Title
Analysis and Design of an Incentive Scheme for a Proof-of-Work Blockchain

Author
Yoo, Seunghyun

Publication Date
2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Analysis and Design of an Incentive Scheme for a Proof-of-Work Blockchain

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Seunghyun Yoo

2021

ABSTRACT OF THE DISSERTATION

Analysis and Design of an Incentive Scheme for a Proof-of-Work Blockchain

by

Seunghyun Yoo
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2021
Professor Leonard Kleinrock, Chair

A blockchain characterized by a distributed, decentralized, and public computer system rapidly grows and realizes the Internet of Value. The fundamental idea is to replicate its state database to every node computer with protocol-specific restrictions to keep consistency and integrity. Transferring controls from a central entity to each individual sounds compelling, but a protocol designer should carefully design an incentive scheme to encourage anonymous participants to do the desired actions.

In this dissertation, we analyze the reward policy on block creation in a proof-of-work blockchain. The reward policy consists of a block reward and transaction fees. A protocol designer intends to incentivize block miners to process transactions until the designed capacity. However, we observe that the hidden costs may distort the reward policy; block miners do not have incentives if processing and network delays are not negligible. We model a transaction fee market, the block mining process with the delays, and the decision-making of block miners under the reward policy. We express the expected mining revenue as a function of the number of transactions in a block. We then identify the operating region of a block reward to mitigate the effect of the hidden costs. We propose a dynamic block reward

of a proportional subsidy for processing transactions as a revised incentive scheme. Thus, blockchain users experience reduced waiting time and less expensive transaction fees.

In addition, we further expand queueing theory, especially for a head-of-line bulk service priority queue with an arbitrary service time distribution. We provide analytical expressions (exact solutions) for critical numbers in a transaction fee market: average waiting time, minimum transaction fee, and average total transaction fee.

The dissertation of Seunghyun Yoo is approved.

Demetri Terzopoulos

Peter L. Reiher

Miloš D. Ercegovac

Leonard Kleinrock, Committee Chair

University of California, Los Angeles

2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Remembering my former advisor Prof. Mario Gerla, I am truly honored and grateful to have an opportunity to be in the NRL (Network Research Laboratory) at UCLA. Without his initial support, encouragement, and his creative mind, I would not have been able to reach this moment.

Once again, I am truly honored and grateful to have Prof. Leonard Kleinrock as an advisor. His endless passion for research, deep intuition, leadership, and true humanity have always been inspiring and shining. His pencil and paper (recently iPad) have set a goal in my life. I would not have been able to make a single inch of progress without his support, guidance, and patience. I am forever indebted to him.

I would also like to express my sincere gratitude to the committee members, Prof. Peter L. Reiher, Prof. Miloš D. Ercegovac, and Prof. Demetri Terzopoulos, for their invaluable feedback, support and understanding. They have flourished the community of UCLA Computer Science students. As a student or sometimes as a teaching assistant, I learned a lot from them.

The Connection Lab, located on the third floor in the Boelter Hall, a.k.a. the birthplace of the Internet, is a wonderful place for research. Sorry for the COVID-19 pandemic, but my colleagues in the lab must witness it. I would like to thank Rüstem Can (thank you for holding regular coffee time with me at 1 a.m. on the roof-top), Chloe, Kevin, Eli, and Willy.

I would like to thank the NRL students: Jorge, Josh, Noor, Pengyuan, Qi, Ruolin, Seungbae, Tuan, and Vince. I would also like to thank the visiting scholars (plus the NRLers): Dr. Jeongcheol Lee, Prof. Seungmin Oh, Prof. Uichin Lee, Prof. Wooseong Kim, Prof. Daiki Nobayashi, Prof. Torsten Braun, Prof. Feng Zhou, Prof. Susumu Ishihara, and others who inspired me. Specially, I would like to express my heartfelt gratitude to the Gerla family: Dr. Margaret Phillips, Marisa Gerla, and Cristina Gerla.

I would like to thank Spyros, Enrique, Taqi, Gulzar, Tianyi, Saswat, Lisa, Arul, Turker,

and many friends at UCLA. I would also like to thank Hakmin Kim and Hyunjoong Kim for the internship at Hyundai. Thanks for joining Kelton Stack Overflow: Jiyong, Sungjoon, and Minkyu serving in US Army. Thanks to Gwanak campus EER06 friends who have/had been in a graduate school in Korea or the United States: Byeongju, Hyungjin, Hyunsoo, Jeongsoo, Joonseok, Junhan, Kimin, Minwon, Sangdoo, Wontaek, and Wootaek. Also, thanks to Gwangyang friends: Seulgi, Wonki, Kihyun, Juha, Jaeyoung, Hyunggyun for checking my condition, especially when I was/am undergoing a series of life challenges.

I would like to thank Prof. Paul Eggert and Prof. Todd Millstein for programming languages. I enjoyed/learned the importance of teaching, and functional programming reorganized my coding style. I would also like to thank Joseph Brown and Computer Science Department staff for helping me in many ways.

I would like to thank my parents, Hongjong Yoo and Mihye Park, for telling me: "We love you as you are." Without your unconditional love, support, and trust, everything would not be possible.

Finally, I would like to thank the supporters of the Connection Lab.

VITA

2014 B.S. in Electrical and Computer Engineering,
Department of Electrical and Computer Engineering,
Seoul National University,
Seoul, Korea.

2018 M.S. in Computer Science,
Department of Computer Science,
University of California, Los Angeles,
Los Angeles, California.

2014 - 2021 Graduate Student Researcher/Teaching Assistant,
Department of Computer Science,
University of California, Los Angeles,
Los Angeles, California.

PUBLICATIONS

Seunghyun Yoo, Seungbae Kim, Joshua Joy, Mario Gerla. "Promoting Cooperative Strategies on Proof-of-Work Blockchain" *2018 International Joint Conference on Neural Networks*, 2018.

Pengyuan Du, Seunghyun Yoo, Qi Zhao, Muhao Chen, Mario Gerla. "Towards Opportunistic Resource Sharing in Mobile Social Networks: an Evolutionary Game Theoretic Approach" *Proceedings of the Nineteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2018.

Seungbae Kim, Jinyoung Han, Seunghyun Yoo, Mario Gerla. "How Are Social Influencers Connected in Instagram?" *Social Informatics - 9th International Conference*, 2017.

Muhammad Ali Gulzar, Matteo Interlandi, Seunghyun Yoo, Sai Deep Tetali, Tyson Condie, Todd D. Millstein, Miryung Kim. "BigDebug: Debugging Primitives for Interactive Big Data Processing in Spark" *Proceedings of the 38th International Conference on Software Engineering*, 2016.

Matteo Interlandi, Kshitij Shah, Sai Deep Tetali, Muhammad Ali Gulzar, Seunghyun Yoo, Miryung Kim, Todd D. Millstein, Tyson Condie. "Titian: Data Provenance Support in Spark" *Proceedings of the VLDB Endowment*, 9(3):216-227, 2015.

CHAPTER 1

Introduction

1.1 Motivation

The idea of decentralization, which transfers controls from a central entity to each individual, is a response to the current problems of a centralized system. Admittedly, the clear ownership of a centralized system provides a tailored experience to its users and enables the system components to be configured in a highly efficient manner. The users generally delegate their controls and often responsibilities to one trusted party for the sake of convenience. However, such delegation may pose potential risks such as a single point of failure, a lack of transparency, and an imbalance between users and service providers.

A blockchain characterized by a distributed, decentralized, and public computer system instantiates the concept of decentralization and further realizes the Internet of Value. The fundamental idea is to replicate its state database to every node computer with some protocol-specific restrictions to keep its consistency and integrity. The remarkable properties are its global openness and transparency, while the system state is tamper-proof. A centralized system does not possess these properties as a system operator can alter the internal state. A blockchain may empower the Internet space by allowing people to define the value in a digital form on the borderless peer-to-peer network of computers.[1]

However, a blockchain requires special care for its successful operation. Unlike a cen-

[1]The (potential) applications are programmable money (cryptocurrency), smart contract (business logic), logistics, document management, copyright management, social media, online games, identification management, verifiable electronic voting system, and many others.

tralized system, an incentive scheme is significant in encouraging anonymous participants to do the desired actions. Improperly designed incentives may lead to degraded system performance or system failure in the worst-case scenario. Since it is also hard to modify the established system components and even some minor parameters (causing political conflicts), a careful study of the incentive scheme is necessary.

A typical incentive scheme for a proof-of-work blockchain defines a block reward and transaction fees that are received by the "miners" who process the user transactions. The block reward has multiple roles: incentivizing running blockchain node computers, subsidizing transaction fees, and issuing new coin into a blockchain economy. The transaction fee paid by users can be either constant or decided by the market mechanism. However, we observed that some blockchain miners do not process user transactions as opposed to intuition. In this work, we build a mathematical foundation to explain why these miners make a seemingly irrational decision and also suggest ways to properly incentivize them to process transactions.

1.2 Contributions

This dissertation studies the dynamics of a transaction fee market, the mining process with processing and network delays, and the reward policy on block creation in a proof-of-work blockchain. We expand queueing theory, especially for a bulk service queue with multiple priority groups, to relate the number of waiting transactions in a pending transaction pool and the transaction fee market.

A block miner claims a block reward R_{B} created by a system and transaction fees R_{TX} paid by users, only if her proposed block survives from the intense mining competition where its probability is p_{survival}. Every miner selects r_i, the number of transactions she includes in a block to maximize her expected mining revenue given by

$$\mathbb{E}[R_{\mathrm{B}} + R_{\mathrm{TX}}] = (R_{\mathrm{B}} + R_{\mathrm{TX}})p_{\mathrm{survival}} \tag{1.1}$$

The main problem is to find the optimum number of transactions r_i^* that maximizes the expected mining revenue and an operating range of block reward R_{B} such that

$$\forall i : r_i^* \geq B \tag{1.2}$$

where B is a bulk size, i.e., the maximum number of transactions allowed in a block.

This work makes the following contributions:

- **(Average Waiting Time for an $M/G^B/1$ Head-of-Line Bulk Service Priority Queue)**[2] We provide an analytical expression (exact) to compute the average waiting time for a head-of-line [Kle76, p.119] bulk service priority queue with an arbitrary service time distribution. A typical method involves evaluating B system stationary probabilities. Our numerical approach bypasses this necessity.

- **(Minimum Transaction Fee)** We transform the distribution of the number of waiting transactions to the distribution of the minimum group index, and we then multiply with the fee distribution as a weight. We also present concrete solutions for an $M/M^B/1$ head-of-line bulk service priority queue. These are exact solutions.

- **(Average Total Transaction Fee)** We derive an analytical expression for the expected value of the total transaction fee, $\mathbb{E}[R_{\mathrm{TX}}]$. It is also an exact solution.

- **(Mining Process with Delays)** We approximate the probability of fork survival, p_{survival}, i.e., winning the mining competition. Finding an exact formula is complicated because of an explosively growing number of combinations. We provide the formulae for lower and upper bounds of the probability of fork survival. We also provide concrete solutions for special cases for two miners and three miners. These are approximation.

[2]Kendall's queueing theory notation is $A/S/c$ where A denotes an arrival time distribution, S denotes a service time distribution and c denotes the number of servers. M stands for Markovian (Poisson process), G stands for a general distribution, and B stands for the size of the bulk when more than one customer is served at a time.

- **(*Three Miner Approximation* method)** We find that the solution for the special case for three miners is useful to approximate the probability of fork survival among many miners with two strategies.

- **(Reward Policy)** We derive the optimum number of transactions in a block r_i^* to explain the behavior of miners. We then identify an operating region of the block reward R_{B} that incentivizes a miner to process transactions to the designed capacity B, and achieve the maximum system throughput μB.

- **(Dynamic Block Reward)** We also propose a revised reward policy of a proportional subsidy R_{S} for processing transactions. We show that the optimum number of transactions increases and the operating region of a block reward gets extended.

1.3 Roadmap of the Dissertation

The rest of the dissertation is organized as follows: Chapter 2 presents the background of a blockchain system architecture. Chapter 3 provides the mathematical analysis of the transaction fee market and the expected waiting time using queueing theory. In Chapter 4, we analyze the block mining process considering processing and network delays. Chapter 5 studies the optimum reward policy on block creation and the corresponding decision-making of block miners, and proposes the dynamic block reward to improve their response.

The above roadmap of our research is motivated by the following. We wish to establish incentives to encourage miners to mine blocks with B, the maximum number of transactions. Consistent with the protocol constraints, this will deliver the most transactions per second while at the same time will maintain lower transaction fees for the users since their transactions are being processed quickly and without cutthroat competition for service. One of our main goals is to provide the best incentives to the miners to mine full blocks. To do so, we note that Eq. 1.1 identifies the expected reward that a miner receives and one system

design variable in this equation that we study is to optimally select the block reward R_{B}, in a way that incentivizes the miner to mine full blocks. Thus we seek to evaluate the terms of this equation. In particular, we must find $\mathbb{E}[R_{\mathrm{TX}}]$ as well as p_{survival}. We devote Chapter 3 to finding $\mathbb{E}[R_{\mathrm{TX}}]$ (whose solution is Eq. 3.78) and Chapter 4 to finding p_{survival} (whose solution is in Eq. 4.79). Then, in Chapter 5, we finally solve for the value of R_{B} that identifies the optimal value for this block reward.

Finally, we conclude in Chapter 6.

CHAPTER 2

Background

2.1 Blockchain System Overview

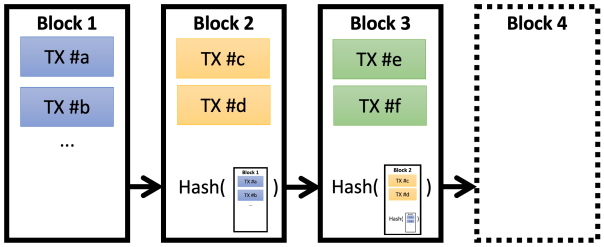

Figure 2.1: An immutable linked list of blocks.

In this section, we briefly introduce the core idea of a blockchain. In short, a blockchain is an immutable linked list of blocks which contain transactions. Every block must include a hash digest of the parent block as shown in Fig. 2.1. Now the data structure obtains the property of immutability. The alteration on the history, either for a malicious purpose or a simple hardware failure, can easily be detected because any single-bit change will result in entirely different hash digests as shown in Fig. 2.2.

This append-only database is replicated over a peer-to-peer network. However, the immutability alone does not guarantee a global consensus. Any peer may append a new block to the end of the linked list. It is virtually impossible for peers, especially anonymous ones, to determine which version is valid among many legitimate blocks.

Bitcoin [Nak08] solves this global consensus problem by using a proof-of-work [Bac02, DN92, JJ99] scheme, which is originally designed to prevent spam emails by imposing some

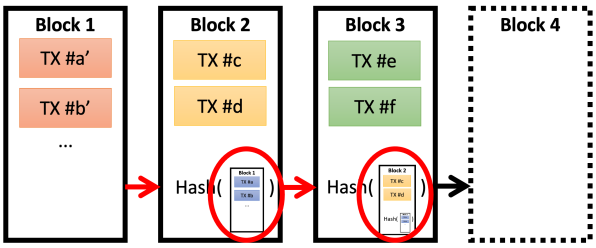

Figure 2.2: Any single-bit change invalidates subsequent hash digests.

computational cost. According to this special rule, a peer must present a random number (nonce) that satisfies a trapdoor function [Yao82] condition to propose a new block. The trapdoor function is easy to verify but very hard to find its inverse. In addition, the protocol specifies that peers should choose the longest chain as a canonical state. To break the global consensus, an attacker needs to possess at least 51% of the total computing power on the network and reiterate mandatory proof-of-work tasks [Nak08, ES14, GKL15, GKK16, PSS17].

Cryptocurrency

Address	Balance
0x0001	100
0x0002	0
0x0003	50
0x0004	25

Computer!

Address	Compiled Assembly Code
0x0001	
0x0002	606060405234156100 0c57fe5b6040516060...
0x0003	8080601f016020809104026020016040519080 1016040528093929190 8...
0x0004	

Figure 2.3: From a cryptocurrency to a global computer.

A transaction is a verifiable message secured by a digital signature.[1] A private key represents the ownership of the specific address in a blockchain.[2] Fig. 2.3 shows an example of what information a blockchain stores. A ledger keeps the balance (and the history) of

[1]ECDSA (Elliptic Curve Digital Signature Algorithm) is popular due to its compactness.

[2]A wallet address is derived from a public key.

each address. A Bitcoin transaction is mostly used for value transfer. It also supports some special instructions for limited-purpose scripting. Ethereum [Woo14] extends this idea to facilitate a general-purpose smart contract [Sza97] platform. A ledger also stores data and even a program code associated with the address. A stack-based virtual machine executes a program code and records state transitions (the program result) to the ledger. In other words, a blockchain is a global computer running on the Internet.

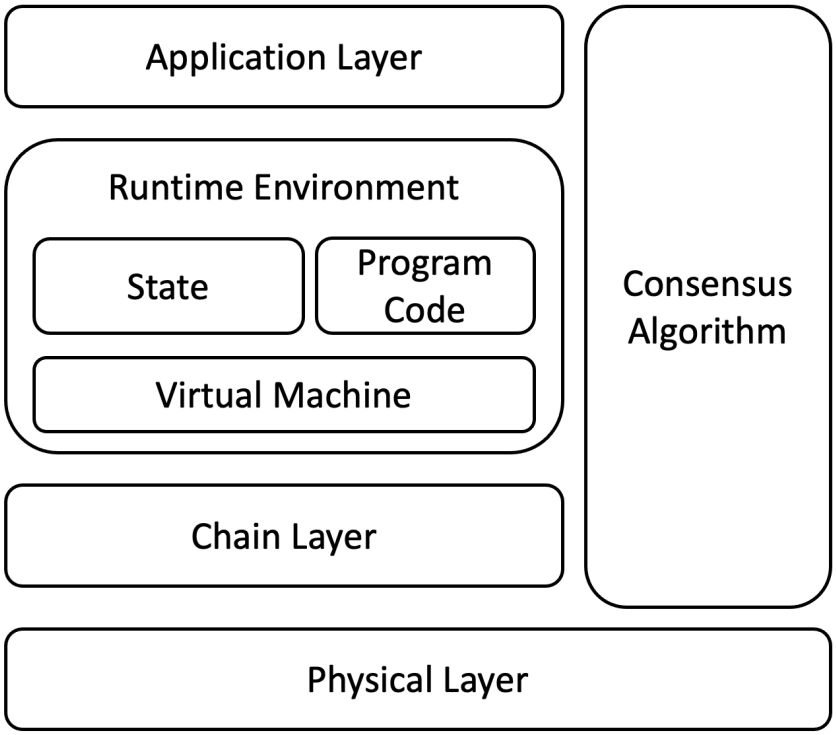

Figure 2.4: Overall blockchain components.

Fig. 2.4 depicts the overall blockchain components. The physical layer provides basic computing resources such as CPU, memory, storage, and network. The chain layer defines a data structure. A linked list of blocks is the most common form. One may also consider an alternative data structure like DAG (Directed Acyclic Graph) [FKS19]. The runtime environment layer consists of the internal state database and the virtual machine that can execute a program code stored in the ledger. A smart contract may take some inputs from

the external world if necessary. A blockchain user may define an advanced service (business logic) upon the application layer by creating a smart contract or combining the existing contracts.[3] Finally, the consensus algorithm ensures that every participant has a global common view of the state.

2.2 Consensus Algorithms

2.2.1 Proof-of-Work

(Mining) Bitcoin [Nak08] relies on a proof-of-work scheme to reach a global consensus in a trust-less environment.[4] A miner prepares a block by selecting and processing transactions from a pending transaction pool where the transactions received from users are temporarily stored. In addition, a miner (i.e., a node who proposes a block) must find a random number, called a *nonce*, that satisfies Eq. 2.1, called a *hash puzzle*. The term *target difficulty* adjusts the difficulty of the hash puzzle.

$$\underbrace{\texttt{SHA256}\left(\texttt{SHA256}\left((\text{nonce})+(\text{data})\right)\right)}_{\text{A numerical value}} < \underbrace{(\text{target difficulty})}_{\text{A numerical value}} \tag{2.1}$$

A hash function such as `SHA256` maps an input to a random number. Due to its randomness, the miner finds a nonce that satisfies the above difficulty condition through trial-and-error. This iterative procedure is called *mining*. The other miners can easily check the validity of the new block by plugging the nonce into Eq. 2.1.

(Block Creation Time) A feedback mechanism regulates the block creation time.[5] If the protocol senses that the block creation time is too fast, it sets the lower threshold

[3]However, a careful review is necessary on the logic of the contract code and its implication.

[4]There are more than 10,000 cryptocurrencies with slight differences as of 2021. We choose Bitcoin as an example because it is the first proof-of-work blockchain.

[5]The block creation time of Bitcoin is approximately 600 seconds. The block creation time of Ethereum is approximately $12 \sim 15$ seconds.

(target difficulty). Thus, the probability of solving a hash puzzle decreases. The opposite adjustment is necessary for the slowed block creation time. Fluctuations in the network hash rate (computing power) and propagation delays may affect the block creation time in a short period [DW13, BKK20]. Over a long period, the feedback mechanism regulates the block creation time reasonably.

(Probabilistic Finality) [GKL15] Since mining is a random process, there can exist multiple new blocks in the peer-to-peer network at the same time, even under the difficulty condition. To resolve such an inconsistent state, miners select the longest chain by default. Combined with the controlled block creation time, the probability that the past blocks are reverted to decreases quickly. One needs to possess 51% of the computing power (hash rate) on the network to supersede the other miners and reiterate mandatory proof-of-work tasks.

(Permission-less) Another attractive characteristic of a proof-of-work consensus is that it does not require authorization to join a blockchain peer-to-peer network. Open membership is achievable because the protocol counts the majority by computing power.

(Block Size) There is a hard limit on the maximum number of transactions in a block. It depends on the size of a digital signature (bytes) and the block size limit (bytes). An increased block size may improve performance in terms of the number of transactions processed per unit time. However, it causes more processing and network delays.

The block size limit naturally forms a transaction fee market. Users pay a fee to place their transactions in a new block. Miners naturally process transactions in order of the highest fees first. The level of transaction fee is sensitive to the demand on the block space as the supply is fixed.

(Reward Policy) It is important to define an incentive scheme properly as the participants are anonymous, and their interests may differ from the intention of a protocol designer. There are mostly two types of incentive rewards: R_{B} (block reward) and R_{TX} (transaction fees). The protocol specifies that a miner whose proposed block gets accepted

by the blockchain receives the block reward from a system plus the associated transaction fees. The block reward (which manifests itself as new coin to the miner) serves the following purposes: encouraging miners to run a blockchain node, subsidizing transaction fees, and issuing new coin.

This dissertation studies the reward policy on block creation in a proof-of-work blockchain.

2.2.2 Proof-of-Stake

While a proof-of-work consensus counts the majority by computing power, a proof-of-stake consensus uses the amount of share. The concept was first introduced by Peercoin [KN12]. It is generally energy-efficient since it does not rely on a proof-of-work style hash puzzle. The idea is based on the belief that most shareholders will not harm the system against themselves. A hybrid form of using both proof-of-work and proof-of-stake was suggested by [BLM14, CDF17]. Ethereum plans to migrate to a proof-of-stake consensus [BRL20]. Cardano [KRD17], Algorand [GHM17] with VDF (Verifiable Delay Functions) [BBB18], and Polkadot [Woo16] are pure proof-of-stake blockchains.

(Block Creation Time) The amount of share determines the probability of finding the next block. The miner who is chosen to create the next block has a probability of being selected in proportion to her stake. The block creation time is typically a constant.

(Finality) The cost of block creation is very cheap compared to that of a proof-of-work consensus. For this reason, a proof-of-stake consensus must tackle a nothing-at-stake problem [Zam15, LAB17, Mar18], i.e., a miner has an incentive to vote on multiple chain forks at the same time. It usually consists of a pre-commit stage and a commit stage. The protocol forbids reverting committed blocks.

(Permission-less) No explicit authorization is required to join a blockchain peer-to-peer network.

(Reward Policy) In a proof-of-stake consensus, a penalty is one way of mitigating the

nothing-at-stake problem. When a node does not behave as expected, e.g., being offline, the protocol allows forfeiting the staked coin. A block reward R_{B} is typically zero or nonzero under an inflationary policy. Block validation gets a reward instead. A transaction fee may exist, but some projects set a fixed amount.

2.2.3 Byzantine Fault Tolerance

This traditional consensus algorithm [Lam82, CL99] can also be used for a blockchain. It is a permission-ed system. Byzantine Fault Tolerance (BFT) requires knowledge of the number of members N whereas a proof-of-work or proof-of-stake consensus counts the majority by computing power or share, respectively.

A distributed computing system should consider two important properties: (1) liveness and (2) safety. The liveness property is associated with whether the system is available (i.e., continues to run). The safety property is associated with whether the system state is consistent. Fig. 2.5 depicts the relationship between the number of failing nodes and the two properties. Let N be the total number of nodes and T be the minimum group of nodes. Set theory tells us that if there are more than $N - T$ failing nodes, it is impossible to find a sufficient number of good nodes ($N - (N - T + 1) < T$). Let m be the number of shared nodes between two groups. The size of the union of the two groups is $1 \leq 2T - m \leq N$, i.e., $2T - N \leq m \leq 2T - 1$. If there are more than $2T - N - 1$ failing nodes, it is possible to find two disjoint groups leading to an inconsistent state. For a blockchain, the safety property is most important. The condition of $N - T \leq 2T - N - 1$, i.e., $T \geq \frac{2}{3}N + \frac{1}{3}$ tells us that 67% of nodes must be honest in order to reach a consistent state (i.e., safe).

Ripple [CM18] and Hyperledger Fabric [ABB18] are blockchains with the BFT consensus algorithm. Only pre-authorized nodes may create a new block. Stellar [LLM19] requires a node to select trustworthy entities. A transaction fee may exist, but a block reward R_{B} hardly exists in this consensus algorithm.

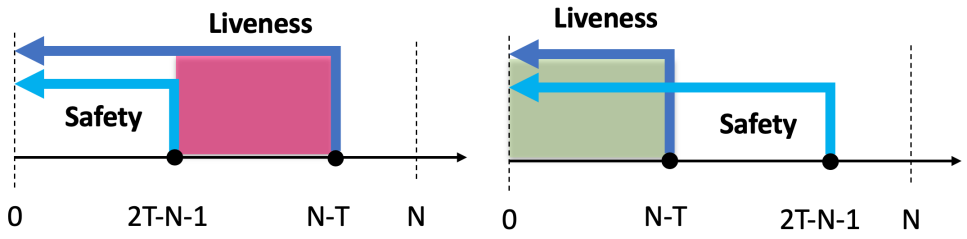

Figure 2.5: Safety and liveness with respect to the number of faulty nodes.

2.2.4 Variants

There are variants of a proof-of-work consensus to mitigate costly busy waiting.

(Proof-of-Elapsed-Time) A proof-of-elapsed-time [Int17] consensus leverages a special hardware feature such as Intel SGX (Software Guard Extensions). The code being executed in a protected area cannot be inspected or tampered with.

(Proof-of-Space) A proof-of-space (proof-of-capacity) [DFK15, ABF14] consensus uses a complicated and slow hash function. Instead of using a CPU or GPU, it precomputes hash values on a disk drive. Looking up the value from the disk drive is much faster than computing the hash function in real-time. Filecoin [BG18] and Chia [CP19] are blockchains with this consensus algorithm.

2.3 Security Threats

There are many unseen security threats and potential vulnerabilities to a blockchain regardless of its consensus algorithm. Some security assumptions on hash functions or digital signature algorithms might not hold in the future. Software defects and human errors are prone to happen repeatedly. They cause catastrophic damage.

In particular, the following attacks apply to a proof-of-work blockchain.

(Double Spending Attack) A 0-confirmation attack, or Finney Attack [Fin11] is one

of double-spending attacks. An attacker finds and hides a block with a transaction that moves the fund of the attacker to somewhere else. The attacker takes an irreversible service in the real world using it and broadcasts the hidden block to override the transaction. The users should be aware of that the probability that a transaction is reverted is not exactly zero.

(Nonlinearity in Mining) Selfish Mining [ES14, GKK16, SSZ16] is an exploit in which a miner generates a fork deliberately to make the other miners waste computing power on the stale block. Block Withholding (BWH) is an attack against the opponent mining pool by pretending to contribute to mining. Fork-After-Withholding (FAW) attack [KKS17] extends the BWH attack, and the gain is greater than or equal to that of a BWH attacker. These attacks weaken the assumption of 51% of computing power.

CHAPTER 3

Transaction Fee Market

3.1 Introduction

In a blockchain system, a transaction fee indicates the urgency of the submitted transaction. It motivates block miners to process transactions. Miners typically process transactions in order of the highest transaction fees first to maximize their mining revenue. Thus, the market mechanism prioritizes expensive user transactions, which prevents attackers' cheaper transactions from pushing the system load beyond the designed capacity. As the transaction fee determines the relative position among other competing transactions, a user should consider the trade-off relationship between her transaction fee and the waiting time required for her transaction to be processed.

We wish to find the average waiting time for a customer as a function of their transaction fee paid. To model this, we consider a head-of-line bulk service queue with multiple priority groups with an arbitrary service time distribution. A bulk service queue requires us to find the system stationary probabilities to obtain the waiting time distribution, and this typically involves the evaluation of some unknown constants. Below, we choose to make use of the Hilbert transform to bypass the necessity of simulations to evaluate those unknown constants.

The standard supply and demand curves provide a basic understanding of how a user determines her transaction fee. The equilibrium price is the point where the two curves intersect. A typical blockchain fee market has an inelastic supply curve due to the block size

limit. The equilibrium price is sensitive to the demand curve. In a head-of-line priority queue, we find that the minimum transaction fee in a block is the same as the equilibrium price. We derive the distribution of the minimum transaction fee in a block from the distribution of the number of customers (transactions) for each priority group. We also derive the formula for the average total transaction fee in a block.

3.2 Related Work

The literature has extensively studied bulk service queues. Bailey solved the limiting distribution of queue length by using an Imbedded Markov chain [Bai54]. Downton obtained the Laplace transform of the waiting time distribution from Bailey's result [Dow55]. Jaiswal solved time-dependent equations for a bulk service queue by defining an additional state (phase) [Jai60]. Neuts considered a case where the server must wait for enough customers to arrive to fill the bulk [Neu67], and Medhi obtained the waiting time distribution [Med75]. Chaudhry [CT81] and Holman *et al.* [HCG81] related the distribution of queue length at random epochs to departure epochs. Chaudhry and Templeton [CT83] covered almost complete variants of bulk service queue problems. The above methods usually need to find stationary probabilities numerically, except for some cases, and [PH86, BC89] developed an optimization technique for finding roots.

Another approach is to solve the recurrence relation known as the Lindley process by the Wiener-Hopf method [Lin52]. Keilson related the bulk service queue problem to the Hilbert problem and obtained the steady state waiting time distribution [Kei62]. Bhat nicely solved the Lindley process and provided a compact expression for distinct roots [Bha64]. Keilson further developed the approach and presented stationary probabilities in terms of Green's function [Kei64].

For a head-of-line priority queue, Cobham presented a recursive formula for the average waiting time for each priority group [Cob54][Kle76, p.119]. Kesten *et al.* showed the Laplace

transform of the waiting time distribution [KRV57].

For a blockchain as an application, Kawase and Kasahara obtained the average waiting time for a bulk service queue by finding roots with an arbitrary service time distribution [KK17] and developed the discussion for a priority queue [KK20]. Huberman *et al.* obtained the average waiting time for a bulk service priority queue with an exponential service time distribution [HLM19]. For a transaction fee market, Houy modeled the fee market by using a linear demand curve and a quadratic supply curve of physical goods [Hou14]. Rizun analyzed the existence of the fee market regarding the block size limit [Riz16]. Basu *et al.* [BEO19] showed that the first price auction can be unstable but proposed a revised bidding method based on the second price auction.

3.3 System Model

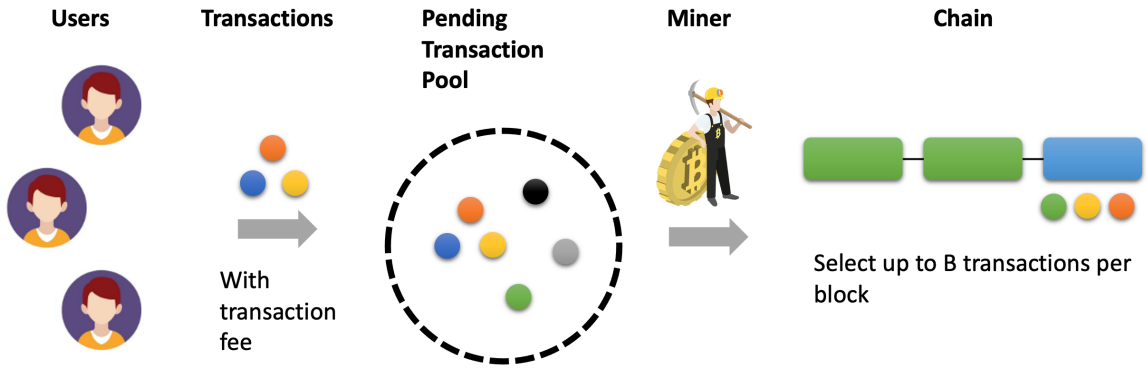

Figure 3.1: A miner chooses up to B transactions from her pending transaction pool.

Fig. 3.1 illustrates a brief overview of the transaction selection process by an individual block miner. A user submits a transaction with a fee to the peer-to-peer (P2P) network. The submitted transactions are propagated to the rest of the network. They eventually reach every miner's pending transaction pool. We assume that every miner has an identical set of pending transactions, i.e., no message is lost. We also assume that transactions

are homogeneous, i.e., the size of each transaction is constant.[1] Now, a miner chooses B transactions from her pending transaction pool for the next block. A rational miner who maximizes her mining revenue chooses the most expensive transactions for her next block.

Bulk Service Queue ($M/G^B/1$)

We assume that a miner always includes B transactions for the next block, except in the case of an occasional lack of a sufficient number of pending transactions. If a pending transaction pool contains k transactions ($0 \leq k < B$), a miner (server) includes k transactions. The service time distribution for a miner to create a block is arbitrary. The number of servers in our queueing model for blockchain mining is 1 as only one chain will survive. The arrival time (assumed to be Poisson arrivals), the service time (arbitrary), and the transaction fees are all independent random variables. That is, we use a bulk service queue model ($M/G^B/1$).

Head-of-Line Priority Queue

A transaction fee can be any real number as long as the user credit balance allows. However, we assume that there are P priority groups as we will assume a discrete set of P transaction fees. For each priority group $x = 1, \cdots, P$, the corresponding transaction fee is g_x and we assume Poisson arrivals at the mean rate of λ_x. We assume that every miner determines the priority based on the transaction fee such that $g_1 < \cdots < g_P$; note that this implies the priority increases with the group index, the P-th group being the highest priority. The discrete case (coarse-grained) can be extended to the continuous case (fine-grained) as $P \to \infty$. We also consider a non-preemptive priority queue since any new transaction does not alter or join the block mining process.

Table 3.1 summarizes the notation.

[1]In practice, the unit of block size limit is a byte. Throughout this work, the block size limit refers to a bulk size B, i.e., the maximum number of transactions in a block.

Table 3.1: Notation

Symbol	Description	Unit
P	Number of priority groups	
x	Priority group index, $x = 1, \cdots, P$	
g_x	Transaction fee used by priority group x	Ƀ
x^*	The smallest index of the group that experiences a finite average waiting time ("saturation" point)	
λ_x	Mean arrival rate for priority group x	$(\texttt{TX})/s$
λ	Mean arrival rate for the system (priority groups of interest)	$(\texttt{TX})/s$
B	Bulk size, the maximum number of transactions in a block	$(\texttt{TX})$
μ	Block rate, the mean service rate (the number of processed transactions per unit time is equal to μB)	$(\texttt{block})/s$
ρ	Utilization factor, $\rho = \frac{\lambda}{\mu B}$	
$\bar{W}_x$	Average waiting time (i.e., time in queue) for priority group x	s
$\bar{T}_x$	Average system time (waiting time plus service time) for priority group x	s
N_x	Random variable denoting the number of waiting transactions in priority group x	$(\texttt{TX})$
R_{TX}	Random variable denoting the sum of transaction fees (total transaction fee) in a block	Ƀ

3.4 Analysis

3.4.1 Average Waiting Time

3.4.1.1 Mean Queue Length

Little's result [Lit61] states the following relationship between the average number of customers $\bar{N}$ in an ergodic system, their average time spent in the system $\bar{T}$, and the mean customer arrival rate λ. We use the term "customer" and "transaction" interchangeably.

$$\bar{N} = \lambda \bar{T} \tag{3.1}$$

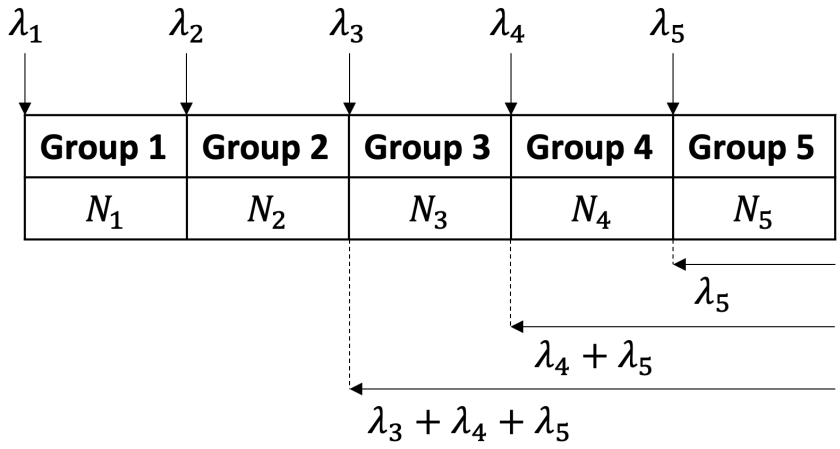

Figure 3.2: Little's result for each priority group ($P = 5$).

If we consider the average number of customers $\overline{N_{\text{queue}}}$ in a queue, by Little's result, the average waiting (i.e., queueing) time $\bar{W}$ is equal to $\overline{N_{\text{queue}}}/\lambda$.

Eq. 3.1 is also useful for a queue with priority groups. Fig. 3.2 shows an example of a queue with five priority groups ($P = 5$). We may consider each priority group a separate queue. Each priority group x has its mean arrival rate λ_x. The arrived customer spends time $\bar{T}_x$ on average in the system (queue plus service) before it departs the system. It forms

a line of customers in the priority group x whose average length is $\lambda_x \bar{T}_x$. Our goal is to find $\bar{W}_x = \bar{N}_x/\lambda_x$.

The number of waiting customers in each priority group N_x is essential to compute average waiting (and system) time. We can obtain the distribution of the number of customers in the highest priority group N_P. However, it is difficult to obtain the distribution for every N_x (except N_P) directly because of entangled dependencies. Instead, we consider the distribution of the cumulative sum of the numbers of customers as an indirect approach. For example, $N_4 + N_5$ shares the same problem structure except the different rate parameter $\lambda_4 + \lambda_5$. As shown in Eq. 3.2, we may obtain the average number of customers in the priority group 3 by solving the two instances of the queue problem.

$$\mathbb{E}[N_3] = \underbrace{\mathbb{E}[N_3 + N_4 + N_5]}_{\text{a solution for } \lambda_3+\lambda_4+\lambda_5} - \underbrace{\mathbb{E}[N_4 + N_5]}_{\text{a solution for } \lambda_4+\lambda_5} \tag{3.2}$$

Now we consider an imbedded Markov chain to model the number of customers in the queue. A state transition occurs whenever the bulk of customers departs the system. Let q_j denote the number of customers in the queue where j is the state transition instant. The server handles at most B customers (transactions) for each block service unit, with v_j equal to the new arrivals during the bulk service period. Thus, we obtain the stochastic recurrence relation as follows:[2]

$$q_{j+1} = \max(q_j - B, 0) + v_j \tag{3.3}$$

We generalize a helper function Δ_k in [Kle75, p.181] as follows:

$$\Delta_{k,B} \triangleq \begin{cases} B & \text{if } k \geq B \\ k & \text{if } k < B \end{cases} \tag{3.4}$$

The value of the helper function $\Delta_{k,B}$ means the number of customers that a server

[2]The use of the recurrence relation is known as the method of Kendall [Ken51].

handles for each service unit. The server cannot handle more than a bulk size B even if there are plenty of customers. To simplify the notation, Δ_k refers to $\Delta_{k,B}$ from now on.

Now Eq. 3.3 can be written as follows:

$$q_{j+1} = q_j - \Delta_{q_j} + v_j \tag{3.5}$$

By squaring both sides of Eq. 3.5 and applying the expectation $\mathbb{E}[\cdot]$, we have:

$$\mathbb{E}[{q_{j+1}}^2] = \mathbb{E}[{q_j}^2] + \mathbb{E}[{\Delta_{q_j}}^2] + \mathbb{E}[{v_j}^2] - 2\,\mathbb{E}[q_j \Delta_{q_j}] - 2\,\mathbb{E}[\Delta_{q_j} v_j] + 2\,\mathbb{E}[q_j v_j] \tag{3.6}$$

Now we let $\tilde{q}$ and $\tilde{v}$ be the limiting number of customers waiting in the queue and the limiting number of new arrivals during the service period, respectively.

The expected value of $\Delta_{\tilde{q}}$ in terms of $\mathbb{E}[\tilde{q}]$ is:

$$\mathbb{E}[\Delta_{\tilde{q}}] = \mathbb{E}[\tilde{q}] + \sum_{k=B}^{\infty} (B-k)\,\mathbb{P}[\tilde{q}=k] \tag{3.7}$$

The expected value of ${\Delta_{\tilde{q}}}^2$ is:

$$\mathbb{E}[{\Delta_{\tilde{q}}}^2] = \sum_{k=0}^{B-1} k^2\,\mathbb{P}[\tilde{q}=k] + B^2 \sum_{k=B}^{\infty} \mathbb{P}[\tilde{q}=k] \tag{3.8}$$

Similarly, the expected value of $\tilde{q}\Delta_{\tilde{q}}$ is:

$$\mathbb{E}[\tilde{q}\Delta_{\tilde{q}}] = \sum_{k=0}^{B-1} k^2\,\mathbb{P}[\tilde{q}=k] + B \sum_{k=B}^{\infty} k\,\mathbb{P}[\tilde{q}=k] \tag{3.9}$$

Since the number of arrivals v_j is independent of q_j, the last two terms $\mathbb{E}[\Delta_{q_j} v_j]$ and $\mathbb{E}[q_j v_j]$ are the products of expectations, $\mathbb{E}[\Delta_{q_j}]\,\mathbb{E}[v_j]$ and $\mathbb{E}[q_j]\,\mathbb{E}[v_j]$, respectively. By taking the above equations together, we obtain:

$$2(B - \mathbb{E}[\tilde{v}])\,\mathbb{E}[\tilde{q}] = \mathbb{E}[\tilde{v}^2] + B^2 - 2B\,\mathbb{E}[\tilde{v}] + 2\,\mathbb{E}[\tilde{v}] \sum_{k=0}^{B-1} (B-k)\,\mathbb{P}[\tilde{q}=k] - \sum_{k=0}^{B-1} (B-k)^2\,\mathbb{P}[\tilde{q}=k] \tag{3.10}$$

If $\mathbb{E}[\tilde{v}] \neq B$, the mean queue length $\mathbb{E}[\tilde{q}]$ is:

$$\mathbb{E}[\tilde{q}] = \frac{\mathbb{E}[\tilde{v}^2] + B^2 - 2B\,\mathbb{E}[\tilde{v}] + 2\,\mathbb{E}[\tilde{v}] \sum_{k=0}^{B-1} (B-k)\,\mathbb{P}[\tilde{q}=k] - \sum_{k=0}^{B-1} (B-k)^2\,\mathbb{P}[\tilde{q}=k]}{2(B - \mathbb{E}[\tilde{v}])} \tag{3.11}$$

We further simplify the numerator part:

$$\mathbb{E}[\tilde{q}] = \frac{\mathbb{E}[(B-\tilde{v})^2] + \sum_{k=0}^{B-1} (B-k)(2\,\mathbb{E}[\tilde{v}] - (B-k))\,\mathbb{P}[\tilde{q}=k]}{2(B - \mathbb{E}[\tilde{v}])} \tag{3.12}$$

For a bulk service queue ($B > 1$), we first need to find B stationary probabilities, $\mathbb{P}[\tilde{q}=k]$ where $k = 0, 1, \cdots, B-1$.

Unstable Conditions

Note that we expect an infinite number of customers when the average number of arrivals $\mathbb{E}[\tilde{v}]$ is equal to B. Further, if $\mathbb{E}[\tilde{v}] > B$, (i.e., again when the system is unstable), the denominator of Eq. 3.12 is a negative number. The numerator of Eq. 3.12 is still a positive number (assuming that $\mathbb{P}[\tilde{q}=k]$ is a positive number). The contradiction is due to the ergodicity assumption. Recall that the state transition occurs at the mean rate of μ. The mean arrival rate λ is now $\mu\,\mathbb{E}[\tilde{v}]$. With the unstable condition $\mathbb{E}[\tilde{v}] \geq B$, we get:

$$\rho = \frac{\lambda}{\mu B} = \frac{\mu\,\mathbb{E}[\tilde{v}]}{\mu B} \geq 1 \tag{3.13}$$

For a priority queue, we may consider groups from x to P. The mean arrival rate for the groups of interest is $\sum_{i=x}^{P} \lambda_i$. The saturation point x^*, which is the smallest index for the group that experiences a finite average waiting time, can be obtained by $\sum_{i=x^*}^{P} \lambda_i < \mu B$.

Special Case ($B = 1$)

Eq. 3.12 is solvable for a single service queue, $B = 1$. We may reduce it to:

$$\mathbb{E}[\tilde{q}] = \frac{\mathbb{E}[\tilde{v}^2] + 1 - 2\,\mathbb{E}[\tilde{v}](1 - \mathbb{P}[\tilde{q}=0]) - \mathbb{P}[\tilde{q}=0]}{2(1 - \mathbb{E}[\tilde{v}])} \tag{3.14}$$

where $\mathbb{E}[\tilde{v}] \neq 1$. Note that $\mathbb{E}[\tilde{v}]$ is equal to ρ (the usual utilization factor in queueing theory). Now $\mathbb{P}[\tilde{q}=0]$ stands for the probability that the queue is idle, or $1-\rho$. Thus, we may further

reduce Eq. 3.14 to:

$$\mathbb{E}[\tilde{q}] = \frac{\mathbb{E}[\tilde{v}^2] + 1 - 2\rho^2 - (1-\rho)}{2(1-\rho)} = \rho + \frac{\mathbb{E}[\tilde{v}^2] - \mathbb{E}[\tilde{v}]}{2(1-\rho)} \tag{3.15}$$

where ρ is the utilization factor. This result is well known in queueing theory.

3.4.1.2 z-Transform

Let us return to the bulk service queue with a bulk size B. A unilateral z-transform is useful to solve Eq. 3.5 as we now show. This method results in a probability generating function for the distribution of the number of customers in a queue.

For a random variable q_j, we apply the unilateral z-transform.

$$Q_j(z) = \sum_{k=0}^{\infty} \mathbb{P}[q_j = k] z^k = \mathbb{E}[z^{q_j}] \tag{3.16}$$

Let us apply expectations for Eq. 3.5.

$$\mathbb{E}[z^{q_{j+1}}] = \mathbb{E}[z^{q_j - \Delta_{q_j}}]\, \mathbb{E}[z^{v_j}] \tag{3.17}$$

Likewise, we apply the unilateral z-transform for a random variable v_j.

$$V_j(z) = \sum_{k=0}^{\infty} \mathbb{P}[v_j = k] z^k = \mathbb{E}[z^{v_j}] \tag{3.18}$$

We obtain the probability generating function.

$$Q_{j+1}(z) = \mathbb{E}[z^{q_j - \Delta_{q_j}}] V_j(z) \tag{3.19}$$

For the limiting random variable $\tilde{q}$ and $\tilde{v}$:

$$Q(z) = \mathbb{E}[z^{\tilde{q} - \Delta_{\tilde{q}}}] V(z) \tag{3.20}$$

The expectation term in Eq. 3.20 is:

$$\mathbb{E}[z^{\tilde{q} - \Delta_{\tilde{q}}}] = z^{-B} \left(Q(z) + \sum_{k=0}^{B-1} \mathbb{P}[\tilde{q} = k](z^B - z^k) \right) \tag{3.21}$$

By rearranging the equation, we obtain:

$$Q(z)\left(z^B - V(z)\right) = \sum_{k=0}^{B-1} \mathbb{P}[\tilde{q} = k](z^B - z^k)V(z) \tag{3.22}$$

The probability generating function for the number of customers in a queue is thus:

$$Q(z) = \frac{V(z)\sum_{k=0}^{B-1} \mathbb{P}[\tilde{q} = k](z^B - z^k)}{z^B - V(z)} \tag{3.23}$$

Now the mean queue length $\mathbb{E}[\tilde{q}]$ can be obtained from the probability generating function $Q(z)$ by taking the first order derivative with $z = 1$.

$$\mathbb{E}[\tilde{q}] = \left[\frac{dQ(z)}{dz}\right]_{z=1} \tag{3.24}$$

We know that $z = 1$ is one of B zeros for the denominator of $Q(z)$. For some cases, one may explicitly find the remaining $B - 1$ zeros and determine the unknown stationary probabilities. In general, using z-transforms does not eliminate the necessity to find the $B - 1$ zeros.

3.4.1.3 Hilbert Transform

We now propose a numerical method to compute the mean queue length without evaluating the stationary probabilities.

Let us once again consider the recurrence relation:

$$q_{j+1} = q_j - \Delta_{q_j} + v_j = \max(q_j - B, 0) + v_j \tag{3.25}$$

The initial number of customers is zero, i.e., $q_0 = 0$. At round 1, we have $q_1 = v_0$. At round 2, we have $q_2 = \max(v_0 - B, 0) + v_1$. At round 3, we have $q_3 = \max(\max(v_0 - B, 0) + v_1 - B, 0) + v_2$. The nested max functions can be rearranged as follows:

$$q_3 = \max\left(v_0 - B + v_1 - B, v_1 - B, 0\right) + v_2 \tag{3.26}$$

In general, the recurrence relation of the number of customers at round j can be expressed:

$$q_j = \max\left(\max_{0\le i\le j-2}\left(\sum_{k=i}^{j-2}(v_k - B)\right), 0\right) + v_{j-1} \tag{3.27}$$

Now let us define an auxiliary variable s_k:

$$s_k = \sum_{i=0}^{k-1}(v_i - B) \tag{3.28}$$

Since the number of arrivals is an independent process, we may ignore the subscript index and only consider the number of additive terms.

$$q_j = \max\left(\max_{0\le i\le j-2}(s_{i+1}), 0\right) + v_{j-1} \tag{3.29}$$

The outermost max function can be applied to each term.

$$q_j = \max_{0\le i\le j-2}(\max(s_{i+1}, 0)) + v_{j-1} \tag{3.30}$$

To simplify the notation, we define $x^+ \triangleq \max(x, 0)$ and we have:

$$q_j = \max_{0\le i\le j-2}\left(s_{i+1}^+\right) + v_{j-1} = \max_{1\le k\le j-1}\left(s_k^+\right) + v_{j-1} \tag{3.31}$$

Now we may use Spitzer's identity (Corollary 1) [Spi56]:

$$\mathbb{E}[\max(s_1^+, s_2^+, \cdots, s_n^+)] = \sum_{k=1}^{n}\frac{\mathbb{E}[s_k^+]}{k} \tag{3.32}$$

Therefore, the mean queue length is:

$$\mathbb{E}[\tilde{q}] = \lim_{j\to\infty}\mathbb{E}[q_j] = \lim_{j\to\infty}\left(\sum_{k=1}^{j-1}\frac{\mathbb{E}[s_k^+]}{k} + \mathbb{E}[v_{j-1}]\right) = \sum_{k=1}^{\infty}\frac{\mathbb{E}[s_k^+]}{k} + \mathbb{E}[\tilde{v}] \tag{3.33}$$

Expectation of the positive part of random variable X

Now we consider the expectation of $X^+ \triangleq \max(X, 0)$.

$$\mathbb{E}[X^+] = \sum_{k=1}^{\infty} k\,\mathbb{P}[X = k] \tag{3.34}$$

From the definition of the expectation, we have:

$$\mathbb{E}[X] = \sum_{k=-\infty}^{\infty} k\,\mathbb{P}[X=k] = \sum_{k=-\infty}^{-1} k\,\mathbb{P}[X=k] + \sum_{k=1}^{\infty} k\,\mathbb{P}[X=k] \tag{3.35}$$

For the positive part of random variable X:

$$\mathbb{E}[X^+] = \sum_{k=1}^{\infty} k\,\mathbb{P}[X=k] = \frac{1}{2}\left(\mathbb{E}[X] - \sum_{k=-\infty}^{-1} k\,\mathbb{P}[X=k] + \sum_{k=1}^{\infty} k\,\mathbb{P}[X=k]\right) \tag{3.36}$$

Now we want to represent the above series in terms of a characteristic function. In general, the characteristic function of the positive part of a random variable is studied in [Pin15, Pin18].

Characteristic Function

In a probability theory, $\phi_X(t)$, a characteristic function of a random variable completely defines its probability distribution:

$$\phi_X(t) \triangleq \mathbb{E}[e^{itX}] = \sum_{k=-\infty}^{\infty} e^{itk}\,\mathbb{P}[X=k] \tag{3.37}$$

where $i = \sqrt{-1}$.

Now we use Euler's formula

$$e^{ix} = \cos x + i \sin x \tag{3.38}$$

and take a first order derivative of $\phi_X(t)$ as follows:

$$\phi'_X(t) = \sum_{k=-\infty}^{\infty} ik(\cos tk + i \sin tk)\,\mathbb{P}[X=k] \tag{3.39}$$

We compute $\phi'_X(-t)$ and find that the sign of the sine term is the opposite.

$$\phi'_X(-t) = \sum_{k=-\infty}^{\infty} ik(\cos tk - i \sin tk)\,\mathbb{P}[X=k] \tag{3.40}$$

Then we consider their difference divided by t:

$$\frac{\phi'_X(t) - \phi'_X(-t)}{t} = -2 \sum_{k=-\infty}^{\infty} \frac{k \sin tk}{t}\,\mathbb{P}[X=k] \tag{3.41}$$

By integrating from 0^+ to ∞ and assuming that we may change the order of integration and summation:

$$\begin{aligned}\int_{0^+}^{\infty} \frac{\phi'_X(t) - \phi'_X(-t)}{t} dt &= \int_{0^+}^{\infty} -2 \sum_{k=-\infty}^{\infty} \frac{k \sin tk}{t} \mathbb{P}[X = k] dt \\ &= -2 \sum_{k=-\infty}^{\infty} \int_{0^+}^{\infty} \frac{k \sin tk}{t} dt \, \mathbb{P}[X = k]\end{aligned} \tag{3.42}$$

From the integration of a `sinc` function [Wei]:

$$\int_{-\infty}^{\infty} \texttt{sinc}(x) dx = \pi \tag{3.43}$$

where

$$\texttt{sinc}(x) \triangleq \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin x}{x} & \text{otherwise} \end{cases} \tag{3.44}$$

Since the integrand is symmetric, we obtain:

$$\int_{0^+}^{\infty} \frac{k \sin tk}{t} dt = \begin{cases} \frac{k\pi}{2} & \text{if } k > 0 \\ 0 & \text{if } k = 0 \\ \frac{-k\pi}{2} & \text{if } k < 0 \end{cases} \tag{3.45}$$

Now we have the following equation:

$$\int_{0^+}^{\infty} \frac{\phi'_X(t) - \phi'_X(-t)}{t} dt = -\pi \left(\sum_{k=1}^{\infty} k \, \mathbb{P}[X = k] - \sum_{k=-\infty}^{-1} k \, \mathbb{P}[X = k] \right) \tag{3.46}$$

By rearranging Eq. 3.46,

$$\sum_{k=1}^{\infty} k \, \mathbb{P}[X = k] - \sum_{k=-\infty}^{-1} k \, \mathbb{P}[X = k] = -\frac{1}{\pi} \int_{0^+}^{\infty} \frac{\phi'_X(t) - \phi'_X(-t)}{t} dt \tag{3.47}$$

The expectation of X can be computed from the characteristic function.

$$\mathbb{E}[X] = \phi'_X(0)/i = \sum_{k=-\infty}^{\infty} ike^{i \cdot 0 \cdot k} \, \mathbb{P}[X = k]/i = \sum_{k=-\infty}^{\infty} k \, \mathbb{P}[X = k] \tag{3.48}$$

By combining Eq. 3.48 and the above result,

$$\mathbb{E}[X^+] = \sum_{k=1}^{\infty} k\,\mathbb{P}[X=k] = \frac{1}{2}\left(\frac{\phi'_X(0)}{i} - \frac{1}{\pi}\int_{0^+}^{\infty}\frac{\phi'_X(t)-\phi'_X(-t)}{t}dt\right) \tag{3.49}$$

The integral form $-\frac{1}{\pi}\int_{0^+}^{\infty}\frac{\phi'_X(t)-\phi'_X(-t)}{t}dt$ can be represented by the Hilbert transform [Ksc06]. The Hilbert transform is defined as:

$$H(f)(t) \triangleq -\frac{1}{\pi}\int_{0^+}^{\infty}\frac{f(t+\tau)-f(t-\tau)}{\tau}d\tau \tag{3.50}$$

Now we have:

$$\mathbb{E}[X^+] = \frac{1}{2}\left(\frac{\phi'_X(0)}{i} + H(\phi'_X)(0)\right) \tag{3.51}$$

Going back to the original discussion, we have the formula for the mean queue length for a bulk service queue ($M/G^B/1$) as follows:

$$\begin{aligned}\mathbb{E}[\tilde{q}] &= \sum_{k=1}^{\infty}\frac{\mathbb{E}[s_k^+]}{k} + \mathbb{E}[\tilde{v}] \\ &= \sum_{k=1}^{\infty}\left(\frac{1}{2k}\left(\frac{\phi'_{S_k}(0)}{i} - \frac{1}{\pi}\int_{0^+}^{\infty}\frac{\phi'_{S_k}(t)-\phi'_{S_k}(-t)}{t}dt\right)\right) + \frac{\phi'_V(0)}{i}\end{aligned} \tag{3.52}$$

The auxiliary random variable S_k is the sum of the number of arrivals with B deductions for every service unit (k times).

The characteristic function of the shifted random variable $X' = X - B$ is:

$$\phi_{X'}(t) = \mathbb{E}[e^{it(X-B)}] = \mathbb{E}[e^{itX}]e^{-itB} = \phi_X(t)e^{-itB} \tag{3.53}$$

The characteristic function of the sum of independent random variables is the product of their characteristic functions. Therefore, we obtain the characteristic function for the random variable S_k in terms of k times convolution:

$$\phi_{S_k}(t) = \left(\phi_V(t)e^{-itB}\right)^k = \phi_V^k(t)e^{-itkB} \tag{3.54}$$

We now have everything to compute the mean queue length for a bulk service queue ($M/G^B/1$). Note that the use of the Hilbert transform does not require B stationary probabilities. All terms are eventually real numbers. However, we need to deal with the two infinite ranges. Let us consider the finite ranges.

$$\mathbb{E}[\tilde{q}] \approx \sum_{k=1}^{k_{max}} \left(\frac{1}{2k} \left(\frac{\phi'_{S_k}(0)}{i} - \frac{1}{\pi} \int_{0^+}^{a_k} \frac{\phi'_{S_k}(t) - \phi'_{S_k}(-t)}{t} dt \right) \right) + \frac{\phi'_V(0)}{i} \tag{3.55}$$

In practice, we choose suitable k_{max} and a_k for approximation and stop the iteration loop if the absolute value of an additional term is less than the threshold (arbitrarily set at 10^{-3}).

For a priority queue with groups $(1, \cdots, P)$, we use the total arrival rate of the groups of interest $\lambda = \sum_{i=x}^{P} \lambda_i$. Let $\mathbb{E}[\tilde{q}; \lambda]$ denote the mean queue length at a service instant for a bulk service queue with the arrival rate λ. For priority group x, we obtain:

$$\mathbb{E}[\tilde{q}_x] = \mathbb{E}\left[\tilde{q}; \sum_{i=x}^{P} \lambda_i\right] - \mathbb{E}\left[\tilde{q}; \sum_{i=x+1}^{P} \lambda_i\right] \tag{3.56}$$

Since Eq. 3.56 is defined for each service instant, the sum of the number of waiting transactions in priority group x and the number of transactions being served in priority group x is

$$\mathbb{E}[\tilde{q}_x] + \lambda_x W_0 \tag{3.57}$$

where W_0 is the average remaining service time.

We apply Little's result to get the average system time for priority group x.

$$\bar{T_x} = \frac{\mathbb{E}[\tilde{q}_x]}{\lambda_x} + W_0 \tag{3.58}$$

Finally, we obtain the average waiting time as follows:

$$\bar{W_x} = \frac{\mathbb{E}[\tilde{q}_x]}{\lambda_x} + W_0 - \bar{T} \tag{3.59}$$

where $\bar{T}$ is the average service time.

Now we compare the average waiting time obtained from a simulation and the truncated computation using Eq. 3.55. The queue simulation code and the numerical integration

code are available in Appendix D. As examples, we consider three arbitrary service time distributions.

- Exponential service time distribution ($\mu' = \frac{1}{600s}$)
- Shifted exponential service time distribution ($\mu' = \frac{1}{300s}, d = 300s$)
- Deterministic service time distribution ($\mu' = \frac{1}{600s}$).

The average service time for each distribution is 600 seconds.[3] As common parameters, the bulk size B is 10, the utilization factor $\rho = \frac{\lambda}{\mu B}$ is 1.15, and the number of priority groups P is 100. However, there are no specific implications regarding the parameter choices. The queue is slightly saturated. For convenience, we assume a uniform distribution of arrival rates for each group x, $\lambda_x = \lambda/P$. The saturation point x^* is 15 as $x^* > P(1 - 1/\rho) + 1 \approx 14.04$. Finally, we need to obtain the characteristic function $\phi_V(t)$ of the number of arrivals for each service unit. The detailed derivations for Eq. 3.60, Eq. 3.61 and Eq. 3.62 are available in Appendix C.

For the exponential distribution:

$$\phi_V(t) = \frac{1}{\frac{\lambda}{\mu}(1 - e^{it}) + 1} \tag{3.60}$$

For the shifted exponential distribution:

$$\begin{aligned} \phi_V(t) &= \sum_{k=0}^{\infty} \left(e^{itk} \frac{\mu}{\lambda + \mu} \frac{1}{e^{\lambda d}} \left(\frac{\lambda}{\lambda + \mu} \right)^k \sum_{m=0}^{k} \frac{(\lambda d + \mu d)^m}{m!} \right) \\ &= \frac{e^{-\lambda d(1 - e^{it})}}{\frac{\lambda}{\mu}(1 - e^{it}) + 1} \end{aligned} \tag{3.61}$$

For the deterministic service time distribution:

$$\phi_V(t) = e^{-\frac{\lambda}{\mu}(1 - e^{it})} \tag{3.62}$$

[3]The distribution of block creation time in Bitcoin is close to the exponential distribution with rate parameter $\frac{1}{600s}$. If there exists a processing delay, one may consider the shifted exponential distribution. If a consensus algorithm uses constant block creation time, one may consider the deterministic service time distribution.

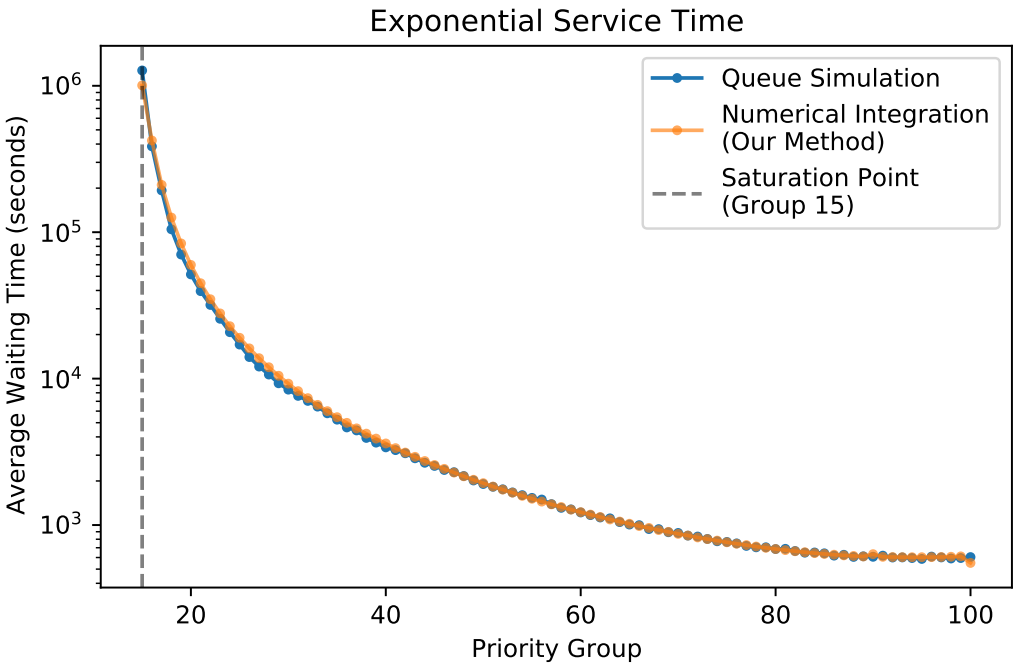

Figure 3.3: Average waiting time for exponential service time ($\mu' = \frac{1}{600s}$).

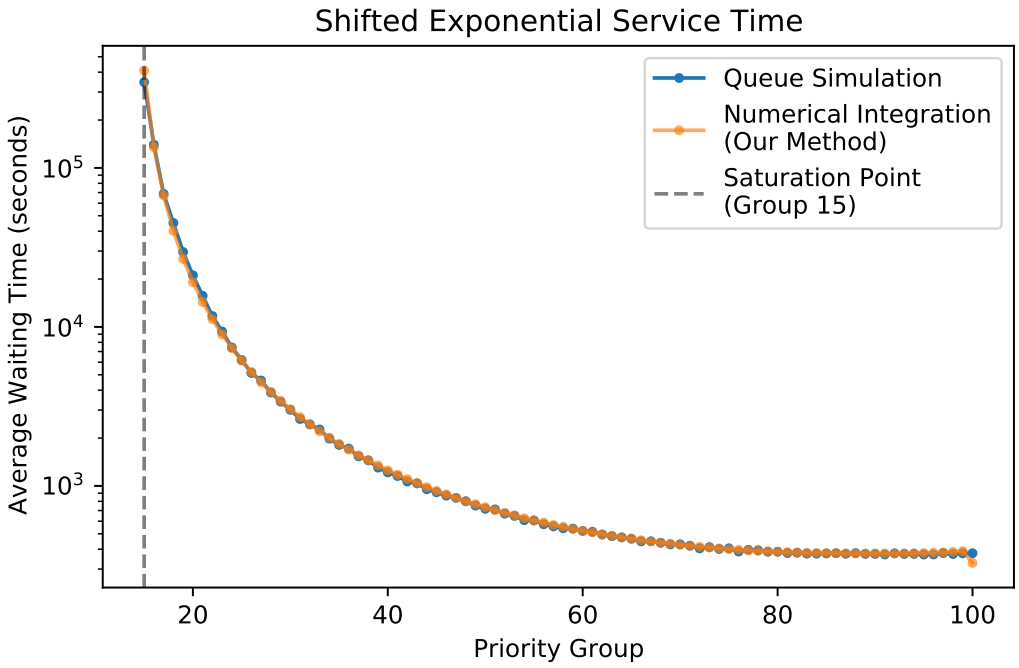

Figure 3.4: Average waiting time for shifted exponential service time ($\mu' = \frac{1}{300s}, d = 300s$).

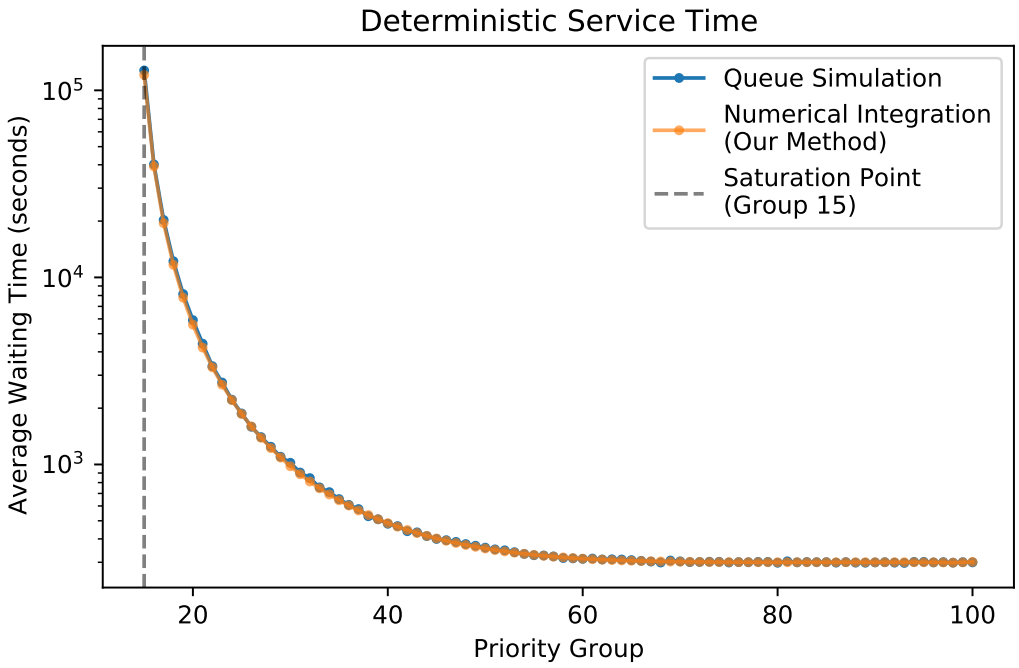

Figure 3.5: Average waiting time for deterministic service time ($\mu' = \frac{1}{600s}$).

3.4.2 Minimum Transaction Fee

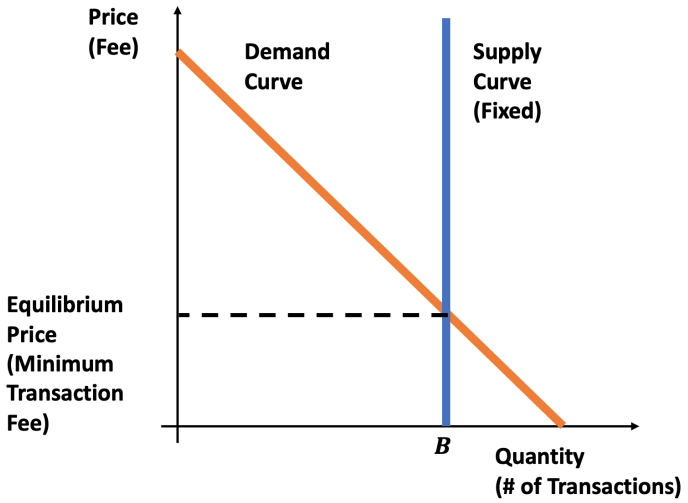

Figure 3.6: Supply and demand curves in a transaction fee market.

Due to the block size limit, a miner cannot include more than B transactions in a block. Fig. 3.6 shows the supply and demand curves in a transaction fee market. The fixed supply curve, $Q = B$, represents the constraint. Though a miner may change her preferred block size, we assume that every miner always includes B transactions in a block if enough transactions are available. The descending demand curve (shown arbitrarily as a straight line as an example) represents the general tendency of users who do not want to submit transactions with expensive fees.

Now we focus on the equilibrium price, the intersection point of the supply and demand curves. The equilibrium price goes higher for the following cases. When a miner prefers to create a block with fewer transactions, the supply curve (the vertical line) shifts to the left. When users need to pay more transaction fees, the demand curve shifts to the right.

We are interested in finding the location of the equilibrium price. The minimum transaction fee in B transactions included by a miner is the same as the equilibrium price. Since a fee distribution may fluctuate over time, it is not convenient to deal with absolute values.

Instead, we use a priority group index $x = 1, \cdots, P$. Suppose that group x contains the cheapest transaction fee. Then the equilibrium price is g_x. The group index can be understood as the percentile $\approx (1 - x/P) \times 100(\%)$, especially if we have fine-grained priority groups with a uniform distribution of arrival rates. It also allows us to obtain the lower bound of the sum of transaction fees ($R_{\text{TX}} > g_x B$) for a fully filled block.

Definition 3.1. *(Minimum Group Index) The minimum group index $k = 0, 1, \cdots, P$ refers to priority group k containing the cheapest transaction fee among B transactions included by a miner. The special case $k = 0$ refers to a partially filled block due to an insufficient number of transactions ($< B$).*

Let X denote a random variable for the minimum group index k. Let N_i denote the number of transactions in priority group i. The event $X \leq k$ is equivalent to the event $\sum_{i=k+1}^{P} N_i < B$. Note that the special case $X \leq 0$ ($X = 0$) implies an insufficient number of transactions, $\sum_{i=1}^{P} N_i < B$. In this case, a miner creates a partially filled block. The event $X \leq P$ always happens.

We now derive the distribution of the minimum group index X, $\mathbb{P}[X = k]$. This probability can be computed as follows:

$$\mathbb{P}[X = k] = \begin{cases} \mathbb{P}[X \leq k] - \mathbb{P}[X \leq k-1] & \text{if } k = 1, \cdots, P \\ \mathbb{P}\left[\sum_{i=1}^{P} N_i < B\right] & \text{if } k = 0 \end{cases} \tag{3.63}$$

Note that the probability $\mathbb{P}[X \leq k] = \mathbb{P}\left[\sum_{i=k+1}^{P} N_i < B\right] = \mathbb{P}\left[N_{k+1} + \cdots + N_P < B\right]$ can be obtained from the distribution of customers in a system. Likewise, the probability $\mathbb{P}[X \leq k-1]$ can be obtained from the same process with a different arrival rate.

For an $M/M^B/1$ queue, the equilibrium difference equations yield the distribution of customers in a system [Kle75, p.138]. The details are available in Appendix A. The stationary

probability that j customers are waiting in the queue is:

$$p_j = \left(1 - \frac{1}{z_0}\right)\left(\frac{1}{z_0}\right)^j \tag{3.64}$$

We numerically solve z_0 for given a utilization factor ρ and a bulk size B:

$$B\rho z_0^{B+1} - (1 + B\rho)z_0^B + 1 = 0 \tag{3.65}$$

where $z_0 > 1$.[4]

Now we consider the total arrival rate for priority groups $k, \cdots, P$, which is equal to $\sum_{i=k}^{P} \lambda_i$. The utilization factor ρ_k is $\sum_{i=k}^{P} \lambda_i/(\mu B)$. We find $z_{0,k}$ such that $B\rho_k z_{0,k}^{B+1} - (1 + B\rho_k)z_{0,k}^B + 1 = 0$ and $z_{0,k} > 1$. We may then evaluate the probability of the block having a total number of transactions less than B.

$$\mathbb{P}[N_k + \cdots + N_P < B] = \left(1 - \frac{1}{z_{0,k}}\right)\sum_{m=0}^{B-1}\left(\frac{1}{z_{0,k}}\right)^m = 1 - \left(\frac{1}{z_{0,k}}\right)^B \tag{3.66}$$

Similarly, we find $z_{0,k+1}$ using the same equation with a utilization factor ρ_{k+1}. Therefore, we obtain the distribution of the minimum group index X for an $M/M^B/1$ queue as follows:

$$\begin{aligned}
\mathbb{P}[X = k] &= \mathbb{P}[X \le k] - \mathbb{P}[X \le k-1] \\
&= \mathbb{P}[N_{k+1} + \cdots + N_P < B] - \mathbb{P}[N_k + \cdots + N_P < B] \\
&= \left(1 - \left(\frac{1}{z_{0,k+1}}\right)^B\right) - \left(1 - \left(\frac{1}{z_{0,k}}\right)^B\right) \\
&= \left(\frac{1}{z_{0,k}}\right)^B - \left(\frac{1}{z_{0,k+1}}\right)^B \\
&= B\rho_k(1 - z_{0,k}) - B\rho_{k+1}(1 - z_{0,k+1})
\end{aligned} \tag{3.67}$$

Now we compare the analytical result and the simulation result for the distribution of the minimum group index X. As an example, we consider two different cases: (1) slightly overloaded queue ($\rho = 1.15$) and (2) underloaded queue ($\rho = 0.85$). The service time

[4]When $B = 1$, we immediately know that $z_0 = 1/\rho$ from a quadratic equation $\rho z_0^2 - (1 + \rho)z_0 + 1 = (\rho z_0 - 1)(z_0 - 1) = 0$

distribution is exponential with the rate $\mu = \frac{1}{600s}$, the bulk size B is 100, and the number of priority groups P is 100. The simulation length is $100,000 \times 600s$. There are no specific implications regarding the parameter choices.

Both Fig. 3.7 and 3.8 show the nice agreement. For the overloaded queue, the saturation point x^* is 15. For the underloaded queue, it is frequent to observe partially filled blocks. The probability of $X = 0$ (not shown in Fig. 3.8) is 0.2967 (simulation) and 0.3022 (analytical).

Finally, given an (empirical) fee distribution g_x, we immediately obtain the distribution of the minimum transaction fee g_X from the distribution of the minimum group index (see Eq. 3.63).

Table 3.2: Distribution of the minimum transaction fee

Weight	Probability
$g_0 \triangleq 0$	$\mathbb{P}[X = 0]$
g_1	$\mathbb{P}[X = 1]$
g_2	$\mathbb{P}[X = 2]$
...	
g_{P-1}	$\mathbb{P}[X = P - 1]$
g_P	$\mathbb{P}[X = P]$

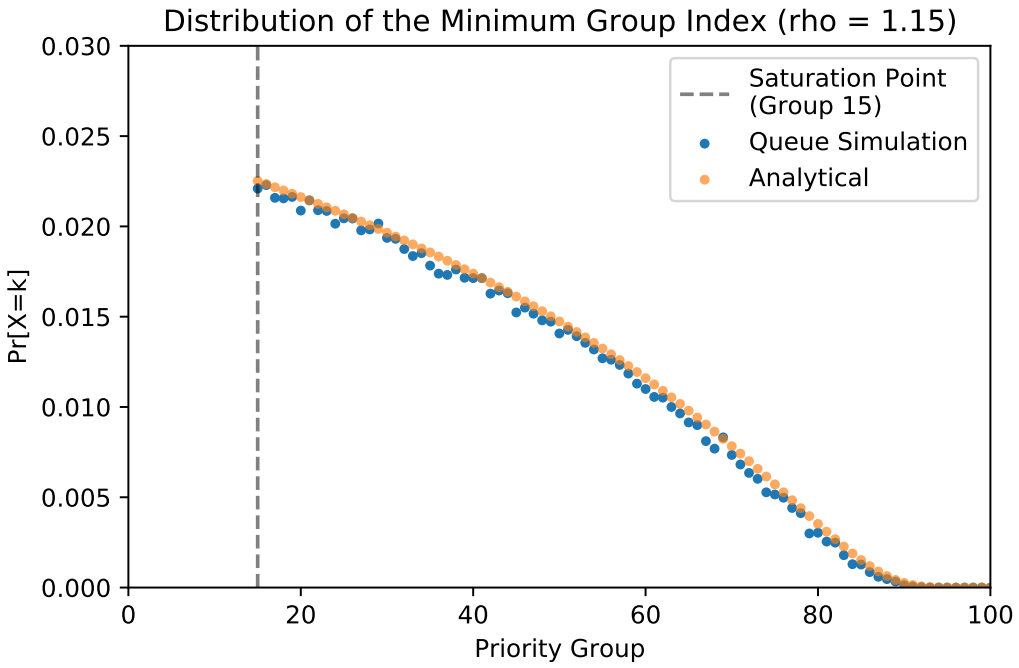

Figure 3.7: Distribution of the minimum group index ($\rho = 1.15$).

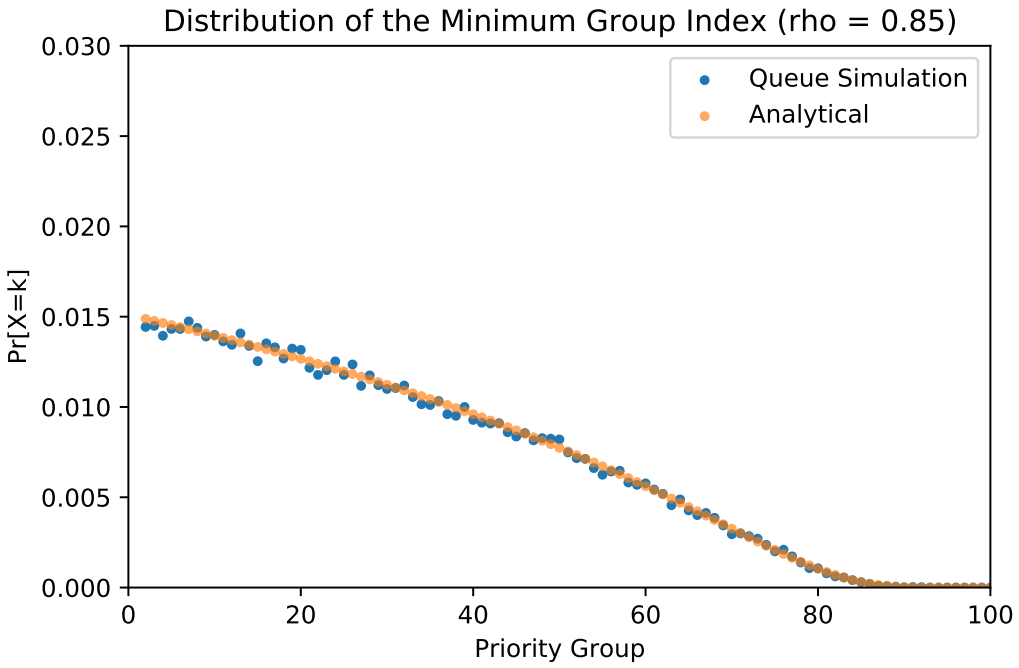

Figure 3.8: Distribution of the minimum group index ($\rho = 0.85$).

3.4.3 Average Total Transaction Fee

Now we consider the total transaction fee R_{TX} collected by a miner. Let N_i' be the number of transactions in priority group i processed by a miner.

$$R_{\mathrm{TX}} = \sum_{i=1}^{P} g_i N_i' \tag{3.68}$$

Getting the distribution of R_{TX} remains an open question as one needs to solve the distribution of the number of transactions in each priority group. The product of the minimum transaction fee g_k and a bulk size B is a lower bound of R_{TX} for a fully filled block ($k \neq 0$).

$$g_k B \leq R_{\mathrm{TX}} \tag{3.69}$$

We find that the law of total expectation yields the expectation of R_{TX}. Recall the expected value of the conditional expected value of A given B is the same as the expected value of A.

$$\mathbb{E}[R_{\mathrm{TX}}] = \mathbb{E}[\mathbb{E}[R_{\mathrm{TX}}|X]] = \sum_{k=0}^{P} \mathbb{E}[R_{\mathrm{TX}}|X = k]\, \mathbb{P}[X = k] \tag{3.70}$$

where X is a random variable denoting the minimum group index.

The minimum group index $X = k$ gives detailed information about the number of transactions.

- When $k = P \Rightarrow N_P \geq B$,
- When $k = 1, \cdots, P-1 \Rightarrow N_{k+1} + \cdots + N_P < B$ and $N_k + \cdots + N_P \geq B$,
- When $k = 0 \Rightarrow N_1 + \cdots + N_P < B$.

For example, $k = P$, a miner selects B transactions from the highest priority group P. When $k = 0$, a miner selects all existing transactions to fill the block (but partially). When

$k = 1, \cdots, P-1$, a miner selects all transactions from priority groups $k+1, \cdots, P$ and $B - \sum_{i=k+1}^{P} N_i$ transactions from priority group k. Thus, the number of transactions in priority group i processed by a miner is:

$$N_i' = \begin{cases} \min\left(N_k, B - \sum_{i=k+1}^{P} N_i\right) & \text{if } i = k \\ N_i & \text{if } i = k+1, \cdots, P \end{cases} \tag{3.71}$$

By combining Eq. 3.68, Eq. 3.70 and Eq. 3.71, we have:

$$\begin{aligned} \mathbb{E}[R_{\mathrm{TX}}] = \mathbb{E}\left[\sum_{i=1}^{P} g_i N_i | X = 0\right] \mathbb{P}[X = 0] \\ + \sum_{k=1}^{P} \left(\mathbb{E}\left[g_k B + \sum_{i=k+1}^{P} (g_i - g_k) N_i | X = k \right] \mathbb{P}[X = k] \right) \end{aligned} \tag{3.72}$$

After a lengthy rearrangement, we obtain the following equation.[5]

$$\begin{aligned} \mathbb{E}\left[R_{\mathrm{TX}}\right] = B \left(\sum_{k=1}^{P} g_k \, \mathbb{P}\left[X = k\right] \right) \\ + g_1 \, \mathbb{E}\left[\sum_{k=1}^{P} N_k | X = 0\right] \mathbb{P}\left[X = 0\right] \\ + \sum_{i=2}^{P} \left((g_i - g_{i-1}) \sum_{j=0}^{i-1} \left(\mathbb{E}\left[\sum_{k=i}^{P} N_k | X = j\right] \mathbb{P}\left[X = j\right] \right) \right) \end{aligned} \tag{3.73}$$

To simplify the equation, we define an auxiliary random variable as follows:

$$M_i \triangleq \sum_{k=i}^{P} N_k \tag{3.74}$$

[5]The detailed derivation is available in Appendix B.

$$\begin{aligned}\mathbb{E}\left[R_{\mathrm{TX}}\right] = B &\left(\sum_{k=1}^{P} g_k \,\mathbb{P}\left[X=k\right]\right) \\ &+ g_1 \,\mathbb{E}\left[M_1 | X=0\right]\mathbb{P}\left[X=0\right] \\ &+ \sum_{i=2}^{P}\left((g_i - g_{i-1})\sum_{j=0}^{i-1}\left(\mathbb{E}\left[M_i|X=j\right]\mathbb{P}\left[X=j\right]\right)\right)\end{aligned} \tag{3.75}$$

Note that the event $X = 0$ (partially filled block) is equivalent to $M_1 = \sum_{k=1}^{P} N_k < B$. Now we can merge mutually exclusive events as follows:

$$\sum_{j=0}^{i-1}\mathbb{E}\left[M_i|X=j\right]\mathbb{P}\left[X=j\right] = \mathbb{E}\left[M_i|X<i\right]\mathbb{P}\left[X<i\right] \tag{3.76}$$

In general, the event $X < i$ is equivalent to the event $M_i < B$. We then rewrite the equation in terms of M_i:

$$\begin{aligned}\mathbb{E}\left[R_{\mathrm{TX}}\right] = B &\left(\sum_{k=1}^{P} g_k \,\mathbb{P}\left[X=k\right]\right) \\ &+ g_1 \sum_{k=1}^{B-1} k\,\mathbb{P}\left[M_1=k, M_1<B\right] \\ &+ \sum_{i=2}^{P}\left((g_i - g_{i-1})\sum_{k=1}^{B-1} k\,\mathbb{P}\left[M_i=k, M_i<B\right]\right)\end{aligned} \tag{3.77}$$

Finally, we obtain the formula for the average total transaction fee. To simplify the equation, we define $g_0 \triangleq 0$.[6]

$$\mathbb{E}\left[R_{\mathrm{TX}}\right] = B\left(\sum_{k=1}^{P} g_k \,\mathbb{P}\left[X=k\right]\right) + \sum_{i=1}^{P}\left((g_i - g_{i-1})\sum_{k=1}^{B-1} k\,\mathbb{P}\left[M_i=k\right]\right) \tag{3.78}$$

For an $M/M^B/1$ queue, we may use the following information.

$$\mathbb{P}\left[X=k\right] = B\rho_k\left(1-z_{0,k}\right) - B\rho_{k+1}\left(1-z_{0,k+1}\right) \tag{3.79}$$

[6]This definition also means that the minimum transaction fee for a partially filled block is 0.

For the partial sum, we have:[7]

$$\sum_{k=1}^{B-1} k\,\mathbb{P}\left[M_i = k\right] = B^2\rho_i\left(z_{0,i} - 1\right) + B\left(\rho_i - 1\right) \tag{3.80}$$

Then we obtain the average total transaction fee for an $M/M^B/1$ queue.

$$\begin{aligned}\mathbb{E}\left[R_{\mathrm{TX}}\right] = B\left(\sum_{k=1}^{P} g_k\left(B\rho_k\left(1 - z_{0,k}\right) - B\rho_{k+1}\left(1 - z_{0,k+1}\right)\right)\right) \\ + \sum_{i=1}^{P}\left(g_i - g_{i-1}\right)\left(B^2\rho_i\left(z_{0,i} - 1\right) + B\left(\rho_i - 1\right)\right)\end{aligned} \tag{3.81}$$

For a single service queue $B = 1$, the above equation is reduced to:

$$\mathbb{E}\left[R_{\mathrm{TX}}\right] = \sum_{k=1}^{P} g_k\left(\rho_k - \rho_{k+1}\right) = \sum_{k=1}^{P} g_k\frac{\lambda_k}{\mu} \tag{3.82}$$

We compare the average total transaction fee from the analytical expression, Eq. 3.81 and one from the simulation. Fig. 3.9 shows the distribution of total transaction fee (simulation), where the vertical lines are indicating the average total transaction fee (simulation and analytical). The simulation uses an $M/M^B/1$ queue with the bulk size $B = 10$. The number of priority groups P is 100. The fee distribution is $g_k = k$.[8] The service rate μ is $\frac{1}{600s}$. The utilization factor ρ is 1.15 (slightly overloaded). The total arrival rate λ is $\rho \cdot \mu B$ and the arrival rate for each priority group is uniform, i.e., $\lambda_x = \frac{\lambda}{P}$. The number of trials (blocks) is $100,000$. Our analytical expression for the average total transaction fee yields 580.75, which matches well with the simulation result of 571.09.

[7] $\sum_{k=1}^{n}\left(k \cdot ar^k\right) = ar\left(\frac{nr^{n+1} - (n+1)r^n + 1}{(1-r)^2}\right)$

[8] One may consider using an empirical fee distribution.

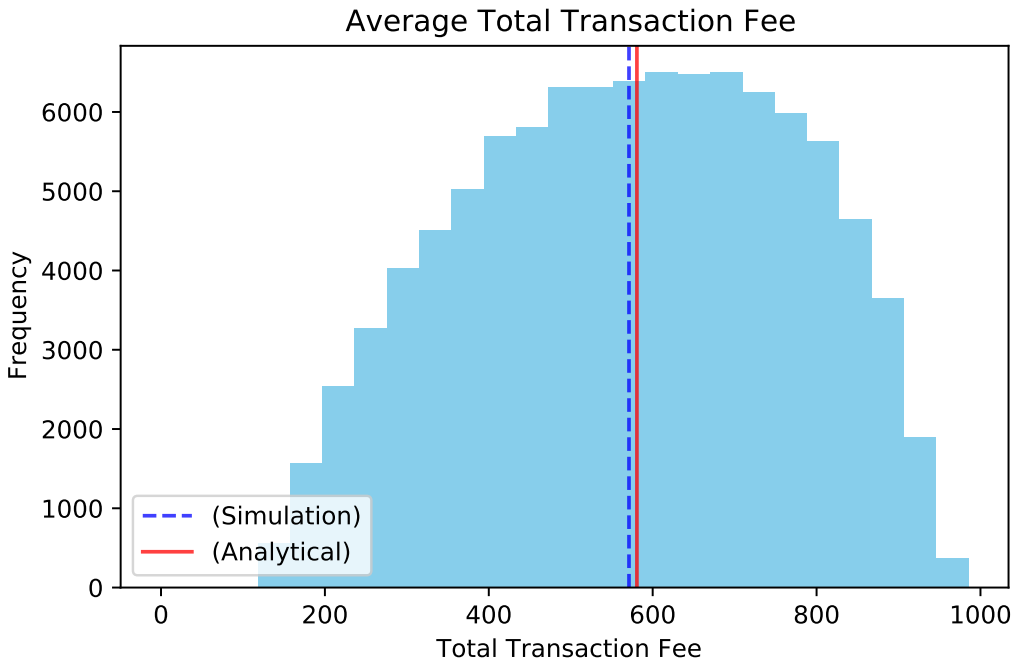

Figure 3.9: Average total transaction fee.

3.5 Conclusion

In this chapter, we analyzed a transaction fee market using queueing theory. We modeled the system as a head-of-line bulk service priority queue with an arbitrary service time distribution. We developed a numerical method to compute the average waiting time without the necessity of finding the stationary probabilities for an $M/G^B/1$ queue, as given in Eq. 3.52. We transformed the distribution of the number of transactions for each priority group to the distribution of the minimum transaction fee, which is an equilibrium price in the transaction fee market, as given in Eq. 3.63. We also derived the formula for the expected total transaction fee collected by a miner, as given in Eq. 3.78. For an $M/M^B/1$ queue, we obtained the concrete solutions for the distribution of the minimum transaction fee and the average total transaction fee, as given in Eq. 3.67 and Eq. 3.81, respectively.

CHAPTER 4

Block Mining Process with Delays

4.1 Introduction

A proof-of-work blockchain relies on the stochastic process of creating a block with protocol specific constraints. Due to the luck-based block proposal method, a blockchain peer-to-peer network occasionally observes multiple versions of chains, called a fork. The most common rule for resolving a fork is that every miner chooses the longest chain of blocks as a canonical state. Combined with the difficulty of block creation, the protocol gives a consistent view to every participant, especially for the past blocks.

Block mining is intense and expensive competition. Every miner wants to extend the chain by finding a new block as soon as possible. The intuition is that the probability of fork survival, i.e., winning the mining competition, is a pure function of computing power (hash rate). However, this only applies in an ideal world where delays do not exist. While the new block is propagated to the blockchain peer-to-peer network, there are inevitable processing and network delays as a miner should verify transactions. The nonzero delays in the real world decrease the probability of fork survival.

Users are sensitive to the probability of fork survival. The users can have strong confidence in probabilistic finality of the included transaction after waiting for a certain number of blocks. Miners are also sensitive to this probability since only winning miners claim a block reward and transaction fees. The miners want to maximize the expected mining revenue with this probability. They determine how many transactions they will put in a block considering

the trade-off relationship between the block size and the probability of fork survival.

In this chapter, we focus on deriving a formula for the probability of fork survival. Due to the complicated nature of the combinatorial problem that emerges, we provide a lower bound and an upper bound of this probability. We then solve special cases for two miners and three miners. We introduce the *three miner approximation* method, which is helpful to approximate the probability of fork survival with an arbitrary number of miners.

4.2 System Model

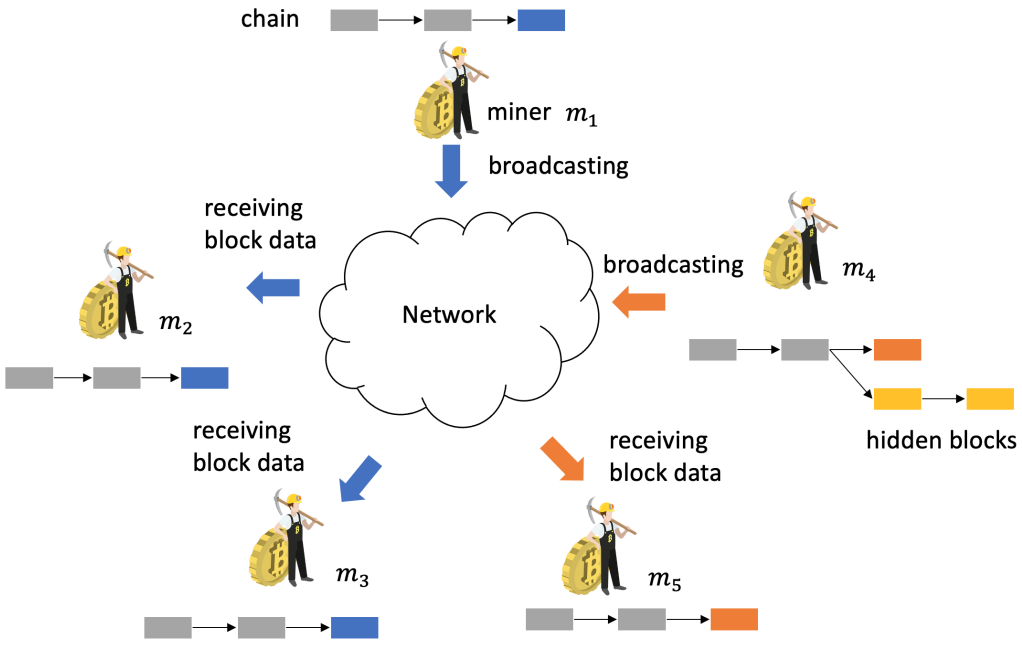

Figure 4.1: Miners are broadcasting their blocks to the network.

Every node in a blockchain peer-to-peer network replicates the append-only database. Once a miner finds a block by solving a hash puzzle, she broadcasts the new block to the peer-to-peer network. A proof-of-work blockchain protocol defines that miners should choose the longest chain as a canonical state.

In most cases, there is no disagreement between (honest) miners as there exists only one block proposal in the network. Fig. 4.1 illustrates that two miners (miner 1 and 4) are broadcasting their new blocks to the network. Every node accepts the block that arrives first. Due to processing and network delays, block arrival timing is different. In the example, miner 2 and 3 accept the block proposed by miner 1, while miner 5 accepts the block proposed by miner 4. Now there exist two proposals in the network. It is natural to observe multiple chain forks even among (honest) miners in probabilistic finality. In our work, we assume that every miner does not hide a new block intentionally.

We divide the block mining process into three parts for our analysis. It consists of block creation, propagation, and acceptance subprocesses.[1] In the block creation subprocess, we incorporate the processing delay. In the block propagation subprocess, we consider the network delay. To model the network delay, we will use a one-hop transmission delay model defined at each pair of nodes. In the block acceptance subprocess, we introduce a timing matrix to describe that a miner accepts a new block based on its arrival timing.

Table 4.1 summarizes the notation.

Table 4.1: Notation

Symbol	Description	Unit
n	Number of miners	
m_i	Miner $i = 1, \cdots, n$	
h_i	Hash rate of miner i (computing power)	$\#/s$
μ_i	Average rate of solving a hash puzzle by miner i (see Eq. 4.1)	$(\texttt{block})/s$
μ_b	Average block creation rate (system parameter)	$(\texttt{block})/s$
d_i	Processing delay at miner i	s
τ_{ij}	One-hop transmission delay from miner i to miner j	s

[1]Mining a block often refers to the block creation subprocess in a narrow sense.

τ'_{ij}	Aggregate delay from miner i to miner j (see Eq. 4.8)	s
X_i	Random variable denoting time spent by miner i solving a hash puzzle.	s
Y_i	Random variable denoting time spent by miner i creating a new block (see Eq. 4.2)	s
$B_{H,i}$	Block proposed by miner i whose height is H	(`block`)
$T_{H,i}$	Timestamp at which miner i reaches chain height H	s
$o_{H,i}$	Time skew in miner i at round H	s

4.3 Block Mining Process

4.3.1 Block Creation Subprocess

A block contains a list of transactions and metadata such as a hash digest of the parent block, a wallet address of a miner, a block height, a (local) timestamp, target difficulty, nonce (a random number), etc.

Definition 4.1. *A chain is a sequence of blocks. The chain height is the number of elements (blocks) in the sequence. The block height is the position of the block in the chain.*

Every block has a very strong link to the parent block. Any single-bit modification invalidates all the subsequent hash digests. Rearranged blocks are considered entirely different ones, even if they have the same list of transactions.

Definition 4.2. *If chains have at least one different element at the same height, a fork occurs.*

In probabilistic finality, it is natural to observe different blocks at the same height.[2]

[2]We do not consider a fork caused by a protocol change. This type of fork should be regarded as a different blockchain instance. This work can be applied to each instance separately.

Nodes have an inconsistent state by definition, especially for the recent blocks. However, every miner has the same sequence of the past blocks, i.e., the intersection of chains. To resolve a fork, miners choose the longest chain.

However, the longest chain rule alone is insufficient for a proof-of-work consensus. Suppose that the cost of block creation is nearly zero. Many new blocks will flood the peer-to-peer network. It will then be impossible for each miner to decide which version she should accept. Therefore, a proof-of-work blockchain must regulate the block creation rate by requiring a miner to solve a hash puzzle as shown in Eq. 4.1.

$$\underbrace{\texttt{Hash}((\text{transactions}), (\text{nonce}), \cdots)}_{\text{a numerical value}} < \underbrace{(\text{target difficulty})}_{\text{a numerical value}} \tag{4.1}$$

In the block creation subprocess, a miner processes transactions for a new block that extends the chain. She then solves the hash puzzle. A hash function maps the input data to a fixed-size random number. It is almost impossible to find its inverse. Due to the randomness of hash values, only the trial-and-error method is available to solve the hash puzzle. Therefore, presenting a nonce that satisfies Eq. 4.1 proves that a miner has spent some time creating a new block. Other miners can easily verify the nonce.

We have the following assumptions for the block creation subprocess.

- (Busy miner) A miner continues the block creation subprocess to append a new block. If the chain is extended by another miner or replaced, she immediately stops solving the current hash puzzle and starts solving a new hash puzzle for the new chain. The random variable X_i denotes the time spent by miner i solving a hash puzzle, i.e., finding a nonce satisfying the difficulty condition.

- (Processing delay) There is additional time d_i for miner i to process transactions to prepare a new block. She needs to verify the signature of each transaction and other conditions (like account balance) for integrity. The random variable Y_i denotes the

time spent creating a new block by miner i, which is given by

$$Y_i = X_i + d_i \tag{4.2}$$

- (No selfish mining) We assume that a miner does not intentionally hide a new block to get more benefits or conduct a double-spending attack. She broadcasts the new block to the rest of the network without any hesitation.

- (No hash split) A miner does not split her computing power to cause a fork. Instead, miners tend to form a group called a mining pool to aggregate their computing power. A mining pool is considered a single mining node with the sum of hash rates.

Block Creation Time Distribution

There are only two possible outcomes for a nonce: invalid (failure) or valid (success). Since the distribution of hash values is uniform, then the probability that any given nonce is valid is given by

$$p = \frac{\text{(Target Difficulty)}}{\text{(Range of a Hash Function)}} \tag{4.3}$$

In a short period of time, we assume that the hash rate h_i of miner i does not change. Solving a hash puzzle follows the binomial distribution $B(N, p)$.[3] However, it requires many trials ($N \to \infty$) as the target difficulty is much smaller than the range of a hash function ($p \to 0$). In this case, as is well known, the binomial distribution then converges to the Poisson distribution with the parameter ($= Np$). Now we know that the time spent by miner i solving a hash puzzle, X_i, follows the exponential distribution with the rate parameter μ_i.

$$X_i \sim \exp(\mu_i) \tag{4.4}$$

Note that random variables X_i ($i = 1, \cdots, n$) are mutually independent.

[3] $\mathbb{P}[x \text{ successes in a sequence of } N \text{ trials}] = \binom{N}{x} p^x (1-p)^{N-x}$

For a collection of miners $\{m_1, \cdots, m_n\}$, the time in solving a hash puzzle follows the exponential distribution with the sum of rate parameters.

$$\min(X_1, \cdots, X_n) \sim \exp(\mu_1 + \cdots + \mu_n) \tag{4.5}$$

If any hash rate changes, it affects the block creation rate. A proof-of-work blockchain adjusts the target difficulty to keep the block creation rate to be μ_b. However, the protocol does not know the actual hash rate of each miner nor the exact block arrival timing. Alternatively, it estimates the average block creation time from timestamps of the recent blocks. For example, Bitcoin uses the recent $2,016$ blocks (≈ 14 days) [Wik]. Ethereum only uses the timestamp of the parent block, but a geometric ratio adjusts the target difficulty quickly [Woo14].

Note that there exists a survival bias in the estimated average block creation time. The hash rate of miner i, h_i, is directly related with μ_i. However, the block creation rate by miner i can differ from μ_i.

4.3.2 Block Propagation Subprocess

After a miner finds a valid nonce for the new block, she advertises the information to her neighbor nodes. The neighbor nodes request the block data if there is any chance that the chain may become the longest one. Every mining node then checks the validity of the received block: nonce, transactions, and other necessary metadata, and repeats the same advertisement to her neighbor nodes.

The analysis of the block propagation subprocess often gets complicated due to many factors such as network topology, message path, and resource utilization that may affect block arrival timing. We choose to simplify this process by assuming that only one-hop transmission delays are required to propagate from miner i to miner j, which are pairwise constants denoted by τ_{ij} where $i, j \in \{1, \cdots, n\}$. This approximation is useful to model the stochastic nature of block arrival timing.

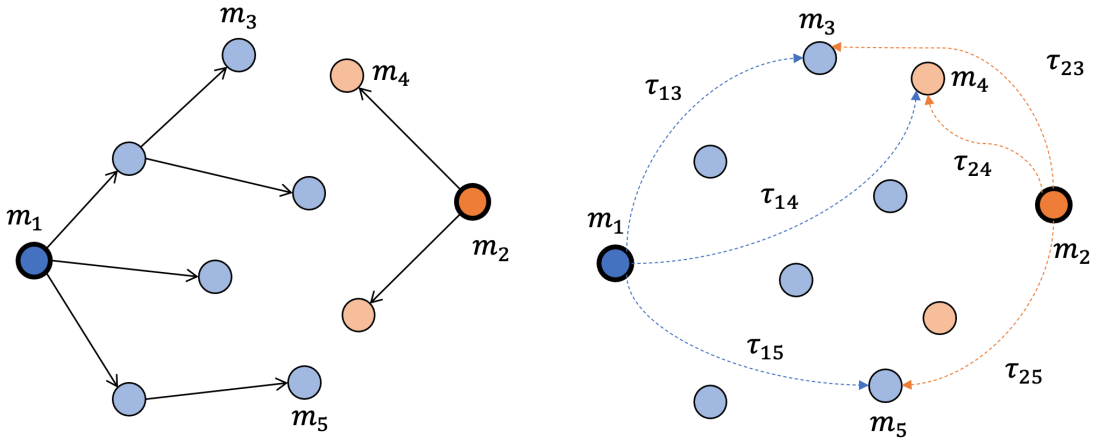

Figure 4.2: One-hop transmission delays between miners.

For one-hop transmission delays, we assume the followings:

- (Nonnegative) One-hop transmission delays are nonnegative real number, i.e., $\tau_{ij} \geq 0$.
- (Zero loopback delay) A one-hop transmission delay from node i to the node itself is zero, i.e., $\tau_{ii} = 0$.
- (Indirect path) An indirect path $i \to j \to k$ takes a longer time than a direct path $i \to k$, i.e., $\tau_{ij} + \tau_{jk} \geq \tau_{ik}$.

Fig. 4.2 shows the two competing chains proposed by miners (m_1 and m_2) in the peer-to-peer network of ten miners ($n = 10$). There are a lot of possible message paths between miners. Even if we suppose that the two miners found the blocks simultaneously, block arrival timing for each miner will be different as there are many factors along the message path. We simplify the situation by the use of $n^2 = 10^2$ constants. For example, miner m_3 receives the block proposed by m_1 first because τ_{13} is smaller than τ_{23}. Miner m_4 receives the block proposed by m_2 first because τ_{24} is smaller than τ_{14}. The same logic applies to miner m_5 who receives the block proposed by m_1 first as $\tau_{15} < \tau_{25}$.

We also assume that there is no message lost along the communication channel.

4.3.3 Block Acceptance Subprocess

Every miner accepts a new block based on its height and block arrival timing. We assume that a miner does not discriminate blocks by other metadata like wallet addresses. While a miner does not accept a stale block (because of late arrival), it may keep it in the cache storage. We also assume that retransmission for the entire chain data is not required.

Definition 4.3. *When miner i accepts block $B_{H,j}$, she begins solving a hash puzzle for the next block, $B_{H+1,i}$, whose parent block is $B_{H,j}$.*

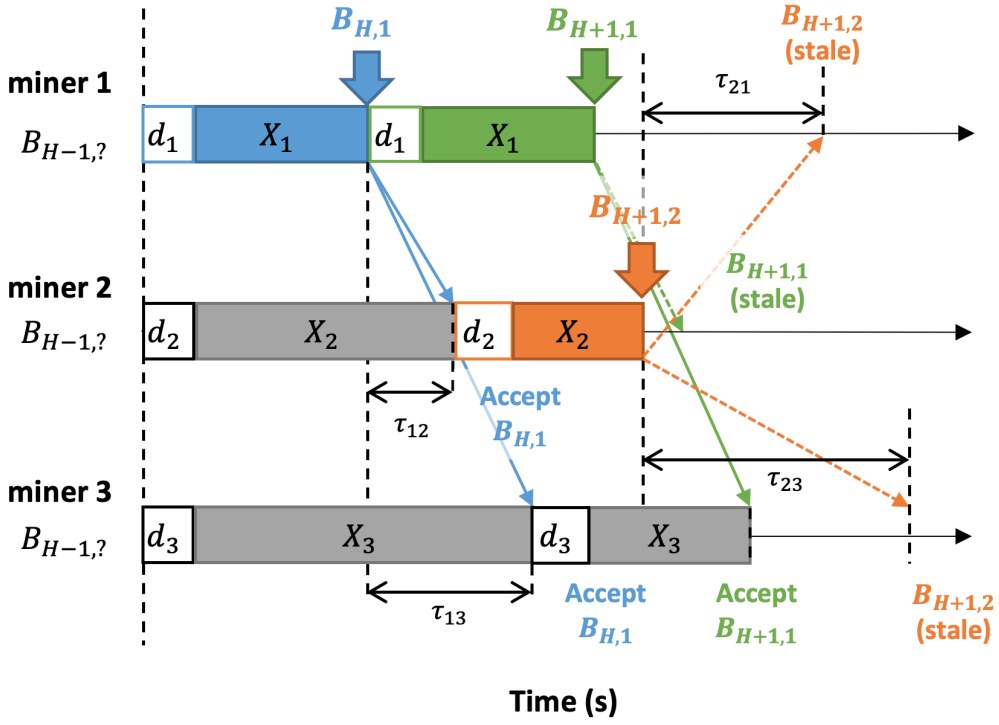

Figure 4.3: Time diagram of the mining process with delays.

Fig. 4.3 depicts a time diagram for three miners. Initially, every miner has the same chain with $B_{H-1,?}$. Their chain height is $H-1$. Miner 1 finds block $B_{H,1}$ and she broadcasts it to the rest of the network. Miner 2 and 3 accept $B_{H,1}$ as their chain height is still $H-1$. Every miner agrees with $B_{H,1}$ (fully accepted). Now miner 1 finds the next block $B_{H+1,1}$,

but miner 2 also finds the next block $B_{H+1,2}$ before the arrival of $B_{H+1,1}$. Miner 1 and 2 are in disagreement. Miner 2 ignores the stale block $B_{H+1,1}$ and miner 1 also ignores the stale block $B_{H+1,2}$. Miner 3 accepts block $B_{H+1,1}$ as it arrives before $B_{H+1,2}$. Now there are two competing chains in the network. The chain with $B_{H+1,1}$ is supported by $\{m_1, m_3\}$ and another chain with $B_{H+1,2}$ is supported by $\{m_2\}$. This inconsistent state may be resolved in the next round $H+2$, or may be continued.

Now we represent timing information in matrix form.

Definition 4.4. *A timing matrix* (a_{ij}) *is an* $n \times n$ *matrix defined as:*

$$a_{ij} \triangleq Y_i + o_{H,i} + \tau_{ij} = X_i + d_i + o_{H,i} + \tau_{ij} \tag{4.6}$$

where Y_i *is a random variable denoting time spent by miner* i *creating a block,* $o_{H,i}$ *denotes time skew in miner* i *at round* H*, and* τ_{ij} *is a one-hop transmission delay from miner* i *to miner* j*. Note that* Y_i *is equal to* $X_i + d_i$*.*

$$\begin{pmatrix} \boxed{a_{11}} & \boxed{\cdots} & \boxed{a_{1n}} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \tag{4.7}$$

The i-th row represents miner i who proposes block $B_{H,i}$. The j-th column represents miner j who accepts the first arrived block at round H. Let A_{ij} denote the event that the i-th row contains the smallest element for the j-th column.[4] We draw a rectangular box at the location of the smallest element for each column. For example, Eq. 4.7 shows the event $A_{11} \cap \cdots \cap A_{1n}$, i.e., every miner accepts block proposed by miner 1, $B_{H,1}$.

The processing delay term d_i and the one-hop transmission delay term τ_{ij} can be expressed as follows:

$$\tau'_{ij} \triangleq \tau_{ij} + d_i \tag{4.8}$$

[4] $A_{ij}^{\complement}$ denotes the complementary event of A_{ij}.

The time skew in miner k can be expressed as follows:

$$o_{H,k} = T_{H-1,k} - \min_{i \in \{1,\cdots,n\}} T_{H-1,i} \tag{4.9}$$

where $T_{H,k}$ is the timestamp at which miner k reaches chain height H.

Theorem 4.5. *Miner i whose block height $H-1$ may extend the chain either by proposing a new block $B_{H,i}$ or by accepting another miner j's block $B_{H,j}$ (the block height is H) if available.*

Proof. The proof is straightforward due to causality. Without receiving a block from another miner, miner i must propose $B_{H,i}$ to extend her own chain. When accepting a block from another miner, the block height must be equal to H. Although the next block $B_{H+c,j}$ ($c = 1, 2, \cdots$) may exist at that time, $B_{H,j}$ must arrive earlier than $B_{H+1,j}$. □

Theorem 4.6. *Let S be the set of rows containing the smallest element for each column in a timing matrix. $|S|$ is the number of chain forks in the network.*

Proof. Miner j accepts the block proposed by miner i, $B_{H,i}$, if and only if

$$a_{ij} = \min_{k \in \{1,\cdots,n\}} a_{kj} \tag{4.10}$$

because a miner cannot accept another block without splitting her computing power. The set S is $\{\arg\min_k a_{kj} \mid j = 1, ..., n\}$. For every element $i \in S$, the content of block $B_{H,i}$ is unique. □

Fig. 4.4 shows an example of three miners. At round $H-1$, every miner has a consistent state. At round H, miner 1 finds block $B_{H,1}$, miner 2 accepts block $B_{H,1}$, and miner 3 finds her own block $B_{H,3}$. Now there are two competing chains supported by $\{m_1, m_2\}$ and $\{m_3\}$, respectively. Eq. 4.11 shows the timing matrix at round $H-1$. Note that the first and third rows contain the smallest elements ($|S| = 2$).

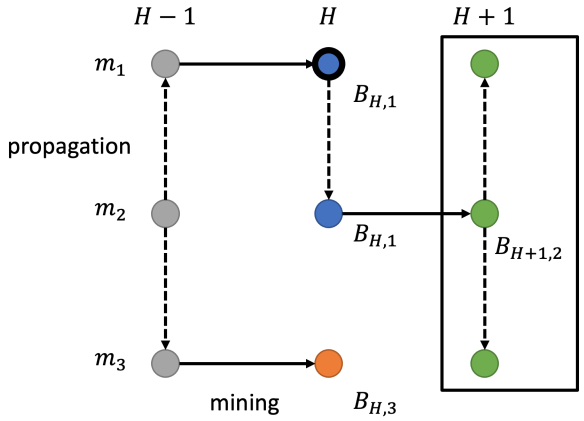

Figure 4.4: Every miner accepts $B_{H+1,2}$ at round $H+1$ and $B_{H,1}$ survives.

$$\begin{pmatrix} \boxed{a_{11}} & \boxed{a_{12}} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & \boxed{a_{33}} \end{pmatrix} \tag{4.11}$$

At round $H+1$, every miner accepts block $B_{H+1,2}$. The chain $(\cdots B_{H-1,?}B_{H,1}B_{H+1,2})$ wins the mining competition against the chain $(\cdots B_{H-1,?}B_{H,3})$. The transactions in block $B_{H,3}$ are discarded. Eq. 4.12 shows the timing matrix at round H. Note that the only second row contains the smallest elements ($|S| = 1$).

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ \boxed{a_{21}} & \boxed{a_{22}} & \boxed{a_{23}} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{4.12}$$

Fig. 4.5 shows a case in which the fork persists. At round $H+1$, miner 2 finds block $B_{H+1,2}$ instead of accepting block $B_{H+1,3}$. Eq. 4.13 shows the timing matrix at round H. Note that the second and third rows contain the smallest elements ($|S| = 2$).

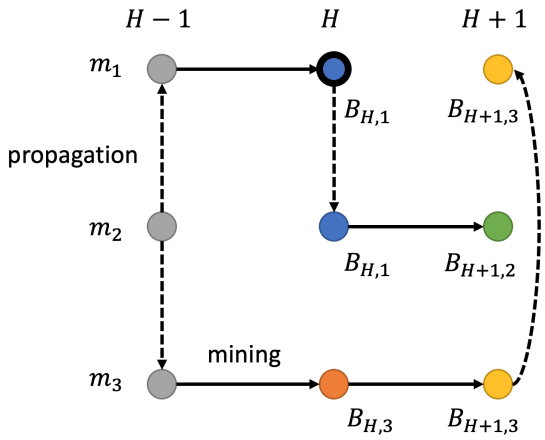

Figure 4.5: The fork persists. $B_{H,1}$ and $B_{H,3}$ have a chance to survive.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & \boxed{a_{22}} & a_{23} \\ \boxed{a_{31}} & a_{32} & \boxed{a_{33}} \end{pmatrix} \tag{4.13}$$

4.4 Probability of Fork Survival

The analysis of the probability of fork survival for an arbitrary number of miners, n, is fairly complicated. Most of the time, the fork will not persist. However, the different block arrival timing and nonuniform hash rates lead to a combinatorial problem.

At the first round H, there are three possible outcomes regarding how the block proposed by miner i, namely $B_{H,i}$, is accepted by miners. Fig. 4.6 shows the possible outcome paths for the proposed block.

- (Fully accepted at round H) Every miner accepts block $B_{H,i}$.
- (Partially accepted at round H) Some miners (not everyone) accept block $B_{H,i}$.
- None of them accept block $B_{H,i}$ at round H.

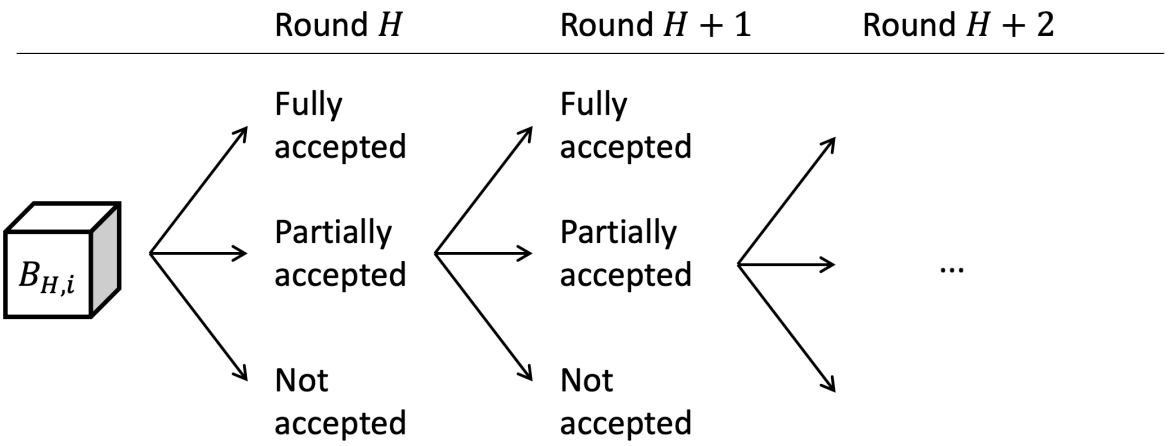

Figure 4.6: The possible outcome paths for the proposed block $B_{H,i}$.

If every miner accepts block $B_{H,i}$ at round H, a chain with $B_{H,i}$ wins the race (fully accepted). In this case, since no miners are working on the other competing chain with $B_{H,j}$ $(i \neq j)$ (the third outcome happens to $B_{H,j}$), there is no chance that the chain with $B_{H,i}$ is defeated by any other possibly longer chain.

If the second outcome happens, there is a question whether $B_{H,i}$ survives or not. Recall the example of three miners ($n = 3$) in Fig. 4.5. At round H, one chain with $B_{H,1}$ is supported by $\{m_1, m_2\}$ and another chain with $B_{H,3}$ is supported by $\{m_3\}$. At round $H+1$, either miner 1 or 2 may extend the chain and $B_{H,1}$ survives.

4.4.1 Probability of Full Acceptance (n miners)

First, let us compute the probability that every miner accepts $B_{H,i}$ within one round (fully accepted). We consider the following two cases.

Case 1) Processing and one-hop transmission delays are zero.

Since processing delay d_i and one-hop transmission delay τ_{ij} are zero, the corresponding timing matrix is $a_{ij} = X_i + o_{H,i}$. Miner $j = 1, \cdots, n$ accepts block $B_{H,i}$ according to the

following condition:

$$a_{ij} = \min_{k\in\{1,\cdots,n\}} a_{kj} \iff X_i + o_{H,i} = \min_{k\in\{1,\cdots,n\}} (X_k + o_{H,k}) \tag{4.14}$$

The right-hand side is independent of j, and thus, every miner finds the same k for the smallest value. Furthermore, time skew does not exist, $o_{H,k} = 0$, as the delays are zero.

Since $X_1, \cdots, X_n$ are mutually independent, the probability, $p_{\text{full},i}$, that every miner accepts block $B_{H,i}$ within one round is:

$$p_{\text{full},i} = \int_0^\infty \prod_{\substack{k=1\\k\neq i}}^{n} e^{-\mu_k x_i} \mu_i e^{-\mu_i x_i} dx_i = \int_0^\infty \mu_i e^{-\sum_{k=1}^n \mu_k x_i} dx_i = \frac{\mu_i}{\sum_{k=1}^n \mu_k} \tag{4.15}$$

The sum of the above probability for each miner i is $\sum_{i=1}^n \left(\frac{\mu_i}{\sum_{k=1}^n \mu_k}\right) = 1$. The probability that a fork occurs is $1 - 1 = 0$, i.e., the network does not suffer a fork.

Case 2) Processing and one-hop transmission delays are nonnegative.

Miner $j = 1, \cdots, n$ accepts block $B_{H,i}$ according to the following condition:

$$a_{ij} = \min_{k\in\{1,\cdots,n\}} a_{kj} \iff \bigwedge_{k=1}^{n} (a_{ij} \leq a_{kj}) = \bigwedge_{k=1}^{n} \left(X_i + o_{H,i} + \tau'_{ij} \leq X_k + o_{H,k} + \tau'_{kj}\right) \tag{4.16}$$

where $\bigwedge$ represents the logical AND for boolean predicates and $\tau'_{ij} = \tau_{ij} + d_i$.

Since Eq. 4.16 should hold for every miner j,

$$\bigwedge_{k=1}^{n} \left(X_k + o_{H,k} \geq X_i + o_{H,i} + \max_{j\in\{1,\cdots,n\}} \left(\tau'_{ij} - \tau'_{kj}\right)\right) \tag{4.17}$$

To simplify the notation, we define:

$$c_{ik} \triangleq \max_{j\in\{1,\cdots,n\}} \left(\tau'_{ij} - \tau'_{kj}\right) \tag{4.18}$$

Theorem 4.7. *$c_{ik} + d_k - d_i$ is always nonnegative, i.e., $c_{ik} \geq d_i - d_k$ where c_{ik} denotes $\max_{j\in\{1,\cdots,n\}} \left(\tau'_{ij} - \tau'_{kj}\right)$.*

Proof. From the definition of c_{ik}, we have $c_{ik} + d_k - d_i = \max_{j\in\{1,\cdots,n\}} (\tau_{ij} - \tau_{kj})$. When $j = k$, $\tau_{ij} - \tau_{kk} = \tau_{ij} \geq 0$ because one-hop transmission delays are nonnegative real numbers. Therefore, we have $\max_{j\in\{1,\cdots,n\}} (\tau_{ij} - \tau_{kj}) \geq 0$ and $c_{ik} + d_k - d_i \geq 0$. □

Since $X_1, \cdots, X_n$ are mutually independent, the probability, $p_{\text{full},i}$, that every miner accepts block $B_{H,i}$ within one round is

$$p_{\text{full},i} = \int_0^\infty \prod_{\substack{k=1 \\ k \neq i}}^{n} e^{-\mu_k \max(0, x_i + o_{H,i} - o_{H,k} + c_{ik})} \mu_i e^{-\mu_i x_i} dx_i \tag{4.19}$$

where $o_{H,i}$ denotes time skew in miner i at round H.

It is often impractical to consider all boundaries of the max functions in Eq. 4.19 for n arbitrary miners. Therefore, we introduce the following bounding approach.

$$\underbrace{e^{-a \max(0,b)} \leq e^{-ab}}_{\text{the equality holds when } b \geq 0} \tag{4.20}$$

where $a > 0$ and b is any real number.

Now we have:

$$\begin{aligned}
\int_0^\infty \prod_{\substack{k=1 \\ k \neq i}}^{n} e^{-\mu_k \max(0, x_i + o_{H,i} - o_{H,k} + c_{ik})} \mu_i e^{-\mu_i x_i} dx_i &\leq \int_0^\infty \prod_{\substack{k=1 \\ k \neq i}}^{n} e^{-\mu_k \left(x_i + o_{H,i} - o_{H,k} + c_{ik} \right)} \mu_i e^{-\mu_i x_i} dx_i \\
&= \int_0^\infty \prod_{\substack{k=1 \\ k \neq i}}^{n} e^{-\mu_k \left(o_{H,i} - o_{H,k} + c_{ik} \right)} \mu_i e^{-\sum_{k=1}^{n} \mu_k x_i} dx_i \\
&= \left(\prod_{\substack{k=1 \\ k \neq i}}^{n} e^{-\mu_k \left(o_{H,i} - o_{H,k} + c_{ik} \right)} \right) \frac{\mu_i}{\sum_{k=1}^{n} \mu_k}
\end{aligned} \tag{4.21}$$

As an approximation for Eq. 4.21, we assume that the difference of the time skew terms is negligible. Theorem 4.7 states that one-hop transmission delays do not make c_{ik} negative and processing delays can make c_{ik} negative. One-hop transmission delays decrease the probability that every miner accepts the proposed block, i.e., increasing the probability of causing a fork. Processing delays are tricky as now c_{ik} can be a negative value depending on the difference of $d_i - d_k$. We can imagine miner i with a significant processing delay d_i, which becomes a penalty to miner i. The other miners take advantage of the delays as they have more time for the next block.

By summing Eq. 4.19 for every miner i, we compute the probability that a fork occurs as follows:

$$1 - \sum_{i=1}^{n} \left(\int_0^{\infty} \prod_{\substack{k=1 \\ k \neq i}}^{n} e^{-\mu_k \max(0, x_i + o_{H,i} - o_{H,k} + c_{ik})} \mu_i e^{-\mu_i x_i} dx_i \right) \tag{4.22}$$

4.4.2 Lower Bound and Upper Bound

The straightforward approach encounters an intractable number of combinations for n arbitrary miners. We choose to derive a lower bound and an upper bound of the probability of fork survival. Recall that there are three possible outcomes of the first round H for a chain with $B_{H,i}$. Clearly, if every miner accepts a block, the block survives. If none of the miners accept a block, the block does not survive. If some miners (not everyone) accept the block, we need to consider the subsequent outcomes of the next round $H + 1$.

Let $p_{\text{full},i}$, $p_{\text{partial},i}$, and $p_{\text{none},i}$ denote the probability of the following outcomes at first round H, respectively.

- $p_{\text{full},i} \triangleq \mathbb{P}[\text{Every miner accepts block } B_{H,i}]$
- $p_{\text{partial},i} \triangleq \mathbb{P}[\text{Some miners (not everyone) accept block } B_{H,i}]$
- $p_{\text{none},i} \triangleq \mathbb{P}[\text{None of them accept block } B_{H,i}]$

($p_{\text{full},i}$) We already computed the probability that everyone accepts the block in Eq. 4.19:

$$p_{\text{full},i} = \int_0^{\infty} \prod_{\substack{k=1 \\ k \neq i}}^{n} e^{-\mu_k \max(0, x_i + o_{H,i} - o_{H,k} + c_{ik})} \mu_i e^{-\mu_i x_i} dx_i \tag{4.23}$$

($p_{\text{partial},i}$) Suppose that the chain with $B_{H,i}$ is partially accepted at round H. Let M be a set of miners who accept $B_{H,i}$. At round $H+1$, if every miner accepts any block proposed by miner $m \in M$, $B_{H,i}$ survives. We cannot simply reuse Eq. 4.23 as the probability depends on the set of miners who accepted block $B_{H,i}$. For an upper bound, there is a chance that

another chain defeats $B_{H,i}$, the probability of fork survival $B_{H,i}$ is always less than or equal to $p_{\text{partial},i}$. We obtain the inequality as follows:

$$\underbrace{p_{\text{full},i}}_{\text{lower bound}} \leq p_{\text{survival},i} \leq p_{\text{full},i} + p_{\text{partial},i} \tag{4.24}$$

($p_{\text{none},i}$) Suppose that none of the miners accept $B_{H,i}$. Since the sum of probabilities is 1, we obtain $p_{\text{full},i} + p_{\text{partial},i} = 1 - p_{\text{none},i}$. In the timing matrix, the i-th row never has the smallest element for every column.

$$\begin{aligned} &p_{\text{none},i} = \mathbb{P}[A_{i1}^{\complement} \cap \cdots \cap A_{in}^{\complement}] = 1 - \mathbb{P}[A_{i1} \cup ... \cup A_{in}], \\ &p_{\text{full},i} + p_{\text{partial},i} = \mathbb{P}[A_{i1} \cup ... \cup A_{in}] \end{aligned} \tag{4.25}$$

The union of events, $A_{i1} \cup ... \cup A_{in}$, can be expressed:

$$\bigvee_{j=1}^{n} \left(\bigwedge_{k=1}^{n} X_k \geq X_i + o_{H,i} - o_{H,k} + \tau'_{ij} - \tau'_{kj} \right) \tag{4.26}$$

where $\bigwedge$ represents the logical AND, $\bigvee$ represents the logical OR for boolean predicates.

Eq. 4.26 implies the following (its converse is not necessarily true).

$$\bigwedge_{k=1}^{n} X_k \geq X_i + o_{H,i} - o_{H,k} + \min_{j \in \{1,\cdots,n\}} \left(\tau'_{ij} - \tau'_{kj} \right) \tag{4.27}$$

Thus, we obtain the (loose) upper bound

$$\mathbb{P}[A_{i1} \cup ... \cup A_{in}] \leq \underbrace{\int_0^{\infty} \prod_{\substack{k=1 \\ k \neq i}}^{n} e^{-\mu_k \max(0, x_i + o_{H,i} - o_{H,k} + c'_{ik})} \mu_i e^{-\mu_i x_i} dx_i}_{\text{(loose) upper bound}} \tag{4.28}$$

where

$$c'_{ik} \triangleq \min_{j \in \{1,\cdots,n\}} \left(\tau'_{ij} - \tau'_{kj} \right) \tag{4.29}$$

We report that the lower bound and the (loose) upper bound are not tight when the delays are large. However, the lower bound is equivalent to the probability of full acceptance.

In Fig. 4.7 we compare the analytical lower bound and the simulation of the timing model. The simulator implements block creation, propagation, and acceptance subprocesses. It measures the empirical probability of fork survival. In the simulation, we consider three miners whose hash rates are $(\mu_1, \mu_2, \mu_3) = (0.50, 0.25, 0.25) \cdot \mu_b$. We assume that the effect of time skew is zero. The aggregate delays are defined as follows:

$$\tau'_{ij} = \begin{cases} d & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{4.30}$$

where $d \in [0, 600]$.[5] The lower bound, Eq. 4.23, matches well with the simulation result (full acceptance), as indeed it should. The probability of full acceptance decreases as the delay increases. However, the lower bound does not capture an effect of interaction between miners. A large miner (50% of the network hash rate) can extend the chain faster than a small miner (25%). The probability of fork survival for the large miner increases as the delay increases, whereas that of the (relatively) small miner decreases.[6]

[5]The relative amount of delay is important. The delay in the Bitcoin network ($\approx 15 \sim 30$ seconds) is small compared to its block creation time ($\approx 600s$). We exaggerate the absolute amount of delay (from 0% to 100% of the block creation time).

[6]This advantage can be one of reasons to form a mining pool.

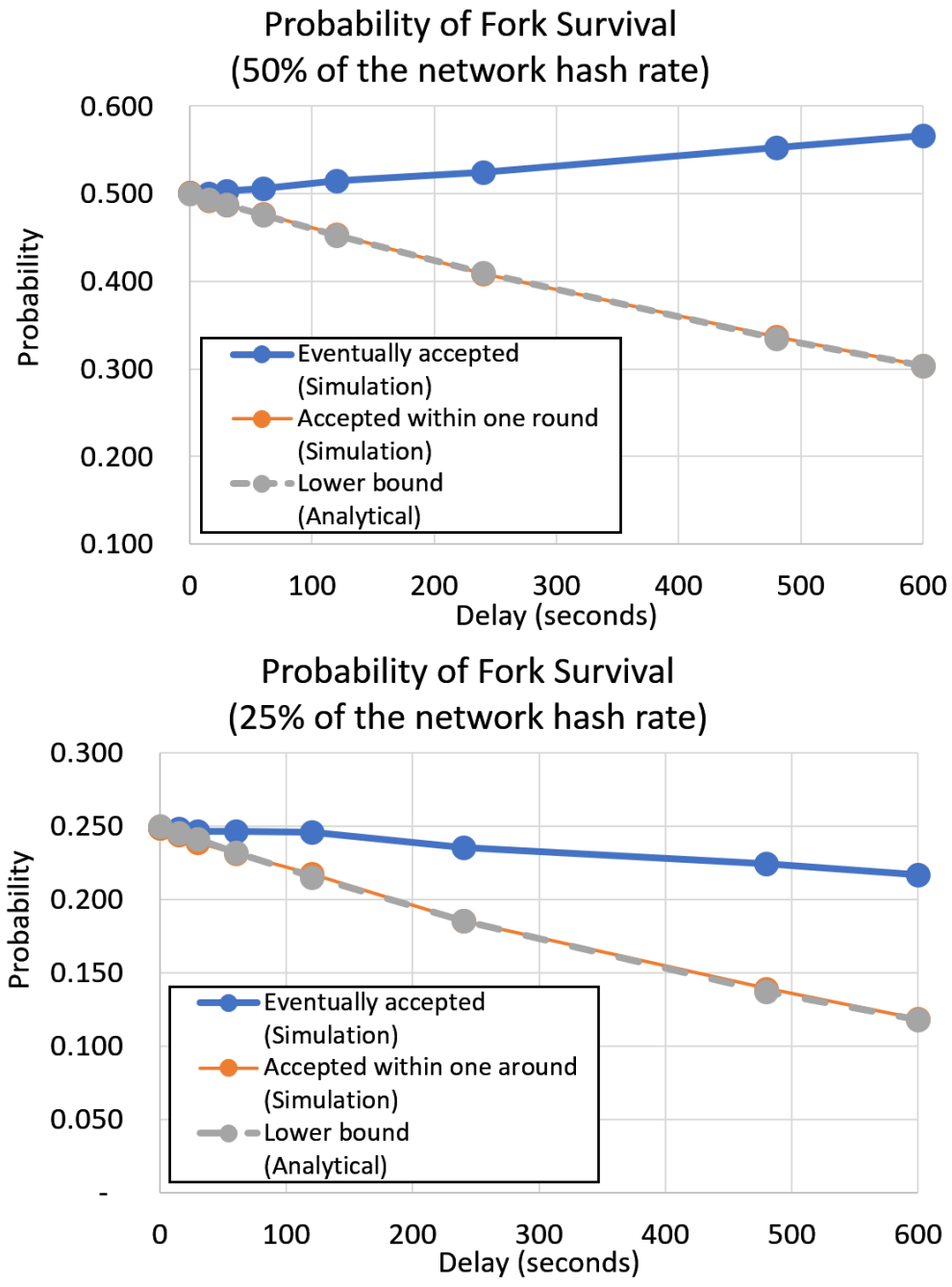

Figure 4.7: Lower bound of the probability of fork survival.

4.4.3 Special Case: two miners (n = 2)

For two miners, we consider a 2×2 timing matrix. There are $2^2 = 4$ combinations in terms of the location of the smallest elements. We consider the following two cases.

Case 1: (Fully accepted) Every miner accepts the proposed block.

Suppose that miner 1, without loss of generality, proposed a block at round H, namely $B_{H,1}$. Only the first row contains the smallest elements. The corresponding timing matrix is:

$$\begin{pmatrix} \boxed{a_{11}} & \boxed{a_{12}} \\ a_{21} & a_{22} \end{pmatrix} \tag{4.31}$$

The probability that every miner accepts $B_{H,1}$ is:

$$p_{\text{full},1} = \mathbb{P}\left[a_{11} < a_{21} \ \wedge \ a_{12} < a_{22}\right] = \int_0^\infty e^{-\mu_2 \max\left(0, x_1 + o_{H,1} - o_{H,2} + c_{12}\right)} \mu_1 e^{-\mu_1 x_1} dx_1 \tag{4.32}$$

i) $o_{H,1} - o_{H,2} + c_{12} \geq 0$, we can reduce Eq. 4.32 to:

$$\int_0^\infty e^{-\mu_2 (x_1 + o_{H,1} - o_{H,2} + c_{12})} \mu_1 e^{-\mu_1 x_1} dx_1 = \frac{\mu_1}{\mu_1 + \mu_2} e^{-\mu_2 (o_{H,1} - o_{H,2} + c_{12})} \leq \frac{\mu_1}{\mu_1 + \mu_2} \tag{4.33}$$

ii) $o_{H,1} - o_{H,2} + c_{12} < 0$, the value of X_1 determines the sign of the max function:

$$\begin{aligned} &\int_0^{-(o_{H,1} - o_{H,2} + c_{12})} \mu_1 e^{-\mu_1 x_1} dx_1 + \int_{-(o_{H,1} - o_{H,2} + c_{12})}^\infty \mu_1 e^{-(\mu_1 + \mu_2) x_1} e^{-\mu_2 (o_{H,1} - o_{H,2} + c_{12})} dx_1 \\ &= \left(-e^{\mu_1 (o_{H,1} - o_{H,2} + c_{12})} + 1\right) + \frac{\mu_1}{\mu_1 + \mu_2} e^{\mu_1 (o_{H,1} - o_{H,2} + c_{12})} \\ &= 1 - \frac{\mu_2}{\mu_1 + \mu_2} e^{\mu_1 (o_{H,1} - o_{H,2} + c_{12})} > \frac{\mu_1}{\mu_1 + \mu_2} \end{aligned} \tag{4.34}$$

Summarizing these two cases,

$$\mathbb{P}\left[a_{11} < a_{21} \ \wedge \ a_{12} < a_{22}\right] = \begin{cases} \frac{\mu_1}{\mu_1 + \mu_2} e^{-\mu_2 (o_{H,1} - o_{H,2} + c_{12})} & \text{if } o_{H,1} - o_{H,2} + c_{12} \geq 0 \\ 1 - \frac{\mu_2}{\mu_1 + \mu_2} e^{\mu_1 (o_{H,1} - o_{H,2} + c_{12})} & \text{if } o_{H,1} - o_{H,2} + c_{12} < 0 \end{cases} \tag{4.35}$$

For miner 2, we repeat the above computation. Only the second row contains the smallest elements.

$$\begin{pmatrix} a_{11} & a_{12} \\ \boxed{a_{21}} & \boxed{a_{22}} \end{pmatrix} \tag{4.36}$$

The probability that every miner accepts $B_{H,2}$ is:

$$\mathbb{P}\left[a_{21} < a_{11} \ \wedge \ a_{22} < a_{12}\right] = \begin{cases} \frac{\mu_2}{\mu_1+\mu_2} e^{-\mu_1(o_{H,2}-o_{H,1}+c_{21})} & \text{if } o_{H,2}-o_{H,1}+c_{21} \geq 0 \\ 1-\frac{\mu_1}{\mu_1+\mu_2} e^{\mu_2(o_{H,2}-o_{H,1}+c_{21})} & \text{if } o_{H,2}-o_{H,1}+c_{21} < 0 \end{cases} \tag{4.37}$$

Case 2) (Partially accepted) Some miners accept the proposed block.

Miner 1 and miner 2 are in disagreement: m_1 accepts $B_{H,1}$ and m_2 accepts $B_{H,2}$. The corresponding timing matrix is:

$$\begin{pmatrix} \boxed{a_{11}} & a_{12} \\ a_{21} & \boxed{a_{22}} \end{pmatrix} \tag{4.38}$$

Equivalently,

$$a_{11} < a_{21} \ \wedge \ a_{22} < a_{12} \tag{4.39}$$

We rewrite Eq. 4.39 in terms of X_1 and X_2:

$$(X_2 > X_1 + o_{H,1} - o_{H,2} + \tau'_{11} - \tau'_{21}) \wedge (X_2 < X_1 + o_{H,1} - o_{H,2} + \tau'_{12} - \tau'_{22}) \tag{4.40}$$

To evaluate the probability, we need to consider the following cases:

i) $o_{H,1} - o_{H,2} + \tau'_{11} - \tau'_{21} \leq o_{H,1} - o_{H,2} + \tau'_{12} - \tau'_{22} \iff \tau'_{11} - \tau'_{21} \leq \tau'_{12} - \tau'_{22}$

a) $o_{H,1} - o_{H,2} + \tau'_{11} - \tau'_{21} \leq 0 \leq o_{H,1} - o_{H,2} + \tau'_{12} - \tau'_{22}$ (common),

The probability that a fork occurs is:

$$\begin{aligned} 1 - &\int_0^\infty \left(\int_{x_1+o_{H,1}-o_{H,2}+\tau'_{12}-\tau'_{22}}^\infty \mu_2 e^{-\mu_2 x_2} dx_2 \right) \mu_1 e^{-\mu_1 x_1} dx_1 \\ &- \int_{-(o_{H,1}-o_{H,2}+\tau'_{11}-\tau'_{21})}^\infty \left(\int_0^{x_1+o_{H,1}-o_{H,2}+\tau'_{11}-\tau'_{21}} \mu_2 e^{-\mu_2 x_2} dx_2 \right) \mu_1 e^{-\mu_1 x_1} dx_1 \end{aligned} \tag{4.41}$$

We evaluate the integrals as follows:

$$1 - \frac{\mu_1}{\mu_1+\mu_2} e^{-\mu_2(o_{H,1}-o_{H,2}+\tau'_{12}-\tau'_{22})} - \frac{\mu_2}{\mu_1+\mu_2} e^{\mu_1(o_{H,1}-o_{H,2}+\tau'_{11}-\tau'_{21})} \tag{4.42}$$

We further simplify Eq. 4.42:

$$\frac{\mu_1(1 - e^{-\mu_2(o_{H,1}-o_{H,2}+\tau'_{12}-\tau'_{22})}) + \mu_2(1 - e^{\mu_1(o_{H,1}-o_{H,2}+\tau'_{11}-\tau'_{21})})}{\mu_1+\mu_2} \tag{4.43}$$

b) $0 \leq o_{H,1} - o_{H,2} + \tau'_{11} - \tau'_{21} \leq o_{H,1} - o_{H,2} + \tau'_{12} - \tau'_{22}$,

$$\frac{\mu_1}{\mu_1+\mu_2} \left(e^{-\mu_2(o_{H,1}-o_{H,2}+\tau'_{11}-\tau'_{21})} - e^{-\mu_2(o_{H,1}-o_{H,2}+\tau'_{12}-\tau'_{22})} \right) \tag{4.44}$$

c) $o_{H,1} - o_{H,2} + \tau'_{11} - \tau'_{21} \leq o_{H,1} - o_{H,2} + \tau'_{12} - \tau'_{22} \leq 0$

$$\frac{\mu_2}{\mu_1+\mu_2} \left(e^{\mu_1(o_{H,1}-o_{H,2}+\tau'_{12}-\tau'_{22})} - e^{\mu_1(o_{H,1}-o_{H,2}+\tau'_{11}-\tau'_{21})} \right) \tag{4.45}$$

Summarizing these three cases,

$$\begin{aligned} &\mathbb{P}\left[a_{11} < a_{21} \ \wedge \ a_{22} < a_{12}\right] \\ &= \begin{cases} \frac{\mu_1(1-e^{-\mu_2 t_2})+\mu_2(1-e^{\mu_1 t_1})}{\mu_1+\mu_2} & \text{if } t_1 \leq 0 \leq t_2 \\ \frac{\mu_1}{\mu_1+\mu_2}\left(e^{-\mu_2 t_1} - e^{-\mu_2 t_2}\right) & \text{if } 0 \leq t_1 \leq t_2 \\ \frac{\mu_2}{\mu_1+\mu_2}\left(e^{\mu_1 t_2} - e^{\mu_1 t_1}\right) & \text{if } t_1 \leq t_2 \leq 0 \end{cases} \end{aligned} \tag{4.46}$$

where $t_1 \triangleq o_{H,1} - o_{H,2} + \tau'_{11} - \tau'_{21}$ and $t_2 \triangleq o_{H,1} - o_{H,2} + \tau'_{12} - \tau'_{22}$.

ii) $o_{H,1} - o_{H,2} + \tau'_{11} - \tau'_{21} > o_{H,1} - o_{H,2} + \tau'_{12} - \tau'_{22} \iff \tau'_{11} - \tau'_{21} > \tau'_{12} - \tau'_{22}$

Since one-hop transmission delays are nonnegative, $\tau'_{11} - \tau'_{21} > \tau'_{12} - \tau'_{22} \iff -\tau_{21} > \tau_{12}$ is not true. Thus, this case never happens.

The following type of disagreement does not happen too.

$$\begin{pmatrix} a_{11} & \boxed{a_{12}} \\ \boxed{a_{21}} & a_{22} \end{pmatrix} \tag{4.47}$$

Equivalently,

$$a_{21} < a_{11} \ \wedge \ a_{12} < a_{22} \tag{4.48}$$

To consider Eq. 4.40, the following condition must hold:

$$\tau'_{12} - \tau'_{22} < \tau'_{11} - \tau'_{21} \iff \tau_{12} < -\tau_{21} \tag{4.49}$$

Since one-hop transmission delays are nonnegative, Eq. 4.49 is not true.

The probability that chain with $B_{H,1}$ survives, $p_{\text{survival},1}$, is approximately:

$$p_{\text{survival},1} \approx p_{\text{full},1} + p_{\text{partial},1} \cdot p_{\text{full},1} + p_{\text{partial},1}^2 \cdot p_{\text{full},1} + \cdots = p_{\text{full},1} \left(\frac{1}{1 - p_{\text{partial},1}} \right) \tag{4.50}$$

From the sum of probabilities is 1, we have:

$$p_{\text{partial},1} + p_{\text{full},1} + p_{\text{full},2} = 1 \iff p_{\text{partial},1} = 1 - p_{\text{full},1} - p_{\text{full},2} \tag{4.51}$$

Note that $p_{\text{full},2} = p_{\text{none},1}$.

Unless $p_{\text{partial},1} = 1$ the probability that chain with $B_{H,1}$ survives is:

$$p_{\text{survival},1} \approx p_{\text{full},1} \left(\frac{1}{1 - p_{\text{partial},1}} \right) = \frac{p_{\text{full},1}}{p_{\text{full},1} + p_{\text{full},2}} \approx \frac{\mu_1}{\mu_1 + \mu_2} \tag{4.52}$$

Similarly, the probability that the chain with $B_{H,2}$ survives, $p_{\text{survival},2}$, is approximately $\frac{p_{\text{full},2}}{p_{\text{full},1}+p_{\text{full},2}} \approx \frac{\mu_2}{\mu_1+\mu_2}$. A fork will be eventually resolved unless the network is isolated.

4.4.4 Special Case: three miners (n = 3)

For three miners, we consider a 3×3 timing matrix. There are $3^3 = 27$ combinations in terms of the location of the smallest elements. We consider the following two cases.

Case 1: (Fully accepted) Every miner accepts the proposed block.

Suppose that miner 1, without loss of generality, proposed a block at round H, namely $B_{H,1}$.

The probability that every miner accepts block $B_{H,1}$ is computed:

$$p_{\text{full},1} = \int_0^\infty e^{-\mu_2 \max(0, x_1 + o_{H,1} - o_{H,2} + c_{12})} e^{-\mu_3 \max(0, x_1 + o_{H,1} - o_{H,3} + c_{13})} \mu_1 e^{-\mu_1 x_1} dx_1 \tag{4.53}$$

To simplify the notation, we define:

$$t_k \triangleq o_{H,1} - o_{H,2} + c_{1k} \tag{4.54}$$

We need to consider the several cases to evaluate the probability $p_{\text{full},1}$:

i) $t_2 \leq t_3$

a) $0 \leq t_2 \leq t_3$

$$\begin{aligned} &\int_0^\infty e^{-\mu_2(x_1+t_2)} e^{-\mu_3(x_1+t_3)} \mu_1 e^{-\mu_1 t_1} dx_1 \\ &\qquad = \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3} e^{-\mu_2 t_2} e^{-\mu_3 t_3} \end{aligned} \tag{4.55}$$

b) $t_2 \leq 0 \leq t_3$

$$\begin{aligned} &\int_0^{-t_2} e^{-\mu_3(x_1+t_3)} \mu_1 e^{-\mu_1 x_1} dx_1 \\ &+ \int_{-t_2}^\infty e^{-\mu_2(x_1+t_2)} e^{-\mu_3(x_1+t_3)} \mu_1 e^{-\mu_1 x_1} dx_1 \\ &= \frac{\mu_1}{\mu_1 + \mu_3} e^{-\mu_3 t_3} \\ &- \left(\frac{\mu_1}{\mu_1 + \mu_3} - \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3} \right) e^{(\mu_1+\mu_3)t_2} e^{-\mu_3 t_3} \end{aligned} \tag{4.56}$$

c) $t_2 \leq t_3 \leq 0$

$$\begin{aligned} &\int_0^{-t_3} \mu_1 e^{-\mu_1 x_1} dx_1 \\ &+ \int_{-t_3}^{-t_2} e^{-\mu_3(x_1+t_3)} \mu_1 e^{-\mu_1 x_1} dx_1 \\ &+ \int_{-t_2}^\infty e^{-\mu_2(x_1+t_2)} e^{-\mu_3(x_1+t_3)} \mu_1 e^{-\mu_1 x_1} dx_1 \\ &= 1 - \left(1 - \frac{\mu_1}{\mu_1 + \mu_3} \right) e^{\mu_1 t_3} \\ &\quad - \left(\frac{\mu_1}{\mu_1 + \mu_3} - \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3} \right) e^{(\mu_1+\mu_3)t_2} e^{-\mu_3 t_3} \end{aligned} \tag{4.57}$$

ii) $t_2 \geq t_3$

a) $0 \leq t_3 \leq t_2$

$$\frac{\mu_1}{\mu_1+\mu_2+\mu_3} e^{-\mu_2 t_2} e^{-\mu_3 t_3} \tag{4.58}$$

b) $t_3 \leq 0 \leq t_2$

$$\frac{\mu_1}{\mu_1+\mu_2} e^{-\mu_2 t_2} - \left(\frac{\mu_1}{\mu_1+\mu_2} - \frac{\mu_1}{\mu_1+\mu_2+\mu_3} \right) e^{(\mu_1+\mu_2)t_3} e^{-\mu_2 t_2} \tag{4.59}$$

c) $t_3 \leq t_2 \leq 0$

$$1 - \left(1 - \frac{\mu_1}{\mu_1+\mu_2} \right) e^{\mu_1 t_2} - \left(\frac{\mu_1}{\mu_1+\mu_2} - \frac{\mu_1}{\mu_1+\mu_2+\mu_3} \right) e^{(\mu_1+\mu_2)t_3} e^{-\mu_2 t_2} \tag{4.60}$$

Summarizing these six cases,

$$p_{\text{full},1} = \begin{cases} \frac{\mu_1}{\mu_1+\mu_2+\mu_3} e^{-\mu_2 t_2} e^{-\mu_3 t_3} & \text{if } 0 \leq t_2 \leq t_3 \\ \frac{\mu_1}{\mu_1+\mu_3} e^{-\mu_3 t_3} - \left(\frac{\mu_1}{\mu_1+\mu_3} - \frac{\mu_1}{\mu_1+\mu_2+\mu_3} \right) e^{(\mu_1+\mu_3)t_2} e^{-\mu_3 t_3} & \text{if } t_2 \leq 0 \leq t_3 \\ 1 - \left(1 - \frac{\mu_1}{\mu_1+\mu_3} \right) e^{\mu_1 t_3} - \left(\frac{\mu_1}{\mu_1+\mu_3} - \frac{\mu_1}{\mu_1+\mu_2+\mu_3} \right) e^{(\mu_1+\mu_3)t_2} e^{-\mu_3 t_3} & \text{if } t_2 \leq t_3 \leq 0 \\ \frac{\mu_1}{\mu_1+\mu_2+\mu_3} e^{-\mu_2 t_2} e^{-\mu_3 t_3} & \text{if } 0 \leq t_3 \leq t_2 \\ \frac{\mu_1}{\mu_1+\mu_2} e^{-\mu_2 t_2} - \left(\frac{\mu_1}{\mu_1+\mu_2} - \frac{\mu_1}{\mu_1+\mu_2+\mu_3} \right) e^{(\mu_1+\mu_2)t_3} e^{-\mu_2 t_2} & \text{if } t_3 \leq 0 \leq t_2 \\ 1 - \left(1 - \frac{\mu_1}{\mu_1+\mu_2} \right) e^{\mu_1 t_2} - \left(\frac{\mu_1}{\mu_1+\mu_2} - \frac{\mu_1}{\mu_1+\mu_2+\mu_3} \right) e^{(\mu_1+\mu_2)t_3} e^{-\mu_2 t_2} & \text{if } t_3 \leq t_2 \leq 0 \end{cases} \tag{4.61}$$

Case 2) (Partially accepted) Some miners accept the proposed block.

Now we consider that some miners (not every miner) accept the proposed block $B_{H,1}$ by miner 1, without loss of generality, at the round H. Suppose that miner 1 and miner 2 accept $B_{H,1}$, but miner 3 does not accept $B_{H,1}$.

$$\begin{pmatrix} \boxed{a_{11}} & \boxed{a_{12}} & a_{13} \\ a_{21} & a_{22} & \boxed{a_{23}} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} \boxed{a_{11}} & \boxed{a_{12}} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & \boxed{a_{33}} \end{pmatrix} \tag{4.62}$$

Since A_{13}, A_{23}, and A_{33} are disjoint sets, the probability that $\{m_1, m_2\}$ accept $B_{H,1}$ can be computed as follows:

$$\begin{aligned}\mathbb{P}\left[A_{11} \cap A_{12} \cap A_{13}^{\complement}\right] &= \mathbb{P}\left[A_{11} \cap A_{12} \cap A_{23}\right] + \mathbb{P}\left[A_{11} \cap A_{12} \cap A_{33}\right] \\ &= \mathbb{P}\left[A_{11} \cap A_{12}\right] - \mathbb{P}\left[A_{11} \cap A_{12} \cap A_{13}\right]\end{aligned} \tag{4.63}$$

Note that $\mathbb{P}\left[A_{11} \cap A_{12}\right]$ is reiteration of Eq. 4.53 except the parameter. To simplify the notation, we define:

$$\begin{aligned}p_{\text{full},1}\left(y_2, y_3\right) \triangleq \int_0^{\infty} & e^{-\mu_2 \max\left(0, x_1 + o_{H,1} - o_{H,2} + y_2\right)} \\ & \cdot e^{-\mu_3 \max\left(0, x_1 + o_{H,1} - o_{H,3} + y_3\right)} \\ & \cdot \mu_1 e^{-\mu_1 x_1} dx_1\end{aligned} \tag{4.64}$$

We rewrite Eq. 4.63 in terms of $p_{\text{full},1}\left(y_2, y_3\right)$:

$$\begin{aligned}&\mathbb{P}\left[A_{11} \cap A_{12} \cap A_{13}^{\complement}\right] \\ &= p_{\text{full},1}\left(\max(\tau'_{11} - \tau'_{21}, \tau'_{12} - \tau'_{22}), \max(\tau'_{11} - \tau'_{31}, \tau'_{12} - \tau'_{32})\right) - p_{\text{full},1}\left(c_{12}, c_{13}\right)\end{aligned} \tag{4.65}$$

where

$$c_{ik} = \max_{j \in \{1,2,3\}} \left(\tau'_{ij} - \tau'_{kj}\right) \tag{4.66}$$

Similarly, suppose that only miner 1 accepts her block $B_{H,1}$.

$$\begin{aligned}&\mathbb{P}\left[A_{11} \cap A_{12}^{\complement} \cap A_{13}^{\complement}\right] \\ &= \mathbb{P}\left[A_{11} \cap A_{12}^{\complement}\right] - \mathbb{P}\left[A_{11} \cap A_{12}^{\complement} \cap A_{13}\right] \\ &= \mathbb{P}\left[A_{11}\right] - \mathbb{P}\left[A_{11} \cap A_{12}\right] - \mathbb{P}\left[A_{11} \cap A_{13}\right] + \mathbb{P}\left[A_{11} \cap A_{12} \cap A_{13}\right]\end{aligned} \tag{4.67}$$

We rewrite Eq. 4.67 in terms of $p_{\text{full},1}\left(y_2, y_3\right)$:

$$\begin{aligned}&\mathbb{P}\left[A_{11} \cap A_{12}^{\complement} \cap A_{13}^{\complement}\right] \\ &= p_{\text{full},1}\left(\tau'_{11} - \tau'_{21}, \tau'_{11} - \tau'_{31}\right) \\ &- p_{\text{full},1}\left(\max(\tau'_{11} - \tau'_{21}, \tau'_{12} - \tau'_{22}), \max(\tau'_{11} - \tau'_{31}, \tau'_{12} - \tau'_{32})\right) \\ &- p_{\text{full},1}\left(\max(\tau'_{11} - \tau'_{21}, \tau'_{13} - \tau'_{23}), \max(\tau'_{11} - \tau'_{31}, \tau'_{13} - \tau'_{33})\right) \\ &+ p_{\text{full},1}\left(c_{12}, c_{13}\right)\end{aligned} \tag{4.68}$$

We need to further classify the combinations based on the set of miners who accept block $B_{H,1}$. The sum of hash rates determines the probability of whether $B_{H,1}$ survives or not.

1. Accepted by $\{m_1, m_2, m_3\}$ (fully accepted)
2. Accepted by $\{m_1, m_2\}$ (partially accepted)
3. Accepted by $\{m_1, m_3\}$ (partially accepted)
4. Accepted by $\{m_1\}$ (partially accepted)

Other sets of miners without including m_1 (an original block miner), e.g. $\{m_2\}$, $\{m_3\}$, $\{m_2, m_3\}$, are infeasible.

Theorem 4.8. *When miner* 1 *does not accept her block,* $B_{H,1}$*, the other miners do not accept* $B_{H,1}$*.*

Proof. From the definition of the timing matrix, we have:

$$\begin{aligned} & a_{11} > \min(a_{21}, a_{31}) \\ \iff & a_{12} = a_{11} + \tau_{12} > \min(a_{21} + \tau_{12}, a_{31} + \tau_{12}) = \min(a_{22} + \tau_{21} + \tau_{12}, a_{33} + \tau_{31} + \tau_{12}) \end{aligned} \tag{4.69}$$

Miner 2, without loss of generality, compares a_{12} with $\min(a_{22}, a_{32}) = \min(a_{22}, a_{33} + \tau_{32})$. There are two possible subcases. If $a_{12} > a_{22} + \tau_{21} + \tau_{12}$, miner 2 finds that a_{22} is smaller; a_{12} is not smallest because one-hop transmission delays are nonnegative. If $a_{12} > a_{33} + \tau_{31} + \tau_{12}$, we need to assume $\tau_{32} < \tau_{31} + \tau_{12}$, i.e., the indirect path $(3 \to 1 \to 2)$ takes the longer delay than the direct path $(3 \to 2)$. Therefore, miner 2 does not accept $B_{H,1}$. □

For the event that two miners $\{m_1, m_2\}$ accept $B_{H,1}$ (partially accepted), either m_1 or m_2 may propose the next block $B_{H+1,1}$ or $B_{H+1,2}$. Miners are possibly in disagreement for

height $H+1$, but note that their parent block is $B_{H,1}$. In the timing matrix at round $H+1$, the smallest element for each column should be found in either the first or second row.

$$\begin{pmatrix} \boxed{a_{11}} & \boxed{a_{12}} & \boxed{a_{13}} \\ \boxed{a_{21}} & \boxed{a_{22}} & \boxed{a_{23}} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{4.70}$$

Let A'_{ij} denote the event that miner j accepts the block proposed by miner i at round $H+1$.

To simplify the notation, we define:

$$\begin{aligned} p_{\star|1} &\triangleq \mathbb{P}\left[A'_{11} \cap A'_{12} \cap A'_{13}\right] \text{ (full acceptance)} \\ p_{\star|12} &\triangleq \mathbb{P}\left[(A'_{11} \cup A'_{21}) \cap (A'_{12} \cup A'_{22}) \cap (A'_{13} \cup A'_{23})\right] \\ p_{\star|13} &\triangleq \mathbb{P}\left[(A'_{11} \cup A'_{31}) \cap (A'_{12} \cup A'_{32}) \cap (A'_{13} \cup A'_{33})\right] \\ p_{\star|123} &\triangleq \mathbb{P}\left[(A'_{11} \cup A'_{21} \cup A'_{31}) \cap (A'_{12} \cup A'_{22} \cup A'_{32}) \cap (A'_{13} \cup A'_{23} \cup A'_{33})\right] = 1 \end{aligned} \tag{4.71}$$

In other words, $p_{\star|12}$ is the probability that every miner accepts any block mined by $\{m_1, m_2\}$.

Now the probability that every miner accepts either one chain with $B_{H+1,1}$ or another chain with $B_{H+1,2}$ is:

$$\begin{aligned} p_{\star|12} &= 1 - \mathbb{P}\left[(A'_{11} \cup A'_{21})^{\complement} \cup (A'_{12} \cup A'_{22})^{\complement} \cup (A'_{13} \cup A'_{23})^{\complement}\right] \\ &= 1 - \mathbb{P}\left[A'_{31} \cup A'_{32} \cup A'_{33}\right] \end{aligned} \tag{4.72}$$

where

$$\begin{aligned} A'_{31} &= \{(X_1, X_2) | X_3 + o_{H+1,3} + \tau'_{31} < \min(X_1 + o_{H+1,1} + \tau'_{11}, X_2 + o_{H+1,2} + \tau'_{21})\}, \\ A'_{32} &= \{(X_1, X_2) | X_3 + o_{H+1,3} + \tau'_{32} < \min(X_1 + o_{H+1,1} + \tau'_{12}, X_2 + o_{H+1,2} + \tau'_{22})\}, \\ A'_{33} &= \{(X_1, X_2) | X_3 + o_{H+1,3} + \tau'_{33} < \min(X_1 + o_{H+1,1} + \tau'_{13}, X_2 + o_{H+1,2} + \tau'_{23})\} \end{aligned} \tag{4.73}$$

To evaluate the probability of the union of A'_{31}, A'_{32}, and A'_{33}, we need to consider three random variables X_1, X_2, and X_3. Suppose that the sampled value of X_3 is x_3 to reduce the dimension. We have the three regions as follows:

$$
\begin{aligned}
\texttt{R1}: & X_1 > \max(0, x_3 + o_{H+1,3} - o_{H+1,1} + \tau'_{31} - \tau'_{11}) \wedge \\
& X_2 > \max(0, x_3 + o_{H+1,3} - o_{H+1,2} + \tau'_{31} - \tau'_{21}) \\
\texttt{R2}: & X_1 > \max(0, x_3 + o_{H+1,3} - o_{H+1,1} + \tau'_{32} - \tau'_{12}) \wedge \\
& X_2 > \max(0, x_3 + o_{H+1,3} - o_{H+1,2} + \tau'_{32} - \tau'_{22}) \\
\texttt{R3}: & X_1 > \max(0, x_3 + o_{H+1,3} - o_{H+1,1} + \tau'_{33} - \tau'_{13}) \wedge \\
& X_2 > \max(0, x_3 + o_{H+1,3} - o_{H+1,2} + \tau'_{33} - \tau'_{23})
\end{aligned}
\tag{4.74}
$$

Fig. 4.8 depicts the three regions (the slices of $X_3 = x_3$) in the 2D plane (X_1, X_2). To consider the boundaries of the union of the events, we sort the above three corner points by the first dimension X_1. Let (a, b), (c, d) and (e, f) be the sorted points such that $a < c < e$. There are $3! = 6$ permutations for the relative order of the second dimension X_2.

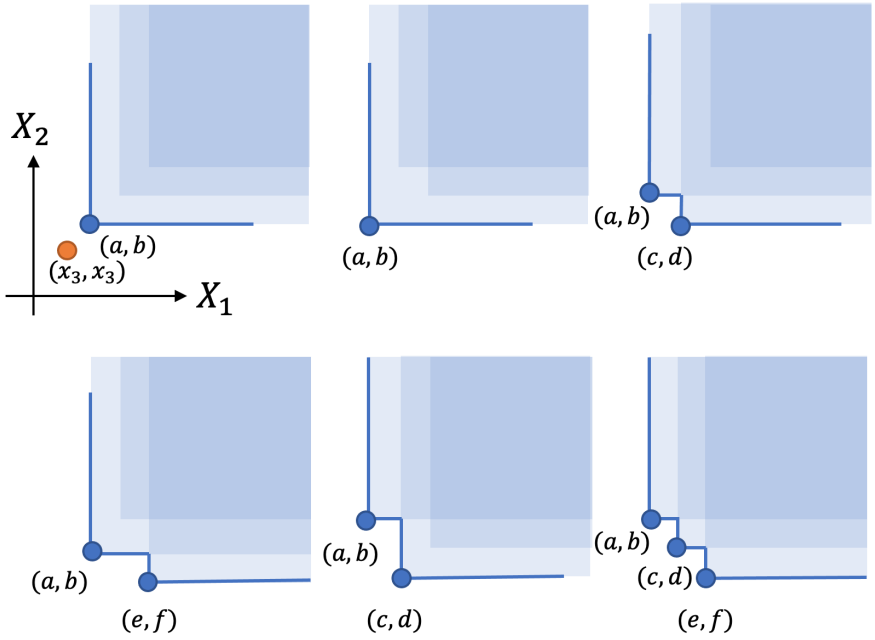

Figure 4.8: Possible boundaries for three miners ($3! = 6$).

Since X_1 and X_2 are mutually independent, we compute the probability of the union of

A'_{31}, A'_{32}, and A'_{33} (conditioned on $X_3 = x_3$) as follows:

$$\mathbb{P}\left[A'_{31} \cup A'_{32} \cup A'_{33} | X_3 = x_3\right]$$

$$= \begin{cases} e^{-\mu_1 a} e^{-\mu_2 b} & \text{if } b < d < f \\ e^{-\mu_1 a} e^{-\mu_2 b} & \text{if } b \leq f \leq d \\ e^{-\mu_1 a} e^{-\mu_2 b} + e^{-\mu_1 c} e^{-\mu_2 d} - e^{-\mu_1 c} e^{-\mu_2 b} & \text{if } d \leq b \leq f \\ e^{-\mu_1 a} e^{-\mu_2 b} + e^{-\mu_1 e} e^{-\mu_2 f} - e^{-\mu_1 e} e^{-\mu_2 b} & \text{if } f \leq b \leq d \\ e^{-\mu_1 a} e^{-\mu_2 b} + e^{-\mu_1 c} e^{-\mu_2 d} - e^{-\mu_1 c} e^{-\mu_2 b} & \text{if } d \leq f \leq b \\ e^{-\mu_1 a} e^{-\mu_2 b} + e^{-\mu_1 c} e^{-\mu_2 d} - e^{-\mu_1 c} e^{-\mu_2 b} + e^{-\mu_1 e} e^{-\mu_2 f} - e^{-\mu_1 e} e^{-\mu_2 d} & \text{if } f \leq d \leq b \end{cases} \tag{4.75}$$

If the delays are small, it is likely that the fork is resolved at the next round $H + 1$. In this case, the three corner points are close to (x_3, x_3). The probability of the events is approximately $\int_0^\infty e^{-\mu_1 x_3} e^{-\mu_2 x_3} \mu_3 e^{-\mu_3 x_3} dx_3 = \frac{\mu_3}{\mu_1+\mu_2+\mu_3}$ and we have:

$$p_{\star|12} = \mathbb{P}\left[(A'_{11} \cup A'_{21}) \cap (A'_{12} \cup A'_{22}) \cap (A'_{13} \cup A'_{23})\right] \approx \frac{\mu_1 + \mu_2}{\mu_1 + \mu_2 + \mu_3} \tag{4.76}$$

However, if the delays are significantly large, the fork may persist over many rounds (more than two rounds). The fork can be resolved at round $H + 2$, $H + 3$, and so on. We further assume that the probability of fork survival for the subsequent rounds is proportional to the sum of hash rates. For example, if the block $B_{H,1}$ is partially accepted by a set of miners $M = \{m_1, m_2\}$, the probability of fork survival conditioned on M is close to $p_{\star|12}$.

We summarize the probabilities in Table 4.2.

Table 4.2: Summary for the probabilities that a set of miners accepts the block

Set of miners, M	M accepts $B_{H,1}$	All accept chain $(\cdots B_{H,1} B_{H+1,k}) : m_k \in M$
$\{m_1, m_2, m_3\}$	$\mathbb{P}\left[A_{11} \cap A_{12} \cap A_{13}\right]$	$p_{\star\|123} = 1$
$\{m_1, m_2\}$	$\mathbb{P}\left[A_{11} \cap A_{12} \cap A^{\complement}_{13}\right]$	$p_{\star\|12} \approx \frac{\mu_1+\mu_2}{\mu_1+\mu_2+\mu_3}$

$\{m_1, m_3\}$	$\mathbb{P}\left[A_{11} \cap A_{12}^{\complement} \cap A_{13}\right]$	$p_{\star\|13} \approx \frac{\mu_1+\mu_3}{\mu_1+\mu_2+\mu_3}$
$\{m_1\}$	$\mathbb{P}\left[A_{11} \cap A_{12}^{\complement} \cap A_{13}^{\complement}\right]$	$p_{\star\|1} \approx \frac{\mu_1}{\mu_1+\mu_2+\mu_3}$

Now we approximate the probability that $B_{H,1}$ survives:

$$\begin{aligned} p_{\text{survival}} \approx & \mathbb{P}\left[A_{11} \cap A_{12} \cap A_{13}\right] \cdot p_{\star|123} \\ & + \mathbb{P}\left[A_{11} \cap A_{12} \cap A_{13}^{\complement}\right] \cdot p_{\star|12} \\ & + \mathbb{P}\left[A_{11} \cap A_{12}^{\complement} \cap A_{13}\right] \cdot p_{\star|13} \\ & + \mathbb{P}\left[A_{11} \cap A_{12}^{\complement} \cap A_{13}^{\complement}\right] \cdot p_{\star|1} \end{aligned} \tag{4.77}$$

We further simplify Eq. 4.77 using the probability of set difference:

$$\begin{aligned} p_{\text{survival}} \approx & \mathbb{P}\left[A_{11} \cap A_{12} \cap A_{13}\right] \cdot \left(p_{\star|123} - p_{\star|12} - p_{\star|13} + p_{\star|1}\right) \\ & + \mathbb{P}\left[A_{11} \cap A_{12}\right] \cdot \left(p_{\star|12} - p_{\star|1}\right) \\ & + \mathbb{P}\left[A_{11} \cap A_{13}\right] \cdot \left(p_{\star|13} - p_{\star|1}\right) \\ & + \mathbb{P}\left[A_{11}\right] \cdot p_{\star|1} \end{aligned} \tag{4.78}$$

In terms of the aggregate delays, we finally have our approximation to p_{survival}:

$$\begin{aligned} & p_{\text{survival}} \approx \\ & \quad p_{\text{full},1}\left(c_{12}, c_{13}\right) \cdot \left(p_{\star|123} - p_{\star|12} - p_{\star|13} + p_{\star|1}\right) \\ & + p_{\text{full},1}\left(\max(\tau'_{11} - \tau'_{21}, \tau'_{12} - \tau'_{22}), \max(\tau'_{11} - \tau'_{31}, \tau'_{12} - \tau'_{32})\right) \cdot \left(p_{\star|12} - p_{\star|1}\right) \\ & + p_{\text{full},1}\left(\max(\tau'_{11} - \tau'_{21}, \tau'_{13} - \tau'_{23}), \max(\tau'_{11} - \tau'_{31}, \tau'_{13} - \tau'_{33})\right) \cdot \left(p_{\star|13} - p_{\star|1}\right) \\ & + p_{\text{full},1}\left(\tau'_{11} - \tau'_{21}, \tau'_{11} - \tau'_{31}\right) \cdot p_{\star|1} \end{aligned} \tag{4.79}$$

where

$$\begin{aligned} p_{\text{full},1}\left(y_2, y_3\right) \triangleq \int_0^\infty & e^{-\mu_2 \max\left(0, x_1 + o_{H,1} - o_{H,2} + y_2\right)} \\ & \cdot e^{-\mu_3 \max\left(0, x_1 + o_{H,1} - o_{H,3} + y_3\right)} \\ & \cdot \mu_1 e^{-\mu_1 x_1} dx_1 \end{aligned} \tag{4.80}$$

In Fig. 4.9, we compare the approximation of the probability of fork survival, Eq. 4.79, and the simulation result. We use the same simulation parameters. We consider three miners whose hash rates are $(\mu_1, \mu_2, \mu_3) = (0.50, 0.25, 0.25) \cdot \mu_b$. We assume that the effect of time skew is zero. The aggregate delays are defined as follows:

$$\tau'_{ij} = \begin{cases} d & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{4.81}$$

where $d \in [0, 600]$. The approximation of p_{survival} captures an effect of interaction between miners. The large miner (50%) takes advantage of the delays whereas the small miners (25%) get some penalties. We further consider an approximation of $p_{\text{full},1}(y_2, y_3)$ by ignoring the max function: $e^{\max(x,0)} = e^x$ if $x \geq 0$. This approximation yields a convenient form as follows:

$$p_{\text{full},1}\left(y_2, y_3\right) \approx \tilde{p}_{\text{full},1}\left(y_2, y_3\right) = \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3} e^{-\mu_2 y_2} e^{-\mu_3 y_3} \tag{4.82}$$

The approximation error can be large, especially when y_2 and y_3 are negative numbers, e.g. $\tau'_{11} - \tau'_{21}, \tau'_{11} - \tau'_{31}$.

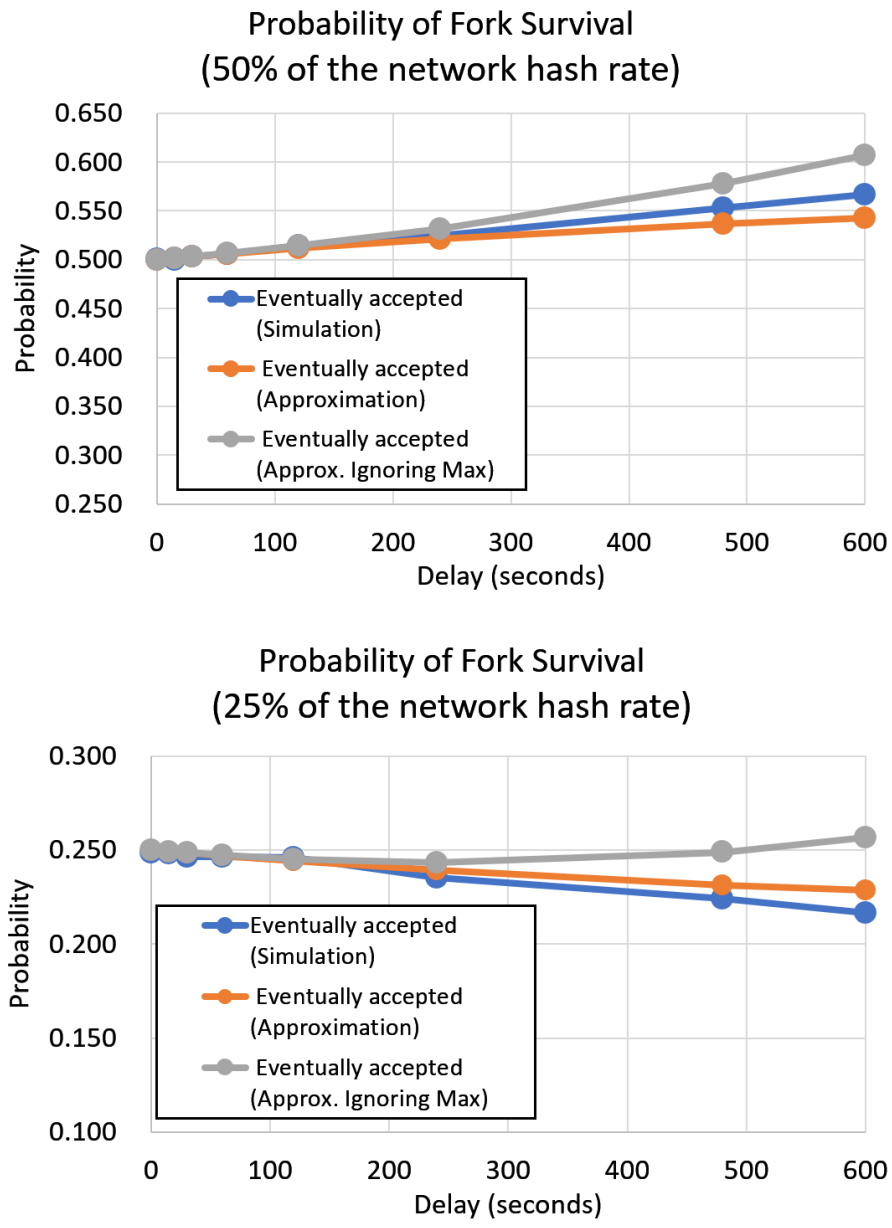

Figure 4.9: The probability of fork survival (special case, n = 3).

4.5 Three Miner Approximation

It is complicated to consider the explosively growing number of combinations for n arbitrary miners. We find that the special case for $n = 3$ helps analyze our problem: an individual miner decides how many transactions she processes for the next block. Let us assume that we have one miner, miner i who processes r_i transactions and that there are in addition two "super" groups of miners: (1) those who never process transactions (i.e., "empty" blocks) and (2) those who prefer to process B transactions (the maximum number of transactions in a block, i.e., "full" blocks). We use the sum of hash rates as a representative for each group. We call this the *three miner approximation* method.

4.5.1 Delay Assumptions

Now we will represent processing delay d_i at miner i and one-hop transmission delay τ_{ij} from miner i to miner j in terms of the number of transactions in a block. Let r_i be the number of transactions in a block proposed by miner i.

$$0 \leq r_i \leq B \tag{4.83}$$

where B is the maximum number of transactions in a block.

Let α_i be the average processing delay per transaction at miner i. Now the processing delay is given by:

$$d_i = \alpha_i r_i \tag{4.84}$$

Let β_{ij} be the average one-hop transmission delay per transaction.

$$\beta_{ij} = \frac{b}{C_{ij}} \tag{4.85}$$

where b is the size of each transaction (bytes), and C_{ij} is a channel capacity (bytes per second) from miner i to miner j.

$$\tau_{ij} = \frac{b}{C_{ij}} r_i = \beta_{ij} r_i \tag{4.86}$$

The aggregate delay τ'_{ij} can be represented in terms of the number of transactions in a block:

$$\tau'_{ij} = d_i + \tau_{ij} = (\alpha_i + \beta_{ij}) r_i \tag{4.87}$$

By using the approximation for $p_{\star|12}$, $p_{\star|13}$, and $p_{\star|1}$ ($p_{\star|123} - p_{\star|12} - p_{\star|13} + p_{\star|1} \approx 0$), we can simplify Eq. 4.79 as follows:

$$\begin{aligned} p_{\text{survival}} \approx & \\ & p_{\text{full},1}\left(\max(\tau'_{11} - \tau'_{21}, \tau'_{12} - \tau'_{22}), \max(\tau'_{11} - \tau'_{31}, \tau'_{12} - \tau'_{32})\right) \cdot \left(\frac{\mu_2}{\mu_1 + \mu_2 + \mu_3}\right) \\ & + p_{\text{full},1}\left(\max(\tau'_{11} - \tau'_{21}, \tau'_{13} - \tau'_{23}), \max(\tau'_{11} - \tau'_{31}, \tau'_{13} - \tau'_{33})\right) \cdot \left(\frac{\mu_3}{\mu_1 + \mu_2 + \mu_3}\right) \\ & + p_{\text{full},1}\left(\tau'_{11} - \tau'_{21}, \tau'_{11} - \tau'_{31}\right) \cdot \left(\frac{\mu_1}{\mu_1 + \mu_2 + \mu_3}\right) \end{aligned} \tag{4.88}$$

where

$$\tau'_{ik} - \tau'_{jk} = (\alpha_i + \beta_{ik}) r_i - (\alpha_j + \beta_{jk}) r_j \tag{4.89}$$

4.5.2 Distribution of Miners

Let G_{bad} be a set of miners who never process transactions and G_{good} be a set of miners who prefer to process B transactions. The sum of hash rates is:

$$h_{\text{good}} = \sum_{i \in G_{\text{good}}} h_i, \quad h_{\text{bad}} = \sum_{i \in G_{\text{bad}}} h_i \tag{4.90}$$

The rate of solving a hash puzzle is proportional to the hash rate.

$$\mu_{\text{good}} = \sum_{i \in G_{\text{good}}} \mu_i, \quad \mu_{\text{bad}} = \sum_{i \in G_{\text{bad}}} \mu_i \tag{4.91}$$

4.5.3 An Example Comparison

In this example comparison, let us assume that there are 15 miners (6 bad miners, 8 good miners, and one joining a group of good miners) whose hash rates are nonuniform (arbitrarily chosen). It is fairly complicated to consider 15 miners using a 15×15 timing matrix. The *three miner approximation* method is useful to obtain the probability of fork survival. We compute the aggregate hash rate of the two groups of miners as follows:

$$\begin{aligned} \frac{\mu_{\text{good}}}{\mu_b} &= \underbrace{0.035}_{\text{oneself}} \\ &\quad + 0.25 + 0.10 + \underbrace{0.035 * 6}_{\text{6 miners}} = 0.035 + 0.560 = 0.595 \\ \frac{\mu_{\text{bad}}}{\mu_b} &= 0.20 + 0.05 + 0.05 + 0.035 + 0.035 + 0.035 = 0.405 \end{aligned} \tag{4.92}$$

Fig. 4.10 compares the empirical probability of fork survival from the simulator and the analytical result from the *three miner approximation* method. An individual miner with 3.5% of the network hash rate experiences a drop in probability as the delay increases. This approximation gives a reasonable value. However, the approximation error can be large when the aggregate delays are large. The different aggregate delays cause the additional errors since these miners cannot be merged into a homogeneous group. Eq. 4.88 will be used in Chapter 5.

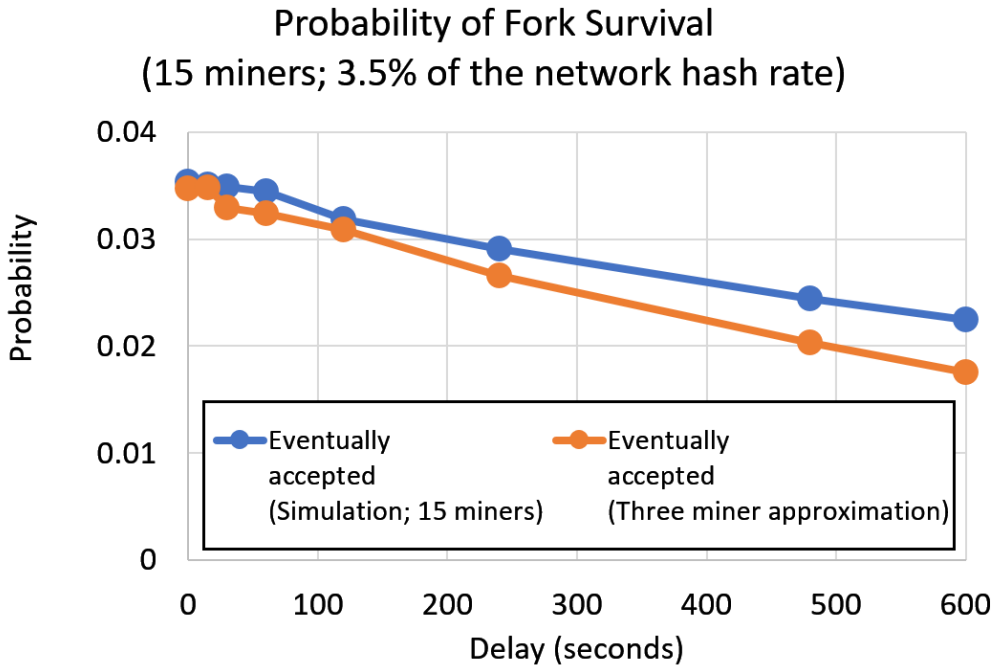

Figure 4.10: Three miner approximation.

4.6 Time Skew

The timestamp at which each miner reaches chain height H is uneven due to the delay. Fig. 4.11 shows that three miners started mining tasks at the exact same time at round $H-1$, i.e., $T_{H-1,1} = T_{H-1,2} = T_{H-1,3}$. Specifically, miner 1 has additional time τ'_{12} for her mining task for round $H+1$ than miner 2. Miner 3 also has additional time τ'_{31} and τ'_{32} than miner 1 and miner 2, respectively. The time skew term further decreases the probability of full acceptance.

We recall from Eq. 4.9, that the time skew in miner i, particularly at round H, can be expressed as follows:

$$o_{H,k} \triangleq T_{H-1,k} - \min_{i\in\{1,\cdots,n\}} T_{H-1,i} \tag{4.93}$$

We find that there exists at least one miner whose time skew term is 0. This can be proven by definition. $T_{H-1,k} - \min_{i\in\{1,\cdots,n\}} T_{H-1,i} = 0$ when $T_{H-1,k}$ is the minimum, i.e.,

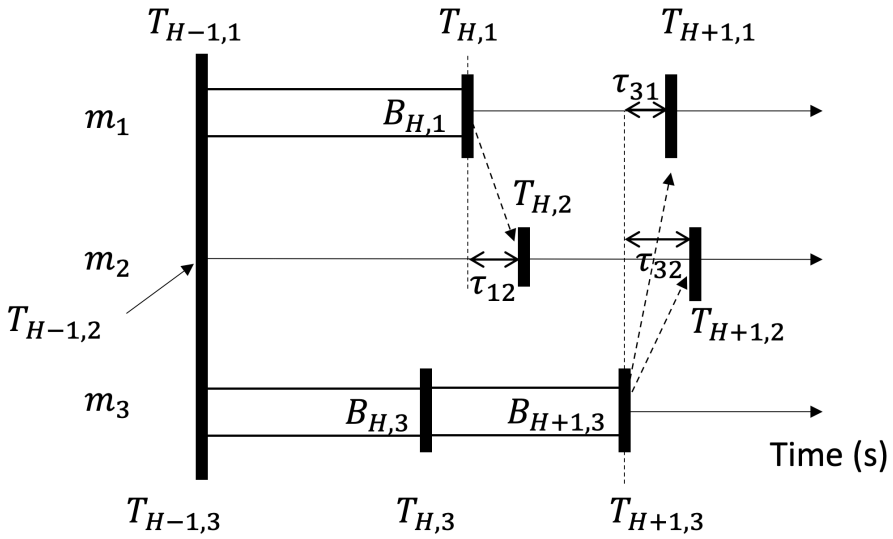

Figure 4.11: Time skew caused by delays.

miner k found block $B_{H-1,k}$ earlier than the other miners.

From the definition of the timing matrix:

$$\begin{aligned} T_{H,k} &= T_{H-1,k} + \min_{j\in\{1,\dots,n\}} a_{jk} \\ &= T_{H-1,k} + \min_{j\in\{1,\dots,n\}} \left(X_j + o_{H,j} + \tau'_{jk}\right) \end{aligned} \tag{4.94}$$

Now we consider:

$$\begin{aligned} o_{H+1,k} &= T_{H,k} - \min_{i\in\{1,\cdots,n\}} T_{H,i} \\ &= T_{H-1,k} + \min_{j\in\{1,\dots,n\}} \left(X_j + o_{H,j} + \tau'_{jk}\right) - \min_{i\in\{1,\cdots,n\}} T_{H,i} \\ &= \min_{j\in\{1,\dots,n\}} \left(X_j + o_{H,j} + \tau'_{jk}\right) - \left(\min_{i\in\{1,\cdots,n\}} T_{H,i} - T_{H-1,k}\right) \end{aligned} \tag{4.95}$$

To simplify the notation, we define:

$$b_k \triangleq \left(\min_{i\in\{1,\cdots,n\}} T_{H,i} - T_{H-1,k}\right) \tag{4.96}$$

The cumulative distribution function of $o_{H+1,k}$ is:

$$\begin{aligned}
\mathbb{P}\left[o_{H+1,k} \leq \omega\right] &= \mathbb{P}\left[\min_{j\in\{1,\ldots,n\}} \left(X_j + o_{H,j} + \tau'_{jk}\right) - b_k \leq \omega\right] \\
&= \mathbb{P}\left[\min_{j\in\{1,\ldots,n\}} \left(X_j + o_{H,j} + \tau'_{jk}\right) \leq \omega + b_k\right] \\
&= 1 - \mathbb{P}\left[\min_{j\in\{1,\ldots,n\}} \left(X_j + o_{H,j} + \tau'_{jk}\right) > \omega + b_k\right] \\
&= 1 - \mathbb{P}\left[\bigwedge_{j=1}^{n} \left(X_j + o_{H,j} + \tau'_{jk} > \omega + b_k\right)\right]
\end{aligned} \tag{4.97}$$

Since $o_{H+1,k}$ is a nonnegative value, we may obtain the expectation from the cumulative distribution function.[7]

$$\begin{aligned}
\mathbb{E}\left[o_{H+1,k}\right] &= \int_0^\infty \left(1 - \mathbb{P}\left[o_{H+1,k} \leq \omega\right]\right) d\omega \\
&= \int_0^\infty \mathbb{P}\left[\bigwedge_{j=1}^{n} \left(X_j + o_{H,j} + \tau'_{jk} > \omega + b_k\right)\right] d\omega \\
&= \int_0^\infty \prod_{j=1}^{n} e^{-\mu_j \max(0, \omega + b_k - o_{H,j} - \tau'_{jk})} d\omega
\end{aligned} \tag{4.98}$$

For a rough approximation:

$$\mathbb{E}\left[o_{H+1,k}\right] \approx \frac{\prod_{j=1}^{n} e^{\mu_j(-b_k + o_{H,j} + \tau'_{jk})}}{\sum_{j=1}^{n} \mu_j} \tag{4.99}$$

Note that processing and one-hop transmission delays $\tau'_{jk} = d_j + \tau_{jk}$ increase the time skew experienced by miner k.

[7]However, b_k and o_H remain in the equation. We leave it for future work to deal with these quantities.

4.7 Conclusion

In this chapter, we analyzed a block mining process with processing and network delays. We introduced a timing matrix that represents block arrival timings between pairwise two nodes. We provided the formulae for the lower and upper bound of the probability of fork survival, i.e., winning the mining competition. We also provided analytical expressions that approximate the probability of fork survival, namely Eq. 4.79, for two miners and three miners. We found that not only the hash rates but also the delays affect the probability of fork survival. We discovered an effect of interaction between miners that a large miner takes advantage of the delays, and the opposite effect exists for small miners. We found that the *three miner approximation* method provides simpler, but approximate expressions for the fork survival probability for many miners with two strategies. We also studied the time skew caused by delays.

CHAPTER 5

Optimal Block Reward Policy

5.1 Introduction

The block reward and transaction fees incentivize miners to participate in the intense mining competition and hopefully to process as many transactions as possible. The protocol defines the block reward, and the transaction fee market determines the level of transaction fees as we studied in Chapter 3.

Any rational miner wants to maximize her expected mining revenue. In an ideal world where processing and network delays are zero, the probability of fork survival, i.e., winning the mining competition, is purely a function of computing power. In this case, the optimal strategy for the miner is to process transactions until she reaches the block size limit. This would be the intention of the protocol designer.

As we studied in Chapter 4, adding more transactions in a block negatively affects the probability of fork survival, especially with significant delays. When a miner processes transactions, she decides how many transactions she will include in a new block. With a large block reward, the optimal strategy might be to skip transactions. Such a reward policy, which is not incentive-compatible, may cause severe performance degradation.

In this chapter, we analyze the behavior of miners. We derive the optimum number of transactions from the probability of fork survival. We then identify the operating region of a block reward that pushes the system throughput to maximum designed capacity. We also propose a dynamic block reward that may further improve the system throughput.

Table 5.1 summarizes the notation.

Table 5.1: Notation

Symbol	Description	Unit
r_i	Number of transactions in a block proposed by miner i	(TX)
r_i^*	Optimum number of transactions for miner i that maximizes her expected mining revenue	(TX)
B	Maximum number of transactions in a block	(TX)
κ	Average transaction fee per transaction	₿/(TX)
α_i	Processing delay per transaction at miner i	s/(TX)
β_{ij}	Transmission delay per transaction from miner i to miner j	s/(TX)
R_{B}	Block reward	₿
$R_{\mathrm{B},0}$	Base block reward (for a dynamic block reward)	₿
R_{S}	Subsidy for processing a transaction (for a dynamic block reward)	₿/(TX)
R_{TX}	Sum of transaction fees (total transaction fee)	₿

5.2 Reward Policy

A proof-of-work blockchain commonly defines a reward policy with the following rewards.[1]

- A block reward R_{B}
- The sum of transaction fees R_{TX}

In this chapter, we assume that the sum of transaction fees R_{TX} is given. Let κ denote the average transaction fee per transaction.

[1]There can be special types of rewards like an *uncle* reward in Ethereum.

In our model, the sum of transaction fees can be expressed as follows:

$$R_{\mathrm{TX}} = \kappa \cdot r_i \tag{5.1}$$

where r_i is the number of transactions processed by miner i.[2]

5.3 Optimum Number of Transactions

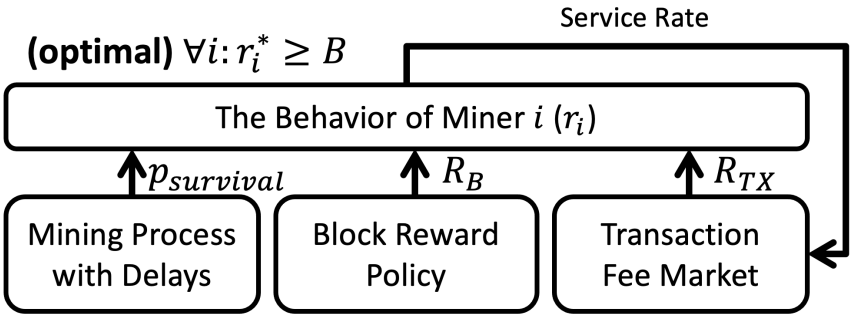

Figure 5.1: The reward policy affects the behavior of miners.

A miner claims the block reward R_{B} and transaction fees R_{TX} once her block is accepted. For the expected mining revenue as shown in Eq. 5.2, one should consider the probability of fork survival, p_{survival}. Fig. 5.1 shows the factors that affect the behavior of miner i, i.e., the number of transactions processed by miner i.

$$\begin{aligned} \mathbb{E}\left[R_{\mathrm{B}} + R_{\mathrm{TX}}\right] &= \left(R_{\mathrm{B}} + R_{\mathrm{TX}}\right) \cdot p_{\mathrm{survival}} \\ &= \underbrace{\left(R_{\mathrm{B}} + \kappa \cdot r_i\right)}_{\text{increasing as } r_i\uparrow} \cdot \underbrace{p_{\mathrm{survival}}(r_i)}_{\text{decreasing as } r_i\uparrow} \end{aligned} \tag{5.2}$$

[2]We also assume that κ is constant. Suppose that no miner processes any transactions; then the corresponding system utilization factor is $\sum_{i=k}^{P} \lambda_i/(\mu \cdot 0) = \infty$. As we studied in Chapter 3, the level of transaction fee will rise and there exists an equilibrium. We leave the study of such a system for future work.

To continue the analysis, we now resort to our *three miner approximation* method. By plugging Eq. 4.89 into Eq. 4.88, we represent p_{survival} as follows:

$$\begin{aligned} p_{\text{survival}}(r_1) \approx \frac{1}{\mu_1+\mu_2+\mu_3}\Bigg(\\ \mu_2 \cdot p_{\text{full},1}\left(\max_{k\in\{1,2\}}((\alpha_1+\beta_{1k})r_1-(\alpha_2+\beta_{2k})r_2), \max_{k\in\{1,2\}}((\alpha_1+\beta_{1k})r_1-(\alpha_3+\beta_{3k})r_3)\right) \\ +\mu_3 \cdot p_{\text{full},1}\left(\max_{k\in\{1,3\}}((\alpha_1+\beta_{1k})r_1-(\alpha_2+\beta_{2k})r_2), \max_{k\in\{1,3\}}((\alpha_1+\beta_{1k})r_1-(\alpha_3+\beta_{3k})r_3)\right) \\ +\mu_1 \cdot p_{\text{full},1}\left((\alpha_1+\beta_{11})r_1-(\alpha_2+\beta_{21})r_2, (\alpha_1+\beta_{11})r_1-(\alpha_3+\beta_{31})r_3\right)\Bigg) \end{aligned} \tag{5.3}$$

where

$$\begin{aligned} p_{\text{full},1}(y_2,y_3) \triangleq \int_0^\infty & e^{-\mu_2 \max(0, x_1+o_{H,1}-o_{H,2}+y_2)} \\ & \cdot e^{-\mu_3 \max(0, x_1+o_{H,1}-o_{H,3}+y_3)} \\ & \cdot \mu_1 e^{-\mu_1 x_1} dx_1 \end{aligned} \tag{5.4}$$

Now we consider a partial derivative of the expected mining revenue of miner 1, Eq. 5.2, with respect to the number of transactions, r_1:

$$\frac{\partial \mathbb{E}[R_{\text{B}}+R_{\text{TX}}]}{\partial r_1} = \kappa \cdot p_{\text{survival}}(r_1) + (R_{\text{B}}+\kappa \cdot r_1) \cdot \frac{\partial p_{\text{survival}}(r_1)}{\partial r_1} \tag{5.5}$$

When the probability of fork survival is independent of delays, i.e., $\frac{\partial p_{\text{survival}}(r_1)}{\partial r_1} = 0$, Eq. 5.5 is nonnegative ($\kappa \geq 0$ and $p_{\text{survival}}(r_1) \geq 0$). The revenue increases with r_1. Thus, miners will process transactions until they reach the hard limit B to maximize their mining revenue. This agrees with intuition.

By assuming the concavity of Eq. 5.2, we may find the optimum number of transactions r_1^* as follows:

$$\left.\frac{\partial \mathbb{E}[R_{\text{B}}+R_{\text{TX}}]}{\partial r_1}\right|_{r_1=r_1^*} = 0 \tag{5.6}$$

We approximate $p_{\text{full},1}(y_2, y_3)$ as Eq. 4.55 which is a common case and we assume that the difference of the two time skew terms is negligible:

$$p_{\text{full},1}(y_2, y_3) \approx \tilde{p}_{\text{full},1}(y_2, y_3) = \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3} \cdot e^{-\mu_2 y_2} e^{-\mu_3 y_3} \tag{5.7}$$

By taking a partial derivative of Eq. 5.7, we have:

$$\begin{aligned} \frac{\partial \tilde{p}_{\text{full},1}(y_2, y_3)}{\partial r_1} &= \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3} \cdot \left(-\mu_2 \frac{\partial y_2}{\partial r_1} - \mu_3 \frac{\partial y_3}{\partial r_1} \right) \cdot e^{-\mu_2 y_2 - \mu_3 y_3} \\ &= \left(-\mu_2 \frac{\partial y_2}{\partial r_1} - \mu_3 \frac{\partial y_3}{\partial r_1} \right) \cdot \tilde{p}_{\text{full},1}(y_2, y_3) \end{aligned} \tag{5.8}$$

Now we consider a partial derivative of the max function in Eq. 5.3.

$$\frac{\partial \max_{k\in\{1,2\}} ((\alpha_1 + \beta_{1k}) r_1 - (\alpha_2 + \beta_{2k}) r_2)}{\partial r_1} = \max_{k\in\{1,2\}} (\alpha_1 + \beta_{1k}) \tag{5.9}$$

By combining these, we have:

$$\begin{aligned} &\frac{\partial \tilde{p}_{\text{survival}}(r_1)}{\partial r_1} = \\ &\frac{\mu_2}{\mu_1 + \mu_2 + \mu_3} \cdot \left(-\mu_2 \max_{k\in\{1,2\}} (\alpha_1 + \beta_{1k}) - \mu_3 \max_{k\in\{1,2\}} (\alpha_1 + \beta_{1k}) \right) \cdot \tilde{p}_{\text{full},1}(y_{2a}, y_{3a}) \\ &+ \frac{\mu_3}{\mu_1 + \mu_2 + \mu_3} \cdot \left(-\mu_2 \max_{k\in\{1,3\}} (\alpha_1 + \beta_{1k}) - \mu_3 \max_{k\in\{1,3\}} (\alpha_1 + \beta_{1k}) \right) \cdot \tilde{p}_{\text{full},1}(y_{2b}, y_{3b}) \\ &+ \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3} \cdot (-\mu_2(\alpha_1 + \beta_{11}) - \mu_3(\alpha_1 + \beta_{11})) \cdot \tilde{p}_{\text{full},1}(y_{2c}, y_{3c}) \end{aligned} \tag{5.10}$$

where

$$\begin{aligned} y_{2a} &\triangleq \max_{k\in\{1,2\}} ((\alpha_1 + \beta_{1k}) r_1 - (\alpha_2 + \beta_{2k}) r_2) \\ y_{3a} &\triangleq \max_{k\in\{1,2\}} ((\alpha_1 + \beta_{1k}) r_1 - (\alpha_3 + \beta_{3k}) r_3) \\ y_{2b} &\triangleq \max_{k\in\{1,3\}} ((\alpha_1 + \beta_{1k}) r_1 - (\alpha_2 + \beta_{2k}) r_2) \\ y_{3b} &\triangleq \max_{k\in\{1,3\}} ((\alpha_1 + \beta_{1k}) r_1 - (\alpha_3 + \beta_{3k}) r_3) \\ y_{2c} &\triangleq (\alpha_1 + \beta_{11}) r_1 - (\alpha_2 + \beta_{21}) r_2 \\ y_{3c} &\triangleq (\alpha_1 + \beta_{11}) r_1 - (\alpha_3 + \beta_{31}) r_3 \end{aligned} \tag{5.11}$$

We assume that a one-hop transmission delay for a loopback is zero, i.e., $\beta_{11} = 0$ and the channel capacity is restricted by the source node, i.e., $\beta_{12} = \beta_{13} = \beta_1$. We further simplify the max functions in Eq. 5.10 as follows:

$$\begin{aligned}
&\frac{\partial \tilde{p}_{\text{survival}}(r_1)}{\partial r_1} = \\
&\frac{\mu_2}{\mu_1+\mu_2+\mu_3} \cdot \left(-\mu_2\left(\alpha_1+\beta_{12}\right) - \mu_3\left(\alpha_1+\beta_{12}\right)\right) \cdot \tilde{p}_{\text{full},1}\left(y_{2a}, y_{3a}\right) \\
&+ \frac{\mu_3}{\mu_1+\mu_2+\mu_3} \cdot \left(-\mu_2\left(\alpha_1+\beta_{13}\right) - \mu_3\left(\alpha_1+\beta_{13}\right)\right) \cdot \tilde{p}_{\text{full},1}\left(y_{2b}, y_{3b}\right) \\
&+ \frac{\mu_1}{\mu_1+\mu_2+\mu_3} \cdot \left(-\mu_2(\alpha_1) - \mu_3(\alpha_1)\right) \cdot \tilde{p}_{\text{full},1}\left(y_{2c}, y_{3c}\right) \\
&= -(\mu_2+\mu_3) \cdot (\alpha_1+\beta_1) \cdot \tilde{p}_{\text{survival}}(r_1) + (\mu_2+\mu_3) \cdot \beta_1 \cdot \frac{\mu_1}{\mu_1+\mu_2+\mu_3} \cdot \tilde{p}_{\text{full},1}(y_{2c}, y_{3c})
\end{aligned} \tag{5.12}$$

Note that Eq. 5.12 is not positive, i.e., processing and one-hop transmission delays decrease the probability of fork survival.

To simplify the notation, we define:

$$\zeta(r_1) \triangleq (\mu_2+\mu_3) \cdot \beta_1 \cdot \frac{\mu_1}{\mu_1+\mu_2+\mu_3} \cdot \tilde{p}_{\text{full},1}(y_{2c}, y_{3c}) \tag{5.13}$$

Note that $\zeta(r_1) \geq 0$ as every term is nonnegative.

Specifically, we represent Eq. 5.5 in terms of the number of transactions r_1 as follows:

$$\frac{\partial\, \mathbb{E}\left[R_{\text{B}} + R_{\text{TX}}\right]}{\partial r_1} = \kappa \cdot \tilde{p}_{\text{survival}}(r_1) - \left(R_{\text{B}} + \kappa \cdot r_1\right)\left((\mu_2+\mu_3) \cdot (\alpha_1+\beta_1) \cdot \tilde{p}_{\text{survival}}(r_1) - \zeta(r_1)\right) \tag{5.14}$$

We next obtain an explicit form for the optimum number of transactions r_1^*:

$$\kappa \cdot \tilde{p}_{\text{survival}}(r_1^*) - \left(R_{\text{B}} + \kappa \cdot r_1^*\right) \cdot \left((\mu_2+\mu_3) \cdot (\alpha_1+\beta_1) \cdot \tilde{p}_{\text{survival}}(r_1^*) - \zeta(r_1^*)\right) = 0 \tag{5.15}$$

By rearranging Eq. 5.15, we have:

$$r_1^* = \frac{1}{(\mu_2+\mu_3)(\alpha_1+\beta_1) - \tilde{\zeta}(r_1^*)} - \frac{R_{\text{B}}}{\kappa} \geq \frac{1}{(\mu_2+\mu_3)(\alpha_1+\beta_1)} - \frac{R_{\text{B}}}{\kappa} \tag{5.16}$$

where $\tilde{\zeta}(r_1^*) \triangleq \frac{\zeta(r_1^*)}{\tilde{p}_{\text{survival}}(r_1^*)} \geq 0$.

5.4 Operating Range of Block Reward

Every rational miner tries to maximize her mining revenue. In an ideal situation where delays do not exist, miners are always willing to process transactions until they reach the hard limit of B. In the real world with inevitable processing and network delays, miners will consider the implication of including transactions to their rewards. As we analyzed it in the previous section, the delays decrease the probability of fork survival, i.e., winning the mining competition. That is, including more transactions does not necessarily mean more expected mining revenue.

In this section, we seek to identify the operating range of the block reward R_{B} such that the optimum number of transactions r_i^* is greater than or equal to B as follows, that is we seek to have the following hold:

$$\forall i : r_i^* \geq B \tag{5.17}$$

That is, we are interested in incentivizing miners to mine full blocks.

From Eq. 5.16 and 5.17, we obtain the following inequality for miner i:

$$R_{\mathrm{B}} \leq \kappa \left(\frac{1}{(\mu_b - \mu_i)(\alpha_i + \beta_i) - \tilde{\zeta}(r_i^*)} - B \right) \tag{5.18}$$

where $\mu_1 + \mu_2 + \mu_3 = \mu_b$.

This is a key result of this research. For every miner i the above condition, Eq. 5.18, should hold.[3] If the processing and one-hop transmission delays are negligible, we get $\alpha_i + \beta_i \approx 0 \Rightarrow R_B < \infty$, i.e., any block reward can be chosen. When the operating range of R_{B} is negative, Eq. 5.18 does not hold ($R_{\mathrm{B}} \geq 0$).

On the other hand, transaction fees may also affect the behavior of miners. Now we

[3]The result in [YKJ18] is a relaxed condition of $B = 0$. The assumption that the probability of winning the mining competition is proportional to the relative hash rate does not reflect the effect of interaction between miners discussed in Chapter 4.

rearrange the condition in Eq. 5.18 and express it in terms of κ:

$$\kappa \geq \frac{R_B\left((\mu_b - \mu_i)(\alpha_i + \beta_i) - \tilde{\zeta}(r_i^*)\right)}{1 - B\left((\mu_b - \mu_i)(\alpha_i + \beta_i) - \zeta(\tilde{r}_i^*)\right)} \tag{5.19}$$

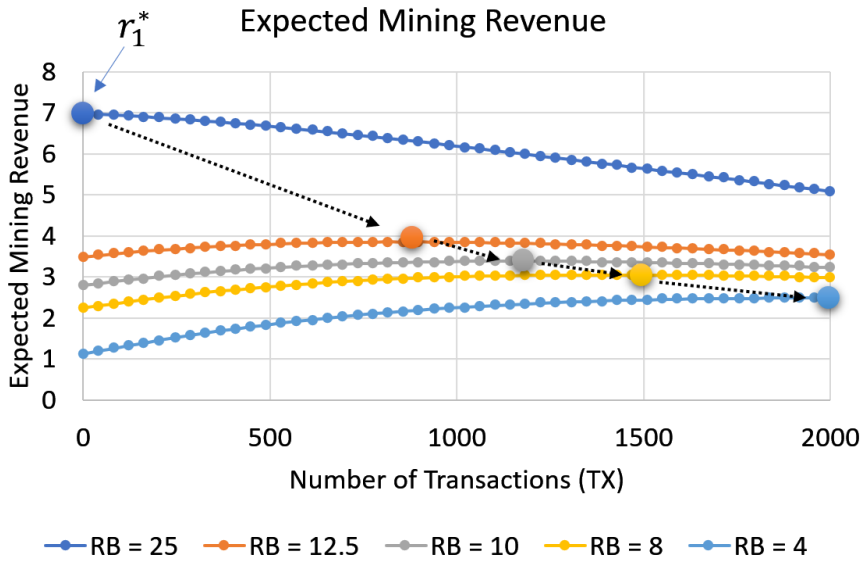

Figure 5.2: The expected mining reward ($\kappa = 0.008$).

Fig. 5.2 shows an example of the expected mining revenue as a function of the number of transactions in a block proposed by miner i. We assume $R_{\text{B}} \in \{25\text{฿}, 12.5\text{฿}, 10\text{฿}, 8\text{฿}, 4\text{฿}\}$, $R_{\text{TX}} = 8\text{฿}$, $(\mu_1, \mu_2, \mu_3) = (0.25, 0.50, 0.25) \cdot \mu_b$, $r_1 \in [0, 2000]$, $r_2 = 0$, $r_3 = 1000$ and $\alpha_i = \beta_i = 0.2$.[4] In this example, we assume that one group of miners (50%) does not process transactions. Another group of miners (25%) processes r_3 transactions.

The solid dots indicate the peak of each mining revenue curve and the optimum number of transactions that achieve that peak. While a large block reward gives more mining reward to a miner, it does not necessarily encourage miners to process transactions. With the large block reward ($R_{\text{B}} = 25\text{฿}$), we observe that the optimum number of transactions is near

[4]We exaggerate transaction fees because of the exaggerated delay. In practice, the delay in Bitcoin is small compared to its block creation time ($\approx 600s$).

0 transactions.[5] With the small block reward ($R_{\mathrm{B}} = 4$Ƀ), we observe that the optimum number of transactions is near B.

5.5 Dynamic Block Reward

In this section, we attempt to revise the current reward policy of giving a block reward R_{B} plus transaction fees R_{TX}. The block reward is constant in a short period, while the coin issuance schedule changes the block reward over a long period.

We now consider a **dynamic** block reward by adding a proportional subsidy R_{S} for each transaction processed, i.e.,

$$\begin{aligned} \texttt{(static block reward)} &:= \quad (R_B + R_{\mathrm{TX}}) \\ \texttt{(dynamic block reward)} &:= \quad (R_{B,0} + R_{\mathrm{S}} \cdot r_1 + R_{\mathrm{TX}}) \end{aligned} \tag{5.20}$$

where $R_{B,0}$ denotes a constant base block reward.

Similarly, Eq. 5.5 becomes:

$$\begin{aligned} &\frac{\partial \, \mathbb{E}\left[R_{\mathrm{B},0} + R_{\mathrm{S}} \cdot r_1 + R_{\mathrm{TX}}\right]}{\partial r_1} = \\ &(\kappa + R_{\mathrm{S}}) \cdot p_{\mathrm{survival}}(r_1) + (R_{\mathrm{B},0} + R_{\mathrm{S}} \cdot r_1 + \kappa \cdot r_1) \cdot \frac{\partial p_{\mathrm{survival}}(r_1)}{\partial r_1} \end{aligned} \tag{5.21}$$

Note that the system creates the subsidy R_{S} and users pay the transaction fees R_{TX}.

Now we obtain the optimum number of transactions as follows:

$$r^*_{1,\mathrm{dynamic}} = \frac{1}{(\mu_2 + \mu_3)(\alpha_1 + \beta_1) - \tilde{\zeta}(r^*_1)} - \frac{R_{\mathrm{B},0}}{\kappa + R_{\mathrm{S}}} \tag{5.22}$$

If we suppose that $\frac{R_{\mathrm{B},0}}{\kappa + R_{\mathrm{S}}} \leq \frac{R_{\mathrm{B}}}{\kappa}$ and $\tilde{\zeta}(\cdot)$ is negligible, we obtain the improved optimum number of transactions under the dynamic block reward policy:

$$r^*_{1,\mathrm{dynamic}} \geq r^*_1 \tag{5.23}$$

[5] Due to the effect of interaction between miners, the expected mining revenue at $r_1 = 0$ is 6.98Ƀ > 6.25Ƀ $(= 25$Ƀ $\cdot 0.25)$.

We also obtain the operating region of the base block reward $R_{\mathrm{B},0}$:

$$R_{\mathrm{B},0} \leq (\kappa + R_{\mathrm{S}}) \left(\frac{1}{(\mu_2 + \mu_3)(\alpha_1 + \beta_1) - \tilde{\zeta}(r_1^*)} - B \right) \tag{5.24}$$

Since the system gives out the additional reward for processing transactions, some miners may abuse the policy. They can create internal transactions to get more rewards (subsidies). It takes almost no cost to create wallet accounts. For example, suppose that a miner has two wallet accounts: `addr1` and `addr2`. Then the miner puts transactions like

$$\begin{aligned} &\text{send } x\text{฿ from } \texttt{addr1} \text{ to } \texttt{addr2} \\ &\text{send } x\text{฿ from } \texttt{addr2} \text{ to } \texttt{addr1} \text{ (the opposite)} \end{aligned} \tag{5.25}$$

in her block.[6]

Let $r_{1,\mathrm{user}}$ be the number of real transactions and $r_{1,\mathrm{miner}}$ be the number of internally generated transactions created by a miner.

$$r_1 = r_{1,\mathrm{user}} + r_{1,\mathrm{miner}} \tag{5.26}$$

Since the net mining transaction revenue from the internal transactions is zero, we have the expected mining revenue as follows:

$$\mathbb{E}\left[R_{\mathrm{B},0} + R_{\mathrm{S}} \cdot r_1 + R_{\mathrm{TX}}\right] = (R_{\mathrm{B},0} + R_{\mathrm{S}} \cdot (r_{1,\mathrm{user}} + r_{1,\mathrm{miner}}) + \kappa \cdot r_{1,\mathrm{user}}) \cdot p_{\mathrm{survival}}(r_1) \tag{5.27}$$

The level of system load determines whether a miner creates internal transactions or not. If the system is running at its full capacity, $r_{1,\mathrm{user}} + r_{1,\mathrm{miner}} = B$ holds. In this case, a miner does not get an additional incentive to create internal transactions. However, if the system is underloaded, i.e., $r_{1,\mathrm{user}} < B$, the miner now has an incentive to create internal transactions up to $B - r_{1,\mathrm{user}}$. Therefore, the system running at full capacity may adopt the dynamic reward policy.

[6]Internal transactions might be indistinguishable from normal transactions as a miner forms a long cyclic path in a graph of transactions.

In summary, the dynamic block reward has the benefits only if the system runs at its full capacity as follows:[7]

- The operating region of a block reward (a base block reward) gets extended.
- The optimum number of transactions increases; the system throughput gets improved.
- The average waiting time and transaction fee get lowered because of less intense demand in the transaction fee market.

5.6 Conclusion

In this chapter, we studied a reward policy on block creation in a proof-of-work blockchain. We analyzed the decision-making of block miners in terms of the number of transactions they are willing to include in a block. Our results showed that (1) in an ideal world where the delays are negligible, block miners process transactions until they reach the designed capacity and (2) in a real-world where the delays are not zero, the optimum number of transactions can be below the block size limit. We also identified the operating range of a block reward that pushes the optimum point beyond the designed capacity (which leads us to full blocks, our desired outcome). Finally, we proposed a dynamic block reward of a proportional subsidy for processing transactions. We showed that it further improves the system throughput as the optimum number of transactions increases and the operating region of a block reward gets extended. Therefore, blockchain users can have a better experience: (1) reduced waiting time and (2) less expensive transaction fees.

[7] In practice, there are a sufficient number of pending transactions in a popular blockchain.

CHAPTER 6

Conclusion

In a decentralized system, it is essential to define an incentive scheme properly to encourage anonymous participants to take the desired actions. However, the ideal and the real world gap is always wider than expected. There are many unknown (and sometimes ambiguous) external factors. In this dissertation, we mathematically analyzed a reward policy in a proof-of-work blockchain. The reward policy that we analyzed is relatively manageable compared to the human world: it consists of (1) a block reward and (2) transaction fees.

In Chapter 3, we used queueing theory to model a transaction fee market as a head-of-line bulk service queue with multiple priority groups. We obtained an analytical expression for the average waiting time for each priority group with an arbitrary service time distribution. We showed that the numerical integration of the formula matches well with the simulation results. We used a Hilbert transform approach which does not require evaluating system stationary probabilities.

We derived the distribution of the minimum group index in this priority queue, which yields the equilibrium price in the transaction fee market. We showed that the minimum transaction fee in a block comes from the minimum group index with the fee distribution as a weight. We also derived the analytical expression for the average total transaction fee in a block. We provided concrete solutions for an exponential service time distribution. We showed that the formulae match well with simulation results.

In Chapter 4, we analyzed a block mining process that included processing and network delays. We aimed to obtain the analytical expression for the probability of fork survival,

i.e., winning the mining competition. We provided lower and upper bound formulae of the probability of fork survival. We also offered the analytical expressions to approximate the probability for special cases for two miners and three miners. We found an effect of interaction between miners that a large miner takes advantage of the delays and small miners experience the opposite effect. We also found that the *three miner approximation* method is helpful to evaluate the probability for many miners with two strategies. We also studied the time skew caused by delays.

Lastly, in Chapter 5, we analyzed the reward policy on block creation in a proof-of-work blockchain. We showed that the processing and network delays (as hidden costs) distort the reward policy. We calculated the number of transactions in a block that maximizes the expected mining revenue. We then identified the operating range of a block reward that encourages miners to create fully filled blocks. We proposed a dynamic block reward of a proportional subsidy for processing transactions. We showed that the revised reward policy further improves the system throughput, and there is no incentive to abuse the policy only if the system runs at its designed capacity. Thus, blockchain users can have a better experience.

APPENDIX A

Number of Customers in an $M/M^B/1$ Queue

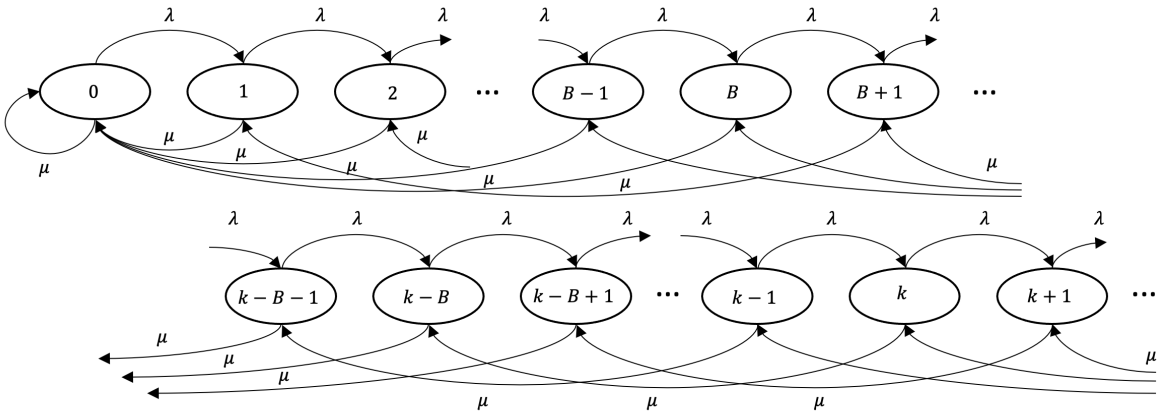

Figure A.1: The state-transition-rate diagram for a bulk service queue ($M/M^B/1$).

In [Kle75, p.138], the distribution of the number of customers in an $M/M^B/1$ queue is solved. Fig. A.1 depicts the state-transition-rate diagram. The state j represents that there are j customers in the queue. We start with a set of equations for the stationary probabilities p_j $(j = 0, 1, \cdots)$.

For the general case we have:

$$(\lambda + \mu)p_j = \mu p_{j+B} + \lambda p_{j-1} \tag{A.1}$$

where $j \geq 1$.

With the boundary condition:

$$\lambda p_0 = \mu(p_1 + \cdots + p_B) \tag{A.2}$$

By solving the above two equations, we have the following probability generating function $P(z)$,

$$P(z) = \frac{\sum_{j=0}^{B-1} p_j(z^j - z^B)}{B\rho z^{B+1} - (1 + B\rho)z^B + 1} \tag{A.3}$$

where $\rho = \frac{\lambda}{\mu B}$.

Clearly, one of the denominator roots is $z = 1$, and B additional roots besides $z = 1$ exist. Since $P(z)$ is a probability generating function, the region of convergence (ROC) is $|z| < 1$. The numerator has one zero at $z = 1$ and $B - 1$ additional zeros. For $P(z)$ to have a bounded value in the ROC $|z| < 1$, then $B - 1$ zeros in the numerator must be the same as $B - 1$ zeros of the denominator. Otherwise, the ROC does not hold. Therefore, besides the denominator root at $z = 1$ the remaining zero z_0 in the denominator must be located at $|z| > 1$ which is outside the ROC. We then numerically solve for the zero z_0 outside the ROC ($z_0 > 1$).

$$B\rho z_0^{B+1} - (1 + B\rho)z_0^B + 1 = 0 \tag{A.4}$$

The stationary probability p_j for every possible state j is then given by

$$p_j = \left(1 - \frac{1}{z_0}\right)\left(\frac{1}{z_0}\right)^j \tag{A.5}$$

where we have used the fact that $P(1) = 1$.

APPENDIX B

Derivation for the Average Total Transaction Fee

Consider $\mathbb{E}[R_{\mathrm{TX}}]$, the expected value of the transaction fee as given by the following equation, Eq. 3.72:

$$\begin{aligned}\mathbb{E}[R_{\mathrm{TX}}] = \mathbb{E}\left[\sum_{i=1}^{P} g_i N_i | X = 0\right] \mathbb{P}[X = 0] \\ + \sum_{k=1}^{P} \left(\mathbb{E}\left[g_k B + \sum_{i=k+1}^{P} (g_i - g_k) N_i | X = k \right] \mathbb{P}[X = k] \right)\end{aligned} \tag{B.1}$$

To simplify the notation, we define $A_{ij} \triangleq \mathbb{E}[N_i | X = j]$ and $p_k \triangleq \mathbb{P}[X = k]$.

$$\mathbb{E}[R_{\mathrm{TX}}] = \tag{B.2}$$

$$(g_1 A_{10} + g_2 A_{20} + \cdots + g_P A_{P0})\, p_0 \tag{B.3}$$

$$+ (g_1 B + (g_2 - g_1) A_{21} + (g_3 - g_1) A_{31} + \cdots + (g_P - g_1) A_{P1})\, p_1 \tag{B.4}$$

$$+ \cdots \tag{B.5}$$

$$+ (g_{P-1} B + (g_P - g_{P-1}) A_{PP-1})\, p_{P-1} \tag{B.6}$$

$$+ (g_P B)\, p_P \tag{B.7}$$

The first line, Eq. B.3, can be expanded:

$$(g_1 A_{10} + g_2 A_{20} + \cdots + g_P A_{P0})\, p_0 \tag{B.8}$$

$$= (g_1)\, A_{10} p_0 \tag{B.9}$$

$$+ ((g_2 - g_1) + g_1)\, A_{20} p_0 \tag{B.10}$$

$$+ \cdots \tag{B.11}$$

$$+ ((g_P - g_{P-1}) + \cdots + (g_2 - g_1) + g_1)\, A_{P0} p_0 \tag{B.12}$$

The second line, Eq. B.4, can also be expanded:

$$(g_1 B + (g_2 - g_1) A_{21} + (g_3 - g_1) A_{31} + \cdots + (g_P - g_1) A_{P1})\, p_1 \tag{B.13}$$

$$= g_1 B p_1 \tag{B.14}$$

$$+ ((g_2 - g_1))\, A_{21} p_1 \tag{B.15}$$

$$+ ((g_3 - g_2) + (g_2 - g_1))\, A_{31} p_1 \tag{B.16}$$

$$+ \cdots \tag{B.17}$$

$$+ ((g_P - g_{P-1}) + \cdots + (g_2 - g_1))\, A_{P1} p_1 \tag{B.18}$$

By grouping terms,

$$\begin{aligned}
&\mathbb{E}[R_{\mathrm{TX}}] = g_1 \left(A_{10} + A_{20} + \cdots + A_{P0}\right) p_0 \\
&+ B \left(g_1 p_1 + g_2 p_2 + \cdots + g_P p_P\right) \\
&+ (g_2 - g_1) \left((A_{20} + A_{30} + \cdots + A_{P0}) p_0 + (A_{21} + A_{31} + \cdots + A_{P1}) p_1\right) \\
&+ (g_3 - g_2) \left((A_{30} + \cdots + A_{P0}) p_0 + (A_{31} + \cdots + A_{P1}) p_1 + (A_{32} + \cdots + A_{P2}) p_2\right) \\
&+ \cdots \\
&+ (g_P - g_{P-1}) \left(A_{P0} p_0 + A_{P1} p_1 + \cdots + A_{PP-1} p_{P-1}\right)
\end{aligned} \tag{B.19}$$

Thus, we obtain Eq. 3.73.

APPENDIX C

Distribution of the Number of Poisson Arrivals

First, we assume Poisson arrivals. Let V denote a discrete random variable for the number of arrivals in an interval and T denote a random variable for the interval or service time. The conditional probability mass function of V is given by

$$\mathbb{P}[V = k|T = \tau] = \mathbb{P}[k \text{ events in interval } \tau] = \frac{(\lambda\tau)^k e^{-\lambda\tau}}{k!} \tag{C.1}$$

where λ is the mean arrival rate.

We may then uncondition on τ using the probability density function $f_T(\tau)$ of the service time for the interval T.

$$\mathbb{P}[V = k] = \int_0^\infty \frac{(\lambda\tau)^k e^{-\lambda\tau}}{k!} f_T(\tau) d\tau \tag{C.2}$$

C.1 Exponential Service Time

The probability density function of the exponential distribution is given by

$$f_T(\tau) = \begin{cases} \mu e^{-\mu\tau} & \text{if } \tau \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{C.3}$$

where $\mu > 0$ is the rate parameter.

Then we obtain the probability mass function of V by plugging Eq. C.3 into Eq. C.2.

$$\begin{aligned}\mathbb{P}[V=k] &= \int_0^\infty \frac{(\lambda\tau)^k e^{-\lambda\tau}}{k!}\mu e^{-\mu\tau}d\tau \\ &= \frac{\lambda^k\mu}{k!}\int_0^\infty \tau^k e^{-(\lambda+\mu)\tau}d\tau\end{aligned} \tag{C.4}$$

We use integration by parts and define $M_{k,0}$.

$$M_{k,0} \triangleq \int_0^\infty \tau^k e^{-\alpha\tau}d\tau = \left[\frac{-\tau^k}{\alpha}e^{-\alpha\tau}\right]_0^\infty + \frac{k}{\alpha}M_{k-1,0} \tag{C.5}$$

where $\alpha = \lambda + \mu > 0$.

We now prove the following equation holds by induction.

$$M_{k,0} = \frac{k!}{\alpha^{k+1}} \tag{C.6}$$

where $\alpha > 0$.

The base case $M_{0,0}$ can be computed:

$$M_{0,0} = \int_0^\infty e^{-\alpha\tau}d\tau = \frac{1}{\alpha} = \frac{0!}{\alpha^{0+1}} \tag{C.7}$$

The induction step is:

$$M_{k+1,0} = \left[\frac{-\tau^{(k+1)}}{\alpha}e^{-\alpha\tau}\right]_0^\infty + \frac{k+1}{\alpha}M_{k,0} = \frac{k+1}{\alpha}\frac{k!}{\alpha^{k+1}} = \frac{(k+1)!}{\alpha^{k+2}} \tag{C.8}$$

By the principle of induction, $M_{k,0} = \frac{k!}{\alpha^{k+1}}$ holds for any nonnegative integer k. Using this with Eq. C.5 yields

$$\int_0^\infty \tau^k e^{-(\lambda+\mu)\tau}d\tau = \frac{k!}{(\lambda+\mu)^{k+1}} \tag{C.9}$$

By plugging Eq. C.9 into Eq. C.4, we obtain:

$$\begin{aligned}\mathbb{P}[V=k] &= \frac{\lambda^k\mu}{k!}\frac{k!}{(\lambda+\mu)^{k+1}} \\ &= \frac{\mu}{\lambda+\mu}\left(\frac{\lambda}{\lambda+\mu}\right)^k\end{aligned} \tag{C.10}$$

Since $\left|\frac{\lambda}{\lambda+\mu}e^{it}\right| < 1$, we obtain the characteristic function of V.

$$\begin{aligned}\phi_V(t) = \mathbb{E}\left[e^{itV}\right] &= \sum_{k=0}^{\infty} e^{itk}\frac{\mu}{\lambda+\mu}\left(\frac{\lambda}{\lambda+\mu}\right)^k \\ &= \frac{\mu}{\lambda+\mu}\sum_{k=0}^{\infty}\left(\frac{\lambda}{\lambda+\mu}e^{it}\right)^k \\ &= \frac{\mu}{\lambda+\mu}\frac{1}{\left(1-\frac{\lambda}{\lambda+\mu}e^{it}\right)} \\ &= \frac{1}{\frac{\lambda}{\mu}\left(1-e^{it}\right)+1}\end{aligned} \tag{C.11}$$

C.2 Shifted Exponential Service Time

The probability density function of a shifted exponential distribution is given by

$$f_T(\tau) = \begin{cases}\mu e^{-\mu(\tau-d)} & \text{if } \tau \geq d \\ 0 & \text{otherwise}\end{cases} \tag{C.12}$$

where $\mu > 0$ is the rate parameter and $d \geq 0$ is the number of shifted units.

$$\begin{aligned}\mathbb{P}[V=k] &= \int_d^{\infty}\frac{(\lambda\tau)^k e^{-\lambda\tau}}{k!}\mu e^{-\mu\tau}e^{\mu d}d\tau \\ &= \frac{\lambda^k\mu e^{\mu d}}{k!}\int_d^{\infty}\tau^k e^{-(\lambda+\mu)\tau}d\tau\end{aligned} \tag{C.13}$$

We use integration by parts and define $M_{k,d}$, which is the generalized version of $M_{k,0}$.

$$M_{k,d} \triangleq \int_d^{\infty}\tau^k e^{-\alpha\tau}d\tau = \frac{d^k}{\alpha}e^{-\alpha d} + \frac{k}{\alpha}M_{k-1,d} \tag{C.14}$$

where $\alpha = \lambda + \mu > 0$.

We will prove the following equation holds by induction.

$$M_{k,d} = \sum_{m=0}^{k}\frac{k!d^{k-m}}{\alpha^{m+1}(k-m)!}e^{-\alpha d} = \sum_{m=0}^{k}\frac{k!(\alpha d)^m}{\alpha^{k+1}m!}e^{-\alpha d} \tag{C.15}$$

where $\alpha > 0$.

The base case $M_{0,d}$ can be computed:

$$M_{0,d} = \int_d^\infty e^{-\alpha\tau} d\tau = \frac{1}{\alpha} e^{-\alpha d} \tag{C.16}$$

The induction step is:

$$\begin{aligned} M_{k+1,d} &= \frac{d^{k+1}}{\alpha} e^{-\alpha d} + \frac{k+1}{\alpha} M_{k,d} \\ &= \frac{d^{k+1}}{\alpha} e^{-\alpha d} + \sum_{m=0}^{k} \frac{(k+1)!(\alpha d)^m}{\alpha^{k+2} m!} e^{-\alpha d} \\ &= \sum_{m=0}^{k+1} \frac{(k+1)!(\alpha d)^m}{\alpha^{k+2} m!} e^{-\alpha d} \end{aligned} \tag{C.17}$$

By the principle of induction, Eq. C.15 holds for any nonnegative integer k.

We thus obtain the probability mass function of V:

$$\mathbb{P}[V = k] = \frac{\mu}{\lambda + \mu} \frac{1}{e^{\lambda d}} \left(\frac{\lambda}{\lambda + \mu} \right)^k \sum_{m=0}^{k} \frac{(\lambda d + \mu d)^m}{m!} \tag{C.18}$$

Note that when $d = 0$ the above equation becomes Eq. C.10.

$$\phi_V(t) = \sum_{k=0}^{\infty} \left(e^{itk} \frac{\mu}{\lambda + \mu} \frac{1}{e^{\lambda d}} \left(\frac{\lambda}{\lambda + \mu} \right)^k \sum_{m=0}^{k} \frac{(\lambda d + \mu d)^m}{m!} \right) \tag{C.19}$$

The characteristic function can further be simplified because:

$$
\begin{aligned}
&\sum_{k=0}^{\infty}\left(\alpha^k\sum_{m=0}^{k}\frac{\beta^m}{m!}\right)\\
&=1+\alpha\left(1+\frac{\beta}{1!}\right)+\alpha^2\left(1+\frac{\beta}{1!}+\frac{\beta^2}{2!}\right)+\cdots\\
&=(1+\alpha+\alpha^2+\cdots)+\frac{\alpha\beta}{1!}(1+\alpha+\alpha^2+\cdots)+\frac{(\alpha\beta)^2}{2!}(1+\alpha+\alpha^2+\cdots)+\cdots\\
&=\frac{e^{\alpha\beta}}{1-\alpha}
\end{aligned}
\tag{C.20}
$$

where $|\alpha|<1$ and $\sum_{m=0}^{\infty}\frac{x^m}{m!}$ is the Maclaurin series of e^x.

By plugging $\alpha=\frac{\lambda}{\lambda+\mu}e^{it}$ and $\beta=\lambda d+\mu d$ into Eq. C.20, we obtain the characteristic function of V.

$$
\begin{aligned}
\phi_V(t)&=\frac{\mu}{\lambda+\mu}\frac{1}{e^{\lambda d}}\frac{e^{\lambda de^{it}}}{1-\frac{\lambda}{\lambda+\mu}e^{it}}\\
&=\frac{e^{-\lambda d(1-e^{it})}}{\frac{\lambda}{\mu}(1-e^{it})+1}
\end{aligned}
\tag{C.21}
$$

Note that Eq. C.21 becomes Eq. C.11 with a special case $d=0$.

C.3 Deterministic Service Time

Let d be a constant service time. The mean service rate $\mu=\frac{1}{d}$ is also a constant. We may consider the Dirac delta function $\delta(\tau-d)$ for the (generalized) probability density function $f_T(\tau)$. The probability mass function of the number of Poisson arrivals for the interval d is given by

$$
\mathbb{P}[V=k]=\frac{(\lambda d)^k e^{-\lambda d}}{k!}
\tag{C.22}
$$

We obtain the characteristic function of V.

$$\begin{aligned}\phi_V(t) = \mathbb{E}\left[e^{itV}\right] &= \sum_{k=0}^{\infty}\left(e^{itk}\frac{(\lambda d)^k e^{-\lambda d}}{k!}\right) \\ &= e^{\lambda d\left(e^{it}-1\right)} = e^{-\lambda d\left(1-e^{it}\right)} \\ &= e^{\frac{\lambda}{\mu}\left(e^{it}-1\right)}\end{aligned} \tag{C.23}$$

where $\left|\frac{\lambda}{\mu}e^{it}\right| < 1$.

Note that the product of Eq. C.23 and Eq. C.11 matches with Eq. C.21 (the sum of independent random variables).

APPENDIX D

Program Code

D.1 Head-of-Line Bulk Service Queue with Priority Groups

```
...

def simulate():
    # [Input]
    events = create_tx_arrival_and_block_mining_events()
    P = 100  # The number of priority groups
    g = fee_distribution() # len(g) == P
    B = 100  # Bulk size

    # [Process]
    # pending_tx_pool: A head-of-line bulk service queue
    #                  with priority groups
    # temp_bucket: We may replenish the pending transaction pool
    #              right after the current mining task is finished.
    pending_tx_pool = create_queue_with_priority_groups()
    temp_bucket = create_queue_with_priority_groups()

    # [Output]
    waiting_time = [[]]  # W
    total_transaction_fee = []  # R_{TX}
    minimum_group_index = []  # X
    queue_length_after = []  # q
```

```
for e in events: # e := (timestamp, event type, [group index])
    e_timestamp = e[0]
    e_type = e[1]
    if e_type == "t":  # TX arrival event
        tx_group = e[2]  # Priority group index
        temp_bucket[tx_group].append(e_timestamp)
    elif e_type == "b":  # Block mining event
        __total_tx_fee = 0
        __minimum_group_index = None
        __remaining_space = B
        # Select up to B transactions from group P to group 1
        i = P - 1
        while i >= 0:
            num_of_available_txs = len(pending_tx_pool[i])
            if num_of_available_txs > 0:
                len_txs = min(__remaining_space, num_of_available_txs)
                txs = pending_tx_pool[i][0:len_txs]  # FIFO
                for tx_ts in txs:
                    waiting_time = prev_block_ts - tx_ts
                    # Record the waiting time of a TX in group i
                    waiting_time[i].append(waiting_time)
                __total_tx_fee += g[i] * len_txs
                __minimum_group_index = i
                pending_tx_pool[i] = pending_tx_pool[i][len_txs:]
                __remaining_space -= len_txs
                if __remaining_space == 0:
                    break
            else:
                pass
            i -= 1
        if __remaining_space > 0: # Partially filled block
            __minimum_group_index = 0
        queue_length = list(map(len, pending_tx_pool))
```

```
            # Record R_{TX}
            total_transaction_fee.append(__total_tx_fee)
            # Record X (the minimum group index)
            minimum_group_index.append(__minimum_group_index)
            # Record the queue length after serving TXs
            queue_length_after.append(queue_length)
            # Now we may replenish the pending transaction pool.
            for i in range(P):
                pending_tx_pool[i].extend(temp_bucket[i])
                temp_bucket[i].clear()
            prev_block_ts = e_timestamp

...
```

D.2 Numerical Integration

```
import cmath
import math
from scipy import integrate


def phi_sk_prime_det(_lambda, d, B, k, t):
    # The first-order derivative of
    # k-times convoluted characteristic function of
    # the deterministic service time distribution.
    i = complex(0, 1)
    return k * (cmath.exp(_lambda * d * (cmath.exp(i*t)-1)-i*t*B) ** k) \
             * (_lambda * d * i * cmath.exp(i*t) - i * B)


def compute_mean_queue_length():
    integration_limit = 1000
```

```
integration_upper = 48
stop_threshold = 1e-4
maximum_iterations = 10000 # won't reach

num_of_groups = 100
bulk_size = 10
total_arrival_rate = 1/600 * 10 * 1.15
service_time = 600

for group_index in range(num_of_groups - 1, 0, -1):
    arrival_rate = total_arrival_rate * \
        (num_of_groups - group_index)/num_of_groups
    bs = [0]
    for k in range(1, maximum_iterations):
        area_sub_sum = 0
        for b in range(1, integration_upper):
            if b == 1:  # to handle a singular point (zero)
                area_sub, _ = integrate.quad(
                    lambda t: 2*phi_sk_prime_det(
                        arrival_rate,
                        service_time,
                        bulk_size,
                        k,
                        t).real/t,
                    a=0,
                    b=1,
                    points=[0],
                    limit=integration_limit)
            else:
                area_sub, _ = integrate.quad(
                    lambda t: 2*phi_sk_prime_det(arrival_rate,
                                                 service_time,
                                                 bulk_size,
```

```
                                                k,
                                                t).real/t,
                        a=(b-1),
                        b=b,
                        limit=integration_limit)
                area_sub_sum += area_sub
            inc = 0.5 * (phi_sk_prime_det(arrival_rate,
                                          service_time,
                                          bulk_size,
                                          k,
                                          0).imag
                         - area_sub_sum/math.pi)/k
            bs.append(bs[-1] + inc)
            delta = None
            if len(bs) >= 2:
                delta = (bs[-1] - bs[-2])
            if abs(delta) < stop_threshold:
                break
        print(group_index, k, bs[-1], arrival_rate * 600, sep="\t")
```

REFERENCES

[ABB18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco, and Jason Yellick. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains." *CoRR*, **abs/1801.10228**, 2018.

[ABF14] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio, and Nicola Galesi. "Proofs of space: When space is of the essence." In *International Conference on Security and Cryptography for Networks*, pp. 538–557. Springer, 2014.

[Bac02] Adam Back. "Hashcash - A Denial of Service Counter-Measure." Technical report, 2002.

[Bai54] Norman TJ Bailey. "On queueing processes with bulk service." *Journal of the Royal Statistical Society: Series B (Methodological)*, **16**(1):80–87, 1954.

[BBB18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. "Verifiable delay functions." In *Annual international cryptology conference*, pp. 757–788. Springer, 2018.

[BC89] G Briere and ML Chaudhry. "Computational analysis of single-server bulk-service queues, $M/G^Y/1$." *Advances in Applied Probability*, pp. 207–225, 1989.

[BEO19] Soumya Basu, David Easley, Maureen O'Hara, and Emin Sirer. "Towards a functional fee market for cryptocurrencies." *Available at SSRN 3318327*, 2019.

[BG18] Juan Benet and Nicola Greco. "Filecoin: A decentralized storage network." *Protoc. Labs*, pp. 1–36, 2018.

[Bha64] U Narayan Bhat. "Imbedded Markov chain analysis of single server bulk queues." *Journal of the Australian Mathematical Society*, **4**(2):244–263, 1964.

[BKK20] R. Bowden, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor. "Modeling and analysis of block arrival times in the Bitcoin blockchain." *Stochastic Models*, **36**(4):602–637, 2020.

[BLM14] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. "Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake." *ACM SIGMETRICS Performance Evaluation Review*, **42**(3):34–37, 2014.

[BRL20] Vitalik Buterin, Daniël Reijsbergen, Stefanos Leonardos, and Georgios Piliouras. "Incentives in Ethereum's hybrid Casper protocol." *International Journal of Network Management*, **30**(5):e2098, 2020. e2098 nem.2098.

[CDF17] Alexander Chepurnoy, Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. "TwinsCoin: A Cryptocurrency via Proof-of-Work and Proof-of-Stake." *IACR Cryptology ePrint Archive*, **2017**:232, 2017.

[CL99] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance." In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, p. 173–186, USA, 1999. USENIX Association.

[CM18] Brad Chase and Ethan MacBrough. "Analysis of the XRP ledger consensus protocol." *arXiv preprint arXiv:1802.07242*, 2018.

[Cob54] Alan Cobham. "Priority Assignment in Waiting Line Problems." *Journal of the Operations Research Society of America*, **2**(1):70–76, 1954.

[CP19] Bram Cohen and Krzysztof Pietrzak. "The chia network blockchain.", 2019.

[CT81] ML Chaudhry and James GC Templeton. "The queuing system $M/G^B/1$ and its ramifications." *European Journal of Operational Research*, **6**(1):56–60, 1981.

[CT83] ML Chaudhry and James GC Templeton. *First course in bulk queues.* 1983.

[DFK15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. "Proofs of space." In *Annual Cryptology Conference*, pp. 585–605. Springer, 2015.

[DN92] Cynthia Dwork and Moni Naor. "Pricing via processing or combatting junk mail." In *Annual International Cryptology Conference*, pp. 139–147. Springer, 1992.

[Dow55] F Downton. "Waiting time in bulk service queues." *Journal of the Royal Statistical Society: Series B (Methodological)*, **17**(2):256–261, 1955.

[DW13] Christian Decker and Roger Wattenhofer. "Information propagation in the Bitcoin network." In *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento, Italy, September 9-11, 2013, Proceedings*, pp. 1–10. IEEE, 2013.

[ES14] Ittay Eyal and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." In *International conference on financial cryptography and data security*, pp. 436–454. Springer, 2014.

[Fin11] Hal Finney. `https://bitcointalk.org/index.php?topic=3441.msg48384`, 2011. [Online].

[FKS19] Pietro Ferraro, Christopher K. King, and Robert Shorten. "IOTA-based Directed Acyclic Graphs without Orphans." *CoRR*, **abs/1901.07302**, 2019.

[GHM17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. "Algorand: Scaling Byzantine Agreements for Cryptocurrencies." In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, p. 51–68, New York, NY, USA, 2017. Association for Computing Machinery.

[GKK16] Johannes Göbel, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay." *Performance Evaluation*, **104**:23–41, 2016.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. "The Bitcoin Backbone Protocol: Analysis and Applications." In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pp. 281–310. Springer, 2015.

[HCG81] DF Holman, ML Chaudhry, and A Ghosal. "Some results for the general bulk service queueing system." *Bulletin of the Australian Mathematical Society*, **23**(2):161–179, 1981.

[HLM19] Gur Huberman, Jacob Leshno, and Ciamac C Moallemi. "An economic analysis of the bitcoin payment system." *Columbia Business School Research Paper*, (17-92), 2019.

[Hou14] Nicolas Houy. "The economics of Bitcoin transaction fees." Working Papers 1407, Groupe d'Analyse et de Théorie Economique Lyon St-Étienne (GATE Lyon St-Étienne), Université de Lyon, 2014.

[Int17] Intel. "Proof-of-Elapsed-Time (PoET)." `https://sawtooth.hyperledger.org`, 2017. [Online].

[Jai60] NK Jaiswal. "Time-dependent solution of the bulk-service queuing problem." *Operations Research*, **8**(6):773–781, 1960.

[JJ99] Markus Jakobsson and Ari Juels. "Proofs of work and bread pudding protocols." In *Secure Information Networks*, pp. 258–272. Springer, 1999.

[Kei62] Julian Keilson. "The general bulk queue as a Hilbert problem." *Journal of the Royal Statistical Society: Series B (Methodological)*, **24**(2):344–358, 1962.

[Kei64] Julian Keilson. "An alternative to Wiener-Hopf methods for the study of bounded processes." *Journal of Applied Probability*, **1**(1):85–120, 1964.

[Ken51] David G. Kendall. "Some Problems in the Theory of Queues." *Journal of the Royal Statistical Society. Series B (Methodological)*, **13**(2):151–185, 1951.

[KK17] Yoshiaki Kawase and Shoji Kasahara. "Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism." In *International Conference on Queueing Theory and Network Applications*, pp. 75–88. Springer, 2017.

[KK20] Yoshiaki Kawase and Shoji Kasahara. "Priority queueing analysis of transaction-confirmation time for Bitcoin." *Journal of Industrial & Management Optimization*, **16**(3):1077, 2020.

[KKS17] Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, and Yongdae Kim. *Be Selfish and Avoid Dilemmas: Fork After Withholding (FAW) Attacks on Bitcoin*, p. 195–209. Association for Computing Machinery, New York, NY, USA, 2017.

[Kle75] Leonard Kleinrock. *Queueing Systems: Volume 1 - Theory*. Wiley Interscience, 1975.

[Kle76] Leonard Kleinrock. *Queueing Systems: Volume 2 - Computer Applications*. Wiley Interscience, 1976.

[KN12] Sunny King and Scott Nadal. "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake." *self-published paper, August*, **19**(1), 2012.

[KRD17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. "Ouroboros: A provably secure proof-of-stake blockchain protocol." In *Annual International Cryptology Conference*, pp. 357–388. Springer, 2017.

[KRV57] Harry Kesten, J Th Runnenburg, and D Van Dantzig. *Priority in Waiting Line Problems I and II*. Koninklijke Nederlandse Akademie van Wetenschappen Amsterdam, 1957.

[Ksc06] Frank R Kschischang. "The Hilbert Transform." *University of Toronto*, **83**:277, 2006.

[LAB17] Wenting Li, Sébastien Andreina, Jens-Matthias Bohli, and Ghassan Karame. "Securing Proof-of-Stake Blockchain Protocols." In Joaquín García-Alfaro, Guillermo Navarro-Arribas, Hannes Hartenstein, and Jordi Herrera-Joancomartí, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2017 International Workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings*, volume 10436 of *Lecture Notes in Computer Science*, pp. 297–315. Springer, 2017.

[Lam82] L. Lamport. "The Byzantine generals problem." *ACM Transactions on Programming Languages and Systems*, **4**:382–401, 1982.

[Lin52] David V Lindley. "The theory of queues with a single server." In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 48, pp. 277–289. Cambridge University Press, 1952.

[Lit61] John D. C. Little. "A Proof for the Queuing Formula: L = λW." *Operations Research*, **9**(3):383–387, 1961.

[LLM19] Marta Lokhava, Giuliano Losa, David Mazières, Graydon Hoare, Nicolas Barry, Eli Gafni, Jonathan Jove, Rafał Malinowsky, and Jed McCaleb. "Fast and Secure Global Payments with Stellar." In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, p. 80–96, New York, NY, USA, 2019. Association for Computing Machinery.

[Mar18] Julian Martinez. "Understanding Proof of Stake: The Nothing at Stake Theory." `https://medium.com/coinmonks/understanding-proof-of-stake-the-nothing-at-stake-theory-1f0d71bc027`, 2018. [Online].

[Med75] J Medhi. "Waiting time distribution in a Poisson queue with a general bulk service rule." *Management Science*, **21**(7):777–782, 1975.

[Nak08] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system." 2008.

[Neu67] Marcel F Neuts. "A general class of bulk queues with Poisson input." *The Annals of Mathematical Statistics*, **38**(3):759–770, 1967.

[PH86] Warren B Powell and Pierre Humblet. "The bulk service queue with a general control strategy: theoretical analysis and a new computational procedure." *Operations Research*, **34**(2):267–275, 1986.

[Pin15] Iosif Pinelis. "Characteristic function of the positive part of a random variable and related results, with applications." *Statistics & Probability Letters*, **106**:281–286, 2015.

[Pin18] Iosif Pinelis. "Positive-part moments via characteristic functions, and more general expressions." *Journal of Theoretical Probability*, **31**(1):527–555, 2018.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. "Analysis of the Blockchain Protocol in Asynchronous Networks." In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pp. 643–673, 2017.

[Riz16] P. Rizun. "A Transaction Fee Market Exists Without a Block Size Limit." 2016.

[Spi56] Frank Spitzer. "A combinatorial lemma and its application to probability theory." *Transactions of the American Mathematical Society*, **82**(2):323–339, 1956.

[SSZ16] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. "Optimal selfish mining strategies in bitcoin." In *International Conference on Financial Cryptography and Data Security*, pp. 515–532. Springer, 2016.

[Sza97] Nick Szabo. "The idea of smart contracts." *Nick Szabo's Papers and Concise Tutorials*, **6**, 1997.

[Wei] Eric W Weisstein. "'Sinc Function' From MathWorld–A Wolfram Web Resource." `https://mathworld.wolfram.com/SincFunction.html`. [Online].

[Wik] Bitcoin Wiki. "Difficulty." `https://en.bitcoin.it/wiki/Difficulty`. [Online].

[Woo14] Gavin Wood. "Ethereum: A secure decentralised generalised transaction ledger." 2014.

[Woo16] Gavin Wood. "Polkadot: Vision for a heterogeneous multi-chain framework." *White Paper*, **21**, 2016.

[Yao82] Andrew C Yao. "Theory and application of trapdoor functions." In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pp. 80–91. IEEE, 1982.

[YKJ18] Seunghyun Yoo, Seungbae Kim, Joshua Joy, and Mario Gerla. "Promoting Cooperative Strategies on Proof-of-Work Blockchain." In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, pp. 1–8. IEEE, 2018.

[Zam15] Vlad Zamfir. "Introducing Casper 'the Friendly Ghost'." `https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/`, 2015. [Online].