BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

SIMULATOR FOR VERIFYING THE PROPERTIES OF DAG-BASED CONSENSUS PROTOCOLS

SIMULÁTOR PRO OVĚŘENÍ VLASTNOSTÍ DAG-BASED CONSENSUS PROTOKOLŮ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — **TOMÁŠ HLADKÝ**
AUTOR PRÁCE

SUPERVISOR — **Mgr. KAMIL MALINKA, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Intelligent Systems (DITS) Academic year 2021/2022

Bachelor's Thesis Specification

Student: **Hladký Tomáš**
Programme: Information Technology
Title: **Simulator for Verifying the Properties of DAG-Based Consensus Protocols**
Category: Security

Assignment:

1. Study the existing of DAG-based consensus protocols.
2. Familiarize yourself with existing blockchain protocol simulators - focus on DAG-based protocols support.
3. Design and extend the existing simulator implementation (see Perešíni et al.) with additional simulation capabilities such as changing the network topology, number of nodes in the network, support for different mining strategies, change of the payoff function, and more. Focus also on the optimization of the simulator implementation.
4. Use the extended simulator to perform basic experiments to verify the performance of the PHANTOM/GHOSTDAG protocols.

Recommended literature:

- Perešíni, M., Benčić, F. M., Malinka, K., & Homoliak, I. (2021). DAG-Oriented Protocols PHANTOM and GHOSTDAG under Incentive Attack via Transaction Selection Strategy. *arXiv preprint arXiv:2109.01102*.
- https://github.com/mpershing/bitcoin_miningsim
- Y. Sompolinsky, S. Wyborski, and A. Zohar, "Phantom and ghostdag: A scalable generalization of nakamoto consensus," Cryptology ePrint Archive, Report 2018/104, 2018, https://eprint.iacr.org/2018/104.pdf
- R. Paulavičius, S. Grigaitis and E. Filatovas, "A Systematic Review and Empirical Analysis of Blockchain Simulators," in IEEE Access, vol. 9, pp. 38010-38028, 2021, doi: 10.1109/ACCESS.2021.3063324.

Requirements for the first semester:

- Items 1 to 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Malinka Kamil, Mgr., Ph.D.**
Consultant: Perešíni Martin, Ing., UITS FIT VUT
Head of Department: Hanáček Petr, doc. Dr. Ing.
Beginning of work: November 1, 2021
Submission deadline: May 11, 2022
Approval date: November 3, 2021

Abstract

In recent years, blockchain has received significant attention in the research community. Since then, several submissions have been proposed to respond to the Proof-of-Work blockchains throughput problem. We study existing Directed Acyclic Graph (DAG) blockchain designs that propose to solve this problem, especially protocols PHANTOM and its optimization GHOSTDAG. They utilize a Bitcoin protocol and propose a random transaction selection, resulting in increased transaction throughput. However, it has been proved by a simulation that actors that use the random transaction selection strategy have less profit than actors who do not follow the protocol and select transactions rationally (i.e., most profitable). That proof has been made on a small network of ten nodes with a circle topology. This article aims to extend, optimize, and automate an existing blockchain simulator. We implement a Bitcoin-like network topology with realistic block propagation latency. Furthermore, we optimize the simulator to run more simulations in parallel and faster, including automation tools that can modify input configurations, perform a combination of runs on multiple CPU cores based on input parameters, and analyze profits and transaction collisions. Finally, we perform experiments to verify malicious actors' advantages in a Bitcoin-like network and create a payoff function to punish this behavior.

Abstrakt

V posledných rokoch sa vo výskumnej komunite venuje blockchainu významná pozornosť. Odvtedy bolo navrhnutých niekoľko návrhov na riešenie problému priepustnosti blockchainov založených na Proof-of-Work. V tejto práci študujeme existujúce návrhy blockchainu s acyklicky orientovaným grafom (DAG), ktoré navrhujú riešenie spomínaného problému, najmä protokol PHANTOM (a jeho optiamlizácia GHOSTDAG). Využívajú bitcoinový protokol a navrhujú náhodný výber transakcií, čo vedie k zvýšeniu priepustnosti transakcií. Simuláciou sa však dokázalo, že aktéri, ktorí využívajú stratégiu náhodného výberu transakcií, majú menší zisk ako aktéri, ktorí nedodržiavajú protokol a vyberajú transakcie racionálne (t.j. najziskovejšie). Tento dôkaz bol vykonaný na malej sieti desiatich uzlov s kruhovou topológiou. Cieľom tohto článku je rozšíriť, optimalizovať a automatizovať existujúci simulátor blockchainu. Implementujeme sieťovú topológiu podobnú bitcoinu s realistickou latenciou šírenia blokov. Okrem toho optimalizujeme simulátor tak, aby sme mohli spúšťať viac simulácií paralelne a rýchlejšie, vrátane automatizačných nástrojov, ktoré môžu upravovať vstupné konfigurácie, vykonávať kombináciu simulácii na viacerých jadrách CPU na základe vstupných parametrov a analyzovať zisky a kolízie transakcií. Nakoniec vykonáme experimenty na overenie výhod škodlivých aktérov v sieti podobnej Bitcoinu a vytvoríme výplatnú funkciu na potrestanie tohto správania.

Keywords

simulator, DAG-based consensus protocol, blockchain, optimizations, payoff function, transaction throughput

Klíčová slova

simulátor, konsenzus protokol založený na štruktúre DAG, blockchain, optimalizácie, payoff funkcia, priepustnosť transakcií

Reference

HLADKÝ, Tomáš. *Simulator for Verifying the Properties of DAG-Based Consensus Protocols*. Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Mgr. Kamil Malinka, Ph.D.

Rozšířený abstrakt

Technológia blockchain v poslednom desaťročí získala veľké množstvo popularity. Na rozdiel od tradičnej databázy sú blockchainy definované decentralizáciou, nemennosťou, transparentnosťou a bezpečnosťou. To otvára nové možnosti pre širokú škálu rôznych oblastí (napr. bankovníctvo a financie, internet vecí (IoT), nehnuteľnosti, zdravotníctvo). Obmedzená priepustnosť blockchainov je však jedným z najvýznamnejších problémov, ktoré im bránia v dosiahnutí ich plného potenciálu v mnohých aplikáciách.

Táto práca sa zameriava na alternatívnu štruktúru blockchainu – Directed Acyclic Graph (DAG), ktorá dokáže spracovať viacero blokov v rovnakej výške a tým zvýšiť priepustnosť transakcií. S touto štruktúrou bolo navrhnutých niekoľko konsenzuálnych protokolov, pričom táto práca študuje niekoľko z nich a zameriava sa na PHANTOM a GHOSTDAG, ktoré zovšeobecňujú Bitcoin. Problém, ktoré tieto dva protokoly obsahujú, je, že autori dôkladne neanalyzovali motiváciu ťažiarov dosiahnuť väčší zisk. Protokol navrhuje stratégiu náhodného výberu transakcií, aby sa tak znížila miera kolíznosti transakcií. Otázkou zostáva, či väčší zisk dosiahnu nečestní aktéri, ktorí sa nedržia protokolu a namiesto toho používajú racionálnu stratégiu výberu transakcií (t.j. výber transakcií s najvyšším poplatkom) typickú pre Bitcoin alebo Ethereum. Podľa výskumu pomocou simulácie bolo dokázané, že nečestní ťažiari zarábajú podstatne viac ako poctiví ťažiari. Vo všeobecnosti, ak má transakcia vyšší poplatok ako ostatné, jej šanca na spracovanie v ďalšom bloku je vyššia, pretože viac ťažiarov má motiváciu zaradiť ju do ťažobného bloku, aby boli odmenení vyššími ziskami. Ďalej sa ukázalo, že nečestní aktéri majú negatívny vplyv na priepustnosť transakcií. Simulácie boli vykonané na jednoduchej kruhovej topológii s desiatimi ťažiarmi.

Naša práca zahŕňa rozšírenie existujúceho simulátora, aby bolo možné simulovať sieť podobnú Bitcoinu s viac ako 7500 ťažiarmi. To zahŕňa aj optimalizáciu a automatizáciu simulátora, keďže simulácia takejto siete je zložitý proces. Je to tiež súčasťou prebiehajúceho výskumu s výskumnou skupinou Security@FIT na Fakulte informačných technológií VUT v Brne. Príspevky našej práce sú nasledovné:

- Analyzujeme existujúcu implementáciu mempoolu a porovnávame výkonnosť dátových štruktúr multi-index, hešovacej tabuľky a červeno-čierneho stromu s ich požiadavkami na použitie.
- Implementujeme novú dátovú štruktúru mempoolu, ktorá pozostáva z kombinácie hešovacej tabuľky a červeno-čierneho stromu a dokazujeme, že táto implementácia zrýchlila proces simulácie.
- Implementujeme automatizačný nástroj na spustenie kombinácie simulácií na viacerých jadrách CPU, postprocesné nástroje na analýzu výsledkov simulácie a skript na úpravu latencie propagácie bloku v sieti peer-to-peer podobnej Bitcoinu.
- Ukazujeme, že jediný nečestný aktér, ktorý vyberá transakcie s najvyšším poplatkom, má významnú výhodu v ziskoch oproti poctivým aktérom, ktorí náhodne vyberajú transakcie v sieti podobnej Bitcoinu so 7592 uzlami.
- Experimentujeme s rôznymi výplatnými funkciami, analyzujeme ich výsledky a navrhujeme spôsob, ako odhaliť a potrestať škodlivé správanie.

Simulator for Verifying the Properties of DAG-Based Consensus Protocols

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mgr. Kamil Malinka, Ph.D.

.....
Tomáš Hladký
May 10, 2022

Acknowledgements

I would like to thank my supervisor Mgr. Malinka Kamil, Ph.D. and my consultant Ing. Perešíni Martin for their valuable guidance and advices. I also acknowledge that this work is also part of ongoing research with the Security@FIT at Brno University of Technology, Faculty of Information Technology where its contributors are: Ing. Martin Perešíni, Ing. Federico Matteo Benčić, Mag., Mgr. Kamil Malinka, Ph.D. and Ing. Ivan Homoliak, Ph.D. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Contents

1 Introduction **3**

2 Blockchain **5**
2.1 What is blockchain 5
2.2 Key features 5
2.3 Structure 6
2.4 Properties 8
2.5 Soft fork 9
2.6 Security risks 9
2.7 Throughput limitations 10

3 DAG-based consensus protocols **12**
3.1 IOTA 12
3.2 Obyte (Byteball) 13
3.3 SPECTRE 13
3.4 PHANTOM and GHOSTDAG 13
3.5 Other DAG-oriented designs 15
3.6 Summary 15

4 Related work **17**
4.1 Verification of blockchain properties 17
4.2 Three-layer architecture 17
4.3 Simulation approach 19
4.3.1 Simulator requirements 20
4.4 Available simulators 20
4.4.1 BTCSim 21
4.4.2 Simbit 21
4.4.3 Shadow-Bitcoin 22
4.4.4 Bitcoin-Simulator 22
4.4.5 Bitcoin-Simulator extension 22
4.4.6 BlockSim:Faria 22
4.4.7 SIMBA (BlockSim:Faria extension) 22
4.4.8 SimBlock 22
4.4.9 SimBlock extension 23
4.4.10 Bitcoin Mining Simulator 23
4.4.11 Conclusion 23

5 Proposed extensions **25**

5.1 Existing versions of simulator . . . 25
5.1.1 Version 1 . . . 25
5.1.2 Version 2 . . . 26
5.1.3 Version 3 extension proposal . . . 26
5.2 Update of simulator architecture . . . 27
5.2.1 Update of existing events . . . 27
5.2.2 Simulator modules . . . 28
5.3 Bitcoin-like network . . . 29
5.4 Optimization . . . 31
5.4.1 Mempool operations . . . 31
5.4.2 Random access . . . 33
5.5 Progress tracker . . . 38
5.6 Automation . . . 38
5.7 Payoff function . . . 43
5.8 Summary . . . 43

6 Evaluation **45**
6.1 Experiment with single malicious miner . . . 45
6.2 Experiment with multiple malicious miners . . . 46
6.3 Payoff function experiments . . . 47

7 Conclusion **52**

Bibliography **54**

A Contents of the included storage media **59**

Chapter 1

Introduction

Blockchain technology has emerged as one of the most disruptive technologies in the last decade. In contrast to a traditional database, blockchains are defined by decentralization, immutability, transparency, and security. This opens new possibilities to a wide range of various fields (e.g., banking and finance, Internet of Things (IoT), real estate, healthcare). However, limited blockchains throughput is one of the most significant problems preventing them from reaching their full potential in many applications.

This work focus on an alternative blockchain structure — Directed Acyclic Graph (DAG), which can process multiple blocks at the same height and thus increase transaction throughput. Several consensus protocols have been proposed with this structure, while this work studies several of them and focuses on PHANTOM and GHOSTDAG that generalize Bitcoin. The issue that involves these two protocols is that the authors did not thoroughly analyze the miners' incentives to reach a greater profit. Protocol proposes a random transaction selection strategy in order to reduce transaction collisions rate. The question is if malicious actors who do not stick to the protocol and instead use a rational transaction selection strategy (i.e., select transactions with the highest fee) typical for Bitcoin or Ethereum earn more profit. In [46] was proved by a simulation that malicious miners earn significantly more profit than honest miners. In general, if a transaction has a higher fee than others, its chance of being processed in the next block is higher because more miners have an incentive to include it in the mining block to be rewarded with higher profits. Further, it shows that malicious actors have a negative impact on transaction throughput. Simulations were done on a simple circle topology with ten miners.

Our work includes an extension to the existing simulator to be capable of simulating a Bitcoin-like network with over 7500 miners. This also includes further simulator optimization and automation, as simulating such a network is a complex process. It is also part of ongoing research with the Security@FIT at Brno University of Technology, Faculty of Information Technology. The contributions of our work are as follows:

- We analyze the existing mempool implementation and compare the performance of multi-index[1], hash table, and red-black tree data structures with their usage requirements.
- We implement a new mempool data structure that consists of a combination of hashtable and red-black tree and prove it speeds up in the simulation process.

[1] https://www.boost.org/doc/libs/1_78_0/libs/multi_index/doc/index.html

- We implement an automation tool to run a combination of simulations on multiple CPU cores, post-process tools to analyze simulation results, and a script to edit block propagation latency in a Bitcoin-like peer-to-peer network.
- We show that a single malicious actor who selects transactions with the highest fee has a significant advantage in profits over honest miners who randomly select transactions in a Bitcoin-like network with 7592 nodes.
- We experiment with different payoff functions, analyze their results, and propose a way to detect and punish malicious behavior.

The organization of this work is as follows: Chapter 2 describes the technical background of blockchain technology, its properties, and problems. Chapter 3 discusses existing DAG-based consensus protocols, compares their features, and further focuses on PHANTOM and GHOSTDAG. Chapter 4 analyzes the possibilities of verifying blockchain properties, describes prerequisites to simulating consensus protocol and provides insight into related work of available simulators. Chapter 5 proposes extensions to the existing simulator and summarizes a comparison with its previous versions. Chapter 6 contains an evaluation to verify properties on a Bitcoin-like network and experiment with payoff functions. Last Chapter 7 provides a summary of results achieved in this work, conclusion, and possibilities for future work.

Chapter 2

Blockchain

This chapter describes the base principles of blockchains, their advantages, potential security risks, and current limitations.

2.1 What is blockchain

The term blockchain was popularized by a person (or group of people) using the pseudonym Satoshi Nakamoto in 2008 on the release of a Bitcoin white paper establishing the blockchain model. A blockchain or distributed ledger technology (DLT) is a distributed database shared among the nodes in the peer-to-peer network. This type of database is append-only. Hence once information is entered, it can never be modified or erased. Each block contains transactions or other digital information verified by the consensus of a majority of the participants in the system. Blocks are then *chained* together and form a blockchain [38, 19].

2.2 Key features

Blockchains are defined by their key features.

Decentralization

Decentralization is one of the essential elements of blockchain technologies. Instead of relying on centralized authority, the network is managed by nodes that communicate through a peer-to-peer network. Every node has its copy of the blockchain. Trusting other nodes is not necessarily required because when any node modifies existing information in its spreading version of the blockchain, most of the nodes in the network will refuse it. The Decision-making process is done through consensus rather than having responsibility in the hands of a single entity [38].

Immutability

Once a new transaction has been inserted into the shared ledger, no single node within the network can alter the information held within it. This is possible because blocks are chained together. Every block has its hash and also a hash of its previous block. If the content of a block gets changed, its hash would be completely different and would not equal the hash in the next block that refers to it. A particular block (i.e., genesis block[1]) that is

[1] https://en.bitcoin.it/wiki/Genesis_block

the first block in the blockchain cannot have a previous hash. The genesis block is often hardcoded into the application that utilizes its blockchain [34].

Security

In order to produce a block, a miner must use a portion of his resources and produce proof — specifically, computational power to produce a hash that meets specific criteria. The more nodes are willing to mine and connect to the network, the bigger the concurrency of block generation is. Miners also want to have a higher mining power to increase their chance of finding a hash. Nakamoto's proposal includes dynamic difficulty change to keep the time of finding a hash the same [43]. This results in more suitable security because an attacker would need to have higher computational power than the majority of the network to perform a successful attack.

Transparency

Each participant has his transparent profile in the blockchain, which is identified by his public address. Any participant can access the holdings and transaction history of any public address [34].

2.3 Structure

A blockchain structure consists of blocks containing transactions organized in a Merkle hash tree.

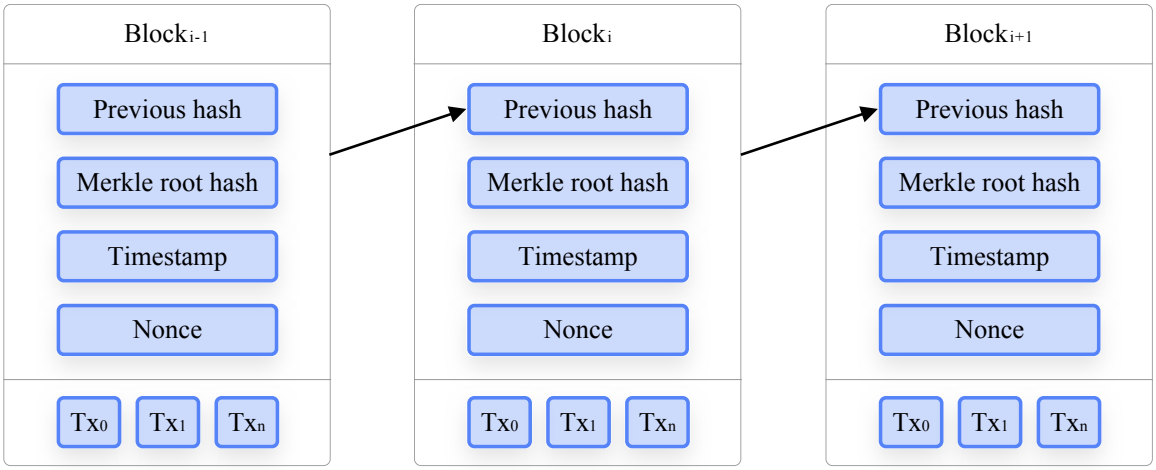

Figure 2.1: Blockchain structure. Blocks are connected together with their block header hashes.

Transaction

The transaction is the elementary part of a blockchain. It represents the movement of funds from one address to another.

Creating a transaction requires the sender to sign it with a private key. The corresponding public key is then used to verify the signature and validate the transaction.

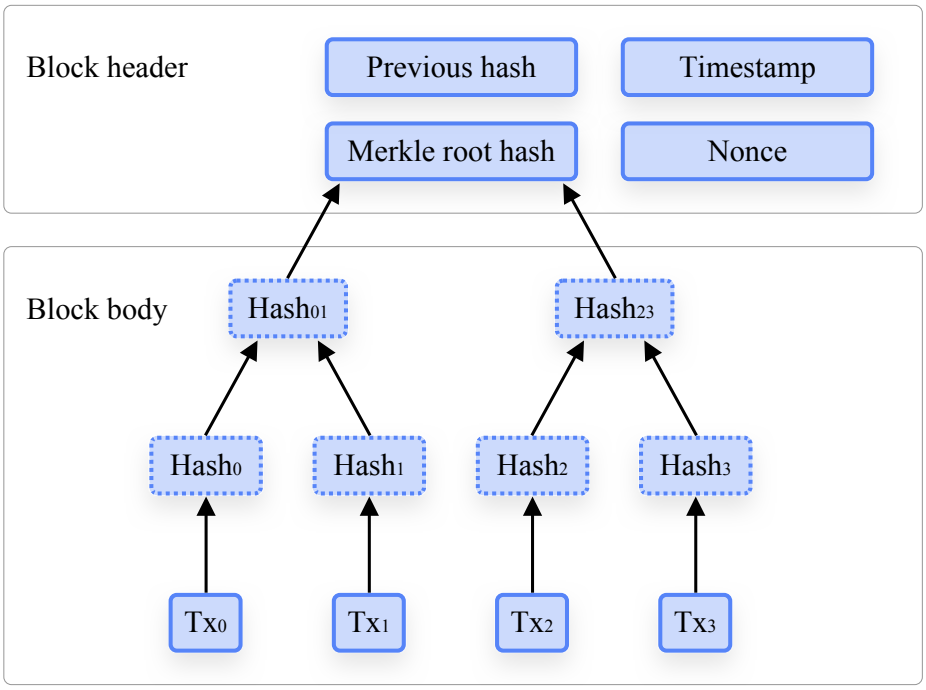

Figure 2.2: Structure of the block, which contains block header and block body.

A sender must send this transaction to at least one node connected to the blockchain network to execute a transaction. If it is valid, it then gets broadcasted among other nodes. Then it can be included in a block. Block has a limited size and can consist only of a limited number of transactions. Each transaction can include a fee which gives miners an incentive to include the transaction in the next processed block. Transactions can be either confirmed or unconfirmed. An unconfirmed transaction can be fully valid but cannot be considered processed because it has not been included in a block, or that block did not reach certain *finality* [38].

Merkle hash tree

Transactions are organized in the form of a Merkle tree, as shown in Fig. 2.2. Transactions are at the leaves position and produce a Merkle root used to keep a snapshot of the state of all transactions in a single block. Because Bitcoin uses the SHA256 hash algorithm, this will result only in 256 bits. It also allows nodes to compare their transactions list quickly instead of verifying transactions individually. This lets building light software clients that do not need to store entire blockchain data to validate their own transactions [38].

Block

A single block consists of a block header and block body. Block body contains all included transactions, from which a Merkle root is calculated. Merkle root is included in the block header with a hash of the previous block header, the timestamp when the block was generated, and a nonce, which is the value that miners are trying to find to mine a block successfully. This structure is shown in Fig. 2.2. Blocks are then chained together with

their block header hashes, as shown in Fig. 2.1. A miner that produces a block is rewarded from two sources [43]:

- **Block reward.** In Bitcoin, each block contains a predefined reward in the form of newly generated coins. This reward is for performing in the blockchain mining and providing computational resources to the Bitcoin network. Its value also continuously decreases to zero, and its purpose is to generate a limited number of new coins rather than create them from nothing. It also motivates newcomers to join mining as transaction fees provide little reward compared to block rewards at the start [38].
- **Transactions fees.** Sum of all fees that are included in each transaction that gets added to the block. More transactions are often waiting to be processed as block size is limited, and the miner chooses what transactions will be included [38].

Mempool

Mempool is a data structure that stores all miner's transactions that are waiting to be processed. When a new transaction is generated, its validity is checked. If the transaction is valid, it gets broadcasted to the current node's peers (i.e., other nodes). Nodes that are also mining blocks include it in their mempool. This process repeats until the transaction propagates throughout the network. When a new block is mined, it also gets similarly broadcasted. If the block is valid, miners remove transactions included in the block from their mempool. Since there is network delay for transaction and block propagation, the content of the mempool may vary slightly from node to node [46].

2.4 Properties

For the context of this work, we present the following properties of blockchains, which may vary according to the needs and design of the consensus protocol.

Finality

Finality is the guarantee that the transaction has been completed and cannot be reversed. It states how many blocks to wait until a transaction is considered valid, but this number may differ in different sources as they may require a higher confidence level. The higher the number of blocks to wait, the higher the chance that the transaction has been processed and is valid [34].

Scalability

Scalability represents how the consensus protocol scales when the number of participants increases. Bitcoin uses the Proof-of-Work consensus protocol and is highly scalable. Therefore, scalability increases concurrency, which results in better security [34].

Throughput

Throughput defines the number of transactions that can be processed per unit of time. This number is derived from block generation time, and the number of transactions contained in a block [34].

2.5 Soft fork

In Bitcoin, when two blocks are found simultaneously and get propagated, both blocks will have the same parent. This situation is called a soft fork and needs to be resolved because blocks are allowed to have only one parent and one child. The consensus protocol resolves this situation, and nodes will finally end up with one accepted block.

Longest chain rule and strongest chain rule

Newly generated blocks are always stored chronologically. Nakamoto's original proposal contains the longest chain rule (see Fig. 2.3), which defines that nodes accept the chain with the largest number of blocks in case of conflict. Since then, this rule has been replaced by the strongest chain rule. There was a security issue where blocks can be easily generated with lower difficulty and thus overwrite the original chain. The strongest chain rule (see Fig. 2.4) involves the amount of work put into each block and thus makes similar attacks even harder and less likely to happen. The chain that results from applying this rule is usually also called the main chain [43, 34].

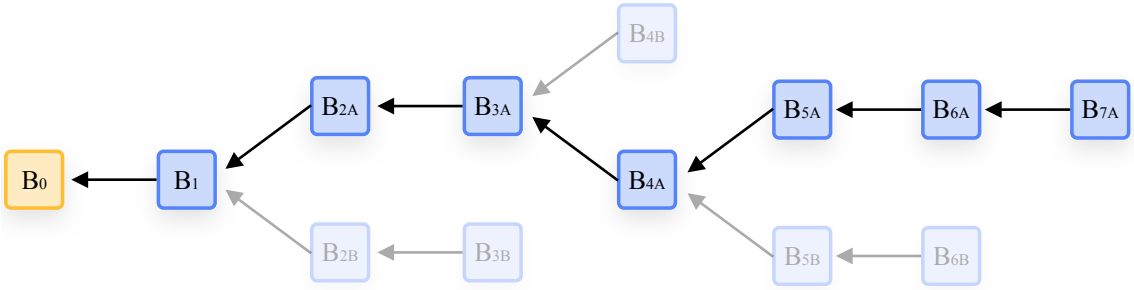

Figure 2.3: The longest chain rule [46].

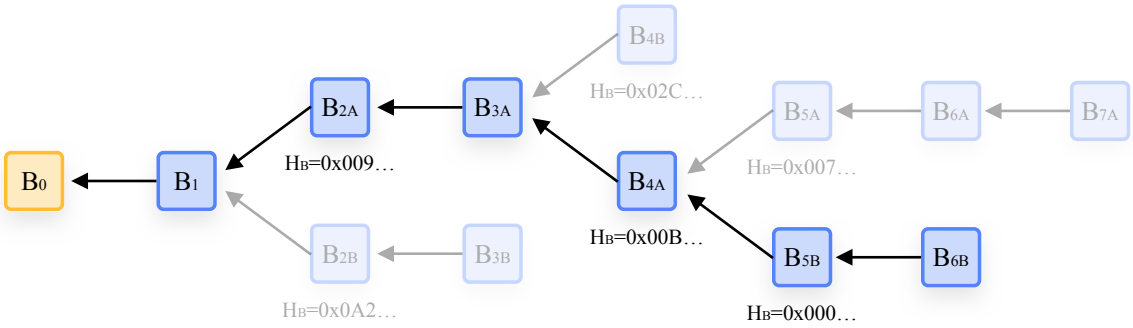

Figure 2.4: The strongest chain rule. The chain is selected by the amount of work put into each block. H_B represents the hash of the current block [46].

2.6 Security risks

While blockchains can handle the problems associated with centralized systems, they contain various other security issues.

51% attack

In a 51% attack, the attacker holds more than 50% of the whole network's computing power and gives him control over the produced results. His chance to produce blocks is higher than others, and thus he can produce blocks faster and block potential transactions that could be included. An attacker can also reverse transactions that he verified and thus perform a double-spending. However, performing such an attack in Bitcoin is challenging because much computational power to overcome the majority is required as difficulty increases [34].

Double-spend attack

This attack can occur when two or more conflicting blocks with the same height result in an inconsistency (i.e., *fork*). Cryptocurrency can be temporarily spent in both conflicting blocks, while later, only one block will be included in the main chain, and the other blocks will be discarded. Discarded blocks are also called *stale blocks*. It is recommended to wait a certain amount of time to prevent this attack. This time should be derived from *finality*. For example, recommended *finality* in Bitcoin is to wait for six blocks since the transaction was sent [34].

Selfish mining

Selfish mining is a vulnerability where an attacker privately builds up his own secret chain. Typically, a miner is expected to announce a block as soon as he finds it. If the rest of the network accepts the block, he gets a reward from this block. By not announcing the block right away, his chain gets ahead of the honest chain. He can continue to mine other blocks while he needs to be at least one block ahead of the honest chain. When the attacker reveals his chain, it will lead to inconsistency (i.e., *fork*). If his chain gets accepted, previously mined blocks will be discarded, and the selfish miner will get the reward from his privately mined blocks. This will lead to a waste of computational resources of other miners [34, 23].

2.7 Throughput limitations

Blockchains suffer from processing throughput bottleneck (see Fig. 2.5). In Proof-of-Work consensus protocols, it is caused by a limited number of included transactions in the block and the time required to generate this block. Bitcoin throughput is 3-7 transactions per second. Ethereum can process 15–25 transactions per second compared to the centralized financial system Visa, which already had a peak of 10,547 transactions per second [33].

This problem can be solved in the following ways by modifying blockchain parameters:

- **Increasing block size** and thus increasing the number of transactions in a single block. However, this leads to centralization as bigger blocks take longer to propagate on the network to all other miners. Smaller miners will be at a disadvantage because while they receive the latest mined block, bigger miners (or mining pools) that have just sent this block can start mining immediately for another block.
- **Decreasing block creation time**, determined by mining difficulty. In Bitcoin, the difficulty is dynamically changing to keep block creation time (λ) to an average of 10 minutes [43]. Decreasing this value will also cause an increase in the stale block rate. A *stale block* is a fully valid block that was successfully mined but was not

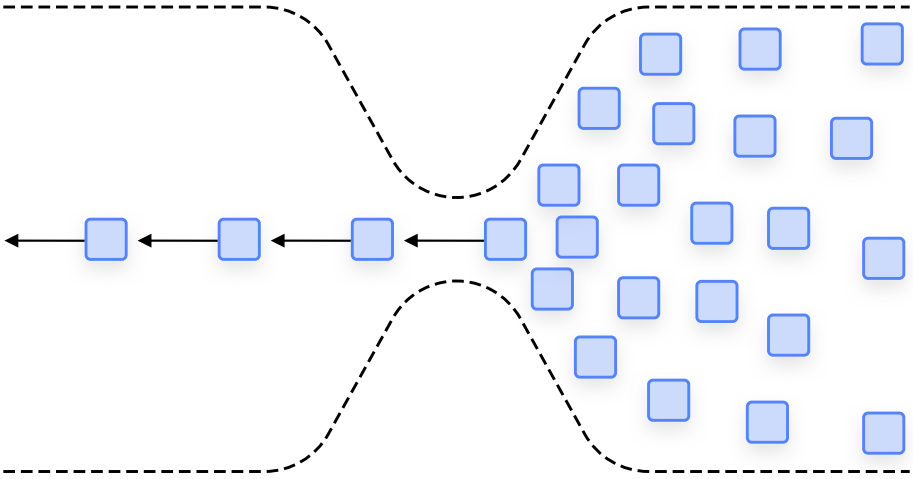

Figure 2.5: Simplified visualization of blockchains throughput bottleneck. It shows that many blocks with different transactions could potentially be generated, but instead, transactions are waiting, and only one block is chosen with a limited number of transactions.

included in the blockchain because some other block at the same height has already been included. Therefore, stale blocks are discarded, which results in wasted power and resources.

As a response to the blockchain throughput problem, several protocols have been proposed (e.g., IOTA [47], Obyte (Byteball [17]), SPECTRE [52], PHANTOM/GHOSTDAG [53]) that change blockchain data structure from single chain to Directed Acyclic Graph (DAG), as displayed in Fig. 2.6. The advantage of this structure over a back-linked list structure is that it can process multiple blocks at the same height and thus can increase transaction throughput. As another benefit, it also decreases the occurrence of stale blocks. We focus on the PHANTOM (and its optimization GHOSTDAG) as they propose a different solution than the other protocols.

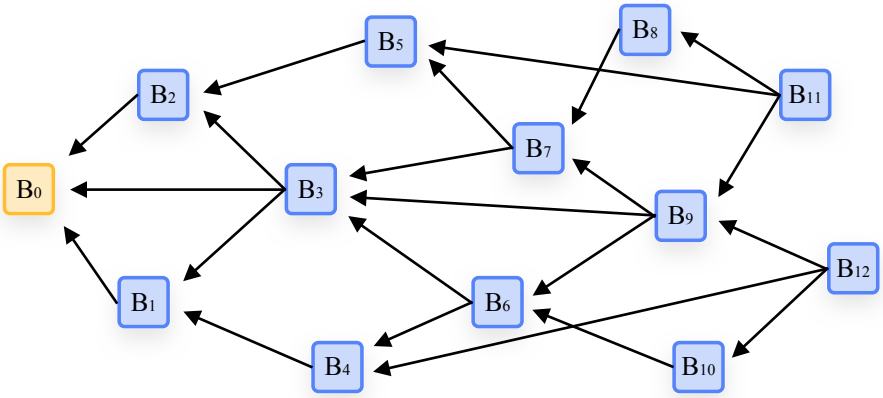

Figure 2.6: A DAG-oriented blockchain structure. Compared to the traditional blockchain structure, there can be multiple blocks at the same height.

Chapter 3

DAG-based consensus protocols

This chapter discusses the existing DAG-based consensus protocols and focuses on PHANTOM and GHOSTDAG [53]. For the context of this work, we can split available protocols into three categories:

- **Block-less.** These protocols do not create blocks but work directly with transactions instead. In this category, we cover IOTA [47] and Obyte [17].
- **Nakomoto's blockchain generalization.** These protocols utilize existing Nakamoto's consensus protocol and propose a new way to increase the limited throughput that Bitcoin suffers. While including whole blocks, this category comes with a new problem where a transaction can be included in more than one block (hereafter transaction collision). Protocols must solve this problem in order to work correctly. This includes SPECTRE [52], PHANTOM, and GHOSTDAG [53].
- **Other.** In contrast to the two previous categories, these protocols involve voting-based consensus. They might also combine other techniques (e.g., sharding or sidechain technique) to bring new features and achieve better performance. However, many of these protocols are still part of ongoing research. We mention some of the available protocols and their proposals [56].

3.1 IOTA

IOTA proposal brings new features than those introduced before. The first of them is the way how network processes transactions. Traditional blockchains like Bitcoin or Ethereum have two roles. Those who send transactions (users) and those who approve them (miners). Transaction fees motivate miners to include transactions to block. Using IOTA DAG, these roles were merged together, and instead of appending whole blocks to the DAG structure, append a single transaction. For the transaction to be verified, a new one must be sent where the sender verifies the two previously sent transactions from other users. In addition, a small amount of power must be used to solve an easily solvable Proof-of-Work puzzle for spam protection. IOTA has been designed to work well for microtransactions. This includes IoT devices that send many small messages to the blockchain. For this reason, there are no fees for transactions like in Bitcoin. Its protocol aims to be cooperative, not competitive. In summary, IOTA has a high throughput and scales the number of nodes in the network thanks to the DAG structure. The more nodes that send transactions to the network, the faster these transactions will be processed [47].

IOTA drawbacks

In order to secure the IOTA network, the protocol uses a centralized element called „Coordinator“. This node ensures the correct verification of transactions by other nodes. It thus prevents a 34% attack (similar to 51% attack) on the network, which would cause that if enough nodes approve a double-spending, then consensus could not be reached. With its centralized element, IOTA is more like Visa and loses unique blockchain feature — decentralization [47].

3.2 Obyte (Byteball)

Obyte (previously named Byteball) also utilizes a graph topology with feeless transactions like IOTA. Nonetheless, it later forms the main chain. Instead of using the Coordinator node to ensure security, Obyte works with twelve nodes called witnesses who are reputable and reliable users with uncovered real-world identities but also points of centralization. These users are expected to behave honestly. The number twelve was mainly chosen because it should provide enough protection against occasional failures of a few witnesses (they might get hacked, behave dishonestly, or go offline for a long time). It is also small enough that users can keep track of all witnesses. As long as the majority of witnesses behave as expected, potential attacks are detected in time and can be prevented. Common nodes can also replace witnesses. However, these changes only happen rarely since most users are required to agree on a new candidate [17, 56].

3.3 SPECTRE

Serialization of Proof-of-work Events: Confirming Transactions via Recursive Elections (SPECTRE) contains a similar mining process presented in Bitcoin. In Contrast to IOTA and Obyte, SPECTRE works with blocks instead of transactions. The key technique proposed in SPECTRE is a recursive weighted-voting algorithm based on the precedence of blocks in the topological sort of the underlying DAG structure. Blocks instead of miners complete the voting procedure. A newly found block is required to submit votes for every pair of blocks. This voting contains information about the ordering of the selected blocks. The Majority vote is used to extract a consistent set of transactions. A weighted-voting algorithm also resolves transaction collisions. SPECTRE also covers protection from hidden mining blocks with beneficial referencing recent blocks. Newly generated honest blocks, which refers to recent honest blocks, get more votes. SPECTRE achieves high throughput and low-latency finality compared to Nakamoto's Consensus. However, this design is still theoretical, and no one has proven it with real-world implementation or simulation of the size of a Bitcoin-like network [52, 56].

3.4 PHANTOM and GHOSTDAG

This section describes transaction selection strategies since PHANTOM and GHOSTDAG bring a new one that is not typically used, the details of these protocols, and their potential advantages and disadvantages.

Rational transaction selection

When a miner generates a block, he includes transactions from his mempool that contains the highest fee to maximize the profit he can earn from a potentially mined block. This approach is typical in blockchains that include transactions with a fee.

Random transaction selection

This approach is proposed in PHANTOM (and its optimization GHOSTDAG). DAG structure allows the creation of multiple blocks at the same height. Because transactions include fees, rational transaction selection would lead to enormous transaction collisions. Content of the blocks at the same height would be only slightly different. As a result, content produced in this DAG structure would be similar to traditional blockchain, which means no increase in throughput. For this reason, the authors of PHANTOM and GHOSTDAG propose a new transaction selection strategy that intends to select transactions randomly and reduce the transaction collisions rate.

PHANTOM

PHANTOM generalizes Nakamoto's Consensus protocol and includes blocks discarded in the longest/strongest chain. The protocol identifies a set of well-connected nodes to detect and exclude blocks created by dishonest nodes (see Fig. 3.1). Then, PHANTOM utilizes a recursive k-cluster algorithm in order to achieve a complete topological order. However, this algorithm has to find the maximal k cluster in the DAG, which is an NP-hard problem and unsuitable for implementation [53].

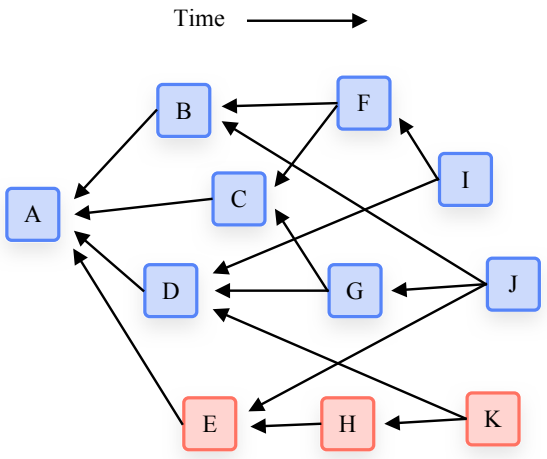

Figure 3.1: An example of the largest cluster of blocks within a DAG structure in PHANTOM. Blocks with blue color belong to honest nodes, and blocks with red color belong to an attacker [53].

GHOSTDAG

Since PHANTOM is impractical for usage, the authors created a greedy approximation algorithm called GHOSTDAG. The solution is to get a large enough k-cluster, not the largest one. GHOSTDAG protocol has been implemented as the underlying technology for

the Kaspa[1] cryptocurrency. Kaspad[2], a full node Kaspa implementation, is currently under active development. Compared to IOTA and Obyte, PHANTOM and GHOSTDAG do not contain any centralized element [53].

Summary

These protocols allow higher throughput by changing block creation rate and block size without affecting protocol security like in Bitcoin. To achieve this, they propose a strategy to reduce a transaction collision. However, the authors do not analyze how this incentive would work in combination with a rational selection strategy. In [46] was proved by simulations that this honest behavior results in extensively less profit compared to malicious actors that do not stick to the protocol and select transactions with the highest fee to increase their profit. In general, if a transaction has a higher fee than others, its chance to be processed in the next block is higher because multiple miners have an incentive to include it in the mining block to be rewarded with higher profit. Further, they show that malicious actors have a negative impact on transaction throughput. Simulations were done on a simple circle topology with ten miners.

3.5 Other DAG-oriented designs

Nano

Nano is a protocol using delegated Proof-of-Stake voting with a combination of Proof-of-Work (to reduce network spam). Unlike other blockchains, Nano uses a *block-lattice* structure where each account has its account chain (see Fig 3.2). Each account holder can select a representative or declare itself as its own representative. When the sender wants to create a transaction, he creates a block marked with *send* that refers to the latest block in his account. Once the block is broadcasted to the network, funds are deducted from the balance of the sender's account. Then the receiver creates a block marked with *receive* in his account chain, which also references the associated *sent* transaction. After that, a fork occurs, and the representative creates a vote referencing the conflicting transaction. Other representatives then join voting, and the transaction process is finished if the election succeeds [39, 56].

IOTA v2

IOTA version 2.0 is currently under development, which should achieve finality in seconds and use Fast Probabilistic Consensus (FPC) voting protocol and thus remove the Coordinator node [48].

3.6 Summary

IOTA and Obyte both contain the centralized element. IOTA states that its Coordinator node is temporary, but there is no proof that this will change except for proposed version 2. Obyte wants to replace witnesses nodes to be managed by the most reliable and independent users/institutions (e.g., universities).

[1] https://kaspanet.org/
[2] https://github.com/kaspanet/kaspad

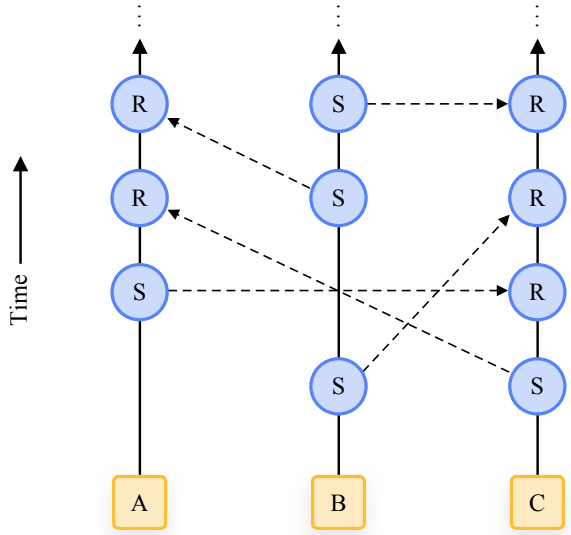

Figure 3.2: An example of Nano's block-lattice structure for individual accounts. Each transfer requires a block marked with send (S) or receive (R) signed by their account chains owner (A, B, C) [39].

We analyzed SPECTRE, PHANTOM, and GHOSTDAG, which generalize Nakamoto's consensus protocol. We further focused on PHANTOM and GHOSTDAG as they propose much higher throughput than Bitcoin. For comparison with other protocols, see [56]. We also discussed that simulations have proved this might not work because malicious actors can earn more profit than others and impact the final throughput. However, these simulations have been done on a simple circle topology with ten miners. Our work aims to extend the used simulator and experiment with a Bitcoin-like network with over 7500 miners.

We also mentioned consensus protocols Nano and IOTA version 2 that do not involve Nakamoto's proposed Proof-of-Work and instead use voting to reach consensus. A comparison of blockchain features of some of the mentioned consensus protocols with Visa is shown in Figure 3.3.

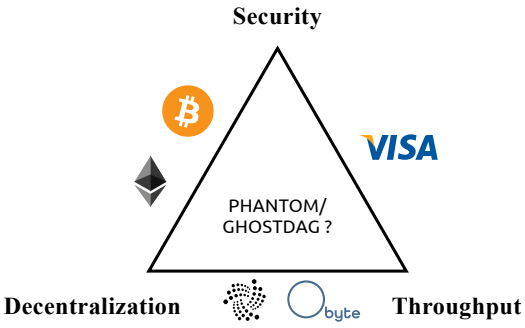

Figure 3.3: Blockchain trilemma showing a comparison of previously mentioned consensus protocols. PHANTOM and GHOSTDAG are shown in the middle, but they involve a different potential problem with unfair rewards, and therefore high throughput is also unsure. Verification of these problems is part of this work with the simulator. IOTA and Obyte are centralized as a consequence of their security issues. Because of that, they are on the side that lacks security, not on the same with Visa.

Chapter 4

Related work

This chapter analyzes available simulators, compares their features, and discusses why the original simulator was chosen as a base for the simulator in [46].

4.1 Verification of blockchain properties

Today, after the successful implementation of blockchain technology in Bitcoin, blockchain has spread to the world, and its unique features give new opportunities to several industries such as banking and finance, the Internet of Things (IoT), real estate, healthcare, and others [49]. It is necessary to continuously test and verify when proposing new changes or modifications to existing blockchains. The challenging point during this process is that some of their properties only show up with a large number of nodes. Since formal modeling cannot cover all properties, another option is to deploy real nodes in a real network, which can be a problem with a large number of nodes (e.g., the Bitcoin network currently has more than 14,000[1] reachable nodes). It would also require minimal blockchain implementation. For this reason, in many cases, simulation remains the only option to verify their properties as it simulates real network behavior. However, the design of such a simulator is complex, and therefore most simulators focus on specific aspects that they implement and only abstract the rest [45].

4.2 Three-layer architecture

To describe a blockchain simulator architecture and functionality, we adapt a three-layer blockchain abstraction (see Fig. 4.1) that includes a network layer, a consensus layer, and a data layer. Original adoption of the multi-layer abstraction in [45] covers an additional two layers — execution (of smart contracts) and application layer. However, these two layers are out of this work's context and have been excluded for simplicity.

1. *The network layer* is at the bottom of the three-layer architecture and consists of a peer-to-peer network that allows decentralized blockchain to operate without a centralized entity. Key features of the peer-to-peer architecture, such as network delay, block synchronization, and peer discovery, are crucial elements to the proper functioning blockchain. Network participants/peers can be divided into full nodes and light nodes. Full nodes are identified by storing the entire blockchain, both

[1] https://bitnodes.io/

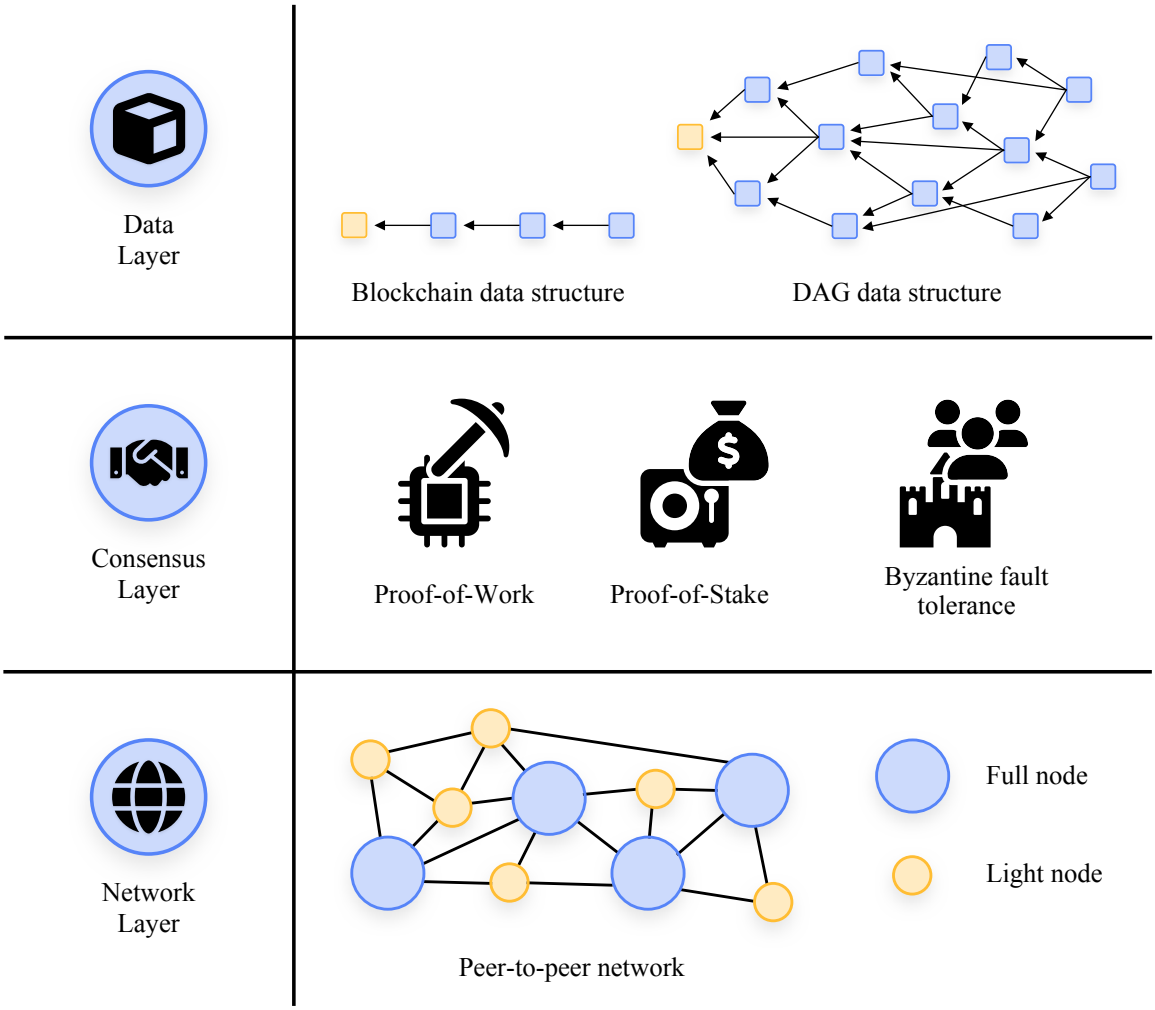

Figure 4.1: Three-layer architecture [45].

its history and newly generated blocks. Furthermore, they receive, broadcast, and verify transactions and blocks and can also perform mining to generate new blocks. Nevertheless, light nodes retain only the block headers, which are used only to validate the authenticity of the transactions. Their functionality depends on full nodes. The storage of block headers takes up considerably less space than the entire blockchain and is therefore used, for example, in a wallet [24, 45].

2. *The consensus layer* describes how nodes in the network should behave in order to reach a decentralized agreement or consensus. Thanks to this mechanism, all nodes have an identical copy of the ledger. Most of the existing consensus algorithms can be divided into three groups. Proof-based, BTF-based, DAG-based [59, 45].

    - *Proof-based* consensus is widespread among public blockchains and includes algorithms such as Proof-of-Work. In order for nodes to reach a consensus, they must include in a message proof that other nodes can verify that a specific participant has used its resources to find a solution. In Bitcoin, this is a solution to a computational problem using the SHA-256 hash algorithm. It is necessary to find a hash that meets certain requirements. The hash value must be less than

the target value, which is determined by difficulty and can be obtained only by adjusting the Nonce value [31]. The difficulty is dynamically adjusted, and thus the duration of finding one block is expected to be 10 minutes. Currently, the difficulty is still increasing. Finding a single block may be challenging, and a chance depends on individual hash power. The first block with valid proof is accepted in the Bitcoin network, and the other blocks with the same height are discarded, and the other participants waste their resources. This race is often associated with winning the lottery. As a result, the Proof-of-Work algorithm offers high security, decentralization, and scalability. However, its disadvantages are low throughput, high confirmation time and high energy consumption [24, 59].

- *BFT-based* is commonly used in private blockchains, and its frequent representative PBFT (Practical Byzantine Fault Tolerance) [14]. Consensus is achieved through communication (exchange of several messages) between the participants. When creating a new block, one leader is selected from the participants, who verifies the validity of the available transactions and groups them into a block, which is then broadcasted to the other participants. Each of them verifies the received block and casts its vote on whether they agree with the new block or not. A majority rule decides the result. Compared to Proof-of-Work, PBFT allows for high throughput and requires only a small amount of energy to operate. Its drawbacks are low network scalability because it needs to exchange more messages for functional communication and may also lead to centralization [59].
- *DAG-based* consensus algorithms use the DAG structure instead of the traditional blockchain structure to organize blocks or transactions. Reaching consensus varies for different DAG-based blockchains and may also combine Proof-based with BFT. We already analyzed existing consensus algorithms in Chapter 3.

3. *The data layer* describes the elementary parts of the blockchain, which are typically block and transaction. Details about block and transaction are described in Section 2.3. Transactions processing also depends on the fee and transaction selection strategy described in Section 3.4. Transactions with low or zero fees can get into the block after a very long time or not at all. It depends on how many transactions are waiting to be processed and how many nodes are willing to process them. Sections 3.1 and 3.2 mention that blockchains such as IOTA or Obyte do not use fees as incentive. [45].

4.3 Simulation approach

The main purpose of the extending simulator is to verify miners' profit and transaction collision rate. Therefore the simulator needs to implement the network layer and data layer. It is also important to note that both layers should not reflect real implementations of blockchain software. We do not simulate attacks on the network, and all simulated actors follow the consensus algorithm. Thus, we can skip the consensus layer and its properties (e.g., Merkle tree hash, SHA256 hash algorithm, block and transaction verifications). A simulation spends most of the time with basic operations with mempool as sort, insert or remove. For this reason, implementation of mempool is essential, and the whole simulator has to be implemented in a compiled language, not interpreted language.

4.3.1 Simulator requirements

The simulator with extension in this work must meet the following requirements:

- It is based on a discrete event simulation model.
- It implements a network layer that includes block propagation with custom latency between peers and custom network topology.
- It implements a data layer that involves transactions in blocks and, for each miner, his own mempool data structure.
- It is implemented in a compiled language.

4.4 Available simulators

This section describes the analysis of available simulators, compares their properties, and establishes the final selection, on which M. Perešíni et al. created the first extension [46] and the second extension, which is processed in this work.

Analysis of available simulators

The list of available simulators was taken from the *A Systematic Review and Empirical Analysis of Blockchain Simulators* (proposed by R. Paulavičius et al.) [45]. This review includes high-quality, peer-reviewed papers published in leading computer science journals, conferences, symposiums, and books. The search performed in the review was made in November 2020 and included 64 publications from which 26 different simulators were selected after a detailed examination. An analysis of available simulators was performed for this work, which could be used as a base.

Nine of 26 available simulators have been excluded that did not have public source codes and one more that was publicly available at the time of the review proposal but not in March 2022.

Besides that, we also added one more simulator to our analysis that is not covered in the review proposal, *Bitcoin mining simulator* [9], created by G. Andresen in May 2015. G. Andresen was an active top third contributor of Bitcoin Core[2] in 2010 – 2016 [1, 2]. He was also the former lead maintainer for a software client. The simulator he created uses parts of the code that is implemented in Bitcoin Core. Specifically, an event scheduler that models discrete-event simulation.

Table 4.1 shows comparison of all 17 selected simulators. From these were selected those that meet all three requirements:

- Use Proof-of-Work.
- Based on the discrete-event simulation model.
- Include a network layer.

This results in ten simulators detailly analyzed below.

[2]Bitcoin Core is an open-source software used to run a node and connect to the Bitcoin network. Satoshi Nakamoto originally released it in early 2009. Nakamoto also continued to maintain the software until he disappeared in late 2010. After that, a series of individuals continued to develop and maintain the project [4].

Table 4.1: Comparison of chosen 17 simulators [45].

Simulator name	BTC/PoW	Discrete-event	Network layer
BTCsim [12]	✓	✓	✓
Simbit [11]	✓	✓	✓
Shadow-Bitcoin [41]	✓	✓	✓
Bitcoin-Simulator [32, 30]	✓	✓	✓
Bitcoin-Simulator extension [50, 51]	✓	✓	✓
VIBES [44, 54]	✓	✓	-
eVIBES [20, 21]	✓	✓	-
BlockSim:Faria [25, 26]	✓	✓	✓
SIMBA (extension) [27, 28]	✓	✓	✓
BlockSim:Alharby [6, 5]	✓	✓	-
SimBlock [55, 10]	✓	✓	✓
SimBlock extension [15, 16]	✓	✓	✓
DAGsim [57, 58]	-	-	✓
Incentive network simulator [29, 13]	✓	-	-
Proof of Prestige Simulator [36, 37]	-	-	-
Core-BTC Network Simulator [7, 8]	✓	-	✓
Bitcoin Mining Simulator [9]	✓	✓	✓

4.4.1 BTCSim

BTCSim is implemented in Python and supports consensus and network layer. It realistically simulates the mining time of the new blocks, which is given by the exponential distribution depending on the miners' hash power. These blocks do not contain any transactions, and their size is also simulated. Therefore, this simulator is not intended to make a profit analysis. Nevertheless, it includes examples of 51% attack, selfish mining, and focus on consensus layer [12].

4.4.2 Simbit

This simulator focuses mainly on the consensus and network layers but also implements the data layer. Thanks to the JavaScript implementation, it allows visualizing the results in real-time directly in a web browser. It offers an example of a selfish mining attack. Its use case is mainly for visual tracking of a simulated Bitcoin-like network [11].

4.4.3 Shadow-Bitcoin

Shadow-Bitcoin is a plugin for Shadow network simulator [35] that runs real applications like Tor to simulate distributed systems with thousands of network-connected processes in a private network. This plugin is implemented in Python, but the Shadow simulator is implemented in C and Rust. Its next advantage is the use of a system call API, and thus unmodified Bitcoin core application does not need to be aware that it is running in Shadow simulation. Although this simulator uses a realistic Bitcoin implementation, it is too complex, and further modifications would require many implementation changes [41].

4.4.4 Bitcoin-Simulator

Bitcoin simulator is built upon NS3[3] discrete-event network simulator and is capable of simulating the Bitcoin network with real region data (i.e., download, upload, latencies, and entire network topology). The focus of this simulator is to study how consensus parameters, network characteristics, and protocol modifications affect the Proof-of-Work blockchain. Simulator implementation works only with blocks that contain no transactions, and block size is adjusted by block size distribution and average transaction size. It also contains data for Litecoin and Dogecoin, and test scenarios for selfish mining and double-spending attack [32].

4.4.5 Bitcoin-Simulator extension

Bitcoin simulator was extended with new attack experiments. This extension also contains two types of actors: miners and an attacker. Attacker strategies are modeled using the Markov decision process [50].

4.4.6 BlockSim:Faria

BlockSim by Faria was created to simulate either the Bitcoin or Ethereum network distributed in 3 places — Ireland, Ohio, and Tokyo. Besides the network layer, it also supports a data layer and simulated blocks, including timestamp, difficulty, nonce, full transactions, and a previous block hash. The simulator implements an algorithm for block mining difficulty adjustment, and the consensus layer can validate blocks and transactions. This simulator can be easily extended with new properties to simulate. Its downside is that it is implemented in Python language [25].

4.4.7 SIMBA (BlockSim:Faria extension)

Extension of BlockSim adds Merkle Trees to get more realistic results and to study how Merkle Trees can improve the performance of nodes when verifying the consistency of transactions in blocks [27].

4.4.8 SimBlock

Implementation of SimBlock is done in Java and focuses on network and consensus layers. It supports rich simulation configuration settings, including a network of regions with download, upload, latency, and regional distribution. Miner settings can be adjusted with block

[3] https://www.nsnam.org/

size, average mining power, the standard deviation of mining power, and possible failure settings. The output that simulator produces can be visualized on a web browser [55].

4.4.9 SimBlock extension

The extended version of SimBlock utilizes a difficulty adjustment algorithm and options to increase and decrease hash rate dynamically [15].

4.4.10 Bitcoin Mining Simulator

Bitcoin Mining Simulator created by Andresen is implemented in C++ and utilizes Boost library [3]. Although it does not implement transactions in blocks or mempool, its implementation is fast and straightforward. The simulator includes a network layer and involves block propagation across the peer-to-peer network. Miner settings — hash power, connections, and latency with other peers can be set in a separate configuration file. It also includes simple configurations with ten or fewer miners to verify their block shares and stale block rate [9].

Table 4.2: Comparison after a detailed examination of potential simulators.

Simulator name	Custom network topology	Transactions in blocks	Compiled language
BTCsim [12]	✓	-	-
Simbit [11]	-	✓	-
Shadow-Bitcoin [41]	✓	✓	✓
Bitcoin-Simulator [32, 30]	✓	-	✓
Bitcoin-Simulator extension [50, 51]	✓	-	✓
BlockSim:Faria [25, 26]	✓	✓	-
SIMBA (extension) [27, 28]	✓	✓	-
SimBlock [55, 10]	✓	-	✓
SimBlock extension [15, 16]	✓	-	✓
Bitcoin Mining Simulator [9]	✓	-	✓

4.4.11 Conclusion

A comparison of these properties in a selection of ten simulators can be seen in Table 4.2. *BTCsim* and *Simbit* cannot be used because they are not implemented in a compiled language. *Shadow-Bitcoin* is the only simulator that meets all three factors in the table. However, this simulator involves real Bitcoin implementation, and it would be extremely complex to build upon it. *BlockSim:Faria* could be an ideal choice, but it is implemented

in Python and therefore does not meet the requirement of a compiled language. *Bitcoin-Simulator* and its extension are built upon NS3, and its network simulation process is complex for our purpose. Although *Bitcoin Mining Simulator* does not implement transactions in blocks, it was selected by M. Perešíni in [46], who further extended it. This simulator was selected because it is implemented in a compiled language, is based on Bitcoin, and is simple enough to be extended. An alternative choice could also be a *SimBlock*.

Chapter 5

Proposed extensions

This chapter compares previous versions of the selected simulator and sets new requirements implemented in this work. Extension of the simulator includes an option to create Bitcoin-like network topology and modifications to the simulator to be able to simulate a network of this size. Furthermore, it describes scripts that are used to analyze simulation outputs that are also part of the simulator extension.

5.1 Existing versions of simulator

This section describes previous versions of the simulator: the original implementation to simulate Bitcoin miners by G. Andresen (hereafter version 1) and its extension by M. Perešíni (hereafter version 2).

5.1.1 Version 1

The original version was released seven years ago to simulate block generation and propagation across the network. The simulator verifies the miner's block reward according to their mining power and stale block rate. It allows setting arguments that modify the number of blocks that will be simulated, block propagation latency, a seed for random generator Mersenne Twister[40] to reproduce the results, number of runs, and configuration file with miners settings. The configuration file contains miners' descriptions of their mining power and connections. Miners can only be connected bi-directionally. Examples for this simulator contain network topologies with ten or fewer miners connected randomly or in a ring topology. Results are analyzed from collected information at the end of the simulation and show how network topology and latency parameters affect miners' profit. This version contains two different events that occur during the simulation process:

- *Block generation.* Miners selected by their mining power generate blocks continuously in time defined by the following probability distribution: $600 \cdot exp(1)$[1]. The number 600 represents 10 minutes, which is an average time to generate a block in Bitcoin [43].
- *Block propagation.* Miners receive blocks from other miners through a peer-to-peer network with a delay specified in the configuration file for each connection.

[1] Exponential distribution with λ equals one.

5.1.2 Version 2

Version 2 extends the original version to fill the needs of simulating DAG-based protocol. It includes implementing the mempool data structure, which requires multiple access for different situations. This problem is detailly described in Section 5.4. Although we work with a DAG structured protocol instead of a traditional Nakamotos blockchain, DAG structure output is not required for our experiments. Block ordering in DAG structure in PHANTOM/GHOSTDAG is an NP-hard problem and must be solved in the consensus layer [53]. We can abstract this problem because we do not simulate the consensus layer. Version 2 stores block information, including transactions stored inside a block, block height calculated from the time it has been mined, and the miner who generates it. Besides that, this extension adds two new events:

- *Initial transaction generation.* Generate a predefined number of transactions for each miner at the start of the simulation. At this moment contents of miners' mempools are equal.
- *Transaction generation.* Generate n transactions each m seconds, where n and m are predefined values in simulator implementation.

The following probability distribution generates transaction fees: $150 \cdot exp(1)$. The number 150 was chosen as an average for transaction fees in abstracted cryptocurrency. Transactions are generated directly into each miner's mempool, and thus transaction propagation is abstracted. Each miner has the same limited mempool capacity, and if newly generated transactions exceed it, transactions with the lowest fee are removed. This also applies to honest actors. Similar to version 1, version 2 collects information during the simulation process to memory and analyzes results in the end. This version simulates four different events in total.

5.1.3 Version 3 extension proposal

We created a distribution of Bitcoin peer connections from data in [42]. To simulate network delay between peers, we made distribution from data collected in Bitcoin live monitoring website [22]. With these data, we created a configuration to simulate a Bitcoin-like network with miners that use different transaction selection strategies based on PHANTOM and GHOSTDAG. However, the length of the simulation process shows that the simulator is not well optimized for the complex network. Moreover, we sum up functional requirements for an extended version of the simulator:

- *Update of simulator architecture.* The previous version was not well organized to make additional extensions. For this reason, this includes changing the organization of modules: their separation, update, and the creation of new ones. Additionally, it also contains changes to three out of four existing events.
- *Bitcoin-like network.* We intend to further verify profit from malicious actors that use rational transaction selection over proposed random transaction selection based on PHANTOM and GHOSTDAG on a network of larger size and topology similar to Bitcoin. In addition, we plan to analyze transaction collisions from results to verify protocol throughput.

- *Progress tracker.* The simulation process in the original version can take hours, days, or even weeks but strongly depends on input parameters. Since the original version was missing any progress tracking, we require this feature to analyze running simulations further.
- *Optimization.* In order to simulate a complex network, we require optimization that accelerates the simulation process and has less memory usage, enabling us to run more simulation processes simultaneously.
- *Automation.* As we run multiple simulations in parallel, we want to automate this process and be able to run a combination of simulations with different configurations and seeds. In addition, we demand automation in the processing of simulation results.
- *Payoff function.* Assume we confirm our hypothesis that malicious actors will also have an advantage in profit gains in the larger peer-to-peer network. In that case, we want to experiment with payoff functions that split transaction rewards that occurred in transaction collisions.

5.2 Update of simulator architecture

Previous versions depended on a C++ boost library [3]. Boost was used for the implementation of the Mersenne twister generator, uniform and exponential distributions, and task scheduler. These features have already been implemented in the C++11 standard library, but the author of version 1 has not supported them. The aim of version 2 was to add required features, not reimplement existing features. Our version does not depend on a boost library because all required features are supported in most compilers.

5.2.1 Update of existing events

In order to simulate a Bitcoin-like network, we updated the three out of four events.

Initial transaction generation

The number of initial transactions count can be specified in the argument before the simulation starts.

Transaction generation

This event was parametrized with transaction count generation interval and its time generation interval. Both intervals use uniform distribution. Condition of checking miners' mempool fulness was extended to check extra simulator argument to remove transactions randomly for honest miners instead of rationally. Besides that, it also contains a condition to stop generating transactions. This was missing in version 2, and this event was repeated a predefined number of times.

Block generation

Added condition to check if a miner has enough transactions to generate a block. If not, the simulation stop and prints the state of each miner's mempool. This was also missing in version 2, which caused a simulation crash. When generating a block, it checks if the miner

is honest or malicious and appropriately selects transactions from his mempool. After that, information about the included transaction is saved into a data file for post-processing. Besides that, simulation progress is printed if block id reached a new simulation step (see Section 5.5 for details). At the end of the generation, mempool states of all miners are logged if it is enabled by argument.

Block propagation

This event meets all requirements for our simulations since version 1. It has not been updated in version 2 or version 3.

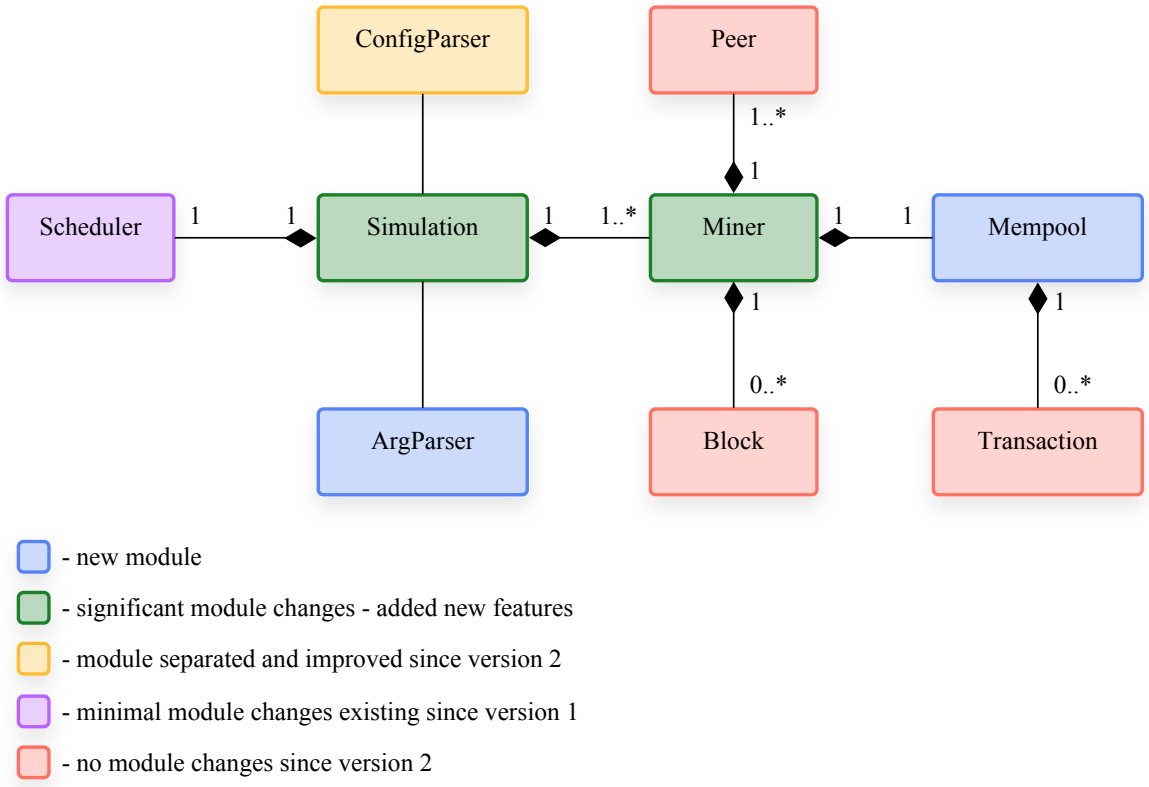

Figure 5.1: Class dependency diagram. It also shows whether modules are new or have been modified from existing versions 1 or 2.

5.2.2 Simulator modules

The structure of version 3 is shown in Figure 5.1. It contains the following modules:

- *ArgParser.* Parse and validate program arguments.
- *ConfigParser.* Parse and validate the input configuration file.
- *Simulator.* The main module of the whole simulation process. After successful argument parsing, simulation starts by calling `runSimulation` method. Its sequence diagram is shown in Figure 5.2.

- *Scheduler.* It contains the simple implementation of an event calendar to run discrete event simulations. Prioritization is not included because it is not required in simulations.
- *Miner.* Implementation of a node in a peer-to-peer network that involves mining.
- *Peer.* Includes reference to a miner and block propagation latency to this miner.
- *Block.* It contains a definition of a block structure (i.e., block id, blockchain height, and transactions).
- *Mempool.* Data structure unique for each miner that holds his transactions that can be included in a block.
- *Transaction.* It defines transaction structure. (i.e., transaction id and fee).

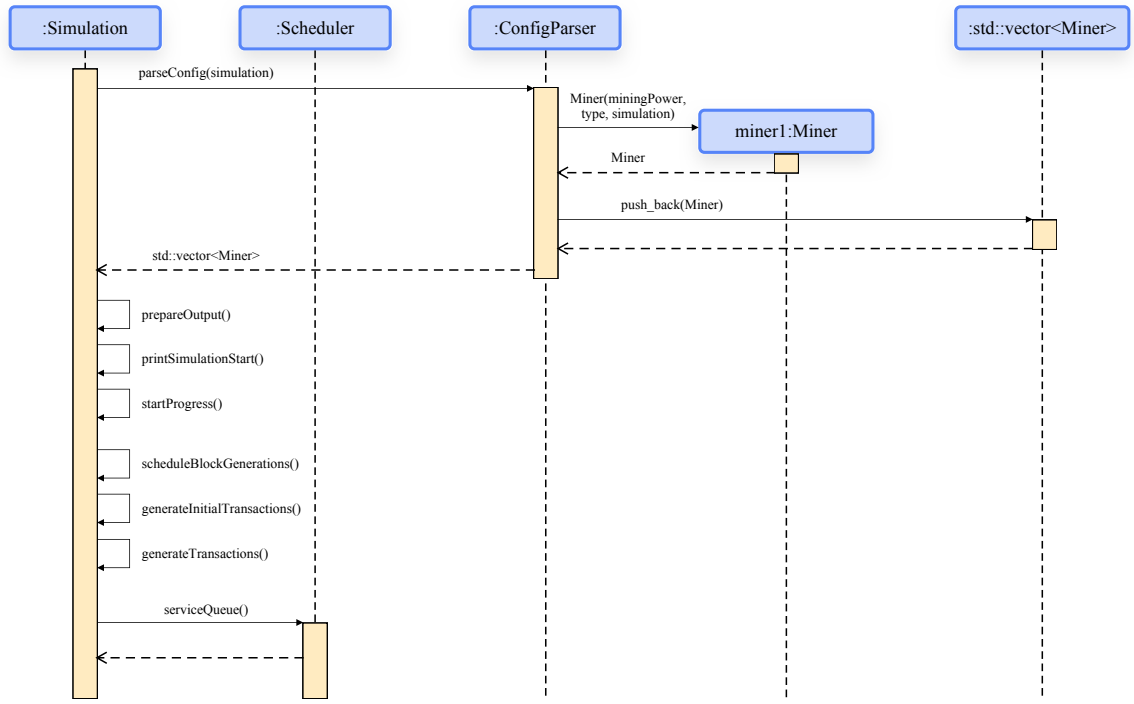

Figure 5.2: Sequence diagram for the `runSimulation` method. This method is called from `main.cpp` file after successful argument parsing. It shows configuration parsing with one miner and further preparation before the simulation starts.

5.3 Bitcoin-like network

We intend to further verify profit from malicious actors that use rational transaction selection over proposed random transaction selection based on PHANTOM and GHOSTDAG on a network of larger size and topology similar to Bitcoin. In addition, we plan to analyze

transaction collisions from results to verify protocol throughput. In our further experiments, we used created a Bitcoin-like network of size 7592[2] nodes. This section describes the proposed distributions used to create such a network of this size.

Number of connections per node

Nodes in the Bitcoin network have a different number of peer connections. We created a discrete distribution from data published in [42]. The result is shown in Fig. 5.3. Results from this distribution are then specified in the configuration file. The tool that generates connections between nodes from the proposed distribution was created by F. Matteo Benčić during the continues research progress and this work.

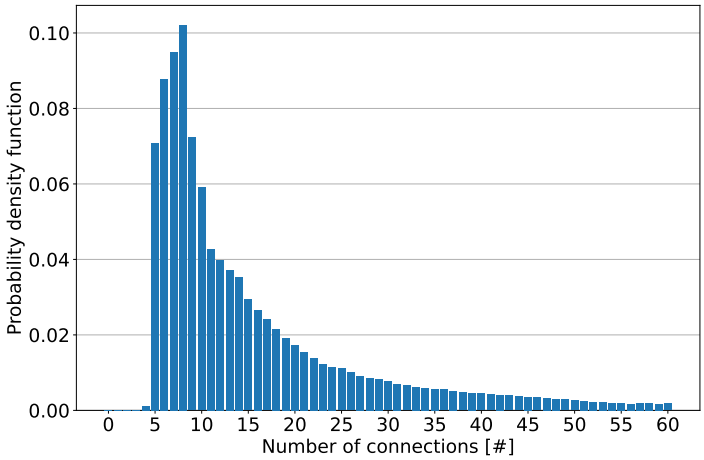

Figure 5.3: Distribution of the number of connections per node created from data published in [42].

Block propagation delay

We created discrete distribution to simulate block propagation delay between nodes. It was produced from 24-hour data published in [22] on 28 February 2022, from a running node in a Bitcoin network. Fig. 5.4 shows a portion of the distribution as blocks get mostly propagated under two seconds. This delay depends on block size. The average block size in Bitcoin is 1MB [18]. In further simulations, we expect that produced blocks are fully-filled with a specified number of transactions and that these blocks have the same size. Thus the time required to receive a block is also the same. Results from the distribution can be stored in the configuration file or can be generated within a simulation. For simplicity, these values state bi-directional latency.

Abstraction of transaction propagation delay

A newly created transaction in Bitcoin takes an amount of time to propagate to the whole network. Version 2 abstracts this, and instead of propagating new transactions, it directly

[2]Number 7592 comes from https://bitnodes.io/, and it states the number of IPv4 reachable nodes in the Bitcoin network on 24 November 2021, when the first topology was created.

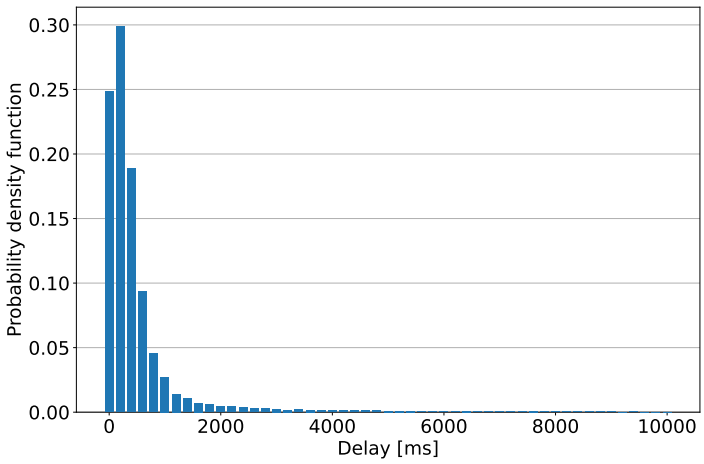

Figure 5.4: Distribution of block propagation network delay. It was created from 24-hour data published in [22] on 28 February 2022. Implemented distribution continues up to 30000 ms. The figure only shows a portion up to 10000 ms for better visibility.

adds these transactions to each miner's mempool. We want to continue experimenting with the current transaction propagation setting. Transaction propagation only slightly affects the random transaction propagation. An honest miner does not depend on transactions with the highest fee. However, a late transaction could negatively affect the result of a malicious miner. Implementing a transaction creation from single or more miners could affect the achieved results. For this reason, we leave this implementation for future work and the next version.

5.4 Optimization

In order to simulate a complex network, we require optimizations that accelerate the simulation and have less memory usage, enabling us to run more simulation processes simultaneously. We focus on optimizing the mempool data structure because simulation spends most of the time manipulating the mempool data structure.

5.4.1 Mempool operations

From previously mentioned simulation events, we can classify mempool operations:

- Initial transaction generation
  - *Direct insert.* Initial transactions are generated with a unique id and inserted into each miner's mempool to be able to start mining.
- Transaction generation
  - *Sorted access and remove (ascending).* When the miner's mempool is full-filled, he has to remove transactions. Because malicious miner wants to earn the most profit, he removes transactions with the lowest fee. By default, this way is also

preferred by honest miners. If honest miners remove transactions randomly, they would have a disadvantage over malicious miners. For a particular situation, when a fee of the newly generated transaction is lower than the fee of each currently stored transaction, the required amount of transactions is removed to make space for new ones. If this situation occurs in reality, malicious miners will keep older transactions with higher fees that have not been included in any block yet. We achieve balanced transaction removal on full mempool capacity for both honest and malicious miners with this approach.

  - *Random access and remove.* Although we do not currently intend to use random transaction removal with honest miners, we reserve this feature if we want to give them a disadvantage or do further investigations.
  - *Direct insert.* Each transaction is generated with a unique id and inserted into each miner's mempool.

- Block generation
  - *Sorted access (descending).* It is required in rational selection strategy by a malicious miner when he generates a block he needs to sort (or has already sorted) a set of transactions from his mempool.
  - *Random access.* Honest miners need to randomly access a transaction set to create a block while using a random selection strategy.
  - *Direct remove.* When an honest or malicious miner creates a block, he must remove included transactions from his mempool.

- Block propagation
  - *Direct remove.* When a mined block is propagated to other miners, they remove already included transactions from their mempool.

Mempool operations summary

By examining existing operations of the mempool, we end up with three different access methods to mempool, and we can classify them as shown in Fig 5.5.

Multi-index mempool implementation

Version 2 implements mempool using multi-index[3], a unique data structure from the C++ boost library. Multi-index allows the creation of a container with multiple index interfaces, as shown in Fig. 5.6. However, it is a generic data structure and is not well-suited for our purpose. Therefore, we need to test other data structures.

Comparison of other data structures

- *Hashtable.* Its average time complexity for search, insert, and remove is $O(1)$. Its only drawback is that it does require sorting all elements when accessing transactions in ascending or descending order.

[3] https://www.boost.org/doc/libs/1_78_0/libs/multi_index/doc/index.html

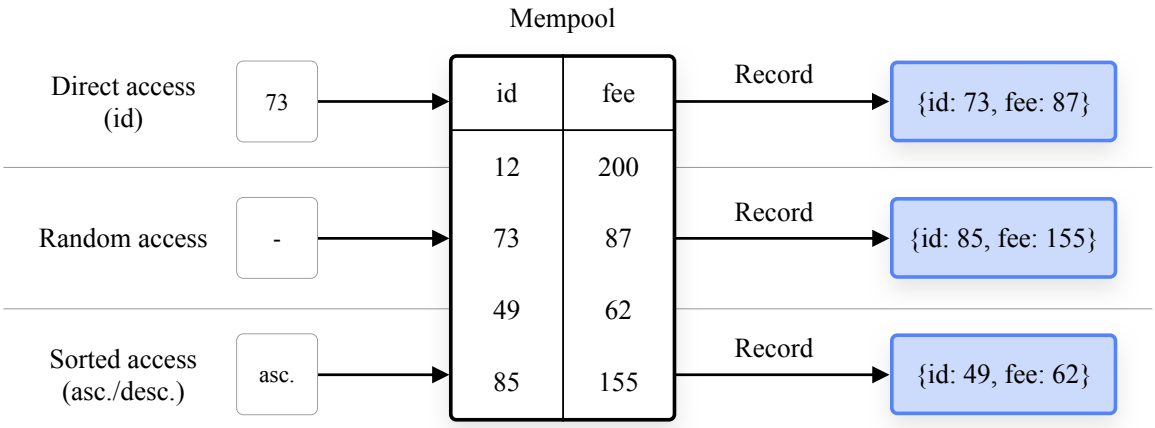

Figure 5.5: Showcase of different access methods to mempool. Direct access returns a transaction specified by id. An id is our simplified unique identifier of the transaction. Random access returns a transaction randomly selected from mempool. Sorted access returns one or more transactions in either ascending or descending order sorted by a fee.

- *Red-black tree.* It provides fast search, insertion, and removal. Its average time complexity for search operation is $O(log(n))$ and implicitly sorts all elements by a key.
- *AVL tree.* In contrast to the Red-black tree, the AVL tree provides a faster search as it is more strictly balanced. However, it is a tradeoff for slower insert and removal. Because mempool structure also requires other operations than fast search, the Red-black tree is a better option.
- *Binary (min/max) heap.* This structure has $O(1)$ average time complexity for search operation for minimum or maximum value depending on whether it is min-heap or max-heap. The tradeoff for this advantage is average time complexity $O(n)$ for the other side. This could be fixed by implementing both min-heap and max-heap and mirroring their content. However, Section 5.4.2 states that a red-black tree has problematic random access, which also applies to the heap. It means that two structures, min-heap, and max-heap, still do not provide efficient access to mempool.

Because we require efficiently implemented all three operations (search, insert, remove), we mainly focus on the hashtable and red-black tree.

5.4.2 Random access

It is important to note that we cannot approximate random access because it would skew the results (e.g., randomly choosing node by node in a red-black tree to access the element). Our simulations require that an honest miner's transaction selection is independent of the selection of other honest miners. Another important note is that random selection will be used most of the time in simulations. Our goal is for malicious miners to have less than 50% mining power in total (i.e., more than 50% involves 51% attack). Therefore, we need an efficient pseudo-random selection algorithm for mempool data structure.

Firstly, we have analyzed our selected data structures. We started by excluding a red-black tree because its average and worst-case time complexity to select an element randomly

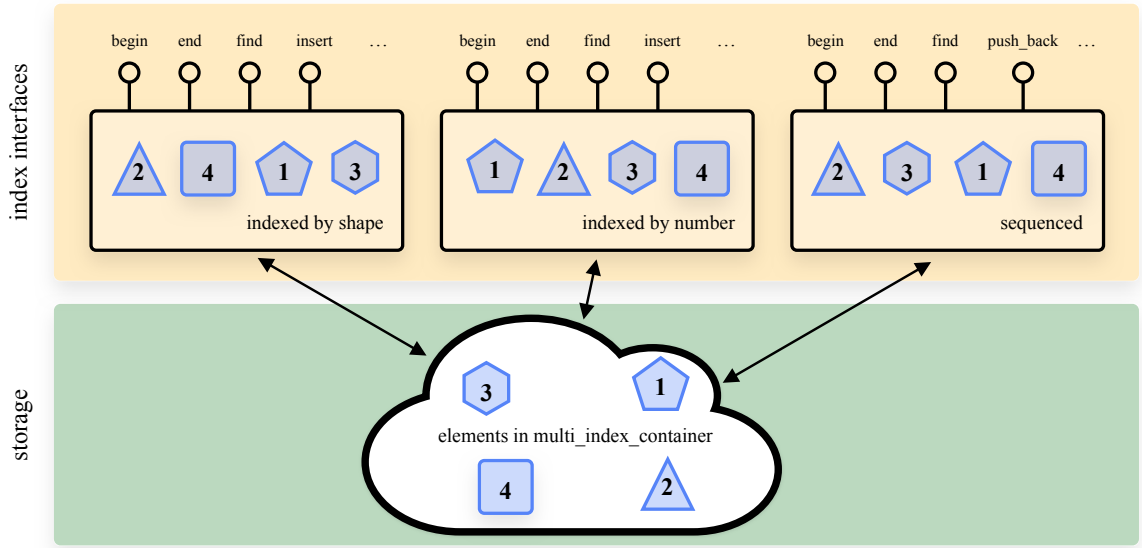

Figure 5.6: Diagram of boost multi-index container. Elements are stored inside a container and be accessed from different index interfaces. These interfaces provide search, insert and remove operations [3].

is always $O(n)$. It requires iterating all elements and then making a selection and thus is not suitable for our algorithm.

For hashtable, its random selection strongly depends on its fullness if it has a limited size. A random index for hashtable is acquired from a unique key for each miner. It can be achieved by combining the miner's id and transaction id. We used hashtable buckets implemented as linked-list. We propose an algorithm *random access* (see Algorithm 1) that can randomly select an element stored in the hashtable. The average time complexity of this algorithm is $O(\frac{m}{n})$, where m is hashtable capacity and n is the number of elements stored in a hashtable. The worst-case time complexity is $O(n)$. Thanks to the miner-unique random hashtable keys, we achieved random selection from hashtables from content stored differently for each miner's hashtable.

We additionally propose the following algorithm modifications:

- *Begin.* Instead of choosing the index randomly, it starts with a zero value. Then, the value is incremented until it finds an element. After that, it randomly selects an element from a linked list if multiple elements are in the bucket. This approach gives skewed results because elements at the end of the hashtable will be selected rarely or never, depending on the fullness of the hashtable. We will use it for further comparison.

- *Begin equal key.* This approach is similar to Begin but while not use miner-unique random hashtable keys. Content of key consists only of transactions, and thus all miners will then have transactions on the same hashtable indexes. We expect this to lead to an enormous increase in the transaction duplication rate. This approach is for comparison with other algorithms.

We further verify multi-index random implementation and compare it with previously proposed algorithms. This was done by simulating a proposed Bitcoin-like network with

Algorithm 1 Hashtable random access

```
1: arrSize: hashtable maximum size
2: htabBuckets: hashtable buckets
3: rng: Mersenne Twister pseudo-random generator
4: function getRandomTx(rng):
5:     index ← randomHtabIndexGenerator(rng)  ▷ Pick random index with maximum value arrSize - 1
6:     max ← (arrSize/2) + 1
7:     down ← index
8:     up
9:     if index = arrSize − 1 then
10:        up ← 0
11:    else
12:        up ← index + 1
13:    end if
14:    for i ← 0 to max do
15:        if down = −1 then
16:            down ← arrSize − 1
17:        end if
18:        if htabBuckets[down] is not empty then
19:            if htabBuckets[down].size() = 1 then
20:                return htabBuckets[down][0]
21:            else
22:                bucketIndex ← random index in size of the bucket - 1
23:                return htabBuckets[down][bucketIndex]
24:            end if
25:        else
26:            down ← down − 1
27:        end if
28:        if up = arrSize then
29:            up ← 0
30:        end if
31:        if htabBuckets[up] is not empty then
32:            if htabBuckets[up].size() = 1 then
33:                return htabBuckets[up][0]
34:            else
35:                bucketIndex ← random index in size of the bucket - 1
36:                return htabBuckets[up][bucketIndex]
37:            end if
38:        else
39:            up ← up + 1
40:        end if
41:    end for
42:    return invalid tx
43: end function
```

5 seconds block propagation delay (for other parameters, see Table 5.1). Fixed 5 seconds

Table 5.1: Table of parameters used in simulations with different random selection algorithm.

Parameter	Value
Malacious miner strategy	rational
Honest miner strategy	random
Malicious miners count	2
Honest miners count	7590
Malicious miner mining power	20%
Block creation time (λ)	20 seconds
Blocks	1000
Block propagation latency	5 seconds
Block size	100 transactions
Mempool capacity	10000 transactions
Transaction generation speed	60 to 160 seconds
Generated transactions	200 to 450

delay is a lot higher than the delay of a proposed Bitcoin-like network in Section 5.3. This is because we want to achieve a higher transaction collision rate on which we are able to compare our algorithms better. Transaction collision rate also depends on other parameters such as block size, mempool capacity, transaction generation size, transaction generation speed, block size, and lambda. Results from this comparison can be seen in Table 5.2.

Table 5.2: Transaction collisions comparison of different random selection algorithms. Transaction throughput for these algorithms is similar. However, the number of transaction collisions varies, resulting in different miners' profits. Simulation with these parameters had higher transaction generation that could be processed in total. The random selection algorithm also affects the number of total processed transactions.

	Random access	Multi-index	Begin	Begin equal key
Transaction throughput	67.1%	67.54%	67.44%	68.1%
Unique tx occurance	23.4%	27.45%	27.34%	15.64%
2 times occurred tx	24.64%	25.76%	25.9%	18.53%
3 times occurred tx	6.19%	5.86%	5.86%	8.14%
4 times occurred tx	1.68%	0.74%	0.75%	3.69%
5 times occurred tx	0.33%	0.08%	0.06%	1.22%
5+ times occurred tx	0.06%	0.013%	0.002%	0.34%
Total processed tx	56.3%	59.91%	59.9%	47.54%

Results show that multi-index implementation uses a similar approach to begin. The results vary very slightly. This approach was also used in [46] because it uses multi-index implementation. Because the used network topology was ten miners connected in a circle,

it was not critical to have a random algorithm as we proposed (see Algorithm 1). However, on Bitcoin-like network topology, it has a higher impact. Besides that, as expected, begin-equal-key has a much higher duplication rate. The performance of Random-access is not better than multi-index or begin but produces more accurate results. We will further use a random-access algorithm for random selection in the hashtable.

Performance comparison

In Fig. 5.7, we compared the performance of multi-index, red-black tree, and hashtable. Multi-index performance is the worst because it is not optimized to our requirements. Hashtable performed slightly better than the red-black tree in these simulations, but it may change as hashtable requires sorting while accessing a transaction with the highest fee or removing with the lowest fee (see Fig. 5.8). The average time complexity of this operation is $O(n * log(n))$. In contrast, the red-black tree has transactions sorted implicitly, and the average time complexity to access or remove sorted transactions is $O(log(n))$.

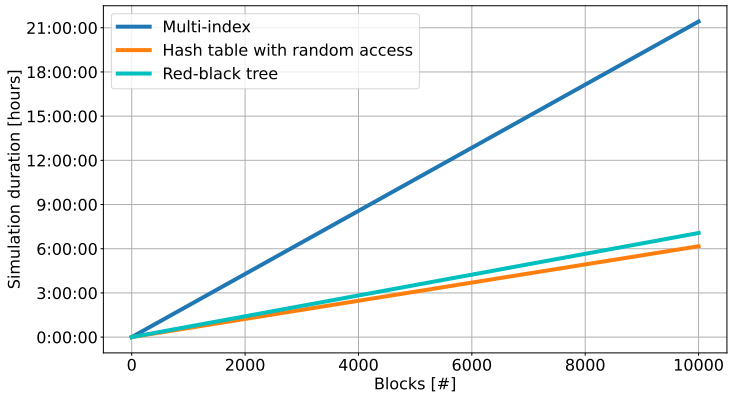

Figure 5.7: Simulation speed comparison of different mempool data structures. Simulations were performed on a new Bitcoin-like network topology.

By analysis of hashtable and red-black data structures, we have made their amortized time complexities comparison for different types of accesses, removal, and insertion shown in Table 5.3.

Combined data structure of hashtable and the red-black tree

Because both hashtable and the red-black tree have their advantages and disadvantages, we decided to create a unique data structure that is created by their combination (see Fig. 5.9). By choosing a combination of these data structures, we benefit from both of them. For direct and random access is used hashtable and for sorted access and removal is used red-black tree. This combined structure has a higher memory usage (10.6 GB compared to 6.5 GB in hashtable in a network of 7592 nodes, each with a full mempool of 10000 transactions). We have made a comparison of combined structure with hashtable when simulating different mempool capacities (see Fig. 5.10). We choose the combined structure as it outperforms other structures in speed.

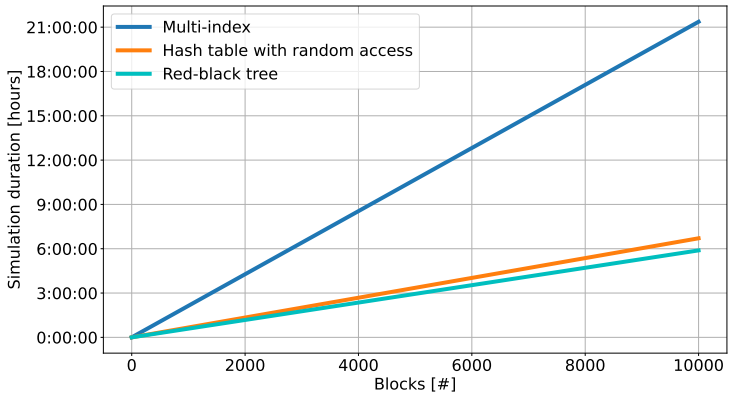

Figure 5.8: Simulation speed comparison of different mempool data structures while simulating only honest nodes and random transactions removal from mempool. Simulations were performed on a new Bitcoin-like network topology.

Table 5.3: Comparison of amortized time complexities of hashtable and red-black tree data structures and their combination for different mempool data access methods. Random access for hashtable and combination depends on n elements currently stored in the hashtable and on its capacity m.

Access	Hashtable	Red-black tree	Combination
Direct	$O(1)$	$O(log(n))$	$O(1)$
Random	$O(\frac{m}{n})$	$O(n)$	$O(\frac{m}{n})$
Sorted	$O(n * log(n))$	$O(log(n))$	$O(log(n))$

5.5 Progress tracker

We propose a progress tracker to track percentage progress. Because simulation can take a long time to complete, we introduced an estimated time calculation to measure simulation speed. This time is calculated from the time it takes to compute one single step. Besides that, it also outputs the percentage fullness of mempool of the first honest and malicious node (if any one of them has at least one representative in simulation). Version 2 was also missing a proper simulation ending. The number of transactions must have been calculated and hardcoded before running a simulation. The simulation also generated transactions even if a specified number of blocks were already created and propagated to other miners. This issue was fixed in a progress tracker.

5.6 Automation

This section describes a change from a real-time data process to a post-process. It also includes two scripts to post-process simulation output data. Further, we discuss the pos-

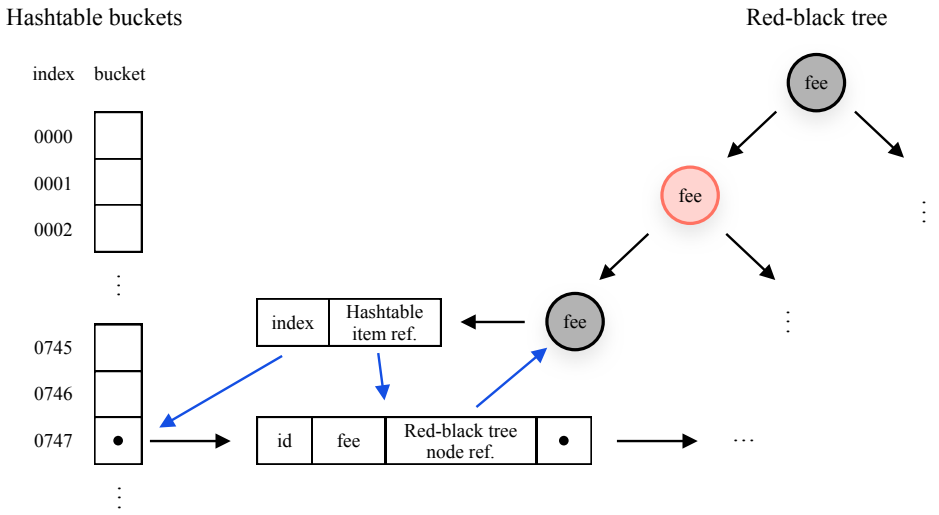

Figure 5.9: Showcase of different access methods to mempool. Direct access returns a transaction specified by id. An id is our simplified unique identifier of the transaction. Random access returns a transaction randomly selected from mempool. Sorted access returns one or more transactions in either ascending or descending order sorted by a fee.

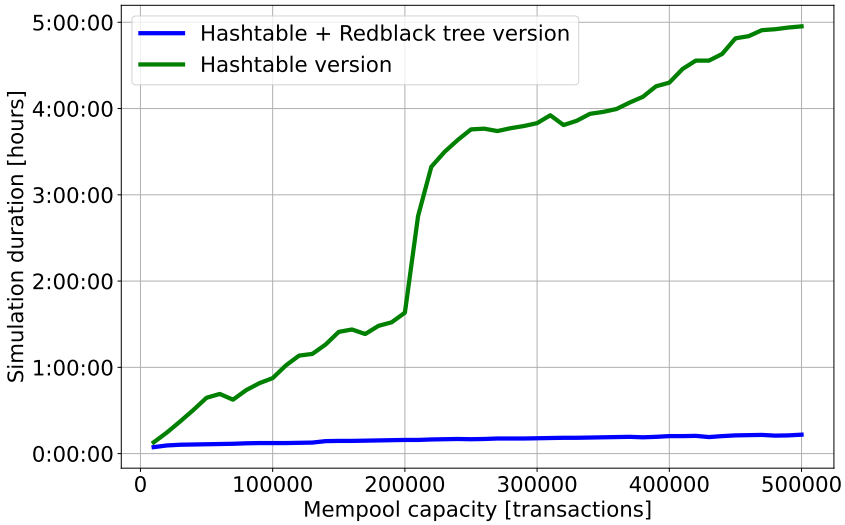

Figure 5.10: Simulation speed comparison with raising mempool capacity. To simulate a mempool of capacity similar to Bitcoin in future experiments, we need to support a few tens and hundreds of thousands of transactions. The performed simulations simulated 1000 blocks.

sibilities of running more simulations in parallel and propose a script for that situations. After that, we propose a script that can edit block propagation latency in created configurations. All proposed scripts in this section are implemented in Python language. See Fig. 5.11 for the external structure of the simulator with mentioned scripts.

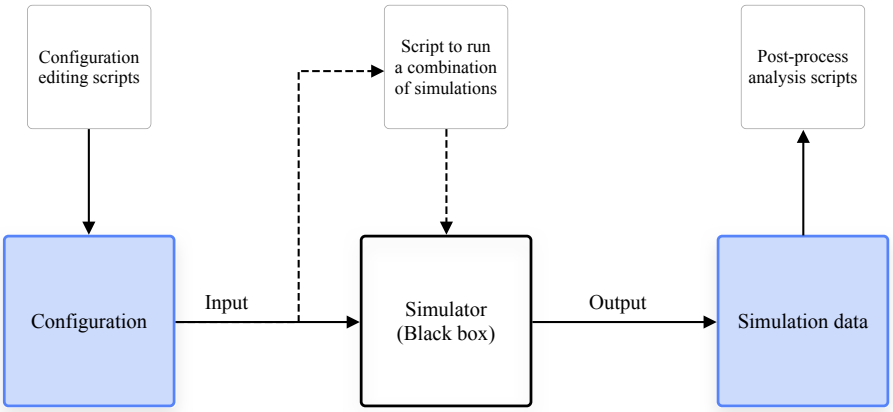

Figure 5.11: The external structure of our extended simulator.

Post-processing

In order to run multiple simulations in parallel, we studied memory usage during the simulation process. The real-time data process causes simulation memory usage to increase linearly in time, as it needs to store data for each block to calculate results when the simulation ends. We propose a post-process method to gather all data during simulation and process results separately after the simulation ends. This approach allows us to analyze gathered data multiple times (e.g., apply different payoff functions), and we do not lose any data if the simulation crashes or does not finish. In addition, memory usage is not increasing during the simulation process, as shown in Fig. 5.12. The simulator with the post-process method generates the following files:

- *Progress.* The content of this file is also mirrored into standard output (stdout) and starts with simulation information in a human-readable format. It further continues with progress information as described in Section 5.5. Once the simulation ends, it also prints its total duration. If an error where the miner does not have enough transactions to generate a block occurs during simulation, the simulation ends sooner than expected, prints an error message, and takes a snapshot of all miners' mempool content.

- *Data.* This file contains all details about generated blocks. When the block is mined, its information is printed in this file before propagation. Block information contains included transaction id, transaction fee, block id, height (depth), and MinerID who created it. Miner ID is an index loaded from the configuration for each miner at the start of the simulation. The content of this file is stored in CSV format. This file is then loaded by post-processing scripts for collision and profit analysis.

- *Metadata.* This file is required for post-processing scripts to make processing easier for the user. Its content includes similar data that are printed at the start of the simulation in progress, but they are in a strict format similar to CFG. This makes parsing this file easier.

- *Mempool data.* By default, the generation of this file is turned off. Its purpose is to make a snapshot of all miners' mempool when a block is generated, which is helpful in simulation debugging.

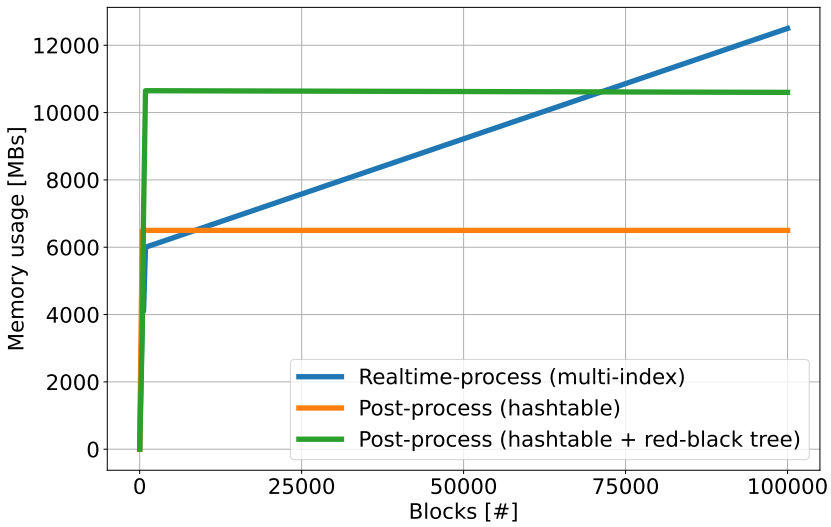

Figure 5.12: Memory optimization result. Comparison of real-time data process and post-process. Realtime-process was simulated with multi-index mempool data structure and post-process with a hashtable and combination of hashtable and red-black tree data structure.

The organization of these files follows a format:

`{type}_{configuration_filename}_{id}.{csv|out|data}`

Where:

- `type` can be either `progress`, `mempool`, `data`, or `metadata`,
- `id` is unique identifier for simulation with same configuration filename.

Collision analysis script

This script aims to analyze simulation output and get the transaction duplication rate. It requires only one argument, a path to a data file generated by simulation. The script automatically parses the name and opens the metadata file. The script also gets a path to a configuration file from the metadata file, storing information about all miners in the network. Thus, this script has all the required information. Output it produces (see Listing 5.1) contains a number of unique transactions and transaction duplication with two or more occurrences, up to five. It also outputs pure duplicated transactions, which states the weighted sum of duplicated transactions and the duplication rate as a relative value. This process can be further automated, and the script allows an optional argument that sets output in CSV format, which is helpful for large-scale post-postprocessing.

Profit analysis script

This script also gets all the required information thanks to the metadata file like the collision analysis script. It is prepared to analyze selected individual profits of honest and malicious

```
Pure duplicated transactions: 68319 (13.66%)
1 tx count (unique): 368314 (73.66%)
2 tx count: 58521 (11.7%)
3 tx count: 4740 (0.95%)
4 tx count: 106 (0.02%)
5 tx count: 0 (0.0%)
More count: 0 (0.0%)
```

Listing 5.1: Example output from collision analysis script. The value of `Pure duplicated transactions` is a complement to transaction throughput.

miners. Besides data argument, it also requires a power threshold argument. This argument stands for minimum mining power for the miner to be included in the analysis. Similar to collision analysis, it also supports the CSV argument that outputs data in CSV format. Other optional arguments include the application of a specific payoff function or printing without extra information such as seed, configuration path, and data path. The output of this script (see Listing 5.2) contains the selected miner's profit and his transaction collision index. See Section 6.3 for details about proposed payoff functions and the transaction collision index.

```
Honest miner #0 profit: 7.64%, txci: 1.1
Honest miner #1 profit: 8.8%, txci: 1.12
Malicious miner #0 profit: 16.44%, txci: 1.38
Malicious miner #1 profit: 18.11%, txci: 1.4
Profit of the remaining miners: 49.01%
```

Listing 5.2: Example output from profit analysis script. The script was started with a mining power threshold argument to output only miners with the most mining power and merge others into one line. The `txci` states the miner's transaction collision index.

Multiple simulation run script

Running an experiment may require running many simulations to cover all different situations. For this reason, it is practical to start multiple simulations at once. Because a single simulation may require much memory to run (depending on input parameters), it is helpful to have the option to limit the number of running simulations. It can be expected that memory will hit the limit before the usage of CPU cores because the single simulation runs on one CPU core. As a result, we propose a script to run a combination of multiple simulations based on input configurations and seeds, as shown in Fig. 5.11. It can be specified with a number of simulation instances to run on an exact number of CPU cores. The advantage of this script is that when a simulation finishes, it automatically starts a new simulation from the waiting queue on a free CPU core. This approach allows us to run a large number of simulations effectively, and it showed as valuable for computational resources used for this work.

5.7 Payoff function

The payoff function aims to punish malicious miners and thus discourage them from similar behavior. If multiple miners use a rational mining strategy, there is competition for the highest-fee transactions, resulting in transaction collisions. The blocks then have similar content, which more closely resembles a traditional blockchain structure. We aim to propose a payoff function to split the profit from these transactions in order to punish malicious behavior. To discourage miners from malicious behavior, their profit depending on their mining power must be less than the profit of miners acting honestly. Thanks to the post-process approach, we can experiment with different payoff functions on already gathered data. Payoff functions experiments are evaluated in Section 6.3.

5.8 Summary

We have extended an existing version of the simulator and improved its performance. The simulation duration has been extensively decreased. We compared its speed with version 2 with parameters used in experiments in Section 6.3. The simulation process took approximately 4 hours and 41 minutes in our extended version. Simulation with the same parameters and network topology took approximately two days and 8 hours. Although simulation duration depends on input parameters, the difference would be even higher if we modified simulation parameters (e.g., mempool capacity of each node).

We are now able to run a simulation on Bitcoin-like network topology and do further experiments. A comparison with previous versions can be seen in Table 5.4.

Table 5.4: Simulator versions comparison.

	Version 1	Version 2	Version 3
Discrete-event simulation	✓	✓	✓
Supports Bitcoin-like network	partially* and extremely slow	partially* and extremely slow	✓
Data processing	Real-time processing	Real-time processing	Post-processing
Used libraries	Boost, C++ Standard Library	Boost, C++ Standard Library	C++ Standard Library
Block implementation	✓	✓	✓
Transaction implementation	-	✓	✓
Mempool structure implementation	-	Boost multi-index	Combination of hashtable and red-black tree
Supports configuration with behavior defined for each miner	-	-	✓
Progress tracker	-	-	✓
Profit analysis	-	simple	advanced
Transaction collision analysis	-	simple	advanced
Supports payoff functions	-	-	✓

* Version 1 and 2 can be started with the Bitcoin-like configuration created for version 3 after removing honest and malicious behavior specifications because they were not supported in the configuration file. Additionally, it must specify block propagation latency for each bi-directional connection. This is optional for version 3 as it will be automatically filled with the value from Bitcoin-like propagation latency (see Section 5.3).

Chapter 6

Evaluation

This chapter includes experiments with the extended version of the simulator. The first experiment verifies that a single malicious node's profit is higher than an honest node in a Bitcoin-like network. The second experiment continues the investigation by increasing malicious nodes count while involving honest nodes with the same mining power as direct competitors. Payoff function experiments focus on experimenting with different payoff functions. In all experiments, we simulate only transaction fee rewards. Newly created blocks do not involve any reward. This approach verifies a blockchain that has already distributed all its supply.

6.1 Experiment with single malicious miner

This experiment investigates whether a single malicious miner has an advantage in making profits by using the rational transaction selection strategy compared to honest miners who follow the random transaction selection strategy in a created Bitcoin-like network of 7592 nodes. Parameters for this experiment are displayed in Table 6.1.

Table 6.1: Table of parameters used in experiment with single malicious miner.

Parameter	Value
Malacious miner strategy	rational
Honest miner strategy	random
Malicious miners count	1
Honest miners count	7591
Malicious miner mining power	0 to 40%
Block creation time (λ)	20 seconds
Blocks	20000
Block propagation latency	5 sec / Bitcoin-like
Block size	20 transactions
Mempool capacity	10144 transactions
Transaction generation speed	60 to 180 seconds
Generated transactions	10 to 240

We simulated a single malicious miner while continuously increasing his mining power relative to the network. His connections to the peers in a network of this size could affect

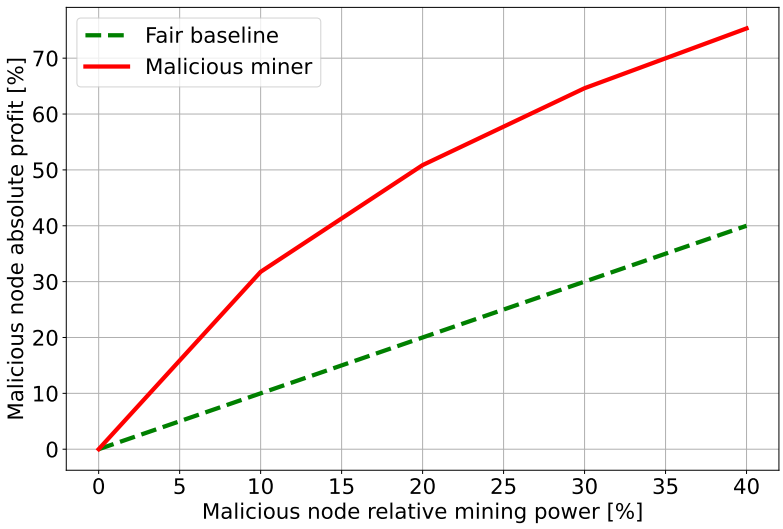

Figure 6.1: Profit relative to mining power of a single malicious miner in Bitcoin-like network with 7592 nodes.

his profits (i.e., connection to well-connected nodes in contrast to poorly connected nodes). Therefore, our experiment involves different placements of the miner in the network, but it has almost no change. His results are described as an average from multiple positions. We simulated the same network topology with Bitcoin-like block propagation latency (see Fig. 5.4) but also with a fixed constant of 5[1] seconds. Results were the same, and latency did not affect the final profit of the single malicious miner. Fig. 6.1 shows the fair baseline, representing profits relative to the miner's mining power. Each miner should be rewarded for his work according to his amount of resources to keep fairness. However, the results show that malicious miner has significantly more profits than other miners.

6.2 Experiment with multiple malicious miners

This experiment focuses on verifying the profits of multiple malicious miners in the network. We simulated a Bitcoin-like network with 7592 nodes while focusing on four of them. Each of these four nodes had 10% mining power relative to the network. We started with four malicious nodes while continuously decreasing their count to zero and increasing the number of honest nodes. The rest of the network consisted of honest nodes. Parameters for this experiment are displayed in Table 6.2. Fig. 6.2 shows that by increasing the number of malicious nodes, their profit decreases as there is greater competition. This also negatively affects the profit of all honest nodes and decreases transaction throughput, as shown in Fig. 6.3.

[1] 5 seconds were selected according to the experiments performed in [46].

Table 6.2: Table of parameters used in experiment with multiple malicious miners.

Parameter	Value
Malacious miner strategy	rational
Honest miner strategy	random
Malicious miners count	4 to 0
Honest miners count	7588 to 7592
Malicious miner mining power	10%
Block creation time (λ)	10 seconds
Blocks	1000
Block propagation latency	Bitcoin-like
Block size	500 transactions
Mempool capacity	10000 transactions
Transaction generation speed	5 to 30 seconds
Generated transactions	500 to 2000

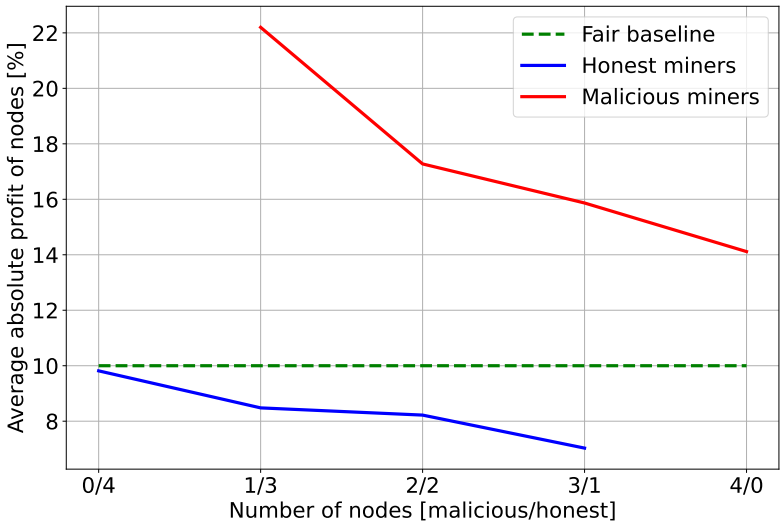

Figure 6.2: Profit earned by honest and malicious nodes from a selection of four with the most mining power. Their profit is averaged for the displayed number of nodes.

6.3 Payoff function experiments

We confirmed the impact of malicious nodes in a Bitcoin-like network. They make more profit compared to honest nodes and reduce transaction throughput. Experiments in this section use the results from an experiment with multiple malicious miners (see Section 6.2) and aim to propose a payoff function to discourage malicious nodes from their behavior.

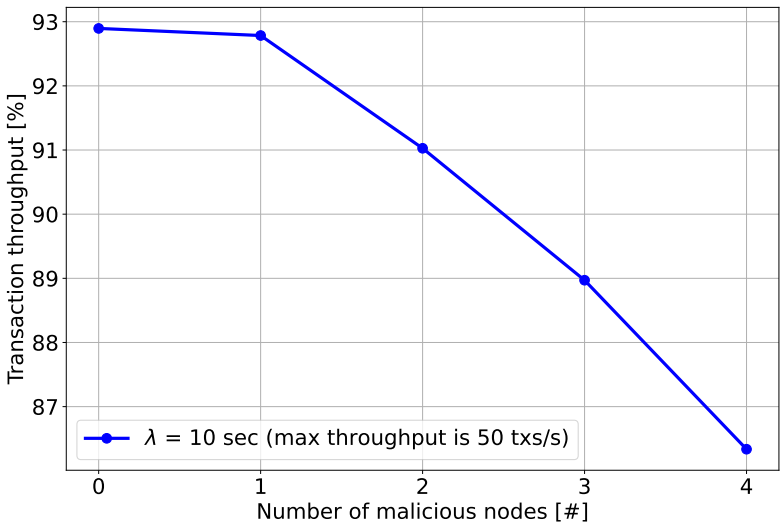

Figure 6.3: Transaction throughput is reduced by increasing the number of malicious nodes. The maximum possible throughput with parameters used in this experiment was 50 transactions per second.

Evenly split

Initially, we experimented with a payoff function that evenly splits profit from colliding transactions to all miners who include one of these transactions in a block. In order to prevent miners from including old transactions, the payoff function could only be applied to transactions whose first occurrence has not reached finality.

Let n be the number of single transaction occurrences in different blocks. A single miner's profit from this transaction after applying an evenly split payoff function is then defined as follows:

$$\mathbb{E} = \frac{1}{n} \tag{6.1}$$

However, the results of experimenting with this payoff function are that it has no impact on the miner's profit.

Halving split and burn

The halving split payoff function is inspired by Bitcoin block reward halving[2]. It aims to reduce duplicate transaction fee rewards as more miners include it in blocks while prioritizing miners who mined it first.

Let f be the fee reward from the single transaction, n the number of the transaction occurrences in different blocks, and i the mined block order, compared to others that also include this transaction. A single miner's profit from this transaction after applying a halving split payoff function is then defined as follows:

[2] https://en.bitcoin.it/wiki/Controlled_supply

$$\mathbb{H} = \frac{f}{2^i} + \frac{f}{n2^n} \tag{6.2}$$

After experimenting with this payoff function, we have added a modification because the standalone halving payoff function only tends to redistribute the profits a little but with no visible change. This modification involves burning part of a fee reward, which also supports the economy of cryptocurrency. Because malicious miners produce mostly colliding transactions instead of unique ones, we can limit their profit by only accepting the first occurrence (i.e., $i = 1$) and discarding (burning) the others after applying the halving split payoff function.

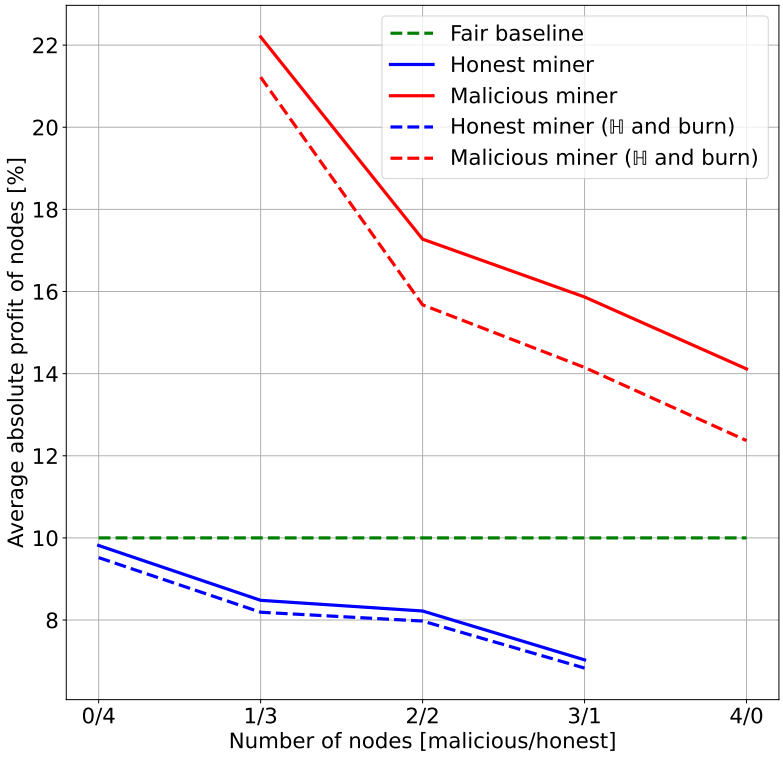

Figure 6.4: Comparison of earned profit by honest and malicious nodes from a selection of four with the most mining power. The full line displays the profit with no payoff function (i.e., the first miner who included it in a block gets a reward), and the dashed line display application of halving split and fee burn. Their profit is averaged for the displayed number of nodes.

Results show that this modification reduces the profits of malicious miners (see Fig. 6.4) but also involves fee reward burning of honest miners. This is problematic because honest miners are penalized for acting honestly. With the increasing number of malicious miners, their burned profit increases because they produce more transaction collisions. The fee reward from the first occurrence still depends on the total number of occurrences.

However, it still does not solve the main problem — malicious miners earn more profit than honest miners.

Transaction collision index

As a different approach, we propose a way to separate honest and malicious nodes. We created a *transaction collision index* (γ), which can be calculated from the number of colliding transactions included in mined blocks. If a transaction has been included in more blocks, its collision will have a greater weight. This approach is shown in Fig. 6.5.

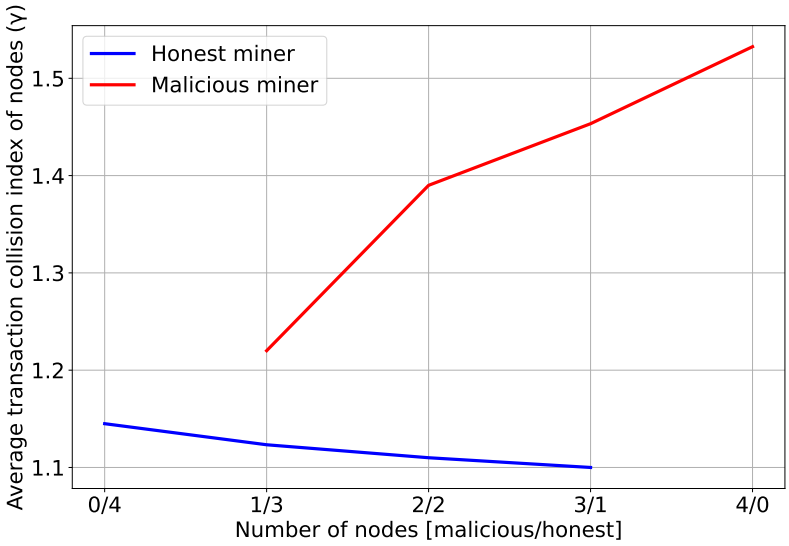

Figure 6.5: Comparison of different transaction collision index values for malicious and honest nodes.

Profit redistribution

Because we can now identify malicious behavior, we created a different payoff function that redistributes 60% of fee rewards earned by malicious nodes to honest nodes based on their mining power. Value 60% can be further modified. This approach can be implemented in the consensus protocol. Just as a change in the difficulty in Bitcoin occurs each 2016th[3] block, similarly could be implemented this payoff function. The results in Fig. 6.6 show that the rational transaction selection strategy used by malicious nodes is less profitable than the random transaction selection strategy after using a payoff function. Thus, miners are discouraged from behaving maliciously.

[3] https://en.bitcoin.it/wiki/Difficulty

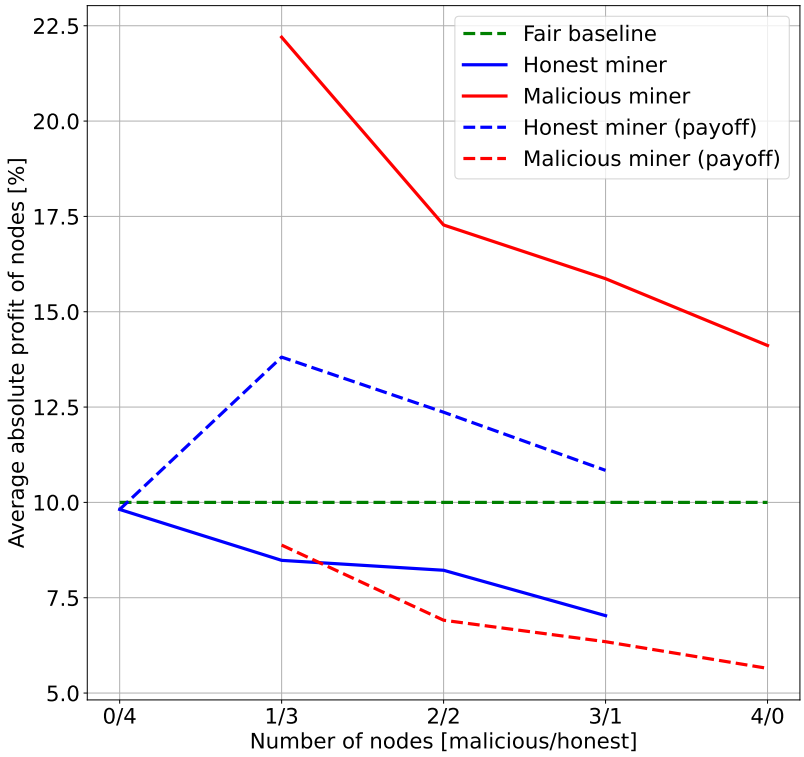

Figure 6.6: Comparison of earned profit by honest and malicious nodes from a selection of four with the most mining power. The full line displays the profit with no payoff function, and the dashed line displays the application of profit redistribution after miners' behavior recognition with the transaction collision index.

Chapter 7

Conclusion

We studied several DAG-based consensus protocols and further focused on PHANTOM and GHOSTDAG. In order to verify their properties, we extended a simulator with capabilities to simulate the network topology of the size of a Bitcoin-like network. This includes tools that can generate and modify configuration files used by a simulator with peer-to-peer network connections and block propagation latency to individual peers. In order to simulate a network of this size, we also optimized an existing simulator. We analyzed and compared different data structures used for a mempool. We chose a combination of hashtable and red-black tree and verified its speed-up. Furthermore, we updated a simulator architecture, changed its data process from real-time to post-process, and created tools to process these data. We also created a tool that can run a combination of multiple simulations based on input configuration, which has also proven valuable for large-scale experimentation.

We also confirmed the original question from [46] by experimenting on a Bitcoin-like network with 7592 nodes. We proved by simulations that the rational transaction selection strategy generates more profit than the random selection strategy (proposed in PHANTOM and GHOSTDAG) with a single and multiple malicious actors. This increase in profit is at the expense of a decrease in profit by honest actors. An increasing number of malicious miners causes a decrease in their profit as there is higher competition. However, the earned profit relative to mining power is still higher than honest miners. We also showed that on a network of this size, their behavior causes a reduction in transaction throughput.

To discourage nodes from malicious behavior, we experimented with payoff functions. These experiments demonstrated that the first two payoff functions we proposed did not impact malicious miners' profit. For this reason, we introduced a payoff function modification of burning fees which shows up that it reduces malicious miners' profit. However, this was not sufficient. Therefore, we came up with a transaction collision index (γ) to recognize miners' behavior. With this informations, we were able to apply a different payoff function which redistributed 60% of the profit earned by malicious miners to honest miners based on their mining power. This approach resulted in honest miners' profits being higher than malicious miners' profits according to their mining power. Consequently, miners no longer have the incentive to have malicious behavior after applying this payoff function because their profits would be lower than acting honestly.

The simulator can be further extended with a transaction propagation event. There is also potential to verify how the simulator can be further optimized by running a single simulation on multiple CPU cores instead of one.

This work was presented at the Excel@FIT event at the Brno University of Technology, Faculty of Information Technology, and was awarded by an expert committee, the SAP company, and the public.

Bibliography

[1] *Bitcoin core* [online]. [cit. 2022-03-17]. Available at: https://en.bitcoin.it/wiki/Bitcoin_Core.

[2] *Bitcoin core source code* [online]. [cit. 2022-03-17]. Available at: https://github.com/bitcoin/bitcoin.

[3] *Boost C++ Libraries* [online]. [cit. 2022-03-17]. Available at: https://www.boost.org/.

[4] *What Is Bitcoin Core?* [online]. River Financial Inc. [cit. 2022-03-17]. Available at: https://river.com/learn/what-is-bitcoin-core/.

[5] Alharby, M. and Moorsel, A. van. BlockSim: A Simulation Framework for Blockchain Systems. *SIGMETRICS Perform. Eval. Rev.* New York, NY, USA: Association for Computing Machinery. jan 2019, vol. 46, no. 3, p. 135–138. DOI: 10.1145/3308897.3308956. ISSN 0163-5999. Available at: https://doi.org/10.1145/3308897.3308956.

[6] Alharby, M. and Moorsel, A. van. *BlockSim Simultor* [online]. April 2019 [cit. 2022-03-17]. Available at: https://github.com/maher243/BlockSim.

[7] Alsahan, L., Lasla, N. and Abdallah, M. *Core-Bitcoin Network Simulator for Performance Evaluation using Lightweight Virtualization* [online]. Oct 2019 [cit. 2022-03-17]. Available at: https://github.com/noureddinel/core-bitcoin-net-simulator.

[8] Alsahan, L., Lasla, N. and Abdallah, M. M. Local Bitcoin Network Simulator for Performance Evaluation using Lightweight Virtualization. *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT).* 2020, p. 355–360.

[9] Andresen, G. *Bitcoin mining simulator simulator* [online]. May 2015 [cit. 2022-03-17]. Available at: https://github.com/gavinandresen/bitcoin_miningsim.

[10] Aoki, Y., Otsuki, K., Kaneko, T., Banno, R. and Shudo, K. Simblock: A blockchain network simulator. In: IEEE. *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* 2019, p. 325–329.

[11] Bowe, S. *Javascript P2P Network Simulator* [online]. Mar 2014 [cit. 2022-03-17]. Available at: https://github.com/ebfull/simbit.

[12] Brune, R. *Bitcoin Network Simulator* [online]. Nov 2013 [cit. 2022-03-17]. Available at: https://github.com/rbrune/btcsim.

[13] Bruschi, F., Rana, V., Gentile, L. and Sciuto, D. Mine with It or Sell It: The Superhashing Power Dilemma. *SIGMETRICS Perform. Eval. Rev.* New York, NY, USA: Association for Computing Machinery. jan 2019, vol. 46, no. 3, p. 127–130. DOI: 10.1145/3308897.3308954. ISSN 0163-5999. Available at: https://doi.org/10.1145/3308897.3308954.

[14] Castro, M., Liskov, B. et al. Practical byzantine fault tolerance. In: *OsDI*. 1999, vol. 99, no. 1999, p. 173–186.

[15] Chin, Z. H. *SimBlock (extension)* [online]. Feb 2020 [cit. 2022-03-17]. Available at: https://github.com/Z-Hau/SimBlock-with-Difficulty-Adjustment.

[16] Chin, Z. H., Yap, T. T. V. and Tan, I. K. T. Simulating Difficulty Adjustment in Blockchain with SimBlock. In: *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*. New York, NY, USA: Association for Computing Machinery, 2020, p. 192–197. BSCI '20. DOI: 10.1145/3384943.3409437. ISBN 9781450376105. Available at: https://doi.org/10.1145/3384943.3409437.

[17] Churyumov, A. Byteball: A decentralized system for storage and transfer of value. [online]. 2016. Available at: https://byteball.org/Byteball.pdf.

[18] Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A. et al. On Scaling Decentralized Blockchains (A Position Paper). In:. February 2016.

[19] Crosby, M., Pattanayak, P., Verma, S., Kalyanaraman, V. et al. Blockchain technology: Beyond bitcoin. *Applied Innovation*. 2016, vol. 2, 6-10, p. 71.

[20] Deshpande, A. and Nasirifard, P. *EVIBES – an Ethereum Blockchain Simulator* [online]. Aug 2018 [cit. 2022-03-17]. Available at: https://github.com/i13-msrg/evibes.

[21] Deshpande, A., Nasirifard, P. and Jacobsen, H.-A. EVIBES: Configurable and Interactive Ethereum Blockchain Simulation Framework. In: *Proceedings of the 19th International Middleware Conference (Posters)*. New York, NY, USA: Association for Computing Machinery, 2018, p. 11–12. Middleware '18. DOI: 10.1145/3284014.3284020. ISBN 9781450361095. Available at: https://doi.org/10.1145/3284014.3284020.

[22] DSN Research Group, K. a. K. *Bitcoin Network Monitor* [online]. 2015. Available at: https://www.dsn.kastel.kit.edu/bitcoin/index.html.

[23] Eyal, I. and Sirer, E. G. Majority is Not Enough: Bitcoin Mining is Vulnerable. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. jun 2018, vol. 61, no. 7, p. 95–102. DOI: 10.1145/3212998. ISSN 0001-0782. Available at: https://doi.org/10.1145/3212998.

[24] Fan, S., Ghaemi, S., Khazaei, H. and Musilek, P. Performance Evaluation of Blockchain Systems: A Systematic Survey. *IEEE Access*. june 2020, PP, p. 1–1. DOI: 10.1109/ACCESS.2020.3006078.

[25] Faria, C. *BlockSim: Blockchain Simulator* [online]. Mar 2018 [cit. 2022-03-17]. Available at: https://github.com/carlosfaria94/blocksim.

[26] Faria, C. and Correia, M. P. BlockSim: Blockchain Simulator. *2019 IEEE International Conference on Blockchain (Blockchain)*. 2019, p. 439–446.

[27] Fattahi, M. and Fard, A. M. *SIMBA: An Efficient Simulator for Blockchain Applications* [online]. Sep 2019 [cit. 2022-03-17]. Available at: https://github.com/nyit-vancouver/SIMBA.

[28] Fattahi, S. M., Makanju, A. and Fard, A. M. SIMBA: An Efficient Simulator for Blockchain Applications. *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. 2020, p. 51–52.

[29] Gentile, L. *Incentive-network-simulator* [online]. Sep 2018 [cit. 2022-03-17]. Available at: https://github.com/lorenzogentile404/incentive-network-simulator.

[30] Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H. et al. On the Security and Performance of Proof of Work Blockchains. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2016, p. 3–16. CCS '16. DOI: 10.1145/2976749.2978341. ISBN 9781450341394. Available at: https://doi.org/10.1145/2976749.2978341.

[31] Ghimire, S. Analysis of Bitcoin Cryptocurrency and Its Mining Techniques. *UNLV Theses, Dissertations, Professional Papers, and Capstones*. 2019. Available at: http://dx.doi.org/10.34917/15778438.

[32] Glykantzis, V. and Gervais, A. *Bitcoin-Simulator, capable of simulating any re-parametrization of Bitcoin* [online]. April 2016 [cit. 2022-03-17]. Available at: https://github.com/arthurgervais/Bitcoin-Simulator.

[33] Hazari, S. and Mahmoud, Q. Improving Transaction Speed and Scalability of Blockchain Systems via Parallel Proof of Work. *Future Internet*. july 2020, vol. 12, p. 125. DOI: 10.3390/fi12080125.

[34] Homoliak, I., Venugopalan, S., Hum, Q., Reijsbergen, D., Schumi, R. et al. *The Security Reference Architecture for Blockchains: Towards a Standardized Model for Studying Vulnerabilities, Threats, and Defenses*. October 2019.

[35] Jansen, R. et al. *The Shadow Simulator* [online]. May 2011 [cit. 2022-03-17]. Available at: https://github.com/shadow/shadow.

[36] Król, M. *Proof of Prestige* [online]. Jun 2018 [cit. 2022-03-17]. Available at: https://gitlab.com/mharnen/pop.

[37] Król, M., Sonnino, A., Al Bassam, M., Tasiopoulos, A. G., Rivière, E. et al. Proof-of-prestige: A useful work reward system for unverifiable tasks. *ACM Transactions on Internet Technology (TOIT)*. ACM New York, NY. 2021, vol. 21, no. 2, p. 1–27.

[38] Lantz, L. and Cawrey, D. *Mastering Blockchain*. O'Reilly, 2021. ISBN 9781492054702.

[39] LeMahieu, C. *Nano: A feeless distributed cryptocurrency network* [online]. 2018. Available at: https://nano.org/en/whitepaper.

[40] Matsumoto, M. and Nishimura, T. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Trans. Model. Comput. Simul.* New York, NY, USA: Association for Computing Machinery. jan 1998, vol. 8, no. 1, p. 3–30. DOI: 10.1145/272991.272995. ISSN 1049-3301. Available at: https://doi.org/10.1145/272991.272995.

[41] Miller, A. and Jansen, R. *Shadow-plugin-bitcoin* [online]. Dec 2014 [cit. 2022-03-17]. Available at: https://github.com/shadow/shadow-plugin-bitcoin.

[42] Mišić, J., Mišić, V. B., Chang, X., Motlagh, S. G. and Ali, M. Z. Modeling of Bitcoin's Blockchain Delivery Network. *IEEE Transactions on Network Science and Engineering.* 2020, vol. 7, no. 3, p. 1368–1381. DOI: 10.1109/TNSE.2019.2928716.

[43] Nakamoto, S. *Bitcoin: A peer-to-peer electronic cash system* [online]. 2009. Available at: http://www.bitcoin.org/bitcoin.pdf.

[44] Nasirifard, P., Schüssler, F. and Stoykov, L. *VIBES: Fast Blockchain Simulations for Large-scale Peer-to-Peer Networks* [online]. Mar 2018 [cit. 2022-03-17]. Available at: https://github.com/i13-msrg/vibes.

[45] Paulavičius, R., Grigaitis, S. and Filatovas, E. A Systematic Review and Empirical Analysis of Blockchain Simulators. *IEEE Access.* 2021, vol. 9, p. 38010–38028. DOI: 10.1109/ACCESS.2021.3063324.

[46] Perešíni, M., Benčić, M. F., Malinka, K. and Homoliak, I. DAG-Oriented Protocols PHANTOM and GHOSTDAG under Incentive Attack via Transaction Selection Strategy. In:. Institute of Electrical and Electronics Engineers, 2021, p. 1–8. Available at: https://www.fit.vut.cz/research/publication/12563.

[47] Popov, S. The tangle. *White paper.* 2018, vol. 1, no. 3, [cit. 2022-03-06].

[48] Popov, S. and Buchanan, W. J. FPC-BI: Fast Probabilistic Consensus within Byzantine Infrastructures. *Journal of Parallel and Distributed Computing.* Jan 2021, vol. 147, p. 77–86, [cit. 2022-03-06]. DOI: 10.1016/j.jpdc.2020.09.002. ISSN 0743-7315. Available at: http://dx.doi.org/10.1016/j.jpdc.2020.09.002.

[49] Rawat, D. B., Chaudhary, V. and Doku, R. Blockchain Technology: Emerging Applications and Use Cases for Secure and Trustworthy Smart Systems. *Journal of Cybersecurity and Privacy.* november 2020, vol. 1, p. 4–18. DOI: 10.3390/jcp1010002.

[50] Sai, A. R. *Bitcoin-Simulator (Bitcoin-Simulator extension)* [online]. Oct 2018 [cit. 2022-03-17]. Available at: https://github.com/ashishrsai/BTC.

[51] Sai, A. R., Buckley, J. and Le Gear, A. Assessing the security implication of Bitcoin exchange rates. *Computers & Security.* 2019, vol. 86, p. 206–222. DOI: https://doi.org/10.1016/j.cose.2019.06.007. ISSN 0167-4048. Available at: https://www.sciencedirect.com/science/article/pii/S0167404818312112.

[52] Sompolinsky, Y., Lewenberg, Y. and Zohar, A. *SPECTRE: A Fast and Scalable Cryptocurrency Protocol* [Cryptology ePrint Archive, Report 2016/1159]. 2016. https://ia.cr/2016/1159.

[53] Sompolinsky, Y., Wyborski, S. and Zohar, A. PHANTOM GHOSTDAG: A Scalable Generalization of Nakamoto Consensus: September 2, 2021. In:. New York, NY, USA: Association for Computing Machinery, 2021, p. 57–70. ISBN 9781450390828. Available at: https://doi.org/10.1145/3479722.3480990.

[54] Stoykov, L., Zhang, K. and Jacobsen, H.-A. VIBES: Fast Blockchain Simulations for Large-Scale Peer-to-Peer Networks: Demo. In:. New York, NY, USA: Association for Computing Machinery, 2017, p. 19–20. Middleware '17. DOI: 10.1145/3155016.3155020. ISBN 9781450352017. Available at: https://doi.org/10.1145/3155016.3155020.

[55] Takayama, S. et al. *SimBlock* [online]. Jun 2019 [cit. 2022-03-17]. Available at: https://github.com/dsg-titech/simblock.

[56] Wang, Q., Yu, J., Chen, S. and Xiang, Y. SoK: Diving into DAG-based Blockchain Systems. *ArXiv preprint arXiv:2012.06128*. 2020, abs/2012.06128. Available at: https://arxiv.org/abs/2012.06128.

[57] Zander, M. *IOTA Tangle Simulation* [online]. Dec 2018 [cit. 2022-03-17]. Available at: https://github.com/IC3RE/DAGsim.

[58] Zander, M., Waite, T. and Harz, D. DAGsim: Simulation of DAG-Based Distributed Ledger Protocols. *SIGMETRICS Perform. Eval. Rev.* New York, NY, USA: Association for Computing Machinery. jan 2019, vol. 46, no. 3, p. 118–121. DOI: 10.1145/3308897.3308951. ISSN 0163-5999. Available at: https://doi.org/10.1145/3308897.3308951.

[59] Zhu, Q., Loke, S. W., Trujillo Rasua, R., Jiang, F. and Xiang, Y. Applications of Distributed Ledger Technologies to the Internet of Things: A Survey. *ACM Comput. Surv.* New York, NY, USA: Association for Computing Machinery. nov 2019, vol. 52, no. 6. DOI: 10.1145/3359982. ISSN 0360-0300. Available at: https://doi.org/10.1145/3359982.

Appendix A

Contents of the included storage media

Files in the root folder are organized as follows:

- `/`
  - `README.md`
  - `bachelor-thesis-xhladk15.pdf`
  - `dag_simulator` ..... simulator implementation
    - `Makefile`
    - `requirements.txt` ..... used Python modules
    - `config-scripts` ..... configuration edit scripts
      - `block_prop_delay_5sec.py` ... edit config to five seconds block propagation delay
      - `block_prop_delay_distr.py` ... edit config to Bitcoin-like block propagation delay
    - `post-process` ..... post-process scripts
      - `collision_analysis.py` ..... collision analysis script
      - `profit_analysis.py` ..... profit analysis script
    - `multiple_sim_run.py` ..... script to start multiple simulations
    - `doc` ..... documentation (needs to be generated)
    - `outputs` ..... store simulation results
    - `configs` ..... configuration files used in payoff functions experiments
    - ...
  - `latex` ..... latex sources code
    - ...
  - `payoff_experiments_results` ..... payoff function experiments data and results
    - ...

Documentation for the simulator can be generated with the command `make doc` in `dag_simulator`, which is then stored in `dag_simulator/doc`.