework of transaction auction process related with the confirmation delay is proposed in [37].

In [58, 59], the authors study the behavior of Bitcoin in periods of heavy load, i.e., the moments when the network has more transactions than can be placed to a block. In particular, they are interested in exploring the unfair behaviors of mining pools that may violate the blockchain neutrality and decide to mine transactions without following the fee per Byte auction outcome. While this phenomenon is crucial for delay sensitive transactions, it does not significantly change the perspective of delay insensitive transactions that are mostly interested into the confirmation probability rather than their delays.

[36] introduces the study of *orphan transactions*, i.e., the transactions with temporal or permanent lack of the parental transaction that they rely on.

In [72], the authors use the time to ruin of the Cramér-Lundberg model to evaluate the confirmation time in Bitcoin network. With respect to the previous mentioned papers, this considers a mean time to absorption analysis and hence the model takes into account the Mempool state at the arrival time of the transaction. Beside this similarity with our contribution, the Cramér-Lundber model assumes a constant arrival flow while we adopt a more realistic random process and our metric of interest is the confirmation probability and not the

expected confirmation time. Cramér-Lundber model is also at the base of the analysis proposed in [46] to estimate the probability that a transaction with a certain fee is confirmed before a certain number of blocks. The author derives bounds for this metric and a diffusion approximation for the model.

Finally, and most importantly, we should notice that all models mentioned so far assume an infinite Mempool size and hence can be used only in condition of stability, i.e., when the intensity of the arrivals is lower than the service capacity. This is not always the case for important blockchains like Bitcoin that can experience long periods of heavy load. To the best of our knowledge, the model proposed in this paper is the first considering a finite Mempool and hence capable of studying the heavy load conditions that cause transaction evictions.

2.2.3 Blockchain throughput prediction

Statistical analysis on blockchain and in particular BTC system have been widely investigated in the recent years. However, most of the research efforts have been devoted to the prediction of the conversion rate to USD or other currencies (see, e.g., [21, 60]).

We focus on studying the cost of transaction fees. Most of the previous works assume a time-homogeneous arrival process, as in [6, 39, 41] which can be reasonable for expensive transactions that are confirmed within one hour from their request. However, when the delay is longer, the fluctuations of the arrival process cause the model with the homogeneity assumption to generate inaccurate predictions.

In addition, [48] provides a similar contribution by demonstrating the stationary analysis of the queueing model and the definition of the customer priority classes. However, the authors focus on a game theoretical framework where they attempt to find correlations between the fee fluctuations and the miners' economical incentive.

Another work [79] analyses the transaction fees in the blockchain networks. However, their research is related to the Ethereum blockchain and particularly the smart contract transactions.

To the best of our knowledge, this is the first study that aims at predicting the intensity of the transaction arrival process by using time series analysis and predicting on the confirmation time based on the offered fee.

Chapter 3

Essential elements of stochastic modeling

3.1 Markov chains

This chapter describes the modeling methods that we used to study blockchain networks. In particular, we focus on Markov processes. These have been used by the community of performance and reliability evaluation to study various type of systems.

3.1.1 Discrete-time Markov Chains (DTMC)

In general, a set of random variables X_t refers to the stochasic process where t usually represents the time. Variable t can be an integer or real number, the former is used for a discrete time process while the latter describes a continuous process (the following subsection will give information about such processes).

A sequence $\{X_n\}_{n\geq 0}$ of random variables which values belong to a set E stands for a discrete-time stochastic process with state space E. In this thesis, we consider the state space to be countable, that is its elements are defined by $i, j, k, \ldots$ For instance, $X_n = i$ can be interpreted as the process X in a moment of time n is in state i or it visits the state i at time n.

Markov Property Sequences of independent and identically distributed (i.i.d.) random variables are stochastic processes, but they are not always interesting as stochastic models because their behavior does not depend on the history of the process. For the sake of variability, some dependence on the past can be allowed, such as deterministic recursive equations. Discrete-time homogeneous Markov chains hold the necessary property, since they can always be interpreted as the stochastic recurrent equation $X_{n+1} = f(X_n, Z_{n+1})$, where $\{Z_n\}_{n\geq 1}$ is an i.i.d. sequence, not related to the initial state X_0.

Probabilistic dependence on the past is carried out only through the past states, but this bounded amount of memory is sufficient to produce a wide va-

riety of behaviors. Consequently, application of Markov chains applications can be found in various fields, such as engineering, physics, biology, sociology, and others, since they perform qualitative and quantitative assessment as well as valuable information for system design. This subsection provides basic definitions of homogeneous discrete-time Markov chains and classical examples to illustrate the theory of the following chapters.

Definition 1. *[13, Dfn 1.1 Ch 2][Homogeneous Markov Chain] Let $\{X_n\}_{n\geq 0}$ be a discrete-time stochastic process with countable state space E. If for all integers $n \geq 0$ and all states $i_0, i_1, \ldots, i_{n-1}, i, j$,*

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = P(X_{n+1} = j \mid X_n = i) \tag{3.1}$$

whenever both sides are explicitly introduced, this stochastic process is a Markov chain *process. In addition, it can refer to a* homogeneous *Markov chain (HMC) when the right-hand side of 3.1 does not depend on n.*

Property 3.1 is the *Markov property.* The matrix $\mathbf{P} = \{p_{ij}\}_{i,j\in E}$, where

$$p_{ij} = P(X_{n+1} = j \mid X_n = i)\,,$$

is the transition matrix of the HMC. As it consists of probabilities and a transition from one state i must be to some other arbitrary state (j), it follows that

$$p_{ij} \geq 0, \sum_{k\in E} p_{ik} = 1$$

for all states i, j. A matrix $\mathbf{P}$ that is indexed by E and satisfies the above properties stands for a stochastic matrix. Notice that since its state space may be infinite this kind of matrices are generally not considered in linear algebra. However, the basic operations of addition and multiplication follow the same formal rules. For example, with $A = \{a_{ij}\}_{i,j\in E}$ and $B = \{b_{ij}\}_{i,j\in E}$, the product $C = AB$ is the matrix $\{c_{ij}\}_{i,j\in E}$, where $c_{ij} = \sum_{k\in E} a_{ik}b_{kj}$. The notation $x = \{x_i\}_{i\in E}$ formally denotes a *column* vector, while x^T is a row vector, the transpose of x. Furthermore, $y = \{y_i\}_{i\in E}$ given by $y^T = x^T A$ is represented by $y_i = \sum_{k\in E} x_k a_{ki}$. The same way, $z = \{z_i\}_{i\in E}$ given by $z = Ax$ is represented by $z_I = \sum_{k\in E} a_{ik}z_k$.

Distribution of an HMC The random variable X_0 is called *initial state*, and its probability distribution ν,

$$\nu(i) = P(X_0 = i)\,, \tag{3.2}$$

is the *initial distribution.* From Bayes's sequential rule, $P(X_0 = i_0, X_1 = i_1, \ldots, X_k = i_k = P(X_0 = i_0)P(X_1 = i_1) \mid X_0 = i_0) \cdots P(X_k = i_k \mid X_{k-1} = i_{k-1}, \ldots, X_0 = i_0$, and hence, from the perspective of the homogeneous Markov property and the definition of the transition matrix,

$$P(X_0 = i_0, X_1 = i_1, \ldots, X_k = i_k) = \nu(i_0)p_{i_0 i_1} \cdots p_{i_{k-1} i_k}\,. \tag{3.3}$$

The data 3.3 for all $k \geq 0$, all states $i_0, i_1, \ldots, i_k$, constitute the *probability law*, or *distribution* of the HMC. Thus, we obtain the following result.

Theorem 1. *[13, Thm 1.1 Ch 2][Distribution of an HMC] The distribution of a discrete-time HMC is determined by its initial distribution and its transition matrix.*

The distribution at time n of the chain is vector ν_n, where

$$\nu_n(i) = P(X_n = i) \tag{3.4}$$

From Bayes's rule of exclusive and exhaustive causes, $\nu_{n+1}(j) = \sum_{i\in E} \nu_n(i)p_{ij}$, that is, in matrix from $\nu_{n+1}^T = \nu_n^T \mathbf{P}$. Iteration of this equality yields

$$\nu_n^T = \nu_0^T \mathbf{P^n}\,. \tag{3.5}$$

The matrix $\mathbf{P^n}$ is called the *n-step transition matrix* because its general term is

$$p_{ij}(m) = P(X_{n+m} = j \mid X_n = i)\,. \tag{3.6}$$

Indeed, using Bayes's sequential rule and the Markov property, find for the right side of the last equality

$$\sum_{i_1,\ldots,i_{m-1}\in E} p_{ii_1}p_{i_1i_2}\cdots p_{i_{m-1}}j\,,$$

and this is the general term of the mth power of $\mathbf{P}$.

The Markov property (3.1) extends to

$$\begin{aligned} P(X_{n+1} = j_1, \ldots, X_{n+k} = j_k \mid X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) \\ = P(X_{n+1} = j_1, \ldots, X_{n+k} = j_k \mid X_n = i) \end{aligned}$$

for all $i_0, \ldots, i_{n-1}, i, j_1, \ldots, j_k$ such that both sides of the equality are determined. Writing

$$A = \{X_{n+1} = j_1, \ldots, X_{n+k} = j_k\}, \quad B = \{X_0 = i_0, \ldots, X_{n-1} = i_{n-1}\}\,,$$

the last equality reads $P(A \mid X_n = i, B) = P(A \mid X_n = i)$, which is consequently equivalent to

$$P(A \cap B \mid X_n = i) = P(A \mid X_n = i) = P(A \mid X_n = i)P(B \mid X_n = i)\,. \tag{3.7}$$

In other words, the future at time n and the past at time n are conditionally independent for the current state $X_n = i$. This also demonstrates that the Markov property does not depend on the direction of time.

Notice that we abbreviate $P(A \mid X_0 = i)$ as $P_i(A)$. Since μ is a probability distribution on E, $P_\mu(A) = \sum_{i\in E} \mu(i)P_i(A)$ denotes the probability of A where the initial state is distributed according to μ.

First-step analysis

Absorption Probability First-step analysis is a method that allows assess various aspects of HMC, including probabilities of absorption by a closed set, i.e., when $\sum_{j\in A} p_{ij} = 1$ for all $i \in A$, the set is considered to be *closed*, and average times before absorption. We will use these results in Chapter 6 where we study blockchains from the user's perspective. This method, which is the core technique for the majority of computations in Markov chain theory, is well demonstrated by the following example.

Example 1 (Gambler's Ruin)**.** *Two players A and B play "heads or tails", where heads occur with probability $p \in (0,1)$, and the successive outcomes form an i.i.d. sequence. Let X_n be the fortune in dollars of player A at time n, then $X_{n+1} = X_n + Z_{n+1}$, where $Z_{n+1} = +1$ (resp., -1) with probability p (resp., $q = 1-p$), and $\{Z_n\}_{n\geq 1}$ is i.i.d. Assume that A bets \$ 1 on heads at each toss, and B bets \$ 1 on tails. The corresponding initial fortunes of A and B are a and b. The game finishes when a player is ruined, and hence the process $\{X_n\}_{n\geq 1}$ is a random walk, aside from that it is limited to $E = \{0,\ldots,a,a+1,\ldots,a+b=c)$. The duration of the game is T, the first time n at which $X_n = 0$ or c, and the probability of winning for A is $u(a) = P(X_T = c \mid X_0 = a)$.*

Instead of calculating just $u(a)$, first-step analysis does

$$u(i) = P(X_T = c \mid X_0 = i)$$

for all states $i \in [0,c]$, and for this, first, it creates a recurrence equation for $u(i)$, splitting the event "A wins" according to what can appear after the first step (first toss) and applying the rule of exclusive and exhaustive causes. If $X_0 = i \in [1,c-1]$, then $X_1 = i+1$ (resp., $X_1 = i-1$) with probability p (resp., q), and the probability of ruin of B starting with A's initial fortune $i+1$ (resp., $i-1$) is $u(i+1)$ (resp., $u(i-1)$). Hence, for $i \in [1,c-1]$ (see a strict proof at the end of the example),

$$u(i) = pu(i+1) + qu(i-1)\,, \tag{3.8}$$

with the limiting conditions

$$u(0) = 0, u(c) = 1\,.$$

The characteristic equation connected to this linear recurrence equation is $pr^2 - r + q = 0$. It has two distinct roots, $r_1 = 1$ and $r_2 = 3$, if $p \neq q$, and a double root, $r_1 = 1$, if $p = q = \frac{1}{2}$. Thus, the general solution is $u(i) = \lambda r_1^i + \mu r_2^i = \lambda + \mu(\frac{q}{p})^i$ when $p \neq q$, and $u(i) = \lambda r_1^i + \mu i r_1^i = \lambda + \mu i$ when $p = q = \frac{1}{2}$. Considering the limiting conditions, it is possible to calculate the values of λ and μ. The answer is, for $p \neq q$,

$$u(i) = \frac{1-(q/p)^i}{1-(q/p)^c}\,, \tag{3.9}$$

and for $p = q = \frac{1}{2}$,

$$u(i) = \frac{i}{c}\,. \tag{3.10}$$

If $p = q = \frac{1}{2}$, *the probability* $\nu(i)$ *that* B *wins when the initial fortune of* B *is* $c - i$ *derives by replacing* i *by* $c - i$ *in expression (3.10) as follows:*

$$\nu(i) = \frac{c-i}{c} = 1 - \frac{i}{c}\,.$$

Check now that $u(i) + \nu(i) = 1$, *implying particularly that the probability that the game eventually finishes.* □

Notice that a generaizaiton of the Gambler's ruin problem will be used in Chapter 8 to study the probability of the eviction of transactions in blockchains.

A confirmation will be given for using the rule of exclusive and exhaustive causes when obtaining a recursive equation (3.8). Indeed, the same proof can be executed for each example of first-step analysis.

Let $Y = \{Y_n\}_{n\geq 0}$ represent the Markov chain derived by delaying $X = \{X_n\}_{n\geq 0}$ by one time unit: $Y_n = X_n + 1$. If $X_0 \in [1, c-1]$, the events "X is absorbed by 0" and "Y is absorbed by 0" are equivalent, and hence

$$\begin{aligned} &P(X \text{ is absorbed by } 0, X_1 = i \pm 1, X_0 = i) \\ &\qquad = P(Y \text{ is absorbed by } 0, X_1 = i \pm 1, X_0 = i)\,. \end{aligned}$$

As far as $\{Y_n\}_{n\geq 0}$ and X_0 are independent provided X_1, the right-hand side of the above equality is represented as follows:

$$P(X_0 = i, X_1 = i \pm 1)P(Y \text{ is absorbed by } 0 \mid Y_0 = i \pm 1)\,.$$

The two chains have the identical transition matrix, and hence in case of sharing the same initial state, they obtain the same distributions. Thus

$$P(Y \text{ is absorbed by } 0 \mid Y_0 = i \pm 1) = u(i \pm 1)\,.$$

Mean Time to Absorption Example 1 solves essentially the identical problem as calculating the probability to reach a state before visiting another. First-step analysis can also be applied to calculate the average time before absorption.

Example 2 (Gambler's Ruin: Continue)**.** *This example extends Example 1. The mean duration* $m(i) = E[T \mid X_0 = i]$ *of the game when the starting fortune of player* A *is* i *meets the recurrence equation*

$$m(i) = 1 + pm(i+1) + qm(i-1) \tag{3.11}$$

for $i \in [1, c-1]$. *Clearly, the coin will be tossed at least once, and then with probability* p *(resp.,* q*) the fortune of player* A *will be* $i+1$ *(resp.,* $i-1$*), and hence* $m(i+1)$ *(resp.,* $m(i-1)$*) more trials will be necessarily on average before one of the players ruins. The limiting conditions are*

$$m(0) = 0, m(c) = 0. \tag{3.12}$$

By solving (3.11) with the limiting conditions (3.12), interpret (3.11) as $-1 = p(m(i+1) - m(i)) - q(m(i) - m(i-1))$. *Writing*

$$y_i = m(i) - m(i-1)\,,$$

we obtaion, for $i \in [1, c-1]$,

$$-1 = py_{i+1} - qy_i \tag{3.13}$$

and

$$m(i) = y_1 + y_2 + \cdots + y_i\,. \tag{3.14}$$

We now solve (3.13) with $p = q = \frac{1}{2}$. *From 3.13,*

$$\begin{aligned} -1 &= \frac{1}{2}y_2 - \frac{1}{2}y_1 \\ -1 &= \frac{1}{2}y_3 - \frac{1}{2}y_2 \\ &\vdots \\ -1 &= \frac{1}{2}y_i - \frac{1}{2}y_{i-1}\,, \end{aligned}$$

and hence, summarizing,

$$-(i-1) = \frac{1}{2}y_i - \frac{1}{2}y_1,$$

that is, for $i \in [1, c]$,

$$y_i = y_1 - 2(i-1).$$

Referencing to the expression (3.14), and seeing that $y_1 = m(1)$, *we derive*

$$m(i) = im(1) - 2[1 + 2 + \ldots + (i-1)] = im(1) - i(i-1)\,.$$

The limiting condition $m(c) = 0$ *gives* $cm(1) = c(c-1)$ *and hence, finally,*

$$m(i) = i(c-i)\,. \tag{3.15}$$

□

First-step analysis leads to the important conditions in the form of a system of linear equations. In the example above, it appears that the system in question has a unique solution, a scenario that persists when the state space is finite, and not the general case with an infinite state space.

Steady-State

Stationarity We demonstrate the central statement of the stability theory of discrete-time HMCs.

Definition 2. *[13, Dfn 5.1 Ch 2][Stationary Distribution] A probability distribution π satisfying*

$$\pi^T = \pi^T \mathbf{P} \tag{3.16}$$

refers to a stationary distribution *of the transition matrix $\mathbf{P}$, as well as of the corresponding HMC.*

The *global balance equation* (3.16) implies that for all states i,

$$\pi(i) = \sum_{j \in E} \pi(j) p_{ij} \,. \tag{3.17}$$

Iteration of 3.16 provides $\pi^T = \pi^T \mathbf{P^n}$ for all $n \geq 0$, and hence, if the initial distribution $\nu = \pi$, then $\nu_n = \pi$ for all $n \geq 0$. Henceforth, if a chain begins with a stationary distribution, it carries the same distribution permanently. Moreover, we see that

$$P(X_n = i_0, X_{n+1} = i_1, \ldots, X_{n+k} = i_k) = \pi(i_0) p_{i_0 i_1} \ldots p_{i_{k-1} i_k} \tag{3.18}$$

is independent on n. Consequently, the chain is *stationary.* It can be said that the chain is in a *stationary regime*, or in *equilibrium*, or in *steady-state.* In summary:

Theorem 2. *[13, Thm 5.1 Ch 2][Steady-State] A chain that starts with a stationary distribution is stationary.*

The balance equation $\pi^T \mathbf{P} = \pi^T$, together with the requirement that π be a probability vector, that is $\pi^T \mathbf{1} = 1$ (where $\mathbf{1}$ is a column vector with all its entries equal to 1), compose when E is finite, $|E|+1$ equations for $|E|$ unknown variables. One of the $|E|$ equations in $\pi^T \mathbf{P} = \pi^T$ is superfluous given the constraint $\pi^T \mathbf{1} = 1$. Clearly, summing up all equalities of $\pi^T \mathbf{P} = \pi^T$ yields the equality $\pi^T \mathbf{P1} = \pi^T \mathbf{1}$, i.e., $\pi^T \mathbf{1} = 1$.

Below there are demonstrated some basic usage examples of Markov chains:

Example 3 (Two-State Markov Chain)**.** *Assume that $E = \{1, 2\}$ and the transition matrix as follows*

$$\mathbf{P} = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 1 & \quad 2 \end{array} \\ \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} \end{array}$$

where $\alpha, \beta \in (0, 1)$. The global balance equations are

$$\begin{aligned} \pi(1) &= \pi(1)(1-\alpha) + \pi(2)\beta \\ \pi(2) &= \pi(1)\alpha + \pi(2)(1-\beta) \,. \end{aligned}$$

This is a dependent system which reduces to the single equation $\pi(1)\alpha = \pi(2)\beta$, to which must be added $\pi(1) + \pi(2) = 1$ expressing that π is a probability vector. We have

$$\pi(1) = \frac{\beta}{\alpha + \beta} \,, \qquad \pi(2) = \frac{\alpha}{\alpha + \beta} \,.$$

□

Example 4 (Symmetric Random Walk)**.** *A symmetric random walk on $\mathbb{Z}$ cannot have a stationary distribution. Clearly, the solution of the balance equation*

$$\pi(i) = \frac{1}{2}\pi(i-1) + \frac{1}{2}\pi(i+1)$$

for $i \geq 0$, with initial data $\pi(0)$ and $\pi(1)$, is

$$\pi(i) = \pi(0) + (\pi(1) - \pi(0))i\,.$$

As $\pi(i) \in [0,1]$, certainly $\pi(1) - \pi(0) = 0$. Thus, $\pi(i)$ is a constant, surely 0 since the total mass of n is finite. Hence for all $i \geq 0$, and consequently, considering the global balance equation, for all $i, \pi(i) = 0$, a contradiction if we want π to be a probability distribution. □

3.1.2 Continuous-time Markov Chains

Poisson Processes

Point Processes This part introduces random point processes of which the simplest example is the homogeneous Poisson process. A random point process is, in general, a countable random collection of points on a real line. In most engineering and operations research applications, a point of such process is the time at which an event occurs, for that reason points are usually refers to events. For example, customers arrivals at a post office or jobs arrivals on a computer's CPU are events in point processes. In biology, an event can be the time of an organism's birth. In physiology, the excitation time of a neuron can also be counted as an event. In the general case, point processes on a straight line appear in stochastic models, where the state of the system changes when some event occurs. Furthermore, we can use the phrase *stochastic systems driven by point processes*, and if the state of the system is discrete, then sometimes it can be called *stochastic discrete-event systems*. The main examples are the Poisson process and Continuous time Markov chains (CTMC).

Definition 3. *[13, Dfn 1.1, Ch 8][Random Point Process] A random point process on the positive half-line is a sequence $\{T_n\}_{n\geq 0}$ of nonnegative random variables such that, almost surely,*

(1) $T_0 \equiv 0$

(2) $0 < T_1 < T_2 < \cdots$

(3) $\lim_{n\uparrow\infty} T_n = +\infty$.

The normal definition of a random point process is less restrictive. Particularly, Condition (2) is relaxed in the more general definition, where multiple points (e.g., simultaneous arrivals) are allowed. When Condition (2) is met, we refer to a *simple point process*. In addition, Condition (3) is not mandatory in the more general definition, where it may happen that $P(\lim_{n\uparrow\infty} T_n < \infty) > 0$:

With positive probability there is an *explosion*, i.e., an accumulation of events in finite time. Conditions (2) and (3) fit the special case of homogeneous Poisson processes that this part focuses on.

The sequence $\{S_n\}_{n\geq 1}$ defined by

$$S_n = T_n - T_{n-1} \tag{3.19}$$

stands for the *interevent* sequence, and sometimes, in the appropriate context, the *interarrival* sequence. For any interval $(a, b]$ in $\mathbb{R}_+$,

$$N((a, b]) \triangleq \sum_{n\geq 1} 1_{(a,b)}(T_n) \tag{3.20}$$

is an integer-valued random variable counting the events occurring in the time interval $(a, b]$. For simplicity, it will be occasionally denoted by $N(a, b]$, omitting the external parentheses. For $t \geq 0$, let

$$N \triangleq N(0, t].$$

Particularly, $N(0) = 0$ and $N(a, b] = N(b) - N(a)$. As the interval $(0, t]$ is closed on the right, the trajectories (or sample paths) $t \mapsto N(t, \omega)$ are right-continuous for almost all samples $\omega \in \Omega$. The sample paths are nondecreasing, have limits on the left at every time t, and jump one unit upwards at each event of the point process. The family of random variables $N = \{N(t)\}_{t\geq 0}$ refers to the *counting process* of the point process $\{T_n\}_{n\geq 1}$. As the sequence of events can be recovered from N, the latter can be called the *point process*.

Counting Process of an Heterogeneous Poisson Process There exist several equivalent definitions of a Poisson process. The one demonstrated here is the most practical.

Definition 4. *[13, Dfn 1.2, Ch 8][Homogeneous Poisson Process] A point process N on the positive half-line stands for a* homogeneous Poisson process *(HPP) with* intensity $\lambda > 0$ *if*

(1) For all times $t_i, i \in [1, k]$, such that $0 \leq t_1 \leq \cdots \leq t_k$, the random variables $N(t_i, t_{i+1}], i \in [1, k-1]$, are independent.

(2) For any interval $(a, b] \subset \mathbb{R}_+$, $N(a, b]$ is a Poison random variable with mean $\lambda(b - a)$.

Henceforth, for all $k \geq 0$,

$$P(N(a, b] = k) = e^{-\lambda(b-a)} \frac{\lceil \lambda(b - a) \rceil^k}{k!}$$

and, particularly,

$$E[N(a, b]] = \lambda(b - a)\,.$$

So, λ is the average density of points.

Condition (1) is the property of *independence of increments* of Poison processes. Particularly, it implies that for any interval $(a, b]$, the random variable $N(a, b]$ is independent of $(N(s), s \in (0, a])$. Consequently, Poisson processes are sometimes refer to *memoryless*. However, it is better to state that the *increments* of HPP have no memory of the past.

Theorem 3. *[13, Thm 1.1, Ch 8][HPPs are i.i.d. Exponentials] The interevent sequence $\{S_n\}_{n\geq 1}$ of an HPP on the positive half-line with intensity $\lambda > 0$ is i.i.d., with exponential distribution of parameter λ.*

The cumulative distribution function (CDF) of an arbitrary intervent time is hence,

$$P(S_n \leq t) = 1 - e^{-\lambda t}\,.$$

Recall that

$$E[S_n] = \lambda_{-1}\,,$$

i.e., the average number of events per unit of time equals the inverse average interevent time.

Long-Run Behavior

The limiting behavior of continuous-time HMCs follows from that of discrete-time HMCs in a usually straightforward manner. We first refer to the ergodic case and then to the absorbing case.

Ergodic Chains

Theorem 4. *[13, Thm 6.1, Ch 8][Long Run Behavior of Ergodic Regular Jump HMCs] Let $\{X(t)\}_{t\geq 0}$ be an ergodic regular jump HMC with state space E and transition semigroup $\{P(t)\}_{t\geq 0}$. Then, for all $i, j \in E$,*

$$\lim_{t\to\infty} p_{ij}(t) = \pi(j)\,, \tag{3.21}$$

where π is the (unique) stationary distribution.

Proof. We use the method of skeletons. Notice that the skeleton $\{X(n)\}_{n\geq 0}$ is an irreducible recurrent HMC. It is positive recurrent, because it has π for a stationary distribution. Although the embedded chain might be periodic, the skeleton $\{X(n)\}$ is not, since the sojourn times of the continuous-time chain in a given state i are i.i.d. exponentials, and this removes periodic behavior for the skeleton. Indeed, two independent continuous-time HMCs with the same transition semigroup $\{\mathbf{P}(t)\}_{t\geq 0}$, but hypothetically different starting distributions, will converge at a finite integer random time, as their skeletons do. Consequently, the result follows by the same coupling argument as in the discrete time case. □

Theorem 5. *[13, Thm 6.2, Ch 8][Ergodic Theorem] Let $\{X(t)\}$ be ergodic and let π be its stationary distribution. Then*

$$\lim_{t\uparrow\infty} \frac{1}{t}\int_0^t f(X(s))\,ds = \sum_{i\in E} f(i)\pi(i), P_\mu \text{ a.s.} \tag{3.22}$$

for all initial distributions μ *and all* $f : E \to \mathbb{R}$ *such that* $\sum_{i \in E} |f(i)|\pi(i) < \infty$.

All corresponding information about Markovian processes and Markov chains with thorough explanation can be found in [13].

3.2 Queueing models

In this section, we introduce some basic notations as well as general insights about queuing models.

In a queueing system, we serve customers or jobs. For instance, in a case of a blockchain network a transaction in the Mempool is a job to be served, i.e., to be consolidated in the following block. If the jobs arrive at time $t_1, t_2, \ldots, t_j$, then the times $T_j = t_j - t_{j-1}$ are called *inter-arrival times*. We usually assume that T_j are independent random variables that are distributed identically. Thus, these variables form the arrival process of a model. The most common arrival process is the *Poisson arrival process*. In this thesis, we always rely on such processes. The inter-arrival times of the Poisson process are known to be i.i.d. exponential random variables.

Applying queueing models, let us answer the following question: *How long does a job need to be served in a system?* In general, we do not know the size of job that is the amount of time it needs to be served in the system. For this reason, we model the job service times as independent random variable. This is also called the *service time distribution*. In addition, when all the jobs are statistically identical we say that the system has a *single class* of jobs while if jobs can be clustered into certain classes in which their elements are statistically identical, then the system is known to be the *multi-class*.

In queuing models, we can have one or more instances that serve the jobs these are usually called *servers*. In general, we assume the servers to be identical. Notice that the variability stays in the job size rather than the speed of the servers. What is more, any queueing system has a maximum number of jobs that it can contain, i.e., the *system capacity*. Naturally, when the system is approaching its saturation point reaching the system capacity and the capacity is infinite, the service time is no longer guaranteed and become somewhat closer to infinity.

Another key parameter of the queueing model is the population size. It refers to the total number of potential jobs that can access the system. Bear in mind that population size is not quite similar to the system capacity as the latter defines the maximum number of jobs that can stay in the buffer and the former refers to the maximum number of jobs that are just willing to enter the system.

Furthermore, the jobs can be served by the system using various scheduling policies and the determination of these *service disciplines* is of importance. It is distinguished following most used disciplines:

- First Come First Served (FCFS)
- Last Come First Served (LCFS)

- Processor Sharing (PS)
- Approximation of the round robin discipline
- Shortest Remaining Processing Time (SRPT).

In our models, we use FCFS within a job class as the one reflecting the behavior inside a blockchain network.

Remark 1 (Kendall notation)**.** *The Kendall notation [42] is a fast and efficient way to describe a queueing system. It has a following base structure* $A/S^C/m/B/K/SD$ *where:*

- *A: inter-arrival distribution*
- *S: service distribution*
- *C: batch size*
- *m: number of servers*
- *B: system capacity*
- *K: population size*
- *SD: service discipline.*

In turn, few notable abbreviations for A and S can be:

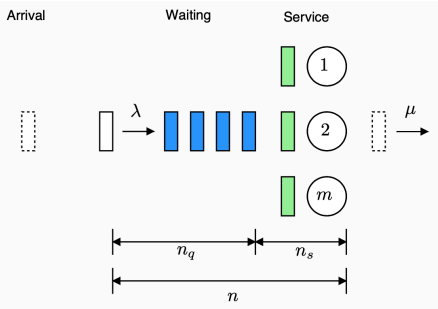
- *M for Exponential*
- *D for Deterministic*
- *G for General.*

Example 5 (Queueing system description)**.** *What are the queuing model parameters provided the following description M/M/3/20/1500/FCFS?*

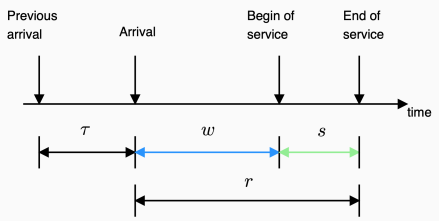
The jobs are entering the system according to Poisson arrival process with exponential service time distribution supported by three servers. Maximum number of jobs in the system is 20 *while the population size is* 1500*. At the end, all the jobs are served with the FCFS discipline.* □

It worth noting that if the system capacity or population size are not specified, we assume them to be infinite. In case of scheduling, discipline FCFS is considered to be default. Thus, M/M/m stands for Poisson arrivals with exponential service time and m servers and batch size 1 working according to the FCFS service discipline. In addition, M/M^B/m would mean that the service time is exponensially distributed with batches of size B. In Chapter 6 we use the latter model with $m = 1$ to study the confirmation time of transactions or the probability of dropping.

Figure 3.1a illustrates a simple queueing system with m servers where the jobs arrive with the rate λ and are served with the rate μ. n_q defines a number



(a) Sketch of a queueing model.



(b) Time description for a queueing model.

Figure 3.1: Queueing system description.

of jobs in the queue and n_s is a number of jobs in the service process while their sum $n_q + n_s = n$ is a total number of jobs in the system (*queue length*).

Figure 3.1b shows the time intervals for the queuing model. τ refers to the inter-arrival time per job so that $\lambda = \mathbb{E}[\tau]^{-1}$, s is the service time per job where $\mu = \mathbb{E}[s]^{-1}$, w is the waiting time per job, and r stands for the response time per job such that $r = w + s$.

3.2.1 Stability and instability of a queueing system

The system is considered to be *unstable* if the number of jobs in the buffer grows continuously and further becomes infinite. Otherwise, we say that the system is *stable*.

For systems with infinite population and infinite buffer capacity we require that the arrival rate must be lower than the maximum throughput of such system:

$$\lambda < m\mu \tag{3.23}$$

Generally speaking, the ratio of λ/μ is called the *load factor* ρ of the queue and represents the relationship between the arrival rate and the service rate. Consequently, Equation (3.23) can be rewritten as $\rho < 1$ providing single server queue.

We study the scenario of the blockchain queueing model with $\rho \geq 1$ in Chapter 8 to assess the probability of confirmation given its offered fee.

3.2.2 M/M/1 queueing system

M/M/1 is one of the most well-known queueing systems. Recall that it characterises with exponential independent inter-arrival times (Poisson arrivals), exponential independent service times, single server, infinite buffer, and FCFS scheduling discipline. The schematic diagram of the process is shown in Figure 3.2. What is more, CTMC underlying the M/M/1 queue is irreducible and

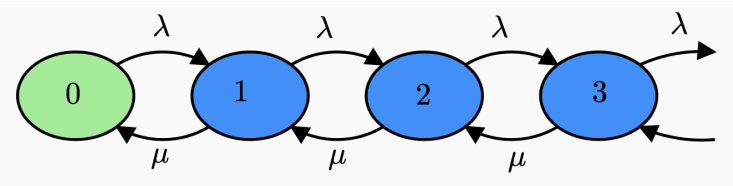


Figure 3.2: State space diagram of M/M/1 queue.

with infinite state space. The ergodicity of the chain is guaranteed if and only if $\rho < 1$. In fact, assuming a reference state 0 we obtain

$$\begin{cases} \pi(0)\lambda = \pi(1)\mu \\ \pi(n)(\lambda+\mu) = \pi(n-1)\lambda + \pi(n+1)\mu \quad n > 0\,. \end{cases}$$

From the former equation we can derive $\pi(1) = \pi(0)\lambda/\mu$. And in general we have:

$$\pi(n) = \pi(0)\left(\frac{\lambda}{\mu}\right)^n \qquad n \geq 0\,.$$

Next, to compute $\pi(0)$ we must impose that $\sum_{n=0}^{\infty}\pi(n) = 1$ so that:

$$\sum_{n=0}^{\infty}\pi(0)\rho^n = \pi(0)\sum_{n=0}^{\infty}\rho^n = \pi(0)\frac{1}{1-\rho}\,. \tag{3.24}$$

where the geometric series converge *if and only if* $\rho < 1$, thus repeating the stability condition of the queueing system.

Finally, equating the result of (3.24) to 1 we obtain:

$$\pi(0)\frac{1}{1-\rho} = 1 \implies \pi(0) = 1-\rho\,.$$

Consequently, in stability the steady-state distribution is seen as follows:

$$\pi(n) = (1-\rho)\rho^n\,,$$

implying that the steady-state probability of observing n jobs in the queue has a geometric distribution with ratio ρ.

This chapter introduced some basic notions and corresponding examples of Markovian processess in order to provide a general understanding of techniques that are used in the following chapters. Summarizing, a Markov process is a stochastic process that meets the Markov property which is often called "memorylessness". Thus, prediction of future states of such process can be done relying on its present state without loss of accuracy compared to real outcomes of the process. There exist two types of Markov processes (also known as *chains*), namely Discrete-time Markov Chains and Continuous-time Markov Chains. Both types of systems are of high applicability in real-world scenarios such that engineering, physics, biology and others. In this thesis we apply each of these techniques to assess the properties of blockchain networks driven by PoW.

Chapter 4

Fairness in PoW private blockchains

4.1 Introduction

Most of academic works study permissioned networks considering consensus methods different from PoW. This is justified by at least two reasons. The first reason is that some scholars prefer to achieve a consensus with algorithms that are less computationally demanding with respect to PoW and the ones with better scalability. The second reason is that in the case of PoW, it would be relatively easy for one of a group of colluding miners to take control of a blockchain by increasing the corresponding HP, e.g., by hiring new machines for a limited amount of time.

However, the motivations for discarding PoW from permissioned networks are still not obvious. Indeed, consider the most popular family of consensus algorithms for permissioned blockchains, namely Byzantine fault tolerance (BFT). On the one hand, it protects authorized miners from the malicious behaviors of a minority of them (*internal trust problem*). On the other hand, BFT does not necessarily guarantee the integrity of the data to the end users (the *external trust problem*).

Indeed, unlike what occurs in the presence of PoW, whenever there is an agreement among the voters (analogy of miners in Bitcoin) of the committee, it is relatively easy for them to change any transaction stored in the blockchain since all the blocks following that with the modified transaction can be instantly regenerated upon agreement of the majority of the "miners".

Example 6. *To provide a better understanding, let us consider a case of a set of companies that maintain a blockchain to store publicly available pollution level data on a certain region in which they operate. Clearly, there is a first level of trust that concerns only the companies participating in the consortium. This can be easily achieved with the voting consensus mechanism. However, at the same time, citizens need to know that the historical data stored in the chain have not been changed; and whether they had been changed, a (possibly high) cost has been paid. BFT does not guarantee this since all companies may be interested in altering past data. Thus, PoW guarantees transparency for the end users, since any change in past data will require the computation of the hashes for the block containing the modified transactions and all the following blocks.*

Another argument in favour of PoW is that although the classic PoW is vulnerable to 50% attacks, the BFT by voting is known to tolerate at most $\lfloor (n-1)/3 \rfloor$ of malicious nodes in a network with weak synchrony conditions [14]. For some cases in permissioned blockchains, it is easier for an attacker to find agreement among other miners than to obtain considerable computational resources as it comes at a cost.

Moreover, to maintain agreement among miners, a network with the BFT consensus mechanism by voting has to use synchronous communications while a PoW blockchain only relies on the timestamps of executing machines [75].

The protocol we propose gives quantitative guarantees for both internal and external trust problems.

Another important aspect of traditional PoW in permissioned networks concerns the balancing of work. In such blockchains, miners do not usually receive a reward for their mining; hence, selfish behavior induces them to reduce the exposed HP with the aim of reducing the energy costs. As a consequence, a different mechanism with respect to rewarding must be adopted to even out the HP used by miners. Notice that this notion of fairness is quite different from that of permissionless networks, where fairness is defined in such a way that the proportion of blocks (and hence the proportion of rewards) obtained by a miner is close to the proportion of his HP [65].

We study the impact of our solution in this regard, showing that an important positive effect, our modified PoW fairly distributes the computational efforts among the miners of the consortium.

4.1.1 Contribution

We introduce a simple PoW mining algorithm for permissioned networks which is based on the use of a sliding window. The main idea is that each miner maintains a control window of size N that stores the information about the consolidators of the latest N blocks in the blockchain. The rule is that a miner m can be present in the window at most once. When a node receives a block from miner m and m is not present in the window, then it behaves as usual (i.e., it verifies the transactions and the hash; and if these are correct, it accepts the new block). Otherwise, the block is rejected. We will discuss how this approach addresses the internal and external trust problems.

We study the security of this protocol with respect to the two major security threats of PoW: the 50% attack, which is particularly dangerous in permissioned networks; and the greedy miner attack in which a miner aims to consolidate more blocks than what is expected from the corresponding HP. Moreover, we provide a quantitative Markovian model of the system to study its fairness, which is here intended as the capability of reducing the gaps among the available HP of the miners.

We observe that these results pose an interesting trade-off. In fact, the total HP of the network is not used because the miners that are present in the window will stop their work; hence, external trust is quantitatively lower than that guaranteed by a plain PoW (which is, however, unable to guarantee internal trust). On the positive side, the HP that is unused, although available, does not become wasted energy. Thus, larger window sizes ensure a high internal trust by protecting the system from the collusion of miners; however, on the other hand, miners reduce the total HP devoted to guarantee the external trust, and vice versa. The quantitative model that we propose allows us to study this trade-off and determine the optimal configuration according to the design needs.

4.1.2 Structure of the Chapter

In Section 4.2, we present the window control blockchain protocol. In Section 4.3, we describe the Markovian model for the performance evaluation of

this protocol and give the algorithm for the computation of the relevant indices. In Section 4.4, we analyse how the window control algorithm reacts to potential security attacks to permissioned networks. In Section 4.5, we analyse the impact of the window control on the fairness of the blockchain network. Finally, Section 4.6 concludes this chapter.

4.2 The problem statement and window-based control

In permissioned networks, we distinguish the problem of trust among the miners and between the set of miners and public observers. While the latter is intrinsically guaranteed by PoW in the sense that even if all the miners agree on modifying a consolidated transaction they have to spend an amount of energy that is publicly known, the former problem requires more attention.

4.2.1 Security vulnerabilities of PoW

Here, we give a brief description of the security threats that affect permissioned blockchain networks:

- *50% attack*. A malicious miner can modify consolidated transactions when it controls at least 50% of the entire computational power of the network [15]. While this attack seems to be very unlikely in public chains with many miners, such as Bitcoin, in case of a restricted pool of miners, this may turn to be a serious threat to the security of the ledger. Indeed, it is possible for one or a small subset of the miners to hire a sufficient amount of computational power so that it can reach the 50% needed to violate the network security. However, in order to conclude the attack successfully, the malicious miner must be able to generate a number of consecutive blocks that coincide with the number of blocks that have been added after the modified block, plus the corrupted block itself.

- *Greedy miner attack*. A malicious miner that controls an amount of HP lower than 50% can consolidate a number of blocks that is still higher than the proportion of the corresponding HP, and this can be done in the following way. Once the attacker mines a sufficient number of blocks to overtake the main chain, he/she does not immediately show his/her progress. Instead, he/she keeps the block unannounced and announces it as soon as some other miner does so with a new block [71]. Specifically, once the hidden fork overtakes the current chain and a new block is announced it then starts to compete with the latter one and naturally wins the competition for the longest chain. Although this problem mainly affects permissionless blockchains in which the rewards per block are expected, such as in Bitcoin, it may also be a problem in permissioned networks. Indeed, the creation of "unnatural" chain forks may reduce the amount of total HP

adopted to guarantee the immutability of the blockchain or, alternatively, may affect the overall system throughput.

4.2.2 Fairness issues in permissioned blockchains

In blockchains, fairness is the property that distributes the efforts required by the distributed ledger evenly among the miners. However, the application of this principle is different for permissioned and permissionless blockchains, especially because the latter have a reward policy to incentivise miners' efforts.

Indeed, the notion of fairness in permissionless blockchains is usually concerned with the fraction of rewards (or consolidated blocks) that a miner possessing a certain HP should statistically receive. This problem has been widely studied in [65] where an entirely new method, called *Fruitchain*, for consolidating blocks has been proposed.

In this chapter, since permissioned networks do not usually adopt a reward mechanism, we propose a different notion of fairness. Ideally, given a certain level of mining difficulty, fairness is achieved when all the miners invest the same amount of HP to the life of the ledger. This is quite difficult to realize since different hardware can be used by miners.

If miners provide different HP, this means that they invest different energy resources (and hence financial efforts) to run the ledger while they all obtain the same service.

The trivial solution could consist of developing a round-robin scheme such that miners consolidate the blocks in turn. However, this solution is ineffective under our assumptions. In practice, the round-robin scheme would allow a miner to totally block the mining process either because of a fault or because of a malicious aim.

4.2.3 Algorithm description: window-based control

In this chapter, we propose a solution to the previously described problems based on a sliding window control algorithm. In this section, we formally describe the algorithm; and in Section 4.3, we show its analysis.

Let us denote the set of M miners that are assumed to have their own identity as $\mathcal{K} = \{m_1, m_2, \ldots, m_M\}$. This means that miners are not anonymous and cannot consolidate new blocks under a different identifier, as could happen in permissionless systems.

Each miner maintains a control window of size N where it stores the identifiers of the latest N blocks' creators. At any time, in the window, at most one block of a miner $m \in \mathcal{K}$ can appear. Upon the announcement of a new mined block, one of these situations may arise:

- If the consolidator of the new block's identifier is already present in the window, then the miners will discard the new block, which is considered an invalid block.

- Otherwise, the block is considered valid under the assumption that all the other conditions are satisfied, e.g., it contains valid transactions and the PoW is solved correctly. The control window is updated with the new miner identifier according to a FIFO policy.

One may propose a delay-based solution where every miner has to comply with certain delay between the proposal of his/her consecutive blocks. However, it remains non-trivial to secure that the delay is long enough because of the random nature of the mining process.

It is worth noting that if there are no forks in the blockchain, then all the miners share the same control window. Otherwise, whenever a fork is solved, the control window must be updated coherently.

Observe that miners whose blocks would be rejected do not even participate in PoW competition. As a consequence, the total hash power that is used by the network is reduced; hence, the electric power consumed by the network also decreases. In Sections 4.4 and 4.5, we will study how this control mechanism may affect network behavior in terms of security and fairness.

Notice that the protocol has two limiting cases. The first limiting case occurs when $N = 0$, i.e., the window mechanism control does not prevent any miners from adding new blocks to the chain. In this case, we obtain the standard PoW-based protocol. The other limiting case occurs when $N = M - 1$. In this case, the mining process follows a round-robin policy in which the blocks are consolidated by the miners in turn. The dimension of the window size allows the definition of intermediate operating conditions, and we will show that it prevents a single miner from taking control of the network (even with more than 50% of the computational power) while it secures the consolidated information with the well-known properties of the PoW algorithm. This tension between the need for a large window size to encourage fair involvement of all miners in the block consolidation process and the need for a small window size to exploit the PoW properties makes the model presented in Section 4.3 crucial for a correct parameterization of the protocol and understanding of its security properties.

4.3 Stochastic Model for the performance evaluation of the algorithm

In this section, we introduce a stochastic model that is based on CTMCs for the sliding window control algorithm described in Section 4.2 and most importantly adopting it to the desired blockchain environment.

The Markov chain underlying the model is ρ-reversible, as described in [54, 56]; hence, this guarantees high numerical tractability. This property allows us to use it to parameterize the model by setting appropriate window sizes. The model considers a single window and is subject to the following assumptions:

- Blocks are generated according to a Poisson process whose rate may depend on the state of the sliding window. This assumption is justified by

the following argument. This process is generated by the superposition of the mining processes of all the miners. Indeed, it is well-known that the PoW requires the computation of a hash and this operation is memoryless. Hence, the time to the next block consolidation for miner m can be assumed to be exponentially distributed with a rate that is proportional to its hash power and that depends on the difficulty parameter set by the network. Moreover, since we can assume that the mining processes are independent, the block generating process is a Poisson process.

- In permissioned blockchains, forks are much rarer than in permissionless networks; therefore, the analyses that we can perform with our model can safely ignore the forks. As a consequence, in our analysis, we assume that there are no forks; hence, all the miners share the same control window.

We denote the window size as N and assume that $N < M$. At each epoch, the state of the window is denoted by vector $\vec{x} = (x_1, x_2, \ldots, x_N)$, where $x_i \in \mathcal{K}$. Moreover, we assume $|\vec{x}|_m = \sum_{i=1}^{N} \delta_{x_i=m}$ be 1 if m is in $\vec{x}$ and 0 otherwise. Finally, as described above, individual miner block generation is assumed to occur according to an independent Poisson process with rate λ_m.

Clearly, the stochastic process $X(t)$ underlying the temporal evolution of $\vec{x}$ is a homogeneous continuous-time Markov chain with finite state space. The transition rates of the CTMC infinitesimal generator are as follows: for $\vec{x}$ and $\vec{x'}$ such that $\vec{x} \neq \vec{x'}$,

$$q\left(\vec{x}, \vec{x'}\right) = \begin{cases} \lambda_m & \text{if } |\vec{x}|_m = 0 \text{ and } \vec{x'} = (m, x_1, \ldots, x_{N-1}) \\ 0 & \text{otherwise.} \end{cases}$$

The state space of $X(t)$ is

$$\mathcal{S} = \{\vec{x} \in \mathcal{K}^N : |\vec{x}|_m \leq 1 \text{ for all } m \in \mathcal{K}\}.$$

The stationary distribution of $X(t)$ can be analytically derived following the lines of [57]. The following theorem provides the exact expression.

Theorem 6. *The stationary distribution $\pi(\vec{x})$ of $X(t)$ is:*

$$\pi(\vec{x}) = \frac{1}{G} \prod_{m \in \mathcal{K}} \lambda_m |\vec{x}|_m \,, \tag{4.1}$$

where $G = \sum_{\vec{x} \in \mathcal{S}} \prod_{m \in \mathcal{K}} \lambda_m |\vec{x}|_m$.

Briefly speaking, the proof of Theorem 6 applies the definition of ρ-reversibility considering those CTMCs that are stochastically similar to their reversed process modulo a state renaming ρ. All corresponding information can be found in [55, 56, 70].

We are interested in real applications where the number of instances of a miner in the window are relevant rather than the order in which they appear.

Corollary 1 provides an analytical expression for such an aggregated stationary probability.

Corollary 1. *Let* $\mathbf{n} = (n_{m_1}, \ldots, n_{m_M})$ *denote an aggregated state with* $n_m \in \{0, 1\}$ *for all* $m \in \mathcal{K}$*, and* $\sum_{m \in \mathcal{K}} n_m = N$*. Let*

$$\mathcal{S}_{\mathcal{K},N} = \left\{ \mathbf{n} : \sum_{m \in \mathcal{K}} n_m = N \text{ and } n_m \in \{0, 1\} \ \forall m \in \mathcal{K} \right\}$$

be the set of all aggregated states.

The stationary probability of observing the aggregated state $\mathbf{n}$ *is:*

$$\pi_A(\mathbf{n}) = \frac{1}{G} N! \prod_{m \in \mathcal{K}} \lambda_m n_m \,.$$

Hereafter, for a model consisting of a set of miners $\mathcal{K}$ and a window size N, we denote the normalizing constant as $G_{\mathcal{K},N}$. Lemma 1 provides the analytical expression for the stationary probability of finding a miner m in the window. Note that this is expressed in terms of the ratio of the normalizing constants of different models.

Lemma 1. *The marginal stationary probability of observing one block of miner* $m \in \mathcal{K}$ *in the window is:*

$$\pi^m_{\mathcal{K},N} = N \lambda_m \frac{G_{\mathcal{K} \setminus \{m\}, N-1}}{G_{\mathcal{K},N}}$$

where $G_{\mathcal{K} \setminus \{m\}, N-1}$ *is the normalizing constant corresponding to a model without miner* m *and a window of size* $N - 1$*.*

The next corollary provides the analytical expression for the throughput of a miner. Note that this is defined as the expected number of mined blocks per unit of time.

Corollary 2. *In the steady state, the throughput for a miner* $m \in \mathcal{K}$ *is given by:*

$$\lambda^*_m = \lambda_m \frac{G_{\mathcal{K} \setminus \{m\}, N}}{G_{\mathcal{K},N}} \,. \tag{4.2}$$

According to our window-based control algorithm, miners whose identifier is present in the window are not joining the mining process. Thus, miner m does not completely use the corresponding HP. Hence, we define the *effective*

HP of miner m as the HP, which is on average devoted to the mining process. Formally, this corresponds exactly to λ_m^* under the convention of measuring the HP in the expected number of consolidated blocks per unit of time.

We can compute the normalizing constant by applying the polynomial convolution algorithm presented in [57].

4.4 Security and performance assessment

In this section, we discuss how the window control algorithm reacts to potential attacks to permissioned networks.

4.4.1 Double spending and greedy miner attacks

From a security perspective, the main advantage of window-based network control is its resistance to attacks that require the consecutive generation of the blocks by a subset of malicious miners. We recall that in permissioned networks, these attacks are possible because we assume that there is a conflict of interests among the miners, where one (or a small subset) of the miners may be interested in changing information that was previously stored in the ledger.

Collusion among malicious miners is possible, and we will consider this possibility. In our evaluation, we assume that malicious miners may collaborate to achieve the same aim, even by sharing their HPs. In other words, a miner whose identifier is present in the window can temporarily transfer its HP to another malicious colluded miner entitled to generate a new block.

As for single-miner threats, 50% and greedy-miner attacks cannot be conducted with the sliding window algorithm. More precisely, we observe the following:

- As noted in Section 4.2, the 50% attack can be a serious problem for permissioned networks based on PoW, which is also one of the reasons for the popularity of BFT in these cases.

  The window control algorithm solves the problem as follows: if there is no collusion among the network miners, then the attacker must produce a certain number of consecutive blocks to conduct a 50% attack. Whenever the window size is positive, this is impossible because the other nodes would reject the proposed fork consisting of consecutive blocks consolidated by the same miner as invalid.

- For a successful greedy miner attack, the selfish miner needs to (i) produce blocks faster than other miners and (ii) make the fork accepted by the others. While the first phase is still doable in a window-controlled network, the second phase cannot be performed since the malicious miner must produce a longer chain of blocks than that actually in use to convince the remaining miners to accept his work. This would require him to mine consecutive blocks, which is again not allowed.

Consequently, it is clear that interested malicious miners will try to mitigate this crucial restriction in order to compromise the past data stored in a blockchain. One feasible solution that they could follow is finding the secret agreement with other miners. Thus, the above vulnerabilities will still occur in the case of several miners who will agree to cooperate. They will act as a mining pool in the network without the sliding window with the only difference that they will try to cheat and deceive others. In addition, if the window size is not smaller than the pool size, the colluded miners will be able to consecutively produce blocks as far as the size of the secret pool. Otherwise, if the window size is smaller than the secret pool size, then its block production is only limited by the fraction of their cumulative HPs.

4.4.2 Security for a single malicious miner

In this section, we consider the threat caused by a single miner that controls different amounts of HP. Recall that, in practice, this is achievable in permissioned networks rather easily since the computational power for the mining processes can be hired by the malicious miner.

From the functional point of view, the fact that the malicious miner (e.g., m_1) cannot consolidate consecutive blocks for any positive window size allows us to conclude that the protocol is safe for any fraction of the total HP controlled by m_1. Furthermore, since all miners' identities are known to the network it is not possible that the same miner would keep mining just using another address.

The impact of window control on the effective HP of the entire network λ_T^* remains to be assessed. Intuitively, this is the total expected HP of the miners that are not present in the window. This can be simply obtained by summing the effective HP of each miner as follows:

$$\lambda_T^* = \sum_{m \in \mathcal{K}} \lambda_m^* \,.$$

Recall that we measure the HP in terms of the expected number of blocks consolidated by a single miner in the unit of time under the condition that the network does not change the difficulty level of the PoW.

Let us consider the scenario in which 20 miners participate in the consolidation process, among which 19 are perfectly balanced, i.e., they expose the same HP. The remaining miner controls a variable fraction of HP that ranges from 5% to 67%. Formally, the vector of hash rates is the following:

$$\boldsymbol{\lambda} = \left(\lambda_1, \frac{100 - \lambda_1}{19}, \ldots, \frac{100 - \lambda_1}{19}\right) .$$

Figure 4.1a shows the effective network HP λ_T^* as a function of the window size. Furthermore, the figure shows that there is a negative dependency between the network effective HP and the window size. On the one hand, larger window sizes result in a more balanced network; however, on the other hand, we have slower blockchain growth and this is where the trade-off between resource

balance and performance appears. Note that if the PoW difficulty adapts to maintain a constant blockchain growth rate, as in the Bitcoin network, then we would compromise the PoW security by requiring a simpler hash computation. To clarify this point, let us consider the mostly unbalanced situation with which that we experimented, i.e., when miner m_1 has 67% of the total HP of the network. Clearly, the speed of the other miners is $(100-67)/19 = 1.74$. If m_1 could know this information, the ideal effective HP would be $1.74 \cdot 20 = 34.74$. With a window size of 1, Figure 4.1a shows that the effective HP is approximately 55. When moving further, the HP falls first to 44 and then gradually falls to almost 20 with a window size of 10. With even larger window sizes, the effective HP drops quickly to zero (with a window size of 20).

Figure 4.1b shows the slowdown of the window-based approach with respect to an ideal situation in which miners can agree to work at the speed of the slowest miner in a perfectly balanced way. In other words, the protocol reduces the number of blocks consolidated per unit of time in the attempt to achieve a fair condition. Since the miners do not explicitly agree on the HP, this is estimated by the use of the sliding window. In our case, the slowdown of the network's effective HP is defined as follows:

$$\mathcal{D} = \lambda_T^* \left(\frac{(100 - \lambda_{m_1}^*)}{M-1} M \right)^{-1} .$$

Figure 4.1b suggests that for large window sizes, the slowdowns of the various scenarios tend to behave as the case of the fully balanced network. Indeed, starting from a window size of 10, the slowdown of every other network is very close to that of the system in which every miner controls 5% of the entire HP. This is explained by the fact that with a window size of 10, we already have a very well-balanced network; hence, the effects of window control on the system are almost indistinguishable from those observable in a perfectly balanced network.

It may be worth emphasizing the fact that when a miner is present in the window, he/she stops his/her mining process; therefore, the HP that is actually used by a node is in general smaller than the available one. Hence, the window control does not increase the energy wasted by the PoW.

Figures 4.1c and 4.1d show the percentages of nodes consolidated by miner 1 and the others, respectively. Notice that when the window size is 19, we have the round-robin discipline; hence, all the miners consolidate 5% of the blocks. We obtain the percentage of consolidated blocks by miner m_i as follows:

$$\frac{\lambda_{m_i}^*}{\sum_{m \in \mathcal{K}} \lambda_m^*} ,$$

i.e., this is proportional to the effective HP used by a miner. Figures 4.1c and 4.1d clearly show that small window sizes are sufficient to smooth out the gap between the HP of m_1 and the others. Indeed, larger window sizes have a strong impact on the effective HP while they give small benefits in terms of smoothing out the differences in miners' HPs. This can also be seen in the

plots of Figures 4.1e and 4.1f. Specifically, note that while large windows have a negative effect on the effective HP of all miners, this is mostly evident for small window sizes and the unbalanced node m_1. For example, if we consider the case of one node controlling 67% of the total HP, his effective HP drops to 13 with a window size of 2.

To conclude, there are two effects of the window, even small windows, on a malicious miner hiring HP to overtake the other nodes: functionally, it prevents the mining of consecutive blocks; and quantitatively, it drastically reduces the imbalance created by this misbehavior.

4.4.3 Security analysis for pools of colluded miners

We analyse the case in which a subset of miners agree on cheating the protocol by changing a past transaction, resulting in a fork of the blockchain that gains the consensus of all the other miners.

Let $\mathcal{K}_C$ be the subset of colluded miners. There are three critical situations that should be considered:

1) $\mathcal{K}_C$ is a minority of all the miners

2) $\mathcal{K}_C$ is a majority of all the miners that controls the minority of the HP

3) $\mathcal{K}_C$ is a majority of all the miners that controls the majority of the HP.

1) $\mathcal{K}_C$ is a minority of all the miners.

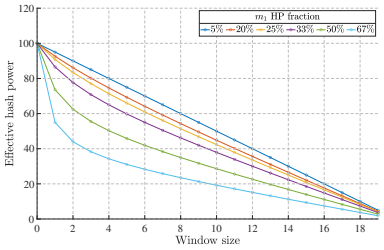
This scenario is a generalization of that considered in Section 4.4.2. Although the system is robust to a single malicious miner, the possibility of collusion complicates the scenario and requires further investigation.

We have to consider two cases:

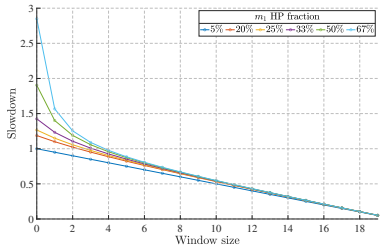
- $|\mathcal{K}_C| > N$. First, we distinguish the network where the number of colluded miners exceeds the window size. Since the colluded miners can transfer their HPs among each other and there is always at least one malicious miner who is not in the window, this case is equivalent to that of the 50% attack in an ordinary PoW blockchain.
- $|\mathcal{K}_C| \leq N$. First, we observe that the miners in $\mathcal{K}_C$ are unable to modify blocks that are more than $N-1$ positions back in the chain, regardless of the percentage of the HP that they control. The blocks in the ledger older than the window size can be considered unmodifiable and hence safe with respect to such an attack.

Example 7. *As an example, consider the permissioned network with a window size of* $N = 5$ *and* 11 *miners,* 5 *of which are colluded and* 6 *are honest. Now, assume that a fraction* $\alpha = 0.6$ *of the total HP is controlled by the colluding miners. Therefore,* $1 - \alpha = 0.4$ *is the HP evenly distributed among the honest party as follows:*

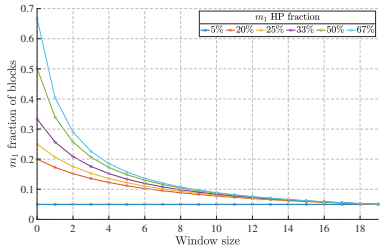
$$\frac{1-\alpha}{6}.$$



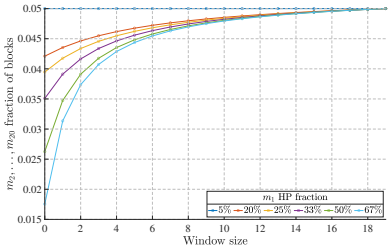
(a) Effective network HP as a function of the window size.



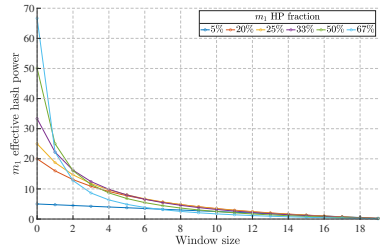
(b) Slowdown as a function of the window size.



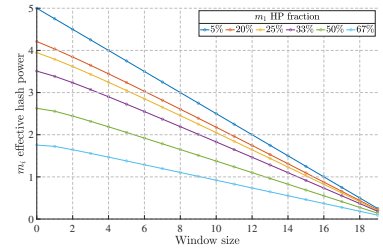
(c) Fraction of consolidated blocks of m_1 as a function of the window size.



(d) Fraction of consolidated blocks of m_i for $m_i \neq m_1$ as a function of the window size.

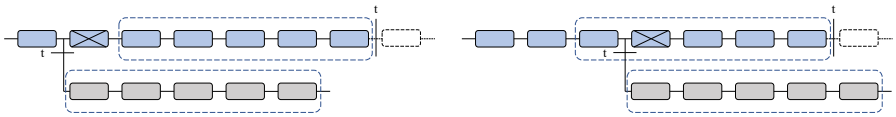


(e) m_1 effective HP as a function of the window size.



(f) m_i effective HP for $m_i \neq m_1$ as a function of the window size.

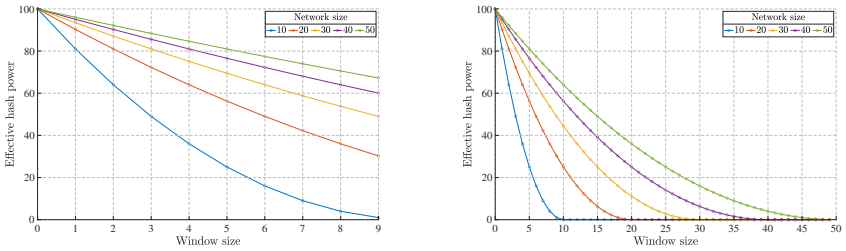
Figure 4.1: Network with one unbalanced node.



(a) Unsuccessful attempt to modify the target block.

(b) Successful attempt to modify the target block.

Figure 4.2: Demonstration of attempts to modify the past blocks of the blockchain of the colluded miner.



(a) Effective network HP as a function of the window size.

(b) Effective network HP as a function of the window size.

Figure 4.3: Different network sizes as the size of the window increases

Since the miners in $\mathcal{K}_C$ can transfer their HPs among each other, the effective hash power of the malicious pool remains constant regardless of the number of malicious miners out of the window. Figures 4.2a and 4.2b represent the cases where at time t the window contains five honest miners and the only remaining honest miner is available to create a new block. In the first example, we see that the colluded pool cannot change the past block marked with a cross. In fact, it is impossible for the malicious pool to create a longer chain than the existing chain because of the rules of the window. Figure 4.2b shows a successful attempt. Clearly, the effective HP of malicious miners is 9 times greater than the HP of the available honest miners. In terms of block creation, this means that by the time the honest miner has one block created, the miners in $\mathcal{K}_C$ will potentially have 9 blocks. However, the window size limits the actions of dishonest miners, and they can produce at most 5 consecutive blocks. It is clear that to have success, the colluded pool needs to overtake the honest party by creating the longest fork. Consequently, since it is impossible for them to produce more than N consecutive blocks in one row, any attempt to rewrite the blocks deeper or equal to N is improbable.

2) $\mathcal{K}_C$ is a majority of all the miners that controls the minority of the HP.

The second case is also worth investigation. We consider that all the honest miners have the same HP.

Table 4.1: Miners' Hash Power

CV	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}	m_{17}	m_{18}	m_{19}	m_{20}
0.0	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000
0.2	4.2995	4.1846	6.0697	6.6619	4.3876	4.8923	4.0834	6.1523	4.8272	4.1176	4.3803	5.0134	4.1967	5.0965	4.1019	4.4555	4.3957	4.7517	6.5653	7.3668
0.5	9.3537	5.8357	1.3516	4.4557	2.0109	5.1976	2.4794	6.9398	3.3745	9.0610	3.4126	1.9169	2.4482	6.5069	4.1149	3.6289	8.1012	5.7067	5.3602	8.7433
1.0	11.1778	10.5201	0.2330	2.0742	1.8475	5.6199	0.3436	2.4887	0.4983	1.0483	15.8871	9.4988	0.4223	10.2551	0.3718	1.7351	4.1179	12.2617	8.5821	1.0168
1.5	0.7740	0.2101	0.3400	14.0988	7.6574	0.2508	18.2027	7.0961	0.3520	0.3511	0.1260	0.3471	22.4673	0.2870	6.9146	0.2040	0.2834	0.2387	19.4122	0.3865

We first notice that if the window size is larger than the number of honest miners, then malicious miners can block the system by simply refraining from mining any new block. Therefore, henceforth, we assume that

$$N \leq |\mathcal{K}_H| < |\mathcal{K}_C|\,,$$

where $\mathcal{K}_H$ denotes the set of honest miners. In this case, the only protection against an attack is the difficulty experienced by the malicious pool to control the majority of the effective HP. In fact, recall that, for the sake of conducting a 50% attack, we need to consider the effective HP.

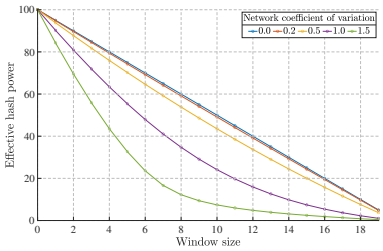
Suppose now that the colluded miners control the fraction of HP such that $\lambda_C < \lambda_H$, where λ_H is the HP of the honest pool. Because of the assumption on the ability of colluded miners to transfer their computational power, we have $\lambda_C^* = \lambda_C$. Since $\mathcal{K}_C$ is working on a fork, only the remaining miners in $\mathcal{K}_H$ compete to access the window. Therefore, $\lambda_H^* < \lambda_H$. In other words, if the window size is too large, we can have the countereffect that we reduce the HP of honest miners too much, allowing malicious miners to succeed in a 50% attack. This observation emphasizes the importance of the proposed quantitative model to analyse the security trade-off that we just described.

To visualize the trade-off, we consider a scenario where the network of honest miners has sizes of 10, 20, 30, 40, and 50. For the sake of simplicity, we assume that they have the same HP. In addition, the window size is smaller than the number of colluded miners; thus, their HP coincides with their effective HP.

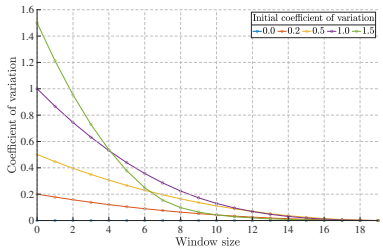
Figures 4.3a and 4.3b show the honest miner network's effective HP as a function of the window size. Consider, for example, a situation with 20 honest miners with an HP of 100 and 25 colluded miners that control an HP of 70. We see that the honest pool maintains the control of the majority of the HP for window sizes up to 3. For larger window sizes, the security of the system can be compromised.

3) $\mathcal{K}_C$ is a majority of all the miners that controls the majority of the HP.

In this case, it is impossible to guarantee the security of the ledger. Indeed, for the same reasons described in the previous case, if $N \geq |\mathcal{K}_H|$, then the mining process can be blocked, while in the opposite case, the malicious pool trivially controls the majority of the effective HP and hence can succeed in a 50% attack. Note that in this situation, both PoW and voting-based agreement algorithms would be vulnerable.

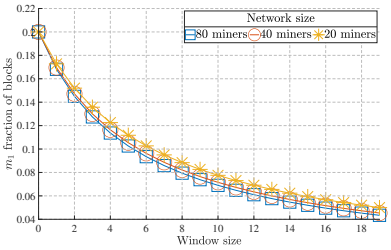


(a) Effective network hash power as a function of the window size for different network coefficients of variation.

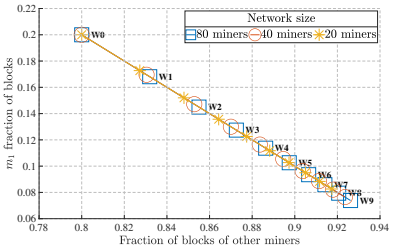


(b) Coefficient of variation of the miner HP fractions as a function of the window size for different network coefficients of variation.

Figure 4.4: Randomly generated scenarios.



(a) Fraction of consolidated blocks of m_1 as a function of the window size.



(b) Fraction of the consolidated blocks of m_1 as a function of the fraction of the consolidated blocks by the others.

Figure 4.5: First scenario with different network sizes.

4.5 Fairness assessment

In this section, we analyse the impact of window control on the fairness of the blockchain network. Recall that, in this context, by fairness, we mean that every miner should invest the same amount of HP to secure the ledger. Therefore, we expect the effective HPs of miners to be closer to each other than their HPs.

Let us consider five scenarios where the HPs of the miners are randomly generated and shown in Table 1. For each scenario, we show the coefficient of variation (CV) of the distribution of miners' HP, where the first corresponds to the ideal case of a perfectly balanced network.

Figure 4.4a shows the network's effective HP as a function of the window size. We notice that, as expected, the window size negatively affects the effective HP, especially for networks with high dispersion of the miners' HP. In other words, if a consortium of miners finds an approximate agreement on the amount of HP to invest (which is clearly the practical scenario), then window control is able to smooth out the unavoidable differences caused by the impossibility of achieving

a good balance with different hardware and the contingent situation that may face a server without reducing the effective HP too much.

Figure 4.4b shows the effect of window control on the coefficient of variation of the effective HPs λ_m^* used by miner m with $m \in \mathcal{K}$. These plots suggest that the small window sizes provide the highest benefits in terms of smoothing out the differences in the effective HPs of the miners. Let us focus on the scenario in which the coefficient of variation is 1. Note that with a window size of 2, the effective HP drops by 20% with respect to the maximum, and the coefficient of variation becomes 0.74. It may seem that the reduction of the HP is high, but that is not the case. Indeed, recall that even if the miners could find an agreement of the HP that they should use, we cannot raise the HP of the slowest, and hence that would become the speed of each individual miner. In the example, the slowest miner m_3 has an HP of 0.233, which would lead to a total network HP of 4.66, which is much lower than the 80 obtained by window control. Clearly, there is a trade-off between the usability of the network and the need for fairness. From another point of view, window control is sufficiently flexible to allow miners some periods of small activity (or inactivity) without drastically reducing the effectiveness of the mining process.

Finally, we consider the network consisting of 19 balanced nodes and node m_1 controlling 20% of the HP of the entire network. We aim to show that the choice of the window size is very robust with respect to the number of miners. In other words, the effects that we observe with a certain window are very similar regardless of the number of nodes. This is shown in Figures 4.5a and 4.5b. Figure 4.5a shows that with a window size of 0, node m_1 generates 20% of the blocks, as expected by its HP. With a window size of 3, this percentage is almost halved independent of the network size. Figure 4.5b confirms that the highest benefits in terms of controlling the system fairness are achieved with small window sizes; and to this aim, it seems useless to exceed a few units.

4.6 Conclusion

In permissioned blockchains, we have to address two levels of security issues. The first concerns the possible lack of trust among the miners, and the second is the lack of trust between the end users and the consortium of miners.

In this chapter, we have proposed a step towards the solution of the problems given by the application of PoW in permissioned blockchains. Indeed, although PoW is able to quantify the energy effort required to change a consolidated transaction, it is very weak in guaranteeing trust among miners. The sliding window control that we introduced does not allow a miner to consolidate two or more consecutive blocks from the latest N blocks, where N is the window size. This does not allow a miner to take control of the mining process by reaching 50% of the total HP. Clearly, if this is a remote possibility for public blockchains such as Bitcoin, for permissioned networks, it represents a high risk. Bear in mind that while public PoW blockchains often contain numerous amount of participants and are generally common, like Bitcoin or Ethereum, the private

ones tend to contain much smaller amount of nodes and usually are not open for everyone as of specificity of such networks that are formed for unique purposes, e.g., by an association of companies in one area.

Moreover, in permissioned blockchains adopting PoW, it is crucial to balance the effective HP provided by the miners. The idea behind this is reaching a certain level of fairness among the miners in terms of energy consumption. This is not a problem in systems where miners are rewarded for their work, but smoothing out the differences in the computational power of different miners in permissioned networks is important to encourage participation in the mining process.

The sliding window algorithm presented in this chapter contributes to increase the security of PoW in permissioned blockchains and their fairness by reducing the effective HP used by the system. Indeed, miners whose identifiers are present in the sliding window stop mining new blocks. This aspect of the algorithm clearly poses a trade-off problem that we addressed with the quantitative model.

The quantitative analysis that we conducted is based on a Markovian model and shows that small window sizes are sufficient to smooth out the differences in the potential HP of a heterogeneous group of miners, thus achieving fairness. Furthermore, the method is robust to the malfunctioning of some of the nodes that may be temporarily unavailable or faulty.

Finally, it is worth noting that its implementation requires minor changes to the existing PoW-based blockchain software and hence represents a viable solution to the above mentioned problems.

Chapter 5

Transaction confirmation time: a system perspective

5.1 Introduction

This chapter extends the state of the art of model-driven analyses and prediction of confirmation delays in blockchain systems driven by PoW. The limited maximum throughput imposed by some invariant properties of blockchains and the auction among the pending transactions lead to intriguing problems in the performance assessment of such systems. Particularly, the confirmation delay depends on the traffic intensity and the offered fees. To study this problem we will use the most famous blockchain, i.e., Bitcoin. More specifically, we highlight the following contributions:

- Comparison between fixed block size and fixed block capacity: most blockchains set a maximum block size (e.g., 1 MB for Bitcoin) and allow for transactions with variable size. This means that a saturated block may contain a variable number of transactions. As opposed to this assumption, almost all models for confirmation delay prediction assume a fixed block capacity, i.e., a constant maximum number of transactions in the block. Is this assumption valid? To answer this question, we introduce an exact model for the analysis of fixed size systems and compare its performance with the state of the art model with fixed capacity. Our findings show that, for Bitcoin, the assumption of fixed capacity is a sound simplification of the real system. This happens because the distribution of the number of transactions in full blocks is highly concentrated around its mean.

- Another simplifying assumption on the system consists in neglecting the batch-service style of blockchains and approximate it by a single service with higher speed. This intuitively corresponds to a "fluidification" of the service process. However, we notice that this approximation is too imprecise and leads to heavy underestimations of the confirmation delay. The model with single service has been previously used for estimating the probability of transaction dropping in saturated systems (i.e., when the arrival intensity is higher than the maximum throughput). We introduce a novel model for the estimation of the dropping probability where the batch service style is preserved and hence provides a higher accuracy with respect to the results available at the state of the art.

In addition, all the results of this chapter as well as of following related chapters are supported with real data-driven numerical evaluation that is introduced in Section 5.3. The data are retrieved using our own Bitcoin node running the full copy of the ledger.

5.1.1 Structure of the Chapter

The chapter is structured as follows. Section 5.2 introduces a model for a queueing system with random batch size that allows us to investigate the validity of the hypothesis of blocks with fixed capacity rather than fixed size. In Section 5.3, we propose a new model for estimating the transaction dropping

probability given the offered fee. What is more, the model is validated with a stochastic simulation and we draw some insights of the dropping process in PoW blockchain. Finally, Section 5.4 proposes some final remarks.

5.2 Comparison of fixed block size and fixed block capacity

The goal of this section is to investigate the difference in the confirmation delay between two blockchain systems with fixed capacity and fixed size but identical maximum throughput.

Recall that, when we assume a fixed block size, this translates into a random capacity since transactions have different lengths in bytes.

Most models for the analysis of confirmation times, assume fixed block capacity although, in practice, blocks have a fixed size. The investigation of the robustness of this simplifying assumption is carried out thanks to two queueing models that will be parameterised with Bitcoin blockchain data.

The section will first consider the queueing model M/M^B/1 with random batch size and a single class. Bear in mind that 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 [32, 45]. Recall that, according to Kendall's notation, 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 in our case represents the number of transactions that a block can fit, and 1 denotes that the system consolidates one block at a time. Then, following the reasoning of [7], we extend the analysis to the case of multiple classes of users. What is more, priority class refers to the set of transactions grouped by its fee per Byte from highest to lowest..

5.2.1 Queueing model for fixed block size

In this section, we extend the state of the art on blockchain confirmation time analysis with the introduction of a queueing model with random service batch size. First, we ignore the priority of the transactions and consider a simple FCFS discipline, then we will extend the result to cope with priorities.

Consider a FCFS queue where jobs arrive in a Poisson stream with rate λ. Service instants occur at exponentially distributed intervals with mean $1/\mu$. At each service instant, an i.i.d. random batch size is chosen, with $K+1$ possible values. These are, in increasing order, $b_0 = 0$ with probability β_0, b_1 with probability β_1, ..., b_K with probability β_K. If the batch size is b_k and there are n jobs present in the queue, a number of jobs equal to $\min(b_k, n)$ are served instantaneously. Choosing a batch size 0 means that even if the queue is non-empty, no jobs are served at that instant.

The average batch size, B, is given by

$$B = \sum_{k=1}^{K} b_k \beta_k \ . \tag{5.1}$$

The queue has a negative trend (is stable) if the following condition is satisfied:

$$\lambda < \mu B \ . \tag{5.2}$$

It is clear that this inequality is the necessary and sufficient condition for ergodicity. However, we shall also provide a formal proof.

For every integer $j = 1, 2, \ldots, b$, where $b = b_K$ is the largest possible batch size, let q_j be the probability that the next random batch will have room for at least j jobs:

$$q_j = \sum_{s=j}^{K} \beta_s \ \ ; \ \ j = 1, 2, \ldots, b \ . \tag{5.3}$$

We have $q_1 = 1 - \beta_0$ and $q_b = \beta_K$.

The sum of all probabilities q_j is equal to the average batch size:

$$\sum_{j=1}^{b} q_j = B \ . \tag{5.4}$$

To establish that result, note that q_j includes β_1 for values of j between 1 and b_1; it includes β_2 for j between 1 and b_2; ...; q_j includes β_K for j between 1 and b. Therefore, the left-hand side of (5.4) coincides with the right-hand side of (5.1).

Let π_n be the steady-state probability that the queue is in state n, i.e., there are n jobs present. Equating the transitions at which the state increases from n to those at which it decreases to n, we obtain the following set of balance equations.

$$\lambda \pi_n = \mu \sum_{j=1}^{b} q_j \pi_{n+j} \ \ ; \ \ n = 0, 1, \ldots \ . \tag{5.5}$$

We shall obtain the general solution to this set of equations in geometric form:

$$\pi_n = C z_1^n \ , \tag{5.6}$$

where C and z_1 are some positive constants. Substituting (5.6) into (5.5), we find that the equations are satisfied as long as z is a zero of the following polynomial of degree b.

$$P(z) = \lambda - \mu \sum_{j=1}^{b} q_j z^j \ . \tag{5.7}$$

In addition, in order that we may obtain a probability distribution, z_1 must be a positive real number in the interval $0 < z_1 < 1$.

We have $P(0) = \lambda > 0$ and $P(1) = \lambda - \mu B < 0$, according to (5.4) and (5.2). Therefore, $P(z)$ has a real zero, z_1, in the interval $(0, 1)$. This provides a normalizable solution to the set of balance equations and allows us to write

$$\pi_n = (1 - z_1)z_1^n \ ; \ n = 0, 1, \dots \ . \tag{5.8}$$

In order to complete the proof of ergodicity and demonstrate that (5.8) represents the unique steady-state distribution of the queue, we must show that $P(z)$ has no other zeros in the interior of the unit disk. To do that, we shall introduce another polynomial, $R(z)$, defined as follows:

$$R(z) = (1 - z)P(z) \ . \tag{5.9}$$

Using the relations (5.3) and cancelling terms, we can write $R(z)$ explicitly as

$$R(z) = \lambda - (\lambda + \mu)z + \mu z \sum_{j=0}^{b} \beta_j z^j \ . \tag{5.10}$$

Definition (5.9) implies that $P(z)$ and $R(z)$ have the same zeros, except that $R(z)$ has an extra zero at $z = 1$. We shall prove the following result.

Lemma 2. *When the inequality (5.2) is satisfied, the real number z_1 appearing in (5.8) is the only zero of $R(z)$, and hence of $P(z)$, in the interior of the unit disc.*

Proof. We invoke Rouché's theorem, which states that if two holomorphic functions, $\phi(z)$ and $\psi(z)$, satisfy $|\phi(z)| > |\psi(z)|$ on a simple closed contour, then $\phi(z)$ and $\phi(z) + \psi(z)$ have the same number of zeros inside that contour. Each zero is counted according to its multiplicity.

We represent $R(z)$ as $R(z) = \phi(z) + \psi(z)$, where

$$\phi(z) = -(\lambda + \mu)z$$

and

$$\psi(z) = \lambda + \mu z \sum_{j=0}^{b} \beta_j z^j.$$

The closed contour is the unit circle. When $|z| = 1$, $|\phi(z)| = \lambda + \mu$. Applying the triangle inequality to $\psi(z)$, we find

$$|\psi(z)| \leq \lambda + \mu \sum_{j=0}^{b} \beta_j = \lambda + \mu \ .$$

Moreover, the inequality is strict everywhere on the contour, except at $z = 1$, where it is an equality.

Note that the derivative of $R(z)$ is positive at $z = 1$. This is because $R'(1) = -P(1) = \mu B - \lambda > 0$. Hence, we can choose a sufficiently small number, ϵ, such

that $R(1-\epsilon) < 0$. This implies $|\phi(1-\epsilon)| > |\psi(1-\epsilon)|$. Modifying the contour slightly in the vicinity of $z = 1$, by making it pass through the point $z = 1-\epsilon$, would ensure that the inequality $|\phi(z)| > |\psi(z)|$ is strict on the entire modified contour.

Function $\phi(z)$ is linear and has a single zero, $z = 0$, inside the contour. Therefore, by Rouché's theorem, $R(z)$ also has a single zero inside the contour. That zero must be z_1, which completes the proof. □

We can extend the result to several priority classes based on the offered fees following the same reasoning of [7]. Let us number the classes with natural number $1, 2, 3, \ldots, K$, where class i has strict priority on class j if $i < j$. Therefore, class 1 is that offering the highest fee per Byte. Let f_i be the frequency of transactions belonging to class i, $\sum_{i=1}^{K} f_i = 1$. Thus, the intensity of arrivals of transactions of class i is $\lambda_i = \lambda f_i$, where λ is the total arrival intensity. The service order among transactions of the same class is FCFS.

We can reason recursively, beginning from the base case $i = 1$. All the transactions belonging to class 1 are completely insensitive to the behavior of classes $j > i$ because of the hard priority between the services. Thus, we can use Equation 5.8 to derive the stationary distribution of the occupancy of class 1 transactions in the Mempool. The mean of the distribution can be obtained as $L_1 = z_1/(1-z_1)$. Notice that the equation differs from the celebrated M/M/1 results since $z_1 \neq \lambda_1/B\mu$. By Little's law, we can obtain the expected response time T_1 for class 1 jobs.

Consider now class i, with $i > 1$ and assume we know the stationary distribution and the expected performance indices of classes $1, \ldots, i-1$. Class i competes with all the classes with higher priority than its own. Let $\Lambda_i = \sum_{j=1}^{i} \lambda_j$. Then, we can use Equation (5.8) with the aggregated arrival intensity Λ_i and derive the expected aggregated occupancy $\overline{L}_i$ thanks to the computation of the associated z_i. At this point, we can obtain the expected occupancy L_i and response time T_i of transactions of class i as follows

$$L_i = \overline{L}_i - \sum_{j=1}^{i-1} \frac{\lambda_j}{\Lambda_i} L_j \,, \quad T_i = \frac{L_i}{\lambda_i} \,.$$

5.2.2 Numerical investigation

Let us consider the model with fixed block size whose capacity distribution is shown in Table 5.1 and the model with fixed block capacity that corresponds to the average of this distribution. For both models, we set $\mu = 1/600$, i.e., an expected inter-block delay of 10 minutes in Bitcoin. This implies that the maximum throughput for the two models is the same. Finally, we consider a very simple model, namely a M/M/1 queueing system, with analogous maximum throughput.

In Figure 5.1, we show the expected occupancy of the Mempool as function of the system's offered load. We can see that the fixed size (green curve) and the

Table 5.1: Distribution of the full block capacity in Bitcoin network as measured between 720489 and 721489 blocks.

Block capacity	Probability
150	0.0058
450	0.0058
750	0.0252
1050	0.0213
1350	0.0349
1650	0.0814
1950	0.0930
2250	0.1494
2550	0.1938
2850	0.1996
3150	0.1376
3450	0.0484
3750	0.0019
4050	0.0019

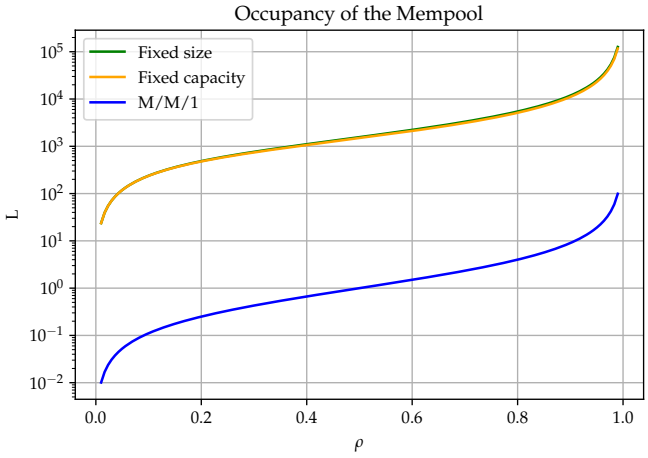


Figure 5.1: Comparison among different models for the prediction of the confirmation delay.

fixed capacity (yellow curve) tend to overlap each other. Recall that, by Little's law, the expected confirmation time is directly proportional to the expected occupancy.

From the numerical point of view, we find the unique root of $R(z)$ as defined in (5.10) in $(0, 1)$ thanks to the bisection algorithm. The case of fixed capacity is solved by using again $R(z)$ as defined in Equation (5.10) with $\beta_{2374} = 1$.

We observe that the models with fixed capacity and fixed size exhibit very similar performance, while the M/M/1 system has a very different behavior.

We conclude that we can safely interchange a model for a system with fixed block size and fixed block capacity, but neglecting the batch service policy and resorting to a M/M/1 approximation can lead to models that are potentially very inaccurate.

Given these observations, the analysis and validation of the multi-class scenario would be redundant with prior findings of the literature, e.g., [7, 39] that have already been done for fixed capacity.

5.3 Dropping policy for fixed block capacity

The analysis presented so far shows that, while the assumption on the fixed capacity of blocks is sound, the replacement of the the batch service with a single service may lead to an inaccurate evaluation of the system performance.

Motivated by this observation, we extend the state of the art of the models for the evaluation of the dropping probability of cheap transactions with the introduction of a model considering batch services. The solution presented here is simpler and more direct than the one already existing in the literature [7].

5.3.1 Bulk services and droppings

In this section, we study an approximate solution of the problem of transaction dropping. As we will see in the experiment section, the accuracy is extremely high and suitable for practical scenarios.

Consider a FCFS queue where jobs arrive at rate λ, service batches of size B are offered at rate μ, and each waiting job is dropped after an exponential random time with parameter γ. That queue is always ergodic, as long as $\gamma > 0$.

The steady-state probabilities, π_n, that there are n jobs in the queue, satisfy the following equations, balancing the up and down flows across a cut between states n and $n + 1$:

$$\lambda\pi_n = (n+1)\gamma\pi_{n+1} + \mu \sum_{i=n+1}^{n+B} \pi_{n+i} \;\; ; \;\; n = 0, 1, \dots \; . \tag{5.11}$$

Define again the generating function

$$g(z) = \sum_{n=0}^{\infty} \pi_n z^n \; . \tag{5.12}$$

Multiplying equation (5.11) by z^n and summing, we obtain

$$\left[\lambda z^B - \mu \sum_{i=0}^{B-1} z^i\right] g(z) = \gamma z^B g'(z) - \mu \sum_{i=0}^{B-1} \pi_i z^i \sum_{j=0}^{B-1-i} z^j \ . \tag{5.13}$$

Thus the steady-state distribution of the queue size is determined by B unknown probabilities π_0, π_1, ..., π_{B-1}. In particular, the average number of jobs in the queue, L, is given by $g'(1)$ which, according to (5.13), is equal to

$$L = \sigma - B\eta + \eta \sum_{i=0}^{B-1} (B-i)\pi_i \ , \tag{5.14}$$

where $\sigma = \lambda/\gamma$ and $\eta = \mu/\gamma$.

The regime that makes dropping necessary is one where $\lambda > B\mu$. Moreover, since miners are reluctant to drop transactions from the Mempool, the dropping parameter is usually rather small. Under those circumstances, the probabilities π_n tend to increase with n, roughly while $\lambda > B\mu + n\gamma$, and when $\lambda < B\mu + n\gamma$, they decrease with n. The idea of the proposed numerical solution is to choose a queue size N which is some multiple of B and also satisfies the inequality $\lambda < B\mu + N\gamma$. The value of π_N is initially fixed arbitrarily (e.g., $\pi_N = 1$). For $n > N$, the probabilities π_n are computed using the recurrences

$$\pi_{n+1} = \frac{\lambda}{B\mu + (n+1)\gamma} \pi_n \ . \tag{5.15}$$

This relies on the observation that when the queue is large, it behaves as if jobs are served one at a time, at rate $B\mu$.

For $n < N$, the probabilities π_n are computed by means of backward recurrences, so as to satisfy the balance equations (5.11):

$$\pi_n = \frac{1}{\lambda} \left[(n+1)\gamma\pi_{n+1} + \mu \sum_{i=n+1}^{n+B} \pi_{n+i}\right] \ . \tag{5.16}$$

Finally, all probabilities π_n are normalized so that their sum is 1.

After some experimentation, a good "rule of thumb" for choosing N is to take the smallest multiple of B which is (a) at least $3B$, and (b) produces a ratio $\lambda/(B\mu + N\gamma)$ no larger than 0.8.

The above algorithm is recapitulated in Figure 5.2.

The numerical solution can, of course, be applied to the higher, non-saturated priority classes since all job types are treated in a uniform manner. However, when $\lambda < B\mu$, the probabilities π_n tend to decay quite quickly. Consequently, the above algorithm should be modified slightly in order to avoid numerical distortions.

We saw in Section 5.2 that in the absence of dropping, the state of a stable queue with bulk services is distributed geometrically, with parameter z which is the zero of a certain polynomial in the interval $(0, 1)$. That zero exists when

1. Choose a suitable queue size, e.g. $N = B \max(3, \lceil(\sigma/(0.8B) - \eta)\rceil)$.
2. Set $\pi_N = 1$.
3. For $n = N+1, N+2, \ldots$, compute π_n according to (5.15), stopping when the resulting value becomes negligibly small.
4. For $n = N-1, N-2, \ldots, 0$ compute π_n according to (5.16).
5. Normalize probabilities by dividing them by their sum.
6. Compute average queue size L according to (5.14).

Figure 5.2: Numerical solution algorithm for the system with transaction dropping.

$\lambda < B\mu$. Hence, the queue behaves like an M/M/1 queue where the offered load $\lambda/B\mu$ is replaced by z. Alternatively, the bulk services are replaced by single ones with parameter $\nu = \lambda/z$. The presence of reneging in such a queue can only speed up the geometric decay.

We can therefore find a queue size N such that $\pi_N \approx z^N(1-z) < \epsilon$, for some sufficiently small ϵ. That is the value of N that should be chosen in step 1 of the algorithm. In step 3, the recurrences (5.15) for $n > N$, should be modified by replacing the effective service rate $B\mu$ with $\nu = \lambda/z$:

$$\pi_{n+1} = \frac{\lambda}{\nu + (n+1)\gamma}\pi_n \ . \tag{5.17}$$

All other steps remain the same.

It remains to study the dropping probability. Given π_i, for $i \geq 0$, the total flow abandoning the system for dropping is

$$\sum_{i=1}^{\infty} \pi_i(i\gamma) = \gamma L \, . \tag{5.18}$$

Thus, the probability of dropping a transaction is given by $p_{drop} = \gamma L/\lambda$, i.e., the ratio between the outgoing flow due to dropping and the overall incoming flow. Indeed, recall that the capacity of the queue is infinite but its occupancy is finite with probability 1 since the stochastic process underlying the queue is always ergodic for $\gamma > 0$.

Given the study of a single class queue with dropping, the extension to the multiple class case can be done following the same observations proposed in Section 5.2.

Table 5.2: Classes of priorities and frequencies as measured from the confirmed transactions.

Priority	Class	Range [S/B]	Dist. in high-load
Highest	1	$[100, \infty)$	0.069
	2	$[60, 100)$	0.235
⋮	3	$[40, 60)$	0.315
	4	$[20, 40)$	0.184
Lowest	5	$[0, 20)$	0.196

5.3.2 Numerical investigation

In order to test the accuracy of the model, we have measured the distribution of the fees per Byte offered by the users in the blockchain of bitcoin. This information is publicly known for confirmed transactions. Thus we have obtained the distribution shown in Table 5.2 considering 5 priority classes. The unit of measure S/B stands for Satoshi per Byte where a Satoshi is 10^{-8} BTC.

In all experiments, we use a block a size $B = 2374$ and $\mu^{-1} = 600$ s.

5.3.3 Description of the simulator

Differently from confirmed transactions, dropped transactions do not leave any trace in the blockchain logs. Thus, the validation of the model must be carried out thanks to a simulation experiment. To this aim, we have developed a simulation model parameterized according to our measurements in the bitcoin blockchain (see Table 5.1 and $B = 2374$) and we have run several tests for different workload conditions.

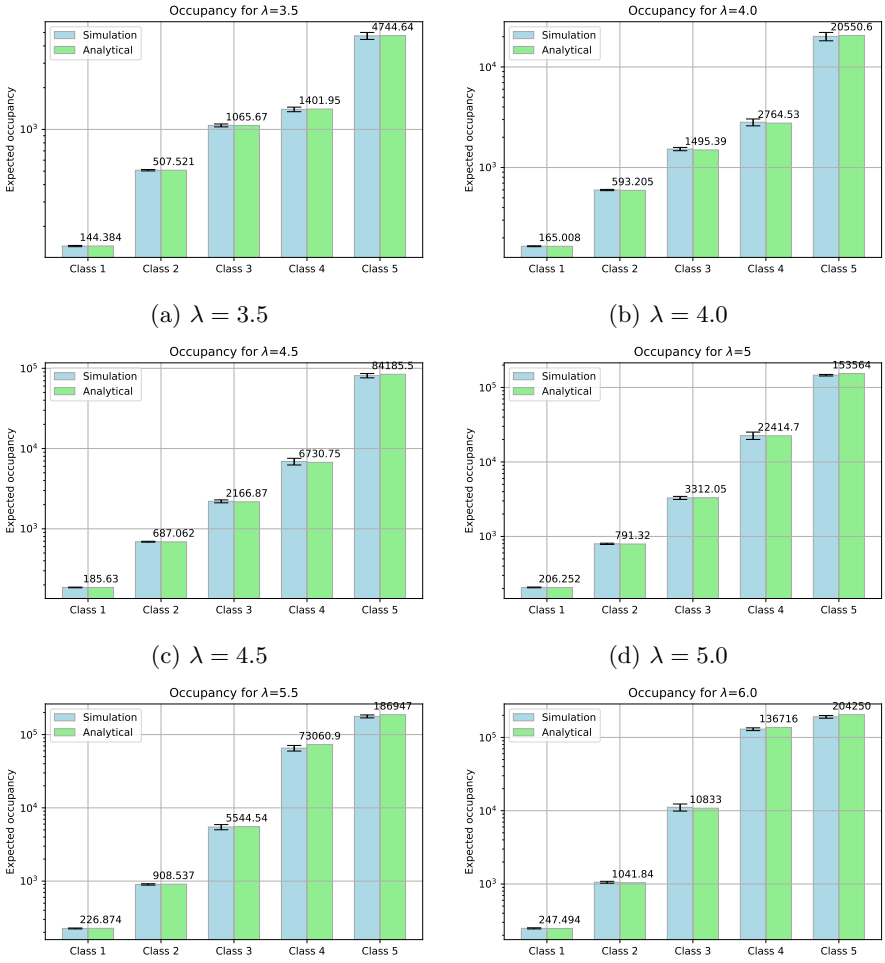
For each considered arrival intensity we run 20 independent experiments each of which consists of 20 millions of events. The first 2 millions are ignored from the statistics to take into account the warm up of the model. Then, we build a confidence interval based on a Student-T distribution with 95% of confidence for the expected occupancy of each class of transactions.

Notice that the simulation experiments are particularly time consuming since the batch service mechanism competing with the dropping policy creates noisy estimates especially in the heavily loaded classes, i.e., those with lower priority. This enhances the importance of a reliable numerical model that can be solved efficiently as that proposed in Sections 5.2 and 5.3.

5.3.4 Results

In Figure 5.3, we show the expected occupancy for the five classes in heavy load obtained by using the analytical model and the stochastic simulation. Bear in mind that the y-axis is in logarithmic scale.

Notice that, without dropping, the stability condition for the queue would be $\lambda < B\mu = 3.96$ tx/s. In these experiments, we begin from $\lambda = 3.5$ tx/s and we reach $\lambda = 6.0$ tx/s.

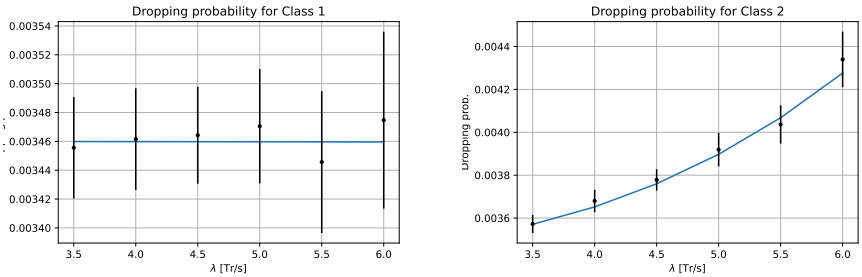


(a) $\lambda = 3.5$ (b) $\lambda = 4.0$

(c) $\lambda = 4.5$ (d) $\lambda = 5.0$

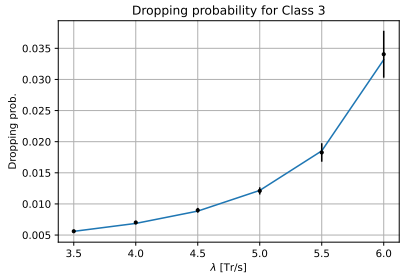
(e) $\lambda = 5.5$ (f) $\lambda = 6.0$

Figure 5.3: Expected occupancy per class under different load factors. Dropping can occur in lowest priority classes.

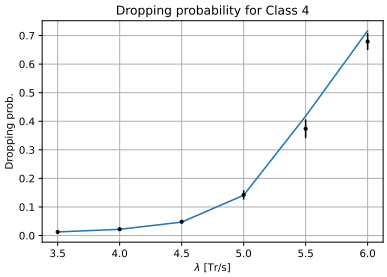


(a) Class 1

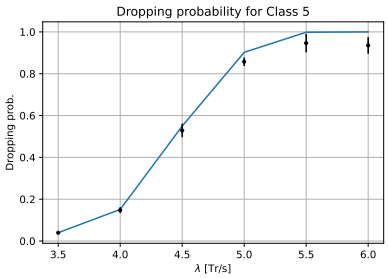
(b) Class 2



(c) Class 3



(d) Class 4



(e) Class 5

Figure 5.4: Dropping probability as function of the aggregated arrival rate for different priority classes.

The plots of Figure 5.3 show a very good agreement in the expected occupancy per class between the results obtained thanks to the analytical model and the estimates obtained with the simulation. With the only exception of Class 5 for $\lambda = 6$ tx/s, all analytical values fall within the confidence intervals of the simulations. This observation is quite important because the solution of the analytical model introduced some approximations that needed to be validated.

With respect to the approximation introduced in [7], the model proposed here has several advantages: (i) it can be used uniformly for saturated and non-saturated classes; (ii) the accuracy remains high both if the class is heavily saturated and if it is lightly saturated, while the experiments presented in [7] showed a drop in the quality of the approximation for the latter case; (iii) differently from the model of [7], the model that we present here can study the dropping policy for all classes, and not just the one with the lowest priority.

From the system perspective, we notice that the progressive increasing of the traffic intensity only marginally affects Class 1 transactions, while it has a huge impact on the last two priority classes.

This observation is supported also by the plots of Figure 5.4. We notice that the dropping probability for the highest priority class is close to 0 independently of the transaction arrival rate. On the other hand, as the priority of classes diminishes, we have more probability of transaction droppings since they are the first to lose jobs. Class 3 is the first for which we start to observe a significant increase of the dropping probability that passes from 0.5% to 3.4%. For Class 5, from $\lambda = 5.5$, we have a probability of dropping close to 1.

Regarding the quality of the approximation for the dropping probabilities, we observe a very good agreement with the simulation model. For the lowest priority class, it seems that the model tend to overestimate the dropping probability for $\lambda > 5.5$ tx/s, but this happens in a regime that, although of theoretical interest, is practically not very relevant for Class 5 since such a dropping probability would be too high to be accepted.

5.4 Conclusion

In this chapter, we have studied some queueing models for the analysis of the average confirmation time and dropping probabilities in PoW blockchain. More specifically, the experiments have been carried out with data retrieved from the bitcoin blockchain.

The first model that we have proposed has been compared with two other models used for the derivation of the transactions' confirmation time given a certain offered fee per Byte. Indeed, while we propose a queueing system with fixed block size (and hence random capacity), in the literature it is usually assumed a fixed block capacity or even a non-batch service (one transaction confirmed per service epoch). Our findings show that while the assumption of a fixed block capacity is robust and gives results that are barely indistinguishable from those assuming fixed block size, the M/M/1 approximation is inaccurate.

The second model that we propose studies the dropping probabilities in

Bitcoin network. With respect to the state of the art, we propose a simpler solution that showed to be accurate in low, moderate and high loads. The fact that the same numerical algorithm can be used both for saturated and unsaturated classes greatly simplifies its usage. The validation of this model has been carried out by resorting to stochastic simulations since the data about dropped transactions are not present in the blockchain. The accuracy of the model both in the prediction of the expected occupancy and the dropping probability is high.

As future work, we plan to extend this approach to other blockchains that have other restrictions than the maximum block size to control the number of transactions that can be stored in a block, e.g., Ethereum blockchain.

Chapter 6

Transaction confirmation time: a user's perspective

6.1 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 [19, 35, 67]. 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. This mechanism causes dynamics worthy of scientific investigation, such as those highlighted in [2, 3, 77].

This chapter 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 noting 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 node, or by using one of the many free online services[1].

6.1.1 Contribution

This chapter 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.

In this chapter we study similar to previous chapter $\mathrm{M}/M^B/1$ queueing model. Recall that the transactions are confirmed in batches and each batch (block) includes as many transactions as minimum between number of non-zero fee transactions in Mempool and designed block capacity.

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 $\mathrm{M}/M^B/1$ 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.

[1]For instance, `http://www.blockchain.com`

We provide the transient solution of such a system based on the technique of generating functions. Theorem 8 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 chapter 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 chapter 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 his/her 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.

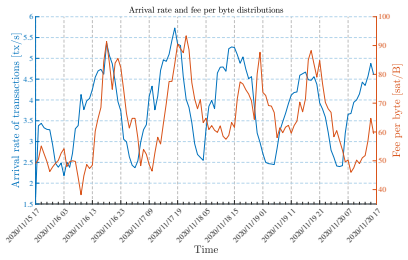
6.1.2 Structure of the Chapter

The chapter is structured as follows. In Section 6.2, we describe our research problem. Moreover, we show some data analysis of BTC blockchain to motivate the significance of our queueing model. The model is presented and solved in Section 6.3. In Section 6.4, we use the results of Section 6.3 to study the expected confirmation time in Bitcoin. What is more, we validate our model both with trace-driven and stochastic simulations. Section 6.5 concludes the chapter.

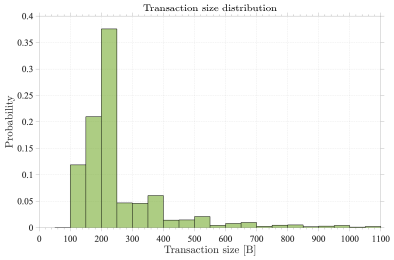
6.2 Problem statement and motivation

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.

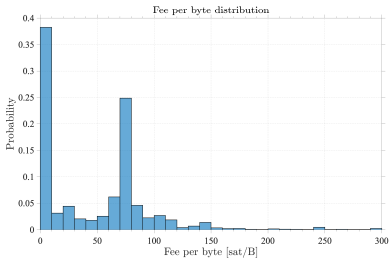
As a consequence, users need a method to tackle the trade-off between the cost of processing the transaction and its confirmation delay.



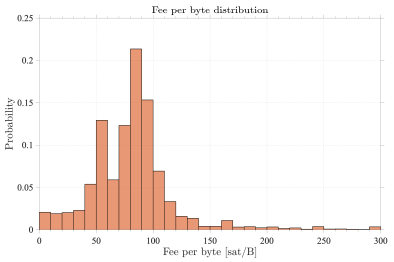
(a) Arrival rate and fee per byte as functions of the time.



(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.

Figure 6.1: Data retrieved from the Bitcoin node.

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* that is part of Bitcoin client functionality[2].

Figure 6.1a 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 hours. 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.

Figure 6.1b 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

[2]`https://bitcoincore.org/en/doc/0.16.0/rpc/util/estimatesmartfee`

sizes larger than 400 bytes.

Figures 6.1c and 6.1d 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 Figure 6.1d), users offer higher fees to solicit miners to select their transactions for inclusion in the next blocks. The majority of the transactions have fee-per-Byte values between 50 and 100 sat/B, with a peak around 85 sat/B.

This chapter 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 2 (The Mempool instances seen by users)**.** *Recall that every blockchain is a peer-to-peer network where information propagates thanks to a controlled flooding mechanism. A transaction is firstly accounted by a user and then it is broadcasted to the others. Technically speaking, there is the possibility that the Mempools seen by the users are not exactly the same. The Bitcoin Network Monitor*[3] *shows that within* 15 *seconds at least* 90% *of the nodes are ready to announce a newly generated transaction (so they have surely received it before) and the block propagation delay is within* 2 *seconds. Thus, these delays are reasonably small to support our assumption coherently with other works in this field [6, 31, 39, 41, 48]. Another aspect that we should bear 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.*

6.3 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:

[3] `https://www.dsn.kastel.kit.edu/bitcoin/`

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 6.3.1.

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 6.3.2.

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 6.3.3 where a numerical procedure for the computation of the expected confirmation time of the transactions is presented.

4. Finally, in Section 6.3.4, we extend our results to transactions offering an arbitrary fee.

6.3.1 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, ... and we have:

$$Pr\{t_{n+1} - t_n \leq x\} = 1 - e^{-\mu x}\,, \quad \forall n \geq 0\,, \tag{6.1}$$

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. Such behavior reflects the the system policy.

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 n-th 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 perpective, we are taking an arrival-before-service approach for the discretisation of the system's time (see, e.g., [45]), thus we have $\eta_0 = Y$.

The collection of random variables $\{\eta_n : n \geq 0\}$ is a DTMC since it trivially satisfies the Markov property [69]. Define the probability that the Mempool will be empty after n steps, given the initial state Y as:

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

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:

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

6.3.2 Solution of the model

The main result of this section is Theorem 8 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.

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

$$P_j^1 = \begin{cases} 1 - \alpha^{B-j+1} & \text{if } j \leq B \\ 0 & \text{if } j > B . \end{cases} \tag{6.2}$$

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

$$P_j^n = \sum_{k=\max(1,j-B)}^{\infty} P_k^{n-1} a_{k-j+B} = \sum_{k=\max(1,j-B)}^{\infty} P_k^{n-1} \beta\alpha^{k-j+B} . \tag{6.3}$$

For $n = 2$, we can easily derive P_j^2. In fact, using Equation (6.3), for $j \leq B$, we have:

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

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

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

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

In general, we rewrite Equation (6.3) for $n \geq 2$ as stated by the following lemma.

Lemma 3. *For $n \geq 2$, the system of Equations (6.3) can be rewritten as:*

$$\begin{cases} \alpha P_j^n = P_{j-1}^n & 2 \leq j \leq B+1 \\ \alpha P_j^n = P_{j-1}^n - \beta P_{j-B-1}^{n-1} & j > B+1 \,. \end{cases} \tag{6.4}$$

Proof. Let us consider $2 \leq j \leq B$. Using Equation (6.2), 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:

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

□

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 $|w| \leq 1$. We can multiply each equation for P_j^n of System (6.4) by w^n and summing up, we obtain for $2 \leq j \leq B+1$:

$$\alpha \left(P_j(w) - P_j^1 w\right) = P_{j-1}(w) - P_{j-1}^1 w \,. \tag{6.5}$$

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

$$\alpha \left(P_j(w) - P_j^1 w\right) = P_{j-1}(w) - P_{j-1}^1 w - \beta w P_{j-B-1}(w) \,. \tag{6.6}$$

Let us introduce the following generating function [45] for $z \in \mathbb{C}$ and $|z| < 1$:

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

and we sum Equations (6.5) and (6.6) multiplied by z^j for $j \geq 2$. 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:

$$\alpha P(z,w) - \alpha z P_1(w) - \alpha w \sum_{j=1}^{\infty} P_j^1 z^j + \alpha w z P_1^1 = zP(z,w) \\ - wz \sum_{j=1}^{\infty} P_j^1 z^j - \beta w z^{B+1} P(z,w) \,. \quad (6.7)$$

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 Equation (6.7) 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 $P(z,w)$:

$$P(z,w) = \frac{w(\alpha - z) h(z) + \alpha z P_1(w) - \alpha w z P_1^1}{\alpha - z + \beta w z^{B+1}} \,, \quad (6.8)$$

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

Lemma 4. *The denominator of the right-hand side of Equation (6.8) has only one zero ξ (that depends on w) in the open unitary disk if the stability condition $\alpha < B/(B+1)$ holds, $|w| \leq 1$ and $\alpha \neq j/(j+1)$ for $j = 1, \ldots, B$.*

Proof. 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 the disk perimeter. Thus, the equation has a unique root ξ that lies 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 [18, Thm 2.1].

Theorem 7 (Theorem 2.1 in [18]). *Let $a > b > 0$ be real numbers and $n > m > 0$ 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:

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

where n is the degree of f and $\overline{z}$ 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 [53, Ch. X]:

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

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 $P_1, \ldots, P_{B+1}$ [53, Thm 42,1].

Lemma 5. *Let $\Psi_{j,\alpha} \triangleq (j + 1)\alpha - j$ for $j = 1, 2, \ldots$ Then, we have:*

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

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

$$a_0^{(j)} = \alpha^{2^{j-2}} \Psi_{j,\alpha} \prod_{k=1}^{j-2} \Psi_{k,\alpha}^{2^{j-2-k}}\,, \tag{6.9}$$

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

where all the unspecified coefficients are set to 0.

Proof of Lemma 5. 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:

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

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:

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

as required. Finally, we have:

$$\begin{aligned}
a_{B-j}^{(j+1)} &= -a_{B+1-j}^{(j)} \overline{a}_1^{(j)} \\
&= -\left[\alpha^{2^{j-2}} (1-\alpha) \prod_{k=1}^{j-2} \Psi_{k,\alpha}^{2^{j-2-k}} e^{i\theta}\right] \left[-\alpha^{2^{j-2}} \Psi_{j-1,\alpha} \prod_{k=1}^{j-2} \Psi_{k,\alpha}^{2^{j-2-k}}\right] \\
&= \alpha^{2^{j-1}} (1-\alpha) \Psi_{j-1,\alpha} \prod_{k=1}^{j-2} \Psi_{k,\alpha}^{2^{j-1-k}} e^{i\theta} \\
&= \alpha^{2^{j-1}} (1-\alpha) \prod_{k=1}^{j-1} \Psi_{k,\alpha}^{2^{j-1-k}} e^{i\theta},
\end{aligned}$$

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

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

Lemma 6. *For $j = B$ and $j = B+1$ we have the following coefficients:*

$$a_0^{(B)} = \alpha^{2^{B-2}} \Psi_{B,\alpha} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-2-k}}, \tag{6.10}$$

$$a_1^{(B)} = \alpha^{2^{B-2}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-2-k}} \left((1-\alpha)e^{i\theta} - \Psi_{B-1,\alpha}\right), \tag{6.11}$$

$$a_0^{(B+1)} = 2\alpha^{2^{B-1}}(1-\alpha)\Psi_{B-1,\alpha}(\cos\theta - 1) \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}}. \tag{6.12}$$

Proof of Lemma 6. The derivation of Equations (6.10) and (6.12) follows the same lines of Lemma 5.

For Equation (6.11), we have:

$$\begin{aligned}
a_0^{(B+1)} = [a_0^{(B)}]^2 - [a_1^{(B)}\overline{a}_1^{(B)}] &= \alpha^{2^{B-1}} \Psi_{B,\alpha}^2 \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}} \\
&- \left(\alpha^{2^{B-2}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-2-k}} \left((1-\alpha)\cos\theta - \Psi_{B-1,\alpha} + i(1-\alpha)\sin\theta\right)\right) \\
&\times \left(\alpha^{2^{B-2}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-2-k}} \left((1-\alpha)\cos\theta - \Psi_{B-1,\alpha} - i(1-\alpha)\sin\theta\right)\right) \\
&= \alpha^{2^{B-1}} \Psi_{B,\alpha}^2 \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}} - \alpha^{2^{B-1}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}} \\
&\times \left((1-\alpha)^2\cos^2\theta + \Psi_{B-1,\alpha}^2 - 2\Psi_{B-1,\alpha}(1-\alpha)\cos\theta + (1-\alpha)^2\sin^2\theta\right).
\end{aligned}$$

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

$$\begin{aligned}
&\alpha^{2^{B-1}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}} \left(\Psi_{B,\alpha}^2 - (1-\alpha)^2 - \Psi_{B-1,\alpha}^2 + 2\Psi_{B-1,\alpha}\right. \\
&\qquad\qquad \left.\times\ (1-\alpha)\cos\theta\right) = \alpha^{2^{B-1}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}} \\
&\times \left(2(1-\alpha)^2 B - 1 + \alpha^2 - (1+\alpha^2-2\alpha) + 2\Psi_{B-1,\alpha}(1-\alpha)\cos\theta\right) \\
&= 2\alpha^{2^{B-1}} \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}} \left((1-\alpha)^2 B - 1 + \alpha + \Psi_{B-1,\alpha}(1-\alpha)\cos\theta\right) \\
&\qquad\qquad = 2\alpha^{2^{B-1}}(1-\alpha)\Psi_{B-1,\alpha}(\cos\theta - 1) \prod_{k=1}^{B-2} \Psi_{k,\alpha}^{2^{B-1-k}},
\end{aligned}$$

as required. □

To conclude our proof, we resort to Marden's theorem [53, 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 $P_1, \ldots, P_{B+1}$. 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:

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

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 Equation (6.9) 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 (6.12) 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 Equation (6.10). Also $a_0^{(B+1)}$ is negative and thus $P_B < 0$ and $P_{B+1} > 0$. This concludes the proof. □

It is worth noting that Lemma 4 does not follow from an immediate application of Rouché theorem as it would be in the domain $|w| < 1$ and $|z| \leq 1$. The lemma 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 $|w| \leq 1$ and $|z| < 1$, ξ must also be a zero of the numerator of Equation (6.8). 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:

$$f(\xi) \triangleq \frac{\alpha w \xi P_1^1 - w(\alpha - \xi)h(\xi)}{\alpha \xi}\,, \tag{6.13}$$

where $w = (\xi - \alpha)/(\beta \xi^{B+1})$. By Lagrange's theorem [78], we can rewrite $f(\xi)$ as:

$$f(\xi) = f(\alpha) + \sum_{t=1}^{\infty} \frac{(\beta w)^t}{t!} \left[\frac{\partial^{t-1}}{\partial x^{t-1}} \left(f'(x) x^{t(B+1)} \right) \right]_{x=\alpha}\,. \tag{6.14}$$

We can easily compute $f(\alpha)$ by substitution using Definition (6.13) 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:

$$\begin{aligned} f'(x)x^{t(B+1)} = &-\frac{\beta}{\alpha}\frac{x^{t(B+1)}}{(1-x)^2} + \frac{B}{\alpha}\frac{x^{(t-1)(B+1)+2}}{(1-x)^2} \\ &+ \frac{\beta - B(\alpha+1)}{\alpha}\frac{x^{(t-1)(B+1)+1}}{(1-x)^2} + B\frac{x^{(t-1)(B+1)}}{(1-x)^2}\,. \end{aligned} \tag{6.15}$$

Lemma 7. *Let:*

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

then, we have:

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

and, for $t \geq 2$:

$$\begin{aligned} g(t) = &\alpha^{B(t-1)} \left(B\alpha - \beta B^2 + \beta(1+B)^2 t \right) \frac{[(t-1)(B+1)]!}{[B(t-1)]!} \\ &- \alpha^{B(t-1)-1}\beta(\alpha - \beta B)\frac{[B(t-1)+t+1]!}{[B(t-1)]!(t+1)}\,{}_2F_1\left[\begin{matrix} 1 & -B(t-1) \\ & t+2 \end{matrix} ; -\frac{\beta}{\alpha} \right] \\ &- \alpha^{Bt-2}\beta B \frac{[t(B+1)]!}{(Bt)!(t+1)}\,{}_2F_1\left[\begin{matrix} 2 & 1-Bt \\ & t+2 \end{matrix} ; -\frac{\beta}{\alpha} \right], \end{aligned} \tag{6.16}$$

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

Therefore, we can write:

$$P_1(w) = (1-\alpha^B)w + \sum_{t=2}^{\infty} \frac{\beta^t w^t}{t!} g(t)\,. \tag{6.17}$$

By taking the derivative of Equation (6.17) 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., [45]). In general, the n-th 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:

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

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 (6.17) needs to be truncated. In Section 6.3.3, 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.

6.3.3 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. Equation (6.17) 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 [5, 6] 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 homogenoeus 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 8 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 (6.17).

Theorem 8. *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:*

$$\begin{cases} M_1^1 = P_1'(1) \\ M_1^{Y+1} = M_1^Y + \frac{T_{Y-1}}{\alpha^{Y-1}} \left(M_1^1 + \frac{\beta}{\alpha} \right) - \frac{T_Y}{\alpha^Y} M_1^1 \,, \end{cases} \tag{6.18}$$

where:

$$T_Y \triangleq \sum_{c=0}^{\lfloor \frac{Y}{B+1} \rfloor} (-1)^{c+1} \binom{Y - Bc}{c} \alpha^{Bc} \beta^c \,. \tag{6.19}$$

Proof. Let us consider $P(z, w)$ as defined by Equation (6.8). Simple algebraic computations show that:

$$M_1(z) \triangleq \left. \frac{\partial P(z,w)}{\partial w} \right|_{w=1} = \beta \frac{z^2}{(1-z)(z-\alpha-\beta z^{B+1})} - \alpha M_1^1 \frac{z}{z-\alpha-\beta z^{B+1}} \,, \tag{6.20}$$

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 $Y \geq 2$:

$$\frac{\partial^Y g_1(z)}{\partial z^Y} = \sum_{k_1=0}^{Y} \sum_{k_2=0}^{Y} \frac{(-1)^{k_1}}{(1-z)^{k+1}} z^{2-k_2} \frac{\partial^{Y-k_1-k_2}}{\partial z^{Y-k_1-k_2}} \frac{1}{z-\alpha-\beta z^{B+1}} \times (-1)^{k_1} k_1! (\delta_{k_2=0} + 2\delta_{k_2=1} + 2\delta_{k_2=2}) \binom{Y}{k_1, k_2, Y-k_1-k_2}, \quad (6.21)$$

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:

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

The following lemma allows us to compute the n-th order derivative that appears in Equation (6.21) evaluated for $z = 0$:

Lemma 8. *The following relation holds:*

$$\frac{\partial^k}{\partial z^k} \frac{1}{z-\alpha-\beta z^{B+1}}\bigg|_{z=0} = \frac{k!}{\alpha^{k+1}} \sum_{\ell=0}^{k} (-1)^{\ell+k+1} \binom{k+1}{\ell+1} \times \sum_{c=\lceil \frac{k-\ell}{B} \rceil}^{\lfloor \frac{k}{B+1} \rfloor} \binom{\ell}{k-(B+1)c, \ell-k+Bc, c} (-\alpha)^{Bc} (-\beta)^c. \quad (6.22)$$

Proof. We use the following identity for the derivative [29]:

$$\frac{\partial^k}{\partial z^k} \frac{1}{f(z)} = \sum_{\ell=0}^{k} (-1)^\ell \binom{k+1}{\ell+1} \frac{1}{[f(z)]^{\ell+1}} \frac{\partial^k}{z^k} [f(z)]^\ell, \quad (6.23)$$

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 n-th order derivative of this expression is:

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

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:

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

Thus, Equation (6.23), evaluated for $z = 0$, becomes:

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

This can be rewritten as in Equation (6.22) as required. □

The following Lemma is used to simplify Equation (6.22) and then derive a recursive expression for the computation of the first moment.

Lemma 9. *Equation (6.22) can be rewritten as:*

$$\left.\frac{\partial^k}{\partial z^k} \frac{1}{z - \alpha - \beta z^{B+1}}\right|_{z=0} = \frac{k!}{\alpha^{k+1}} \left(\sum_{c=0}^{\lfloor \frac{k}{B+1} \rfloor} (-1)^{c+1} \binom{k-Bc}{c} \alpha^{Bc} \beta^c \right) . \quad (6.24)$$

Proof. By Lemma 8, we have:

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

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,

$$\begin{aligned}
&\sum_{\ell=k-Bc}^{k} (-1)^{\ell+k+Bc} \binom{k+1}{\ell+1} \binom{\ell}{k-(B+1)c, \ell-k+Bc, c} \\
&\quad = \sum_{h=0}^{Bc} (-1)^{h+2k} \binom{k+1}{h+k-Bc+1} \binom{h+k-Bc}{k-(B+1)c, h, c} \\
&\quad = \sum_{h=0}^{Bc} (-1)^{h} \binom{k+1}{h+k-Bc+1} \binom{h+k-Bc}{k-(B+1)c, h, c} \\
&\quad = \binom{k-Bc}{c} \binom{k}{Bc} (k+1) \sum_{h=0}^{Bc} (-1)^{h} \frac{1}{(h+k-Bc+1)} \binom{Bc}{h}.
\end{aligned}$$

Now, by applying the following identity [29]:

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

we obtain:

$$\begin{aligned}
&\binom{k-Bc}{c} \binom{k}{Bc} (k+1) \sum_{h=0}^{Bc} (-1)^{h} \frac{1}{(h+k-Bc+1)} \binom{Bc}{h} \\
&\qquad = \binom{k-Bc}{c} \binom{k}{Bc} (k+1) \frac{(Bc)!(k-Bc)!}{(k+1)!} = \binom{k-Bc}{c}.
\end{aligned}$$

This concludes the proof of the Lemma. □

Let us consider again the computation of the Y-th 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 Equation (6.20):

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

In this case, we have:

$$\frac{\partial^Y g_2(z)}{\partial z^Y} = Y\frac{\partial^{Y-1}}{\partial z^{Y-1}}\frac{1}{z-\alpha-\beta z^{B+1}} + z\frac{\partial^Y}{\partial z^Y}\frac{1}{z-\alpha-\beta z^{B+1}}\,. \tag{6.25}$$

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

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

In conclusion, for $Y \geq 2$:

$$\begin{aligned}M_1^Y = \beta\sum_{k_1=0}^{Y-2}\frac{1}{\alpha^{k_1+1}}\sum_{c=0}^{\lfloor\frac{k_1}{B+1}\rfloor}(-1)^{c+1}\binom{k_1-Bc}{c}\alpha^{Bc}\beta^c \\ -\frac{M_1^1}{\alpha^{Y-1}}\sum_{c=0}^{\lfloor\frac{Y-1}{B+1}\rfloor}(-1)^{c+1}\binom{Y-1-Bc}{c}\alpha^{Bc}\beta^c\,.\end{aligned} \tag{6.26}$$

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 $Y \geq 1$:

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

as required. □

Remark 3. *In this remark, we discuss the numerical stability and complexity of the recursive algorithm that uses the equations of Theorem 8 to compute the average time to absorption.*

We first notice that the binomial coefficient in Equation (6.19) 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 Equation (6.18). 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*. It can be assumed as a general behavior. 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 8 let us define the 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 the transaction arrival. More precisely, the theorem provides the first moment that is the mean value while Equation (6.17) does so for the following moments instead. The latter one with approximation can be of use to determine variance and other crucial parameters.

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

6.3.4 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 $\overline{F}$ 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 $\rho = \lambda/(B\mu)$.

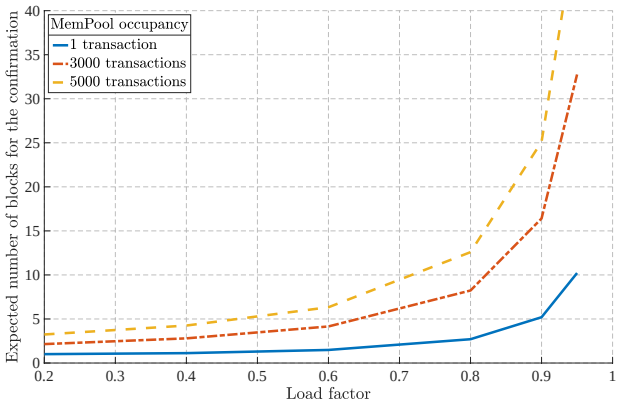


Figure 6.2: Expected number of blocks for the confirmation with different number of transactions in the MemPool as function of load factor.

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 Figures 6.1c and 6.1d 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.*

6.4 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.

6.4.1 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.

Figure 6.2 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 unfavourable working condition that moves the expected delay from 10 to 32 block consolidations in the cases of 1 or $3,000$ 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.

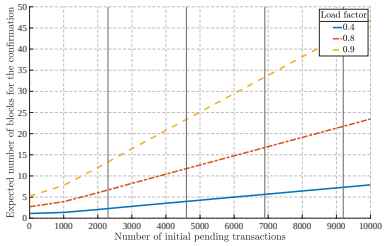
Another important observation is that expected confirmation time tends to grow to infinity as the load factor approaches 1. This is explained by the fact that when $\rho \geq 1$, the Markov chain underlying the model is not positive recurrent; therefore, starting from any initial state, there is a positive probability of having an infinite consolidation delay. In chapter 8 we study the model to determine the probability of confirmation in case of $\rho \geq 1$.

6.4.2 Impact of the initial Mempool state on the expected confirmation time

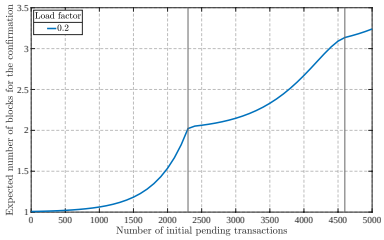
In the previous experiment, we have already discussed the importance of the Mempool state at the tagged transaction arrival. This observation is even more evident thanks to the plots of Figure 6.3a. It is interesting to observe that the function describing the expected confirmation delay given the initial number of pending transactions has abrupt changes in its growth for values corresponding to integer multiples of the maximum block capacity, which is in our case 2300 transactions. This is clearly shown by Figures 6.3b, 6.3c and 6.3d. 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.

6.4.3 Impact of the transaction fee on the expected confirmation time

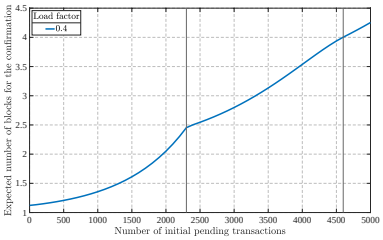
From a practical point of view, the main goal of the model is that of supporting the decision on which fee to offer to confirm a transaction with a given expected delay. Figures 6.4a and 6.4b show the expected confirmation delay as function of the offered fee for the tagged transaction, in moderate and heavy load at the time of initial transaction arrival to Mempool. For the plot of Figure 6.4a, we have used the fee distribution of Figure 6.1c, while for that of Figure 6.4b we have used the distribution of Figure 6.1d. 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



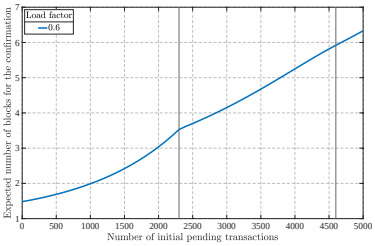
(a) Expected number of blocks for the confirmation as a function of number of initially pending transactions for different load factors.



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

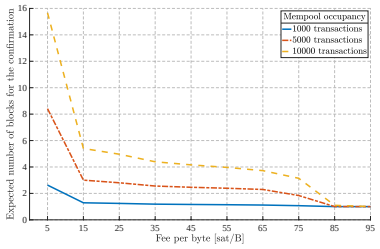


(c) Expected number of blocks for the confirmation as a function of number of initially pending transactions for $\rho = 0.4$.

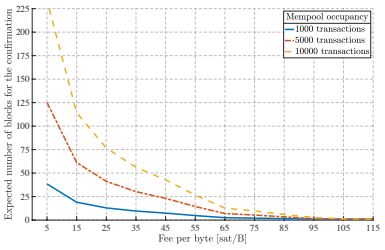


(d) Expected number of blocks for the confirmation as a function of number of initially pending transactions for $\rho = 0.6$.

Figure 6.3: 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.

Figure 6.4: Expected confirmation delay for different transaction fees per Byte.

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.

6.4.4 Validation of the model

In this subsection, we present some experiments aimed at showing the robustness of the model. We answer for the following three questions:

1. Is it accurate to estimate the expected confirmation delay by considering a blockchain with smaller block size but same load factor?

2. Does the replacement of a random block size with its mean have a strong impact on the expected consolidation delay?

3. 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.

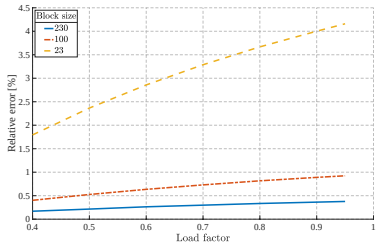
1. Re-scaling of the block size

Currently, the BTC blockchain block can host on average approximately $2,300$ transactions. Other blockchain networks, as that of Bitcoin cash, allow for bigger blocks and hence there is space for more transactions.

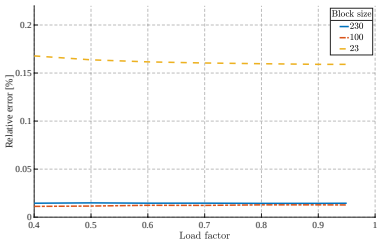
If we imagine to cluster the transactions with approximately the same fee per byte into macro-transactions, we simplify the model by considering smaller blocks as well as a statistically smaller population in the Mempool. However, we can maintain an identical load factor. Clearly, the modified system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

In Figure 6.5, 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 $B = 2,300$ 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.



(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.

Figure 6.5: Analysis of the relative error obtained by re-scaling.

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

Table 6.1: 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.

2. Robustness of the deterministic maximum block size assumption

In the model of Section 6.3, we assume that B is fixed. As shown in Figure 6.3b, 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 Figure 6.3b 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 6.1). 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 hours 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.

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 Figure 6.3b, respectively. Offered fees per byte are sampled from the distributions depicted by Figure 6.1d 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 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[4] are almost sure to be confirmed. We assume that the Mempool contains $6,000$ transactions. Figure 6.7a 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.

Figure 6.7b 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 unfavourable 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.

6.4.5 Validation of the prototype of the confirmation time estimator

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 hours. 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

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

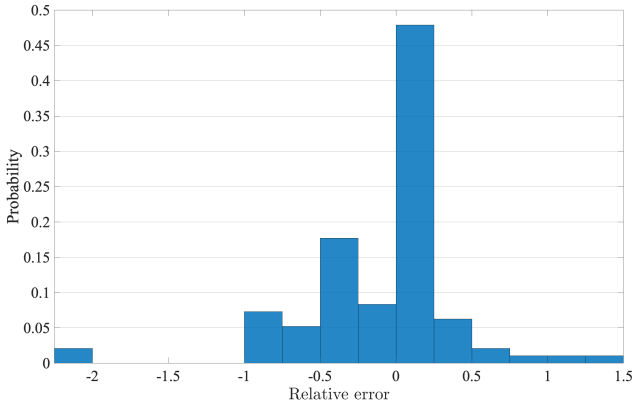


Figure 6.6: Empirical probability density function of relative error of actual and predicted confirmation delays.

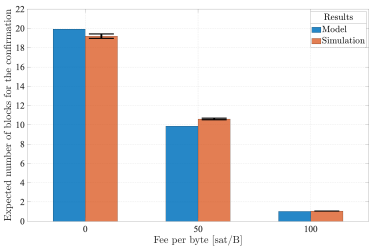
blocks for the confirmation. Thus, it is normal to observe a certain discrepancy between real and analytical data for a few transactions. This is shown in Figure 6.6. The histogram shows the distribution of the relative errors between model-based outcomes and actual measurements for sample set of transactions. 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.

6.5 Conclusion

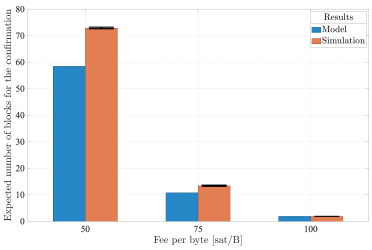
In this chapter, we propose a transient analysis of a $M/M^B/1$ queueing model that allows the definition of a new method for estimating the expected transaction confirmation time in blockchain based on PoW. The model uses three key parameters: the observed state of the Mempool, the current arrival intensity 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 shown that this has a strong impact on the estimations. In fact, the stationary analysis proposed in [6, 39, 41] 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



(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.

Figure 6.7: Comparison between model results and trace-driven simulation estimates.

the $M/M^B/1$ model, with large B and in heavy load can exhibit a quite slow convergence.

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 8 to predict the average number of blocks required to confirm a transaction given its fee.

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 bisection methods).

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, one viable improvement could be to predict the arrival rate during the consolidation delay. In fact, this is the subject to the following chapter.

Finally, it seems promising 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 crucial to derive the expression of the waiting time in the continuous model that may be non-trivial in general.

Chapter 7

Workload prediction methods

7.1 Introduction

In this chapter, we study the problem of predicting the traffic intensity in BTC blockchain with the aim of parameterising a simulation model that studies the expected confirmation time of transactions. After collecting data regarding the transaction arrival process at our BTC node, we use these traces to train two possible prediction models: one based on *Facebook Prophet* model [73], and the other is the well-known Autoregressive Integrate Moving Average (*ARIMA*) model. Both models provide confidence intervals in the prediction of the arrival process and allow us to consider pessimistic-, average- and optimistic-case scenarios. After comparing the two predictive models and choosing one that demonstrates better results in terms of the *absolute error*, we study by simulation the transaction confirmation time as a function of the offered fees and compare the results obtained with the real trace as input with those obtained by using the predicted trace as input.

The chapter is structured as follows. Section 7.2 gives a brief description of the applied prediction models. In Section 7.3, we examine the ARIMA and Prophet forecasts accuracy after certain hours from the transaction arrival and the accuracy of the predictions on the expected confirmation time using Monte Carlo simulations. Finally, Section 7.4 concludes the chapter and provides an insight for future work.

7.2 Background

This section provides some information about two prediction models, namely ARIMA and Prophet by Meta (ex-Facebook), that are used to forecast the throughput of the transactions.

7.2.1 Background on the ARIMA model

ARIMA(p,d,q) model [12] is one of the most widely used models for statistical forecasting a time series of observations X_t. The ARIMA equation is a linear (i.e., regression-type) equation in which the predictors consist of lags of the dependent variable and lags of the forecast errors. The general model can be written as

$$(1 - \phi_1 L - \cdots - \phi_p L^p)(1 - L)^d X_t = c + (1 + \theta_1 L + \cdots + \theta_q L^q)\varepsilon_t$$

where L is the lag-operator, i.e. $L^k a_t = a_{t-k}$ and ε_t is a white noise. The value p refers to the "AutoRegressive" component and represents the number of lagged observations included in the model. The "Integrated part" of the ARIMA model indicates that the data values have been replaced with the difference between their current and previous values, i.e. $(1 - L)x_t = x_t - x_{t-1}$. The value d is a number of times that the raw observations are differenced. In general, differencing refers to the transformation applied to non-stationary time series in order to make them stationary by attempting to remove the deterministic components

such as trends or periodicities. The value q, stands for the size of the "Moving Average" window for the forecast errors. Automatic identification of the orders p, d, q and statistical estimation of the parameters $\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q$ can be done easily (see [12]).

The data are collected every 10 minute and our time series exhibit seasonality with frequency of $144 = 24 \times 6$ which is exactly 24 hours in 10-minutes terms. In our experiment, using the Akaike Information Criterion, we identify a special instance of the ARIMA model, namely a multiplicative seasonal model [12]:

$$(1 - \phi_1 L - \phi_2 L^p)(1 - L)(1 - L^{144})X_t = (1 + \theta_1 L + \theta_2 L^2)\varepsilon_t.$$

7.2.2 Background on the Facebook Prophet model

The Prophet model [73] is a modular regression model with interpretable parameters that can be adjusted in order to optimize the prediction response.

The authors use a decomposable time series model with three key components, namely trend, seasonality, and holidays. The model may be represented as follows:

$$X_t = g_t + s_t + h_t + \varepsilon_t \,.$$

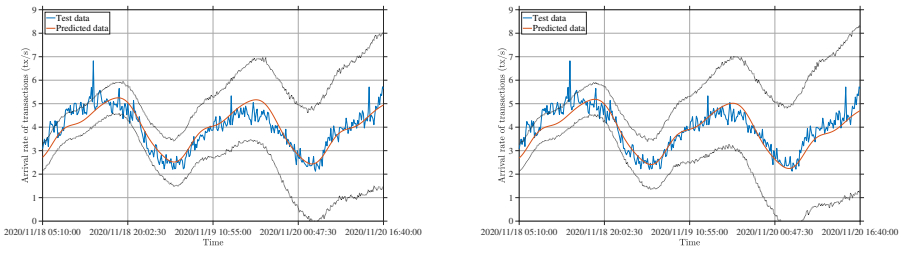
where g_t refers to the trend function that simulates non-periodic changes in the value of the time series, s_t describes periodic changes of the series, that is any seasonality effects, and h_t stands for the effects of holidays which occur on rather irregular pattern over one or more days. The error term ε_t is still white noise and represents any idiosyncratic changes of the model.

What is more, one of the features of g_t can be changepoint prior scale. The changepoints allow to incorporate trend changes in the growth models and stand for the points in time at which the trend is supposed to change its vector. It can be set manually otherwise it will be done automatically. This feature modulates the flexibility of the automatic changepoint selection. Larger values will allow many changepoints and small ones - few.

The authors frame the forecasting problem as a curve-fitting exercise, which differs from the models that account for the temporal dependence structure in the data. Although they miss some inferential benefits of using a generative model, e.g., the ARIMA model, their approach provides several practical advantages such as the fast fitting, ability to use irregular time data, flexible tuning of the trend, and seasonality behavior.

7.3 Evaluation of the accuracy in performance predictions

This section consists of two parts. First, we study the accuracy of the ARIMA and Prophet predictions on the time series of the transaction arrivals in the BTC blockchain. This allows us to obtain a punctual value of the prediction after τ hours from the last considered arrival of transaction and its confidence



(a) The comparison at changepoint prior scale of 0.06.

(b) The comparison at changepoint prior scale of 0.07.

Figure 7.1: Comparison of the actual arrival rate of transactions and the predicted response based on the Prophet model with different changepoint prior scale instances and confidence interval of 95%.

interval. Thus, for each epoch, we have a predicted expected value, a lower bound that represents the optimistic scenario and an upper bound leading to the pessimistic scenario.

The second contribution of the section is the estimation of the accuracy of the predictions on the expected confirmation time by means of Monte Carlo simulations of the confirmation process. The simulation uses as input three values of the confidence interval (lower, upper and central) to obtain an optimistic, pessimistic and expected estimation of the confirmation time.

It worth noting that, while the expected confirmation delay is monotonic increasing with respect to the arrival rate, the relation between waiting time and intensity of the arrival process is not linear and hence the intervals obtained in the confirmation delay predictions are not symmetric with respect to the prediction obtained using the expected arrival rate.

7.3.1 Comparison of time series prediction models

This section describes the accuracy of the estimates as well as their insights obtained by the aforementioned prediction models.

In order to collect the time series, we have installed a BTC mining node and logged the transactions announced at its Mempool. We have collected the data for five days and obtained our dataset that was coherent with the information available on specialised websites but with higher granularity. Additionally, we analysed the distribution of transaction fees of the Bitcoin clients in heavy load conditions.

In order to train the models, we divided our dataset in two parts with the same size: the first one has been used to train the models, while the second part has been used to assess the accuracy of the prediction.

For both the models, we use prediction intervals with a coverage of 95%.

Figures 7.1 and 7.2 show predictions of the transaction arrival intensity provided by the Prophet and ARIMA models, respectively. What is more, Fig-

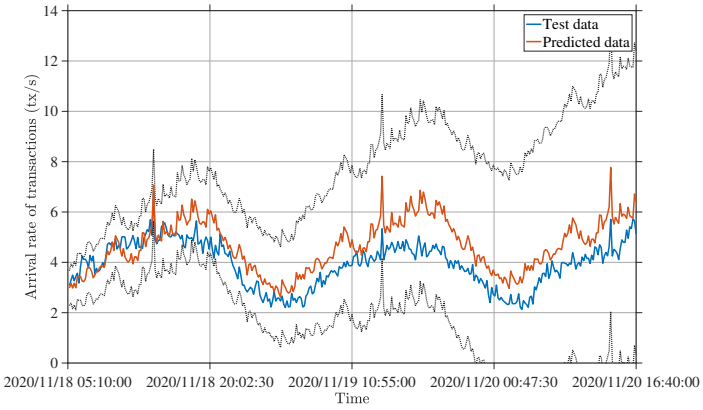


Figure 7.2: Comparison of the actual arrival rate of transactions and the predicted response based on the ARIMA model.

ures 7.1a and 7.1b illustrate the prediction deviation due to the choice of different changepoint prior scale values, namely, 0.06 and 0.07 accordingly. Thus, the outcome of the Prophet model at the parameter 0.07 gives the best prediction, according to our experiment. In our assessment, we will use the best results.

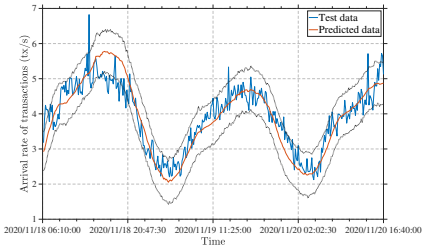
For both the plots, we used the first 2.5 days of data to train the model, and then we predicted the future arrivals. We show the test data of our dataset (blue line), the prediction of the model (red line) and the confidence intervals (grey lines). As expected, as the prediction time is moved far in the future, the confidence interval becomes wider. However, for practical applications, predictions are useful when performed within approximately 10 or 12 hours, otherwise it is very likely that the transaction is delay tolerant.

Even before formally testing the accuracy of the predictions with an error measure, we may notice that Prophet seems to give a better accuracy in this context.

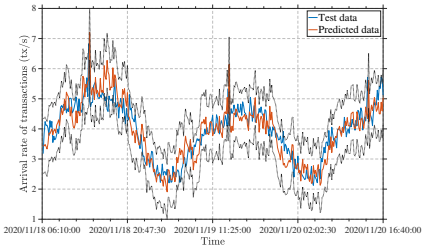
Now, we consider the predictions of the Prophet and ARIMA model at fixed time intervals. More precisely, given an interval τ, at each time t we use all the data up to t to train the model, and forecast the value of the time series at time $t + \tau$.

Figure 7.4 shows the comparison of the predictions obtained with the Prophet and ARIMA models for different values of τ. We can see that, although the ARIMA predictions tend to be more noisy, both the models show rather good predictions of the test data.

More precisely, in Table 7.1 we compute the absolute errors of the Prophet and ARIMA predictions. According to our experiments, the Prophet outperforms ARIMA, especially for short term predictions. Henceforth, in the following section we will carry out our experiments by using the Prophet model.

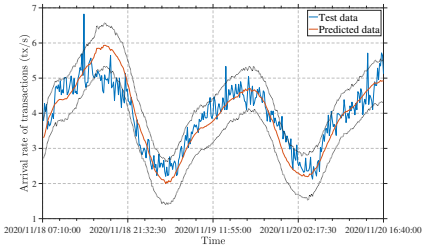


(a) Comparison of the actual arrival rate of transactions and the predicted response for $\tau = 1$ hour ahead based on the Prophet with changepoint prior scale of 0.07.

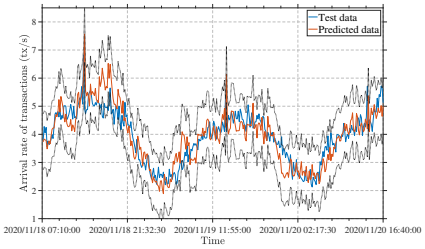


(b) Comparison of the actual arrival rate of transactions and the predicted response for $\tau = 1$ hour ahead based on the ARIMA model.

Figure 7.3: Comparison of the Prophet and ARIMA prediction models at prediction horizon $\tau = 1$ hour and confidence interval of 0.95.



(a) Comparison of the actual arrival rate of transactions and the predicted response for $\tau = 2$ hours ahead based on the Prophet with changepoint prior scale of 0.07.

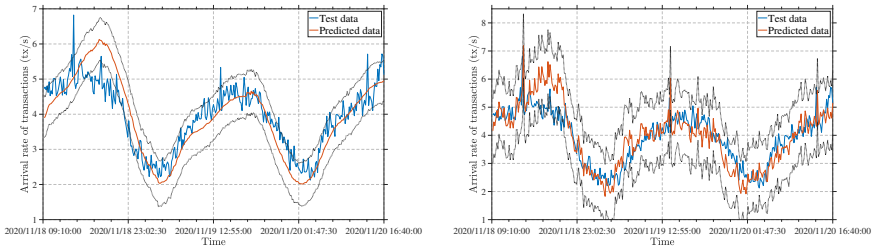


(b) Comparison of the actual arrival rate of transactions and the predicted response for $\tau = 2$ hours ahead based on the ARIMA model.

Figure 7.4: Comparison of the Prophet and ARIMA prediction models with prediction horizon $\tau = 2$ hours and confidence interval of 0.95.

Table 7.1: Mean Absolute Errors of Prophet and ARIMA models with different size of the prediction horizon.

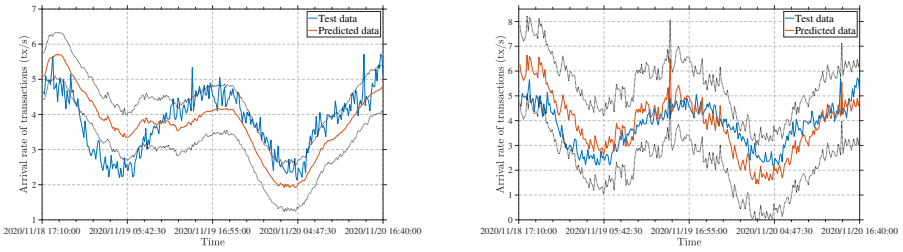
Prediction horizon τ in hours	Prophet Error	ARIMA Error
1	0.3120	0.3668
2	0.3366	0.3896
4	0.3966	0.4219
12	0.6333	0.6416



(a) Comparison of the actual arrival rate of transactions and the predicted response for $\tau = 4$ hours ahead based on the Prophet with changepoint prior scale of 0.07.

(b) Comparison of the actual arrival rate of transactions and the predicted response for $\tau = 4$ hours ahead based on sponse for $\tau = 4$ hours ahead based on the ARIMA model.

Figure 7.5: Comparison of the Prophet and ARIMA prediction models with prediction horizon $\tau = 4$ hours and confidence interval of 0.95.



(a) Comparison of the actual arrival rate of transactions and predicted response for $\tau = 12$ hours ahead based on the Prophet prediction approach by Facebook with changepoint prior scale of 0.07.

(b) Comparison of the actual arrival rate of transactions and predicted response for $\tau = 12$ hours ahead based on the ARIMA model.

Figure 7.6: Comparison of the Prophet and ARIMA prediction models with prediction horizon $\tau = 12$ hours and confidence interval of 0.95.

7.3.2 Simulations

In this section, we are interested in determining the accuracy of the estimation of the expected confirmation time using the Prophet prediction model to determine the arrival intensity of the transactions. The comparison is performed having the real data obtained from the running blockchain node and the data predicted by the model. Moreover, in order to estimate the confirmation time of the tagged transaction using the predicted arrival rate we resort to Monte Carlo simulations whose structure can be summarised as follows:

- We consider a fixed sequence of transaction arrivals. This can be trace-driven by our retrieved dataset or obtained by the model (optimistic-, average- or pessimistic-case scenarios).
- The generation of the blocks occurs at random time intervals, exponentially distributed with average 10 minutes. This follows from the memoryless characteristic of the mining process and from the invariant properties of the BTC blockchain.
- At a block generation instant, the most valuable transactions of the Mempool are confirmed and removed from the queue. We assume that the block contains $2,300$ transactions. Transaction fees are chosen probabilistically using the distribution of Figure 2.4d.
- Initially, the Mempool is populated with a fixed amount of transactions. These transactions offer a fee per Byte according to the distribution of Figure 2.4d. Notice that, although this is an approximation since the cheapest transactions tend to remain in the Mempool, the comparison remains fair since the initial Mempool population is the same for all the scenarios.

More precisely, we number the transactions from $-M$ to ∞, where M is the initial Mempool size, transaction 0 is the tagged transaction whose confirmation time is measured, and transaction denoted by $i > 0$ are those arriving after the tagged one.

Transaction t_i is denoted by a pair (τ_i, f_i), where τ_i is the arrival time and f_i the offered fee. For $i \leq 0$, $t_i = 0$. f_i is sampled from the distribution of Figure 2.4d independently of τ_i. τ_i, for $i > 0$ are obtained from the real traces or from the predictions of Prophet. Notice that, in practice, the fees may be dependent from the system state (Mempool size, intensity of the arrival process) but in this context we use the simplifying assumption of independence since we mainly focus on the accuracy of the predictive power of the Prophet model.

Let $\mathcal{T}$ be the set of transactions and $X_1, X_2, \ldots$ be the sequence of block consolidation times, and assume $X_0 = 0$. Then, $X_{i+1} - X_i$, $i \geq 0$, are i.i.d. exponential random variables with mean 10 minutes.

The state of the simulation model is described by a collection of transactions in the Mempool, denoted by $\mathcal{M}_i$, where the subscript i expresses that the state is associated with the instant immediately after the consolidation of block i.

The set of transactions arriving during the consolidation of the $(i+1)$-th block, but after the consolidation of the i-th, can be denoted by:

$$\mathcal{A}_i = \{t_i \in \mathcal{T} : \tau_i > X_i \wedge \tau_i \leq X_{i+1}\}.$$

Now, let $\mathcal{F}(\mathcal{M})$ the set of at most $2,300$ transactions with the highest fee present in $\mathcal{M}$.

Thus we have the following recursive relation:

- $\mathcal{M}_0 = \{t_i \in \mathcal{T} : \tau_i \leq 0\}$
- $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \mathcal{A}_i \setminus \mathcal{F}(\mathcal{M}_i \cup \mathcal{A}_i)$.

Hence, the confirmation time T_c for the tagged transaction is given by:

$$T_c = \min\{i : t_0 \notin M_i\}.$$

The Monte Carlo simulation experiment consists of $10,000$ samples of T_c for a fixed fee f_0. Then, the expected confirmation time is obtained by averaging the sample values. The experiments have been repeated 30 times and the estimates have been used to determine the confidence interval for the expected confirmation delay. To avoid confusion, we omit the confidence interval from the plot. For a confidence of 95% we have a maximum relative error of 7%.

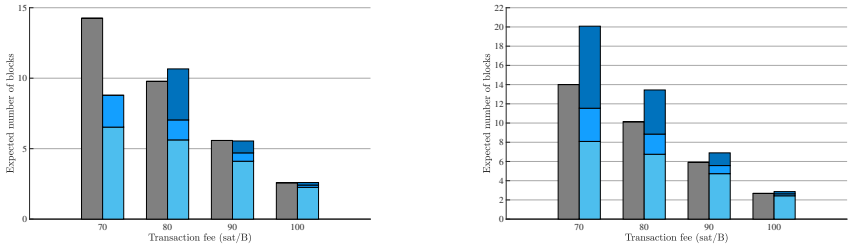
For each scenario that we consider, the tagged transaction offers a certain fee per Byte that controls the confirmation time: the higher the fee, the quicker the process.

According to the trace of arrival that we use, we obtain four estimates: the first using the real data, the second using the average prediction of Prophet model, the third and fourth using the traces given by the lower and upper bounds of the confidence intervals determined by the model. These two latter scenarios can be interpreted as pessimistic and optimistic cases in terms of confirmation delay.

Fig. 7.7a and 7.7b show the expected number of blocks required for the transaction confirmation for different offered transaction fees. The grey bars refer to the expected number of blocks obtained with use of the real data while the combined bars consisting of the light, normal and dark blue bars that are the optimistic-, average- and pessimistic- case scenarios, respectively, derived from the predicted data.

Notice that in the scenario of Figure 7.7a, the data were derived from the time series shown in Figure 7.1b, and the arrival time of the tagged transaction is 2020/11/18 05:10. Thus, the first half of the dataset was used to train the model. While in the second scenario (Figure 7.7b), the arrival time for the tagged transaction is 2020/11/19 08:30:00, and hence the training data include all the series up to that epoch, i.e., 70% of the dataset.

The inspection of Figure 7.7a shows that the pessimistic case scenario for 70 sat/B is absent: this happens because the transaction is dropped before its



(a) The simulation results at 50% of the training data.

(b) The simulation results at 70% of the training data.

Figure 7.7: Simulation results based on the actual data (grey bar) compared to the results of the Prophet predicted response (blue bars) with the optimistic, average and pessimistic cases and the initial Mempool occupancy of $10,000$ transactions and different amount of the training data.

confirmation (usually after 72 hours of residence in the Mempool). A second observation is that, especially in heavy-load (70 and 80 sat/B), the distance between the optimistic prediction and the average is smaller than that from the pessimistic and the average. This is due to the non-linearity of expected response time of a queueing system with respect to the arrival intensity. Finally, for this scenario, we notice that while the prediction obtained with the dataset is always within the optimistic and pessimistic cases, it seems to be closer to the latter. Indeed, Fig. 7.1b shows that predicted values for the first period of time are rather underestimated by the model. To confirm this explanation, we can look at the beginning of the next prediction interval (2020/11/19 08:30:00) when the prediction accuracy is higher. In this case, there is a good matching between the predicted average confirmation time and that obtained by using the real dataset (see Figure 7.7b).

7.4 Conclusion

In this chapter, we have applied two different time series forecast models, namely the Prophet by Facebook and ARIMA, in order to predict the arrival rate of the transactions at the Mempool of the Bitcoin network. According to our experiments, the Prophet model provides more accurate predictions in terms of the absolute errors.

Moreover, we have investigated if these predictions can be used to parameterise a model aimed at estimating the expected confirmation time of a transaction given its offered fee. We have shown two scenarios and in both cases we obtained valuable predictions that can be used to study the trade-off between the blockchain running costs and the quality of service.

Although our study has been carried out for the BTC blockchain, it can be extended to any similar system where transactions are chosen from the Mempool

according to an auction (e.g., Ethereum blockchain).

As future work it would be important to compare the approach proposed here with other forecasting models, e.g., based on machine learning.

Chapter 8

Reliability in blockchains: droppings

8.1 Introduction

Recall that blockchain transactions are added to blocks are selected from a queue of pending transactions according to an auction-based policy. Each block is formed with the transactions offering the highest fees according to the dropping policy. However, when the number of pending transactions exceeds a certain threshold the least valuable transactions are evicted and will never be included in the blockchain.

Notice that this eviction is different from that of invalid transactions due, e.g., to double spending. Henceforth, we will focus on dropping due to saturation of resources. In addition, the eviction of a transaction is not notified to the owners. This creates a serious problem of reliability for the applications based on blockchains. Since transactions offering higher fees have lower probability of being evicted, this poses an important trade-off to the applications: *What is the lowest fee that should be offered to ensure a confirmation probability higher than a given threshold?*

In this chapter, we provide a model to answer to this question. To the best of our knowledge, this is the first model that focuses on the reliability analysis of PoW blockchains, intended as a measure of the confidence that a user can have about the inclusion of his/her transactions in the ledger.

The model is parameterized with publicly known information about the state of the blockchains: the intensity of the transaction arrival process, the distribution of the fees offered by the transactions, the capacity of a block, the rate of block generation and the occupancy and offered fee of the transactions in the Mempool.

The model is solved with an efficient numerical algorithm that, compared to simulation based solutions, allows for the optimization of the trade-off between reliability and running costs of blockchain based applications with the lower computational effort.

The chapter is structured as follows. Section 8.2 introduces the analytical model. In Section 8.3 we provide the analysis of the transaction confirmation estimation for our model. Finally, Section 8.4 concludes the paper and proposes future research direction.

8.1.1 Problem statement and practical relevance

Given the auction governing the transaction confirmations, it is natural to study the trade-off between reliability, i.e., probability for a transaction of being eventually confirmed, and running costs in terms of offered fees. We take a user perspective, so that our model accounts for all publicly available information that can help make an effective decision: the instantaneous intensity of the transaction arrival process per class of fee per Byte, the distribution of the fees and the population and the Mempool. The main question is *If we send a transaction offering f as fee per Byte, what is the probability of being eventually confirmed?* Clearly, the model can be used also to solve the inverse problem, that is *What is*

the minimum fee per Byte that we should offer to have a confirmation probability higher than a certain threshold?

The problem has practical importance especially for those applications using PoW blockchains that do not have constrains on the confirmation time but need to be sure that, up to a certain probability, the transactions will be eventually included in the ledger.

For instance, this is the case of applications that use the blockchain to store monitored data collected by IoT systems. Without a proper estimation of the confirmation probability, the application could incur into unnecessary running costs. Moreover, the lack of notification of the transaction droppings would require the application to consult the Mempools to understand if their transactions are still present, and this is often very costly or unfeasible.

So, wherever it is applicable, the model can be used to estimate the confirmation probability of the transaction already in the Mempool and the cost of another transaction that replaces the former and whose confirmation probability is higher. In fact, given the auction of transactions, the new more expensive transaction would be confirmed before the older one and, if it spends the same cryptocurrency output, immediately invalidates it because of double spending.

8.2 A Gambler's ruin based model to estimate the dropping probability

In this section, we present a model for the estimation of the transaction dropping probability given its offered fee per Byte and the state of the Mempool. We will formulate the problem as the probability of absorption in a CTMC.

8.2.1 Modeling assumptions and notation

Let K be the maximum number of transactions that can be stored in the Mempool. We assume that transaction bids are i.i.d. random variables with CDF $F(x)$, where F is not necessary continuous but can be càdlàg (for this reason we consider ties), $\overline{F}(x) \triangleq 1-F(x)$. Transactions arrive according to a homogeneous and independent Poisson process with intensity λ and blocks are generated on average every μ^{-1} seconds with an independent exponentially distributed delay. The number of transactions in a block is at most B: if the Mempool contains less than B blocks, the block is generated with all available transactions. The model exploits the memoryless property of the mining process, i.e., if miners are working on a candidate block in which the less valuable transaction offers f_1 and a transaction with a bid higher than f_1 arrives, the latter immediately replaces the cheapest one in the candidate block that returns to the Mempool if some space is available, or is evicted otherwise.

Thanks to the independence and the exponential distributions of the delays, the stochastic process underlying the system is a CTMC. Figure 8.1 shows the underlying process for a toy example system with $B = 2$ and $K = 5$. First, let us consider the case of a transaction t offering strictly less than all other

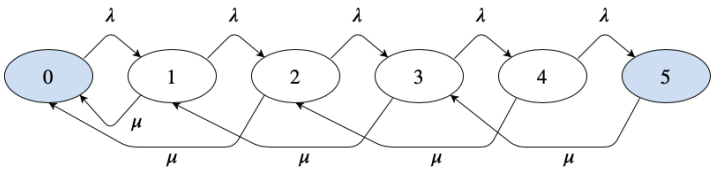


Figure 8.1: CTMC underlying the model for $B = 2$ and $K = 5$. The grey filled states are absorbing states.

transactions in the system. Suppose that at its arrival epoch τ the Mempool contains i transactions. Then, all the transactions present in the Mempool and those arriving after τ but before the confirmation or eviction of t will be confirmed before t. The problem consists in computing the probability that t is confirmed or evicted. From the perspective of t, this means that if the CTMC is absorbed in state 0, then t is confirmed, while if the process is absorbed in state K, then t is evicted.

Thus, our goal is that of computing the probability of absorption in state 0 or 5 given the initial state i seen at the arrival epoch of t. We will generalize this reasoning for the situations in which t makes a general bid in the following subsection.

We introduce $\alpha \triangleq \lambda/(\mu + \lambda)$, i.e., the probability that a transaction arrives before next block consolidation, i.e., $0 < \alpha < 1$.

8.2.2 Model analysis

Let p_i be the probability that a transaction offering the lowest possible fee per Byte is dropped given that at its arrival epoch the Mempool contains i transactions. We write the system of equations for the probability of absorption in state K from state i as follows [38]:

$$\begin{cases} p_0 = 0 \\ p_i = \alpha\, p_{i+1} & 1 \le i \le B \\ p_i = (1-\alpha)\, p_{i-B} + \alpha\, p_{i+1} & B < i < K \\ p_K = 1\,. \end{cases} \tag{8.1}$$

Let β denote the probability that B arrivals occur before a block is consolidated,

$$\beta = \alpha^B (1 - \alpha). \tag{8.2}$$

Theorem 9 gives the expression of the probability of eviction for a transaction arriving when the Mempool occupancy is i, as a function of α, β, K and B. Henceforth, binomial coefficients with negative upper index are assumed to have value 0.

Theorem 9. *For $0 \leq i \leq K$, the solution of the system of equations (8.1) is:*

$$p_i = \frac{T_i}{T_K}\,, \tag{8.3}$$

where

$$T_i = \frac{1}{\alpha^{i-1}} \sum_{l=0}^{m_i} \beta^l \binom{l(B+1)-i}{l} \tag{8.4}$$

and

$$m_i = \Big\lfloor \frac{i-1}{B+1} \Big\rfloor.$$

Proof. It is easy to see that $p_0 = 0$ and $p_k = 1$ are verified. Let us define

$$T \triangleq \sum_{l=0}^{\lfloor \frac{K-1}{B+1} \rfloor} \frac{(1-\alpha)^l}{\alpha^{K-1-lB}} \binom{l+lB-K}{l}.$$

We first consider $1 \leq i \leq B$. In this case, since

$$\Big\lfloor \frac{i-1}{B+1} \Big\rfloor = 0\,,$$

hence

$$p_i = \frac{\sum_{l=0}^{0} \frac{(1-\alpha)^l}{\alpha^{i-1-lB}} \binom{l+lB-i}{l}}{T} = \frac{\frac{1}{\alpha^{i-1}}}{T} = \frac{\alpha^{1-i}}{T} = \alpha\, p_{i+1}\,.$$

Let now consider $K(B+1) \leq i < (K+1)(B+1)$ for some $K \geq 1$. We prove that (8.3) is a solution of equation

$$p_i = (1-\alpha)\, p_{i-B} + \alpha\, p_{i+1}\,.$$

Indeed, we have:

$$\begin{aligned}
& (1-\alpha)\, T\, p_{i-B} + \alpha\, T\, p_{i+1} \\
= \;& (1-\alpha) \sum_{l=0}^{K-1} \frac{(1-\alpha)^l}{\alpha^{i-lB-B-1}} \binom{l+lB-i+B}{l} \\
& + \alpha \sum_{l=0}^{K} \frac{(1-\alpha)^l}{\alpha^{i-lB}} \binom{l+lB-i-1}{l} \\
= \;& \sum_{l=0}^{K-1} \frac{(1-\alpha)^{l+1}}{\alpha^{i-lB-B-1}} \binom{l+lB-i+B}{l} \\
& + \sum_{l=0}^{K} \frac{(1-\alpha)^l}{\alpha^{i-lB-1}} \binom{l+lB-i-1}{l}.
\end{aligned}$$

The latter expression can be further simplified as follows:

$$
\begin{aligned}
& \left(\sum_{l=0}^{K-1} \frac{(1-\alpha)^{l+1}}{\alpha^{i-lB-B-1}} \binom{l+lB-i+B}{l} \right. \\
& \quad \left. + \sum_{l=0}^{K-1} \frac{(1-\alpha)^{l+1}}{\alpha^{i-lB-B-1}} \binom{l+lB-i+B}{l+1} \right) + \frac{1}{\alpha^{i-1}} \\
= & \left(\sum_{l=0}^{K-1} \frac{(1-\alpha)^{l+1}}{\alpha^{i-lB-B-1}} (-1)^l \right. \\
& \quad \left. \times \left[\frac{(i-l-lB-B)_l}{l!} - \frac{(i-l-lB-B)_{l+1}}{(l+1)!} \right] \right) + \frac{1}{\alpha^{i-1}} \\
= & \left(\sum_{l=0}^{K-1} \frac{(1-\alpha)^{l+1}}{\alpha^{i-lB-B-1}} (-1)^l \frac{(i-l-lB-B)_l}{l!} \right. \\
& \quad \left. \times \left(1 - \frac{(i-lB-B)}{l+1} \right) \right) + \frac{1}{\alpha^{i-1}} \\
= & \left(\sum_{l=0}^{K-1} \frac{(1-\alpha)^{l+1}}{\alpha^{i-lB-B-1}} (-1)^l \frac{(i-l-lB-B)_l}{l!} \right. \\
& \quad \left. \times \frac{(l+lB+B-i+1)}{l+1} \right) + \frac{1}{\alpha^{i-1}} \\
= & \left(\sum_{l=0}^{K-2} \frac{(1-\alpha)^{l+1}}{\alpha^{i-lB-B-1}} (-1)^l \frac{(i-l-lB-B)_l}{l!} \right. \\
& \quad \left. \times \frac{(l+lB+B-i+1)}{l+1} \right) + \frac{1}{\alpha^{i-1}} \\
= & \left(\sum_{l=1}^{K-1} \frac{(1-\alpha)^{l}}{\alpha^{i-lB-1}} (-1)^l \frac{(i-l-lB)_l}{l!} \right) + \frac{1}{\alpha^{i-1}} \\
= & \sum_{l=0}^{K-1} \frac{(1-\alpha)^{l}}{\alpha^{i-lB-1}} \binom{l+lB-i}{l} .
\end{aligned}
$$

This proves the statement since, by substituting, we obtain:

$$
(1-\alpha)\, p_{i-B} + \alpha\, p_{i+1} \;=\; \frac{\sum_{l=0}^{k-1} \frac{(1-\alpha)^{l}}{\alpha^{i-lB-1}} \binom{l+lB-i}{l}}{T} \;=\; p_i \,.
$$

□

Notice that, if we define $T \triangleq T_K$, we can rewrite System (8.1) as:

$$\begin{cases} p_0 = 0 \\ p_i = \frac{1}{T}\alpha^{1-i} & 1 \leq i \leq B \\ p_i = (1-\alpha)p_{i-B} + \alpha p_{i+1} & B < i < K \\ p_K = 1\,. \end{cases}$$

In practice, Theorem 9 suffers a problem of numerical stability because of the presence of the binomial coefficients that may reach high values both at the numerators and denominator. While truncation and summation cancellation could be partly covered by applying Fox&Glynn approach [23], in this chapter we propose another computationally efficient method based on theory of difference equations.

We call cases $0 < i \leq B$ *initial conditions*, and $B < i < K$ the *general difference equation*.

Given the general difference equation, we can derive the characteristic polynomial [27] as follows:

$$P(x) = \alpha x^{B+1} - x^B + (1-\alpha)\,.$$

From the roots of $P(x)$, we will derive important properties of our system, as well as an alternative way to compute p_i that does not require the evaluation of large binomial coefficients. To this aim, we need the following lemma.

Lemma 10. *If $\alpha \neq B/(B+1)$, the characteristic polynomial $P(x)$ has distinct roots.*

Proof. A root r of $P(x)$ has multiplicity higher than one if and only if $P(r) = 0$ and $P'(r) = 0$. We have:

$$P'(x) = \alpha(B+1)x^B - Bx^{B-1}\,.$$

$P'(x)$ has $B-1$ roots in 0 and another one in $B/((B+1)\alpha)$. Clearly, 0 is not a root of $P(x)$ and

$$P(\frac{B}{(B+1)\alpha}) = -\left(\frac{B}{(B+1)\alpha}\right)^B \left(\frac{1}{B+1}\right) + (1-\alpha)\,.$$

We seek the relations between α and B that makes this quantity equal to 0. This corresponds to find the real roots of $Q(\alpha)$ in the interval $(0,1)$ with:

$$Q(\alpha) = \alpha^{B+1} - \alpha^B + \left(\frac{B}{B+1}\right)^B \left(\frac{1}{B+1}\right)\,.$$

This polynomial can be factorized as:

$$\begin{aligned} Q(\alpha) &= \left(\alpha - \frac{B}{B+1}\right)^2 \left(\alpha^{B-1} + \sum_{j=1}^{B-1} \frac{B^{j-1}(B-j)}{(B+1)^j} \alpha^{B-j-1}\right) \qquad (8.5) \\ &= \left(\alpha - \frac{B}{B+1}\right)^2 \left(\frac{(B+1)^2(\alpha^{B+1} - \alpha^B) + (B+1)\left(\frac{B}{B+1}\right)^B}{(\alpha(B+1) - B)^2}\right). \end{aligned}$$

Notice that the root $B/(B+1)$ has multiplicity 2 but is excluded by the hypothesis of the theorem, and the second factor, as expressed in Equation (8.5), is a sum of terms whose coefficients are all strictly positive. By Descartes' rule of signs, the second factor does not admit any positive real root.

In conclusion, for $\alpha \neq B/(B+1)$, there cannot be any root of $P'(x)$ that is also a root of $P(x)$. In the following, we will notice that $\alpha = B/(B+1)$ is a critical value for the stability of the system when $K \to \infty$. □

Henceforth, we assume $\alpha \neq B/(B+1)$. We may study the solution also for this special case for which Equation (8.6) does not hold since $P(x)$ has multiple roots in 1, but we omit it for the sake of brevity.

Hence, $P(x)$ admits $B+1$ distinct real or complex roots, namely $\{x_1, \ldots, x_{B+1}\}$. The complex roots come in pairs of conjugate numbers, and one trivial root is 1. Without loss of generality, let us assume $x_1 = 1$.

According to theory of difference equations (see, e.g., [27]), since all roots of $P(x)$ are different by Lemma 10, the solutions can be written as:

$$p_i = \sum_{j=1}^{B+1} C_j^* x_j^i \,, \qquad (8.6)$$

where $C_k^* \in \mathbb{C}$ are coefficients to be determined thanks to the B initial conditions and the case $i = 0$. Thus, we need to solve the system:

$$\begin{cases} C_1^* + C_2^* + \ldots + C_{B+1}^* = 0 & i = 0 \\ C_1^* x_1^i + C_2^* x_2^i + \ldots + C_{B+1}^* x_{B+1}^i = \frac{1}{T}\alpha^{1-i} & 1 \leq i \leq B \,. \end{cases}$$

Define $C_i \triangleq C_i^* T$, then we can compute p_is following an algorithm below:

(1) Compute the roots $\{x_1, \ldots, x_{B+1}\}$ of $P(x)$.

(2) Solve the following system of linear equations in $\mathbb{C}$:

$$\begin{cases} C_1 + C_2 + \ldots + C_{B+1} = 0 \\ C_1 x_1^i + C_2 x_2^i + \ldots + C_{B+1} x_{B+1}^i = \alpha^{1-i} & 1 \leq i \leq B \,. \end{cases} \qquad (8.7)$$

(3) To avoid the computation of T with Equation (8.4), we can use the observation $p_K = 1$ to write:

$$C_1^* x_1^K + C_2^* x_2^K + \ldots + C_{B+1}^* x_{B+1}^K = 1 \,,$$

and hence, multiplying both hand sides by T:

$$T = C_1 x_1^K + C_2 x_2^K + \ldots + C_{B+1} x_{B+1}^K \,. \tag{8.8}$$

(4) Compute all p_i as:

$$p_i = \frac{\sum_{j=1}^{B+1} C_j x_j^i}{T} = \frac{\sum_{j=1}^{B+1} C_j x_j^i}{\sum_{j=1}^{B+1} C_j x_j^K} \,. \tag{8.9}$$

8.2.3 The case of infinite Mempool

In this section, we study the case $K \to \infty$ as in [40]. A misconception may suggest that if $K \to \infty$, then there is no transaction dropping. However, from a theoretical point of view, if $\lambda > \mu B$ constantly, this implies that the intensity of the arrival process is higher than the capacity of service. Thus, transactions tend to form a backlog that grows with time and some of them may never be confirmed. The assumption $K \to \infty$ has clearly an important theoretical importance but it also represents an optimistic model of real systems. In fact, if for a blockchain the Mempool size K has a dropping probability close to the case studied in this section, then it is useless to increase the value of K with the aim of reducing the dropping probability.

First, notice that the stability condition $\lambda < B\mu$ is equivalent to $\alpha < B/(B+1)$. The following Theorem states what happens to the roots of $P(x)$ when this condition is (not) satisfied.

Theorem 10. *The number of roots φ strictly inside the unit disk of $P(x)$ are:*

- *if $\alpha \leq B/(B+1)$, then $\varphi = B-1$*
- *if $\alpha > B/(B+1)$, then $\varphi = B$.*

Proof. To prove this result, we resort to [18][Thm 2.1] stating that the trinomial $bx^n - ax^m + a - b$ has a number of zeros strictly inside the unit disk equal to $m - \gcd(m,n)$ if $a/b \geq n/m$, and m if $a/b < n/m$. The result immediately follows by the observation that, in $P(x)$, $b = \alpha$, $a = 1$, $n = B+1$ and $m = B$. □

In our model, we have to consider two cases.

Stable system: $\alpha < B/(B+1)$ The model with infinite buffer has an underlying CTMC that will eventually be absorbed in state 0 from every state i with probability 1 since the intensity of the workload is lower than the maximum service capacity. Formally, $P(x)$ has $B-1$ roots strictly inside the unit disk, one is $x_1 = 1$ and let us call the remaining one x_{B+1}. This root must be real, because if it were complex, also its conjugated would be on the perimeter of or outside the unit disk. Moreover, we can also observe that it must lay strictly outside the unit disk because -1 is not a root of $P(x)$ and 1 cannot have multiplicity 2 by Lemma 10. Therefore, $T \to \infty$ because all roots strictly inside the unit

disk vanish for $K \to \infty$ and $x_{B+1}^K \to \infty$. Since for all finite i, the numerator of Equation (8.9) is finite, then we conclude that $p_i \to 0$. This means that the probability of not being absorbed in state 0 is 0, as the intuition suggested. Thus, we can write:

$$K \to \infty \wedge \alpha < \frac{B}{B+1}\,, \qquad p_i = 0\,.$$

Unstable system: $\alpha > B/(B+1)$ This is the most interesting case. In fact, while the workload intensity is higher that the maximum service capacity, if i is sufficiently close to 0 we may still have a high probability of being absorbed in state 0. Formally, all roots of $P(x)$ lay strictly inside the unit disk with the exception of $x_1 = 1$. This implies that all the terms of $C_i x_i^K$ vanish for $K \to \infty$ with the exception of x_1, i.e., $T = C_1$. Therefore, we have:

$$K \to \infty \wedge \alpha > \frac{B}{B+1}\,, \qquad p_i = \frac{1}{C_1}\sum_{j=1}^{B+1} C_j x_j^i = 1 + \sum_{j=2}^{B+1} \frac{C_j}{C_1} x_j^i\,.$$

8.2.4 A toy example

In order to support the intuition behind the results presented so far, we introduce a toy example. Let us consider a blockchain in which blocks consist at most of 3 transactions ($B = 3$), the intensity of the arrival process is $\lambda = 1.4$ tx/s and blocks are generated with rate $\mu = 0.6$ blocks/s. The blockchain is able to process $\mu B = 1.8$ tx/s, and $\alpha = 1.4/(1.4 + 0.6) = 0.7$ and $B/(B+1) = 3/4 = 0.75$. Therefore, if the Mempool has infinite capacity $K \to \infty$, we are in the case of a stable system, i.e., the probability of dropping is 0 regardless to the state seen by a transaction at its arrival. If the Mempool capacity is, e.g., $K = 50$ we must first find the roots of the characteristic polynomial $P(x)$ that turn out to be:

$$x_1 = 1\,,\ x_2 \simeq -0.354 - 0.501j\,,\ x_3 \simeq -0.354 + 0.501j\,,\ x_4 \simeq 1.137\,.$$

Notice that, beside $x_1 = 1$ that is common to all possible $P(x)$, we have only one root outside the unit disk, x_4, that is real and positive and two complex conjugate roots inside the unit disk. The next step consists in finding the coefficients by solving the linear system of Equation (8.7). We obtain:

$$\begin{array}{ll} C_1 \simeq -3.500\,, & C_2 \simeq -0.1561 - 0.054889j\,, \\ C_3 \simeq -0.156 + 0.05489j\,, & C_4 \simeq 3.812\,. \end{array}$$

Finally, we compute $T \simeq 2337.29155$ with Equation (8.8). The probability of dropping given the initial number of transactions found in the Mempool are shown in Figure 8.2.

Let us assume now $\lambda = 2$, and hence $\alpha = 2/(2 + 0.6) \simeq 0.769$ *tx/s*, i.e., greater that $B/(B+1)$. In this case, the system with infinite Mempool is unstable. In fact, the roots of the polynomial all lay strictly inside the unit

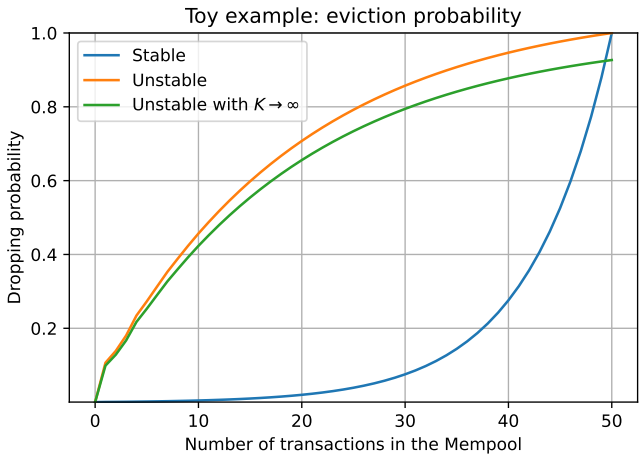


Figure 8.2: Comparison of the dropping probabilities for the three cases studied for the toy example.

circle, with the exception of $x_1 = 1$. Finally, if $K \to \infty$ we have $T = C_1$ and the dropping probability can be expressed in closed form as a function of the three roots of $(1-\alpha)x^{-B} + \alpha x = 1$ that lie inside the unit circle, namely $\tilde{C}_1 \simeq 0.950$, $\tilde{C}_2 \simeq -0.325 + 0.459j$ and $\tilde{C}_3 \simeq -0.325 - 0.459j$. Indeed, it follows from [40] that $p_i = 1 - \text{Real}\left(\sum_{l=1}^{B} \kappa_l \cdot \tilde{C}_l^{i+B-1}\right)$ where $\kappa_l = \prod_{i \neq l}(1 - \tilde{C}_i)/(\tilde{C}_j - \tilde{C}_i)$, i.e., $\kappa_1 \simeq 1.07$, $\kappa_2 \simeq -0.035 + 0.044j$ and $\kappa_3 \simeq -0.035 - 0.044j$. Figure 8.2 shows that the case $K \to \infty$ is a lower bound for the dropping probability for unstable systems. The bound becomes tighter for larger values of K and the result should be used to assess the reliability of the system given a Mempool size with respect to the ideal case.

8.2.5 Computational aspects

The heaviest computational effort for the model solution is the computation of the $B + 1$ roots of $P(x)$. In our implementation, we used the state of the art solution for this problem, i.e., the Aberth's method combined with multiprecision [11] in its implementation MPSolve[1].

Since all roots of $P(x)$ are distinct, the algorithm converges cubically [10] and its parallel version can handle sparse polynomials of degree up to one million, far above our needs.

Finally, the solution of linear system (8.7) has an asymptotic complexity of $\mathcal{O}(B^3)$.

[1] `https://github.com/robol/MPSolve`

8.2.6 The model for transactions offering a general fee

So far, we have reasoned on transactions offering the lowest possible fee, i.e., 0. The model can be easily extended to account for arbitrary fees thanks to the observation that any transaction is insensitive to all transactions offering a fee per Byte strictly lower that its own. Let X be the non-negative random variable modeling the fee per Byte offered by a transaction. Transaction T arrives at time t_0 at the blockchain offering f as fee per Byte. The transactions in the Mempool that are confirmed before T may be possibly dropped are those whose fee per Byte is strictly higher f. Transactions offering exactly f may be evicted before T because they are older. The transactions arriving after t_0 compete with T if and only fee their offered fee per Byte is higher or equal than f, i.e., the perceived arrival process, from the point of view of T, has intensity $\lambda Pr\{X \geq f\}$.

Summing up, a transaction offering f fee per Byte can evaluate its probability of being dropped as follows:

(1) Count the number of transactions i_f offering a fee per Byte strictly higher than f inside the Mempool.[2]

(2) Compute the intensity of the perceived arrival process $\lambda_f = \lambda Pr\{X \geq f\}$.[2]

(3) Use the algorithm presented in Section 8.2.2 using λ_f as arrival rate to obtain p_{i_f} that represents the probability of dropping for the transaction.

Recall that increasing the value of f has two positive effects: the decrease of the number of transactions in the Mempool seen by the new transaction as well as the decrease of persisting number of arriving transactions. Thus, such impact helps with the reduction of the dropping probability.

8.3 Experiments

In this section, we study the accuracy of the model with respect to the prediction of confirmation for transactions in Bitcoin.

8.3.1 Methodology

The model that we propose can be seen as a probabilistic binary classifier [33] that takes the state of the blockchain and the fee per Byte offered by a transaction t and returns the probability for t to be confirmed (Class 0) or dropped (Class 1). The classifier cannot be deterministic because of the intrinsic randomness of the blockchain system variables: the arrival process, the fees offered by the arriving transactions and the random times of block consolidations. We describe the methodology of validations in three steps: (i) analysis of the data set, (ii) parameterization of the model and (iii) performance analysis of the probabilistic classifier.

[2] All information required is publicly available through web services such as `www.blockchain.com`.

Analysis of the dataset. We use a dataset containing the Mempool occupancy in vMB and transaction counts for the last 5 years for Bitcoin. We consider two systems: one with infinite Mempool size that never drops transactions, and the standard one of Bitcoin Core. Transactions are clustered in 40 classes based on the fee per Byte offered. Class 1 is that with the lowest priority (offering between 0 and 1 sat/B) and Class 40 contains the transactions offering more than $2,000$ sat/B. The sampling time is of 1 minute.

Let $F_c(t)$ and $M_c(t)$ the number of transactions at minute t belonging to class c in the infinite and real Mempools, respectively. The $D_c(t) = F_c(t) - M_c(t)$ is always non-negative and denotes the number of transactions present in the fictitious Mempool that have been dropped in the real one. At minute t the number of dropping is then $d_c(t) = (D_c(t) - D_c(t-1))^+$, where x^+ is x if $x > 0$ or 0, otherwise. $D_c(t) - D_c(t-1)$ can be negative if at minute t we observe a block confirmation in which the miner with the infinite Mempool found some space in the block that could have potentially hosted some transactions of class c that had been previously dropped.

From these data, we infer a transaction dropping or confirmation event as follows. Consider a transaction t arriving at time t_0 and finding a backlog of N transactions of the same class in front of it, i.e., $N := M_c(t_0)$. All transactions with lower class (offering lower fees) are irrelevant to determine the behavior of t and are ignored. At each minute $t > t_0$, N is decreased by the number of class c dropped transactions $d_c(t)$ (since the oldest are chosen), or by the number of transactions of class c that entered a block. Assume that at t_1 we have $N < 0$. Then, transaction may have been dropped if at t_1 we do not have a block consolidation, and it is confirmed otherwise.

In this way, from the dataset, we compute, for each arrival epoch, if a transaction offering a certain fee per Byte has been confirmed or not.

Parameterization of the model. From the dataset, we obtain the other statistics of interest to configure our model in a trivial way. So we consider the size of the block B, the maximum capacity of the Mempool in number of transactions K, the state of the Mempool i_f at the transaction arrival time and the arrival rate λ_f of transactions offering more than f as fee per Byte (see Section 8.2.6). While the former two parameters are stable for long periods, the latter two change for each considered transaction. For all our experiments, we have considered $B = 2,100$ transactions and $K = 180,000$.

Performance analysis of the probabilistic classifier. In order to validate our model, we resort to the computation of Brier Score (see, e.g. [34]). Given a set of R observations obtained from the dataset $o_1, o_2, \ldots, O_R$, where $o_i = 0$ if the transaction is confirmed and $o_i = 1$, otherwise and the corresponding probability of dropping estimated by the model $q_1, q_2, \ldots, q_R$, Brier score can be computed as:

$$BS = \frac{1}{N} \sum_{i=1}^{R} (q_i - o_i)^2 \,,$$

Table 8.1: Brier scores for heavy and moderate loads.

	Heavy load	Moderate load
Transaction class	$[1, 12] sat/B$	$[1, 5] sat/B$
Fraction confirmed	0.39	0.64
Fraction dropped	0.61	0.36
BS	0.134	0.161
BS_{Ref_1}	0.465	0.431
BS_{Ref_2}	0.242	0.232
BSS	**0.447**	**0.306**

that can be interpreted as mean square error of the forecast. Then, we consider two simple probabilistic predictors as reference models. Ref_1 is a simple random predictor without any knowledge of the system representing a predictor with no skill. Ref_2 is an ideal predictor that knows a priori the fraction of transactions of Class c that will be dropped and assigns this value as probability of dropping to all transactions of Class c. Ref_2 is not implementable in practice since it uses information available a posteriori, but can be approximated by assuming that, for sufficiently long periods, in case of similar workloads, the fraction of transactions offering a certain fee that are dropped does not vary much. Therefore, the dropping probability used by Ref_2 could be inferred by historical data. As expected, Ref_2 outperforms Ref_1 and will be the term of comparison for our model. Therefore, the Brier Skill Score (BSS) is defined as:

$$BSS = 1 - \frac{BS}{BS_{Ref_2}} .$$

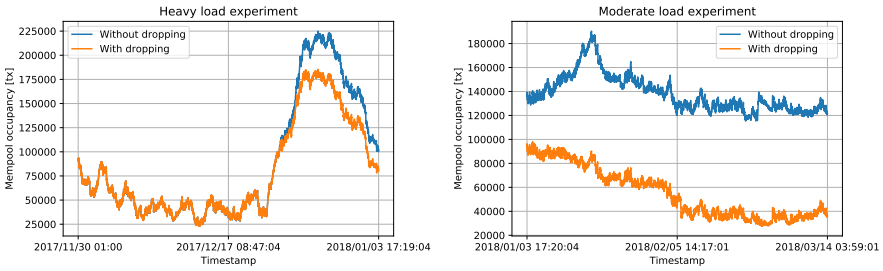
Values of BSS between 0 and 1 denote a better performance of our model with respect to Ref_2.

We consider two scenarios for our test: *heavy load*, where the arrival intensity is higher than the maximum service capacity and *moderate load*, where the stability is satisfied but we are close to the saturation point.

8.3.2 Heavy load conditions

During heavy load periods, we observe many droppings of the cheapest transactions. If we consider a class with very low fee, the experiment would show 100% of dropping probability, with a perfect accuracy of our model.

To make the scenario more challenging for the model, we consider a class of transactions that is not very cheap. A condition of heavy load occurred between 2017/11/30 and 2018/01/03. Figure 8.3a shows the Bitcoin Mempool occupation during these days. We notice that the populations at the infinite and finite Mempools are basically overlapped at the beginning of the observation period, but then a sudden increase in the traffic intensity brings to a high number of droppings.



(a) Heavy load. (b) Moderate load.

Figure 8.3: Mempool occupancy in a system with and without dropping measured in number of transactions.

We study the class of transactions offering between 1 and 12 sat/B. From the dataset, we collect 100 samples uniformly distributed in the considered time interval according to the methodology described in Section 8.3.1.

Table 8.1 shows the results of this experiment. In heavy load, around 60% of the transactions in the consider time interval are dropped despite their offered fee. We may notice that the model that we propose outperforms both the classifier with no skill Ref_1, and Ref_2 based on the assumption of the knowledge of the probability of dropping for the data set.

8.3.3 Moderate load conditions

To study the moderate load condition, we consider a cheaper class of transactions, i.e., the one including transactions offering fees between 1 and 5 sat/B. Figure 8.3b shows the traces of the Mempool occupation during the time interval between 2018/01/03 and 2018/03/14, i.e., immediately following the heavy load condition previously studied. Although the two traces are not overlapped, we can observe that the infinite Mempool trace is a translation of the other, and this denotes that not so many droppings are being done in the time frame (differently from Figure 8.3a).

Table 8.1 shows the results of this experiment. Although there seems to be not much difference in the number of droppings between moderate load and heavy load, the reader should consider that we are studying a class of cheaper transactions. We may notice, also in this case, that the model that we propose outperforms both the reference classifiers.

8.3.4 Reliability Analysis as Function of the Mempool state

Reliability becomes crucial especially when the perceived arrival intensity λ_f is higher than the maximum system service capacity $B\mu$. Let $\rho = \lambda_f/(B\mu)$ be the perceived load factor of the system. Recall that, by rising the offered fee per Byte f, the perceived arrival rate decreases and hence also the perceived load

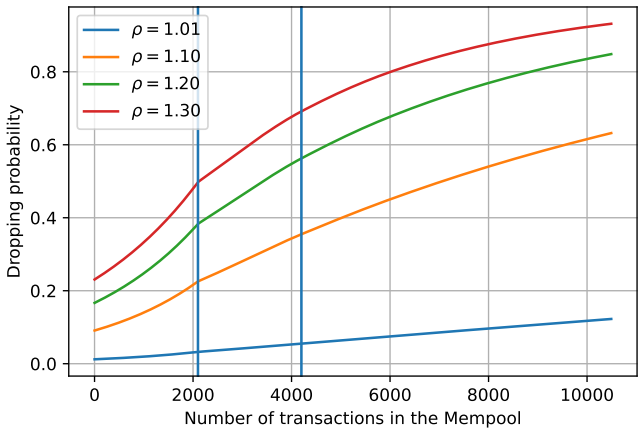


Figure 8.4: Dropping probability for different load factors.

factor. Figure 8.4 shows the impact of this fee modulation as function of the occupancy of the Mempool. The vertical lines show the first two multiplications of the block capacity. It is interesting to observe that this function appears to be convex only when the transaction may be included in the first block, if this is impossible, then it becomes concave.

8.4 Conclusion

In this chapter, we have proposed a model to predict the confirmation or dropping probabilities of transactions in PoW blockchain. The model exploits the auction-based mechanism underlying the confirmation process of these blockchains to derive the the dropping probabilities from the offered fees. The analysis relies on theory of difference equations and its main step is the computation of the roots of a certain characteristic polynomial. The advantage of this white box model relies on the fact that no historical data are necessary to train the model. This is a major drawback of machine learning models applied to this context since dropped transactions do not leave traces in the blockchain logs and hence, if for Bitcoin we have a dataset that allows us to infer these events, for other blockchains analogous historical data are unavailable.

To the best of our knowledge, reliability analysis of the transaction confirmation process in PoW blockchain is a novel aspect in the field of blockchain studies. However, the increasing popularity of these distributed ledgers combined with their limited throughput due to their intrinsic security design, are going to pose the problem of the minimization of the application running costs while maintaining a certain level of reliability. Our contribution is a first step

toward an automatic optimization of this trade-off.

Future works include integration of this model with workload predictors (see, e.g., Chapter 7) to allow the long-term classification of transactions.

Chapter 9

Conclusion

In this thesis, we examine the blockchain network driven by the most actively used family of consensus protocols, namely PoW, in order to study the performance, reliability and fairness of such systems. While performance and reliability are addressed from the end-user and the system perspectives (see Chapters 5, 6 and 8), the fairness is a property that concerns miners (see Chapter 4).

Regarding the fairness in private blockchain domain, we introduce a solution based on the sliding window approach that is easy to implement and integrate in existing systems. This solution offers a feasible method of tackling with two network security issues. On the one hand, our algorithm limits the miners' ability of the new block consolidation such that the 50% and the greedy miner's attacks cannot be executed in such network. On the other hand, thanks to our quantitative Markovian model, we can balance the HPs of the miners to ensure fair sharing of the mining load. What is more, we discuss the possible scenarios of blockchain behaviors considering different size of the sliding window and various fractions of fair and unfair miners in the network.

For what concerns the performance and reliability assessment of the main public blockchains, we introduce and then examine outcomes of our analytical models for estimation of the optimal transaction confirmation time and, as a corner case, assess the probability of being eventually confirmed or evicted from the Mempool for transactions with fairly low fees. Such models are based on the biggest and the most well-known public blockchain, Bitcoin. In addition, to enhance our model we assess several forecasting methods to provide comprehensive prediction of transaction arrival rate (see Chapter 7). Such contribution is of importance for any user producing the transactions as (i) one is always keen to know the optimal fee he/she should propose to guarantee the confirmation of his/her transactions at each moment of time after certain confirmation delay and (ii) considering arbitrary confirmation delays (and consequently relatively low transaction fees) one would like to know the probabilities of such transactions to be eventually included in a block (e.g., storing data from IoT devices).

It is worth noting that all the obtained results are fully applicable to every other blockchain network with similar key properties such as PoW.

The initial purpose of our thesis was to provide mathematically tractable analysis of blockchain networks with PoW consensus mechanism. However, we may state another crucial contribution: the evidence that the queuing models can be and should be of high applicability for the purposes of the blockchain evaluation as it provides fast and powerful mechanism to study the reliability and performance of the networks overcoming the more common approaches, such as Monte Carlo simulations. In addition, we believe that the aggregated positive effect of this thesis can be the first step forward to a greater framework to study, develop new and improve existing blockchain protocols.

Finally, although PoW is widely used and acknowledged family of blockchain protocols, the application of analytical models and the genuine framework must not be limited by these blockchains only. Clearly, the PoS as well as many other promising solutions developing now pose a great challenge to the queuing systems. Nevertheless, the modeling systems based on Markov chains should be able to grant a valuable mathematical interpretation of such blockchains providing a great scope of analytical insights. The latter could be a viable subject to future work.

Bibliography

[1] Jameela Al-Jaroodi and Nader Mohamed. Blockchain in industries: A survey. *IEEE Access*, 7:36500–36515, 2019.

[2] Maher Alharby, Roben Castagna Lunardi, Amjad Aldweesh, and Aad van Moorsel. Data-driven model-based analysis of the Ethereum verifier's dilemma. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks, DSN*, pages 209–220. IEEE, 2020.

[3] Maher Alharby and Aad van Moorsel. The impact of profit uncertainty on miner decisions in blockchain systems. *Electronic Notes on Theoretical Computer Science*, 340:151–167, 2018.

[4] Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. On fairness in committee-based blockchains. In *Proc. of the 2nd International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2020)*, pages 4:1–4:15, 2020.

[5] Norman T.J. Bailey. On queueing processes with bulk service. *J. of the Royal Statistical Society, Series B*, 16(1):80–87, 1954.

[6] Simonetta Balsamo, Andrea Marin, Isi Mitrani, and Nicola Rebagliati. Prediction of the consolidation delay in blockchain-based applications. In *Proc. of Int. Conf. on Performance Engineering (ICPE)*, pages 81–92, 2021.

[7] Simonetta Balsamo, Andrea Marin, Ivan Mitrani, and Nicola Rebagliati. Prediction of the consolidation delay in blockchain-based applications. In *Proc. of ICPE '21: ACM/SPEC International Conference on Performance Engineering*, pages 81–92. ACM, 2021.

[8] Jiabin Bao, Debiao He, Min Luo, and Kim-Kwang Raymond Choo. A survey of blockchain applications in the energy sector. *IEEE Systems Journal*, 15(3):3370–3381, 2020.

[9] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In *Proc. of Int. Conf. on Financial Cryptography and Data Security*, pages 142–157. Springer, 2016.

[10] Dario A. Bini. Numerical computation of polynomial zeros by means of Aberth's method. *Numerical Algorithms*, 13(2):179–200, 1996.

[11] Dario A. Bini and Leonardo Robol. Solving secular and polynomial equations: A multiprecision algorithm. *J. of Computational and Applied Mathematics*, 272, 2015.

[12] George E.P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[13] Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media, 2013.

[14] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proc. of the 3rd Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186, 1999.

[15] Kaylash Chaudhary, Vishal Chand, and Ansgar Fehnker. Double-spending analysis of bitcoin. In *Proc. of 24th Pacific Asia Conference on Information Systems, (PACIS 2020)*, page 210, 2020.

[16] Jacob Willem Cohen. *The single server queue*, volume 8 of *North-Holland series in Applied Mathematics and Mechanics*. North-Holland, revised edition, 1979.

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

[18] Karl Dilcher, James D. Nulton, and Kenneth B. Stolarsky. The zeros of a certain family of trinomials. *Glasgow Mathematical Journal*, 34:55–74, 1992.

[19] David Easley, Maureen O'Hara, and Soumya Basu. From mining to markets: The evolution of bitcoin transaction fees. *Journal of Financial Economics*, 134(1):91–109, 2019.

[20] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *Proc. of 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 45–59, 2016.

[21] Mahboubeh Faghih Mohammadi Jalali and Hanif Heidari. Predicting changes in Bitcoin price using grey system theory. *Financial Innovation*, 13(6), 2020.

[22] Jean-Michel Fourneau, Andrea Marin, and Simonetta Balsamo. Modeling energy packets networks in the presence of failures. In *Proc. of 24th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS*, pages 144–153. IEEE Computer Society, 2016.

[23] Bennett L. Fox and Peter W. Glynn. Computing poisson probabilities. *Communications of the ACM*, 31(4):440–445, 1988.

[24] Brian Fralix. On classes of bitcoin-inspired infinite-server queueing systems. *Queueing systems*, 95:29–52, 2020.

[25] Lucia Gallina, Sardaouna Hamadou, Andrea Marin, and Sabina Rossi. A probabilistic energy-aware model for mobile ad-hoc networks. In *Proc. of Analytical and Stochastic Modeling Techniques and Applications - 18th International Conference, ASMTA*, volume 6751 of *Lecture Notes in Computer Science*, pages 316–330. Springer, 2011.

[26] Stefan Geissler, Thomas Prantl, Stanislav Lange, Florian Wamser, and Tobias Hossfeld. Discrete-time analysis of the blockchain distributed ledger technology. In *2019 31st International Teletraffic Congress (ITC 31)*, pages 130–137. IEEE, 2019.

[27] Saber Goldberg. *Introduction to difference equations*. Dover Publications, 1986.

[28] Aditya Gopalan, Abishek Sankararaman, Anwar Walid, and Sriram Vishwanath. Stability and scalability of blockchain systems. *Proc. of ACM on Measurement and Analysis of Computer Systems*, 4(2):art. n. 35, 2020.

[29] Henry Wadsworth Gould. *Combinatorial Identities*. 1972.

[30] Rachid Guerraoui and Jingjing Wang. On the unfairness of blockchain. In *Proc. of Int. Conf. on Networked Systems (NETYS 2018)*, pages 36–50. Springer, 2018.

[31] Rowel Gundlach, Martijn Gijsbers, David Koops, and Jacques Resing. Predicting confirmation times of bitcoin transactions. *ACM Perf. Eval. Review*, 48(4):16–19, 2021.

[32] Peter G. Harrison and Andrea Marin. Product-forms in multi-way synchronizations. *Comput. J.*, 57(11):1693–1710, 2014.

[33] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning. Data mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

[34] Jose Hernandez-Orallo, Peter Flach, and Cesar Ferri. A unified view of performance metrics: translating threshold choice into expected classification loss. *Journal of Machine Learning Research*, 13:2813–2869, 2012.

[35] Gur Huberman, Jacob D. Leshno, and Ciamac Moallemi. Monopoly without a monopolist: An economic analysis of the bitcoin payment system. *Review of Economic Studies*, 0:1–30, 2021.

[36] Muhammad Anas Imtiaz, David Starobinski, and Ari Trachtenberg. Investigating orphan transactions in the bitcoin network. *IEEE Transactions on Network and Service Management*, 18(2):1718–1731, 2021.

[37] Shuai Wang Fei-Yue Wang Juanjuan Li, Yong Yuan. Transaction queuing game in Bitcoin blockchain. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 114–119. IEEE, 2018.

[38] Samuel Karlin and Howard M. Taylor. *A first course in stochastic processes*. Academic Press, Second edition, 1968.

[39] Shoji Kasahara and Jun Kawahara. Effect of bitcoin fee on transaction-confirmation process. *Journal of Industrial & Management Optimization*, 15(1):365, 2019.

[40] Guy Katriel. Gambler's ruin probability - a general formula. *Statistics & Probability Letters*, 83(10):2205–2210, 2013.

[41] Yoshiaki Kawase and Shoji Kasahara. Priority queueing analysis of transaction-confirmation time for bitcoin. *J. of Industrial & Management Optimization*, 16(3):1077–1098, 2020.

[42] David G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chains. *The Annals of Mathematical Statistics*, 24(3):338–354, 1953.

[43] Kashif Mehboob Khan, Junaid Arshad, and Muhammad Mubashir Khan. Investigating performance constraints for blockchain based secure e-voting system. *Future Gener. Comput. Syst.*, 105:13–26, 2020.

[44] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Proc. of 37th Annual Int. Cryptology Conf - Advances in Cryptology (CRYPTO 2017)*, pages 357–388. Springer, 2017.

[45] Leonard Kleinrock. *Queueing Systems Volume 1: Theory*. Wiley, 1975.

[46] David Koops. Predicting the confirmation time of bitcoin transactions. *preprint*, 2018. Available on Arxive https://arxiv.org/pdf/1809.10596.pdf.

[47] Sergio Demian Lerner. Decor+ hop: A scalable blockchain protocol. Available at `https://scalingbitcoin.org/papers/DECOR-HOP.pdf`, 2015.

[48] Juanjuan Li, Yong Yuan, and Fei-Yue Wang. Analyzing bitcoin transaction fees using a queueing game model. *Electronic Commerce Research*, pages 1–21, 2020.

[49] Zhijie Li, Haoyan Wu, Brian King, Zina Ben Miled, John Wassick, and Jeffrey Tazelaar. A hybrid blockchain ledger for supply chain visibility. In *Proc of 17th Int. Symp. on Parallel and Distributed Computing (ISPDC)*, pages 118–125, 2018.

[50] Damiano Di Francesco Maesa and Paolo Mori. Blockchain 3.0 applications survey. *Journal of Parallel and Distributed Computing*, 138:99–114, 2020.

[51] Ivan Malakhov, Carlo Gaetan, Andrea Marin, and Sabina Rossi. Workload prediction in btc blockchain and application to the confirmation time estimation. In *Performance Engineering and Stochastic Modeling*, pages 3–21. Springer, 2021.

[52] Ivan Malakhov, Andrea Marin, and Sabina Rossi. Analysis of the confirmation time in proof-of-work blockchains. 2022. Available at SSRN: https://ssrn.com/abstract=4031244.

[53] Morris Marden. *Geometry of Polynomials*, volume 3 of *Mathematical Surveys and Monographs*. AMS, 1966.

[54] Andrea Marin and Sabina Rossi. Autoreversibility: Exploiting symmetries in Markov chains. In *Proc. of 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS*, pages 151–160. IEEE Computer Society, 2013.

[55] Andrea Marin and Sabina Rossi. On the relations between lumpability and reversibility. In *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, pages 427–432. IEEE, 2014.

[56] Andrea Marin and Sabina Rossi. On the relations between Markov chain lumpability and reversibility. *Acta Informatica*, 54(5):447–485, 2017.

[57] Andrea Marin, Sabina Rossi, Dario Burato, Andrea Sina, and Matteo Sottana. A product-form model for the performance evaluation of a bandwidth allocation strategy in WSNs. *ACM Trans. Model. Comput. Simul.*, 28(2):13:1–13:23, 2018.

[58] Johnnatan Messias, Mohamed Alzayat, Balakrishnan Chandrasekaran, and Krishna 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.

[59] Johnnatan Messias, Mohamed Alzayat, Balakrishnan Chandrasekaran, Krishna P Gummadi, Patrick Loiseau, and Alan Mislove. Selfish & opaque transaction ordering in the bitcoin blockchain: the case for chain neutrality. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 320–335, 2021.

[60] Mohammed Mudassir, Shada Bennbaia, Devrim Unal, and Mohammad Hammoudeh. Time-series forecasting of bitcoin prices using high-dimensional features: a machine learning approach. *Neural Computing and Applications*, 2020.

[61] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[62] Muhammad Hassan Nasir, Junaid Arshad, Muhammad Mubashir Khan, Mahawish Fatima, Khaled Salah, and Raja Jayaraman. Scalable blockchains - A systematic review. *Future Gener. Comput. Syst.*, 126:136–162, 2022.

[63] Marcel F. Neuts. The busy period of a queue with batch service. *Operations Research*, 13(5):815–819, 1965.

[64] Kelly Olson, Mic Bowman, James Mitchell, Shawn Amundson, Dan Middleton, and Cian Montgomery. Sawtooth: An introduction, 2018.

[65] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM symposium on principles of distributed computing*, pages 315–324, 2017.

[66] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *Proc. of 31st International Symposium on Distributed Computing (DISC 2017)*, volume 91, pages 39:1–39:16, 2017.

[67] Hao Qiu and Tong Li. Auction method to prevent bid-rigging strategies in mobile blockchain edge computing resource allocation. *Future Gener. Comput. Syst.*, 128:1–15, 2022.

[68] Saulo Ricci, Eduardo Ferreira, Daniel Sadoc Menasche, Artur Ziviani, Jose Eduardo Souza, and Alex Borges Vieira. Learning blockchain delays: A queueing theory approach. *ACM Perf. Eval. Review*, 46(3):122–125, 2019.

[69] Sheldon M. Ross. *Stochastic processes, 2nd ed.* Wiley series in probability and statistics. 2008.

[70] Sabina Rossi and Andrea Marin. On discrete time reversibility modulo state renaming and its applications. *EAI Endorsed Transactions on Self-Adaptive Systems*, 1(3), 2015.

[71] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Proc. of Int. Conf. on Financial Cryptography and Data Security*, pages 515–532. Springer, 2017.

[72] Ivo Stoepker, Rowel Gundlach, and Stella Kapodistria. Robustness analysis of bitcoin confirmation times. *SIGMETRICS Perf. Eval. Review*, 48(4):20–23, 2021.

[73] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[74] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *Prof. of Int. Work. on Open Problems in Network Security*, pages 112–125. Springer, 2016.

[75] Marko Vukolić. Rethinking permissioned blockchains. In *Proc. of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 3–7, New York, NY, USA, 2017. ACM.

[76] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, 7:22328–22370, 2019.

[77] Sam M. Werner, Paul J. Pritz, Alexei Zamyatin, and William 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*, pages 127–134, 2019.

[78] Edmund Taylor Whittakerm and George Neville Watson. *A course of modern analysis*. Cambridge at the university press, 1920.

[79] Abdullah A. Zarir, Gustavo A. Oliva, Zhen M. Jiang, and Ahmed E. Hassan. Developing cost-effective blockchain-powered applications: A case study of the gas usage of smart contract transactions in the ethereum blockchain platform. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(3):1–38, 2021.