BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2022
http://excel.fit.vutbr.cz

# Simulator for Verifying the Properties of DAG-based Consensus Protocols

Tomáš Hladký*


**Abstract**
We study existing Directed Acyclic Graph (DAG) blockchain designs that propose to solve a blockchains throughput 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 create or edit input configurations, perform a combination of runs on multiple CPUs 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.

**Keywords:** Simulator — DAG-based consensus — Blockchain — Optimizations — Payoff function — Transaction throughput

**Supplementary Material:** Downloadable Code

*xhladk15@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

## 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 opened new possibilities to a wide range of various fields (e.g., banking and finance, Internet of Things (IoT), real estate, healthcare).

Blockchains suffer from a processing throughput bottleneck. It limits the number of transactions processed per unit of time. 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 [1] 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 [2]. 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, larger 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 ($\lambda$) to an average of 10 minutes. Lowering 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 included in the blockchain because another block at the same height has already been included. Therefore, stale blocks are discarded, which results in wasted power and resources. Stale block occurrence is resolved by consensus.

As a response to the blockchain throughput problem, several protocols have been proposed (e.g., IOTA [3], Obyte (Byteball) [4], SPECTRE [5], PHANTOM and GHOSTDAG [6]) that change blockchain data structure from single chain to Directed Acyclic Graph (DAG), as displayed in Fig. 1. The advantage of this structure over a back-linked list structure is that it can process multiple blocks at the same height and solves the limited throughput problem. As another benefit, it also decreases the occurrence of stale blocks. We focus on the PHANTOM (and its optimization GHOSTDAG) [6] as they propose a different solution than the other DAG-based consensus protocols. They generalize Bitcoin protocol with Proof-of-Work and present a random transaction selection instead of rational transaction selection to reduce the probability that the transaction will be included in more than one block (hereafter transaction collision). However, the authors of these protocols do not analyze how this incentive would work in combination with a rational selection strategy typical for blockchains that include transactions with a fee. In [7] 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 (also called a rational strategy). In general, if a transaction has a higher fee than others, its chance of being 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. The simulations were performed 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 optimizations and automation, as simulating such a network is a complex process. The contributions of our work are as follows:

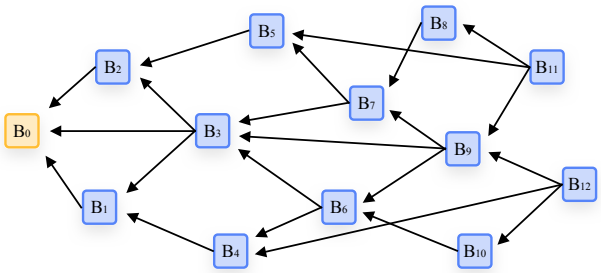


**Figure 1.** A DAG-oriented blockchain

- 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.
- We implement automation tools to create a Bitcoin-like peer-to-peer network with proper latencies between peers, a script to run a combination of simulations on multiple CPU cores, and post-process tools to analyze simulation results.
- 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 create a payoff function to detect and punish malicious behavior.

## 2. Related Work

### 2.1 IOTA

IOTA proposes new features compared to traditional blockchains. 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 into one, 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

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

sent transactions from other users. 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 potentially 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.

In order to secure the IOTA network, the protocol uses a centralized element called "Coordinator". This node ensures correct verification of transactions by other nodes and thus prevents other network attacks, resulting in slower transaction processing. IOTA, with its centralized node, loses unique blockchain feature - decentralization [3].

### 2.2 PHANTOM and GHOSTDAG

Compared to IOTA, this DAG structure includes blocks instead of transactions and involves transaction collision problems. This problem causes processed transactions to be included in multiple blocks. However, they do not contain any centralized element. Both protocols propose a strategy to reduce the transaction collision problem by using random transaction selection on mining. PHANTOM utilizes a recursive k-cluster algorithm to achieve a complete topological order of a DAG structure, which is an NP-hard problem and unsuitable for implementation. The authors created a greedy approximation algorithm called GHOSTDAG to get the same results. Nevertheless, solving the maximum k-cluster SubDAG problem is not necessary for the context of this work, and it is abstracted in our further simulations.[7]

### 2.3 Existing version of simulator

The original version of the simulator is proposed in [7], and it is already an extension to the existing Bitcoin mining simulator[2]. This first version was created by G. Andresen and uses part of the code that is implemented in Bitcoin Core[3]. Specifically, the event scheduler module which is a base for discrete-event simulation. The simulator was used to study network block propagation and stale block rate. In [7] was extended by M. Perešíni with honest and malicious transaction selection strategies and mempool implementation, which is a data structure that stores all miners' transactions that can be included in future blocks.

[2] https://github.com/gavinandresen/bitcoin_miningsim

[3] https://bitcoin.org/en/bitcoin-core/

## 3. Simulator extension proposal

We created a distribution of Bitcoin peer connections from data in [8]. To simulate network delay between peers, we made distribution from Bitcoin live monitoring website [9]. 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:

- *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.

### 3.1 Proposed extensions

This section describes details of proposed extensions for the simulator.

#### 3.1.1 Bitcoin-like network

The simulator was extended with new parameters to run a required Bitcoin-like network simulation, including maximum mempool size, block size, and the option

to define transaction generation size and generation time with uniform distribution. Besides that, we introduce scripts that can create and edit Bitcoin-like network configurations with peer connections (see Fig. 2) and latencies (see Fig. 3). In our further experiments, we will use created Bitcoin-like network of size 7592[4] nodes.

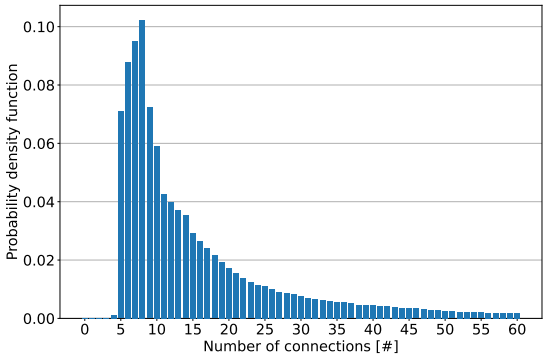


**Figure 2.** Distribution of the number of connections per node created from data published in [8].

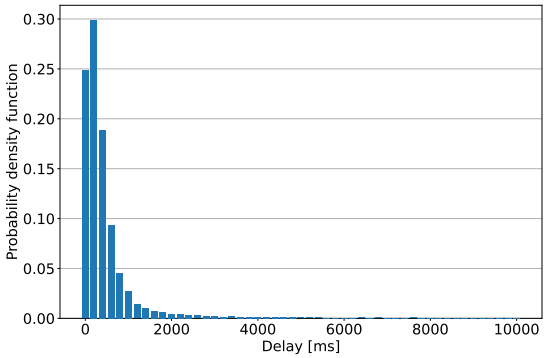


**Figure 3.** Distribution of block propagation network delay. It was created from 24-hour data published in [9] on Feb 28, 2022. Implemented distribution continues up to 30000 ms. The figure only shows a portion up to 10000 ms for better visibility.

### 3.1.2 Progress tracker

We propose a progress tracker to track percentage progress and output the estimated remaining time to measure simulation speed. The previous version was missing a proper simulation ending, which is now implemented in the progress tracker. It also outputs the percentage fullness of mempool of the first honest and malicious node.

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

### 3.1.3 Optimization

The primary purpose of this simulator is to study how miners' transaction selection strategy affects their profits and analyze transaction collision rates. We do not simulate attacks on the network, and all simulated actors follow the consensus algorithm. Thus, we can skip the consensus implementation. Simulation spends most of the time with basic operations with mempool such as access, insert or remove. For this reason, we focus on optimizing the mempool data structure. By examining existing operations of the mempool, we end up with three different access categories to mempool, and we can classify them as shown in Fig. 5. Direct access is required to remove already included transactions from mempool when the mined block gets broadcasted to others miners. Random access is required for honest miners when creating a block. Malicious miners who include the highest fee transactions use sorted access in descending order. We also want to simulate situations when the miner's mempool gets full. In response, malicious miners update their mempool with new transactions by removing transactions with the lowest fee, which is an ascending order of sorted access. Honest miners can either remove transactions randomly or rationally and keep those with a higher fee. However, if honest miners remove transactions randomly, they would have a disadvantage over malicious miners.

**Table 1.** 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 | RB tree | Combination |
|---|---|---|---|
| Direct | $O(1)$ | $O(log(n))$ | $O(1)$ |
| Random | $O(\frac{m}{n})$ | $O(n)$ | $O(\frac{m}{n})$ |
| Sorted | $O(n*log(n))$ | $O(log(n))$ | $O(log(n))$ |

As a result, we need to create a data structure that supports all three access types. The original version implements mempool using multi-index, a unique data structure from the C++ boost library[5]. Multi-index allows the creation of a container with multiple index interfaces, including mentioned three access categories. However, it is a generic data structure and is not well-suited for our purpose. Therefore, we need to test

[5]https://www.boost.org/

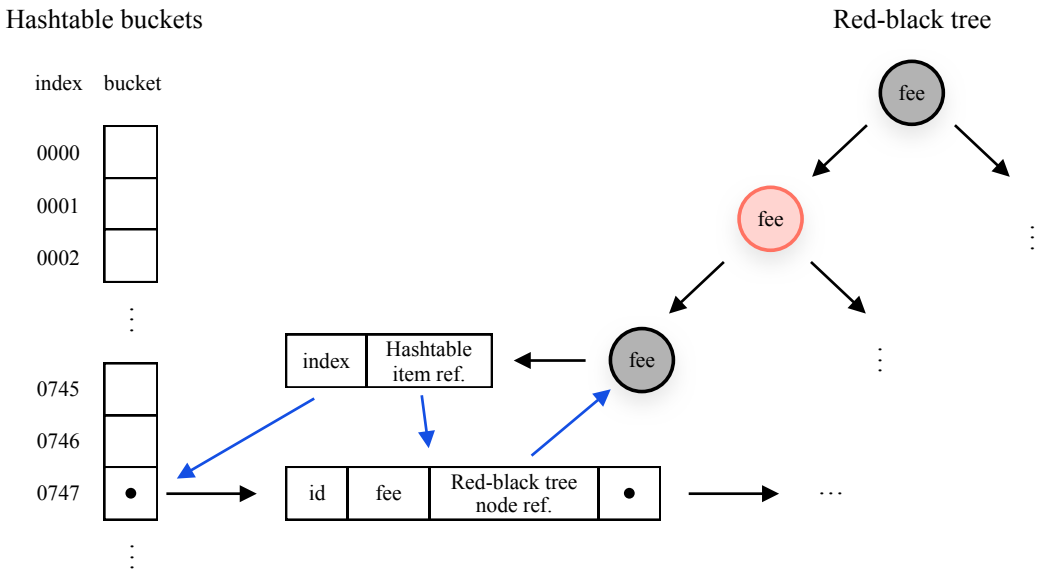


**Figure 4.** Mempool implementation created by combining hashtable and red-black tree data structures. Complete transaction information (i.e., id and fee) are stored in the hashtable with reference to the node in the red-black tree. These transactions are also stored in the red-black tree with a fee as a key. Node content includes an index to a hashtable bucket and a reference to an item of a linked list stored in the bucket.

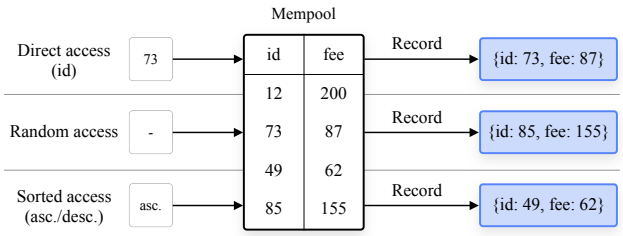


**Figure 5.** Showcase of different access methods to mempool. Direct access returns a transaction specified by id. An id is our simplified unique identifier of the transaction. Random access returns a transaction randomly selected from mempool. Sorted access returns one or more transactions in either ascending or descending order sorted by a fee.

other data structures. We focused on the red-black tree and hashtable and have made their amortized time complexities comparison shown in Table 1. Random access in the red-black tree requires going through all stored elements. It is important to note that we cannot approximate random access because it would skew the results (e.g., randomly choose a path for the red-black tree to access the element). Initially, we analyzed and excluded the red-black tree as this structure is effective only when there is a low number of random accesses. Performance of hashtable starts to decreases when the simulation is set with parameters with higher values as it requires much sorting.

Instead, we chose a combination of these data structures to benefit from both of them (see Fig. 4). 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. 6). We choose the combined structure as it outperforms other structures in speed.

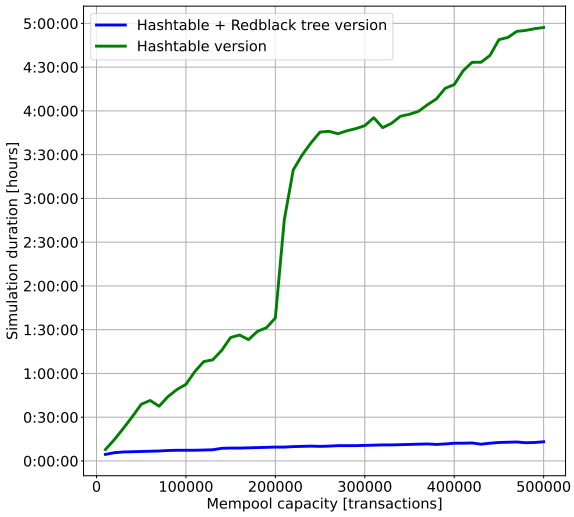


**Figure 6.** Simulation speed comparison with raising mempool capacity. To simulate a mempool of capacity similar to Bitcoin in future experiments, we need to support a few tens and hundreds of thousands of transactions.

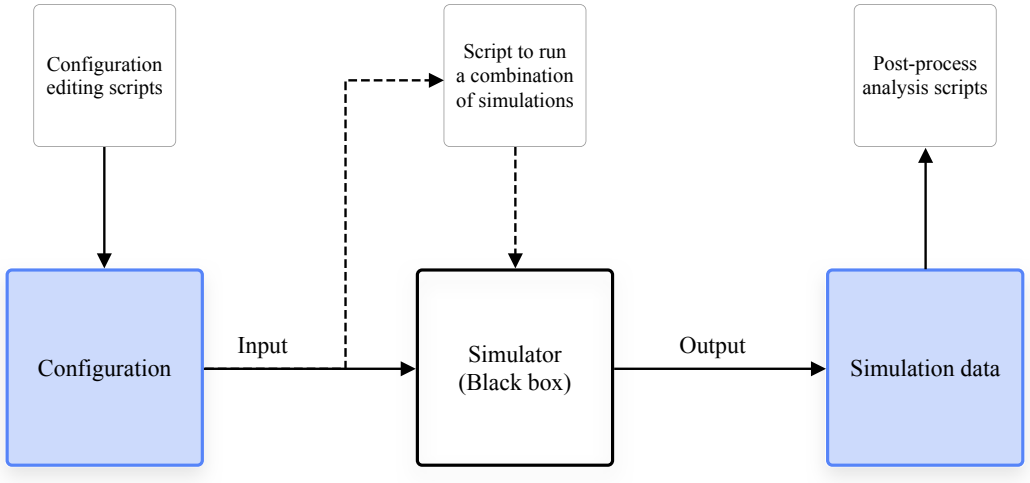


**Figure 7.** The external structure of our extended simulator

#### 3.1.4 Automation

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, 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 crash or does not finish. In addition, memory usage is not increasing during the simulation process. Therefore, we propose a script that runs a combination of multiple simulations based on input configurations and seeds, as shown in Fig. 7. We can specify the number of simulation instances where each instance uses one CPU core. 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 research.

To post-process gathered data, we implemented two scripts: *profit analysis* and *collision analysis*. Both scripts require only one argument, a path to a data file generated by simulation. Scripts automatically parse the name and open the metadata file. The simulator generates metadata, data, progress, and mempool stats files for every simulation. The metadata file contains all details about the simulation, including a path for the used configuration. Thus, post-process scripts have all the required information. This process can be further automated, and both scripts allow an optional argument that sets output in CSV format, which is helpful for large-scale post-processing.

#### 3.1.5 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, which may result in transaction collisions. We want to propose a payoff function to split the profit from these transactions. Thanks to the post-process approach, we can experiment with different payoff functions on already gathered data. Payoff functions experiments are evaluated in section 4.2.

## 4. Experiments on Bitcoin-like network

We performed two experiments. 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 focuses on multiple malicious nodes and proposes a payoff function to punish their behavior and distribute their profit to honest nodes.

### 4.1 Experiment 1

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 7,592 nodes. Parameters for this experiment are displayed in Table 2.

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 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

**Table 2.** Table of parameters used in experiment 1.

| 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 ($\lambda$) | 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 |

**Table 3.** Table of parameters used in experiment 2.

| 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 ($\lambda$) | 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 |

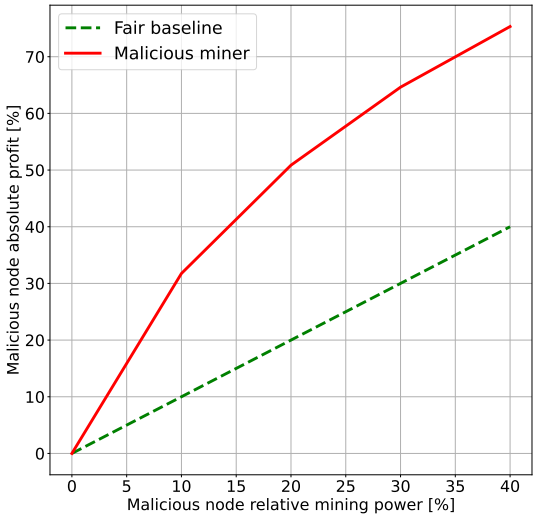


**Figure 8.** Profit relative to mining power of a single malicious miner in Bitcoin-like network with 7592 nodes.

network topology with Bitcoin-like block propagation latency (see Fig. 3) but also with a fixed constant of 5[6] seconds. Results were the same, and latency did not affect the final profit of the single malicious miner. Fig. 8 shows the fair baseline, representing profits relative to the miner's mining power. Each miner should be rewarded for his work according to his number of resources to keep fairness. However, the results show that malicious miner has significantly more profits than other miners.

## 4.2 Experiment 2

This experiment focuses on verifying the profits of multiple malicious miners in the network and applying the payoff function to reduce their profits. 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 3. Fig. 9 shows that by increasing the number of malicious nodes, their profit decreases as there is greater competition. This also negatively affects all honest nodes.

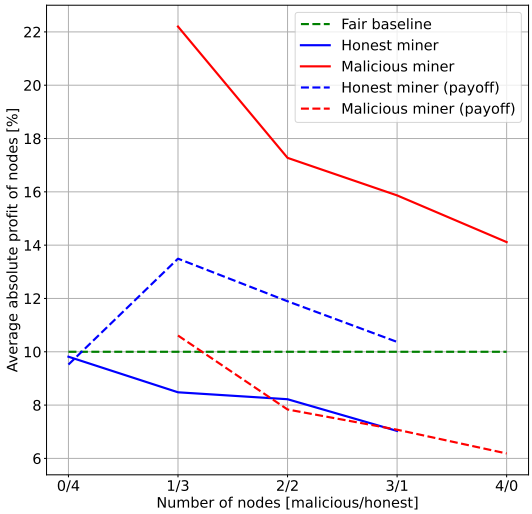


**Figure 9.** Profit earned by either honest or malicious nodes from a selection of four with the most mining power. Their profit is averaged for the displayed number of nodes of two or more.

We propose a payoff function to punish this behavior. For this situation, we created a *transaction collision index* ($\gamma$). This index can be calculated from a number of colliding transactions included in a mined

[6] 5 seconds were selected according to the experiments performed in [7].

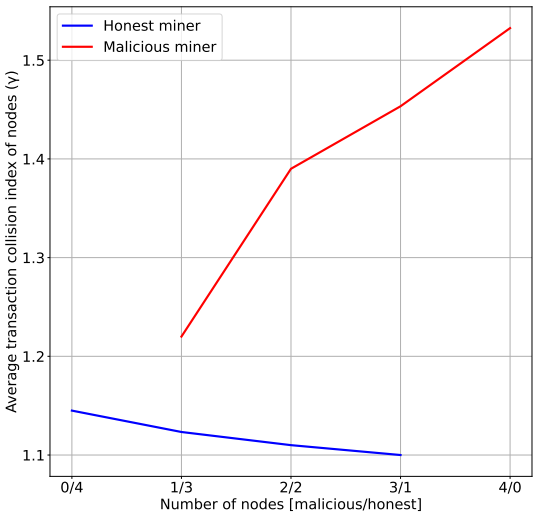


**Figure 10.** Comparison of different transaction collision index values for malicious and honest nodes.

block. If a transaction has been included in more blocks, its collision will have a greater weight. This approach can separate honest and malicious nodes, as shown in Fig. 10. Then, we applied a payoff function which includes distributing 50% of rewards earned by malicious nodes to honest nodes based on their mining power (see Fig. 9). The results show that the rational transaction selection used by malicious nodes is less profitable than random transaction selection after using a payoff function.

## 5. Conclusions

We extended a simulator with capabilities to simulate the network topology of the size of a Bitcoin-like network. We also confirmed the original question from [7] by experimenting on a Bitcoin-like network with more nodes. We proved by simulations that the rational transaction selection strategy generates more profit than the random selection strategy (proposed in PHANTOM and GHOSTDAG [6]) with a single malicious actor. Finally, we created a payoff function to discourage nodes from rational transaction selection.

We want to do further experiments with network latencies similar to the Bitcoin network and simulate more malicious miners in future work. Further, we plan to continue to experiment with payoff functions and study their impact on the profit of malicious actors.

## Acknowledgements

I would like to thank my supervisor Mgr. Kamil Malinka, Ph.D. and my consultant Ing. Martin Perešíni for their invaluable guidance and support. I also acknowledge that this work is part of the ongoing research on 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.

## References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[2] Shihab Hazari and Qusay Mahmoud. Improving transaction speed and scalability of blockchain systems via parallel proof of work. *Future Internet*, 12:125, 7 2020.

[3] Serguei Popov. The tangle, 2018.

[4] Anton Churyumov. Byteball: A decentralized system for storage and transfer of value, 2016.

[5] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Report 2016/1159, 2016.

[6] Yonatan Sompolinsky, Shai Wyborski, and Aviv Zohar. *PHANTOM GHOSTDAG: A Scalable Generalization of Nakamoto Consensus: September 2, 2021*, page 57–70. Association for Computing Machinery, New York, NY, USA, 2021.

[7] Martin Perešíni, M. Federico Benčić, Kamil Malinka, and Ivan Homoliak. Dag-oriented protocols phantom and ghostdag under incentive attack via transaction selection strategy. pages 1–8. Institute of Electrical and Electronics Engineers, 2021.

[8] Jelena Mišić, Vojislav B. Mišić, Xiaolin Chang, Saeideh Gholamrezazadeh Motlagh, and M. Zulfiker Ali. Modeling of bitcoin's blockchain delivery network. *IEEE Transactions on Network Science and Engineering*, 7(3):1368–1381, 2020.

[9] KASTEL at KIT DSN Research Group. Bitcoin network monitor, 2015. https://www.dsn.kastel.kit.edu/bitcoin/index.html.