



VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION SYSTEMS

Aurimas Šakalys

**EVALUATING THE INTEROPERABILITY OF PROOF OF BURN AND
OTHER CONSENSUS ALGORITHMS THROUGH SIMULATIONS**

**“PROOF OF BURN” KONSENSUSO ALGORITMO SĄVEIKUMO SU KITAIS KONSEN-
SUSO ALGORITMAIS TYRIMAS PASITELKiant SIMULIACIJAS**

Master's Graduation Thesis

Information and Information Technologies Security, state code 6211BX014

Master of Informatics Sciences

Vilnius, 2025

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION SYSTEMS

Aurimas Šakalys

**EVALUATING THE INTEROPERABILITY OF PROOF OF BURN AND
OTHER CONSENSUS ALGORITHMS THROUGH SIMULATIONS**

Master's Graduation Thesis

Information and Information Technologies Security, state code 6211BX014

Master of Informatics Sciences

Supervisor _____ prof. habil. dr. Antanas Čenys

Consultant _____

Consultant _____

Vilnius, 2025

<table><tr><td>Vilnius Gediminas Technical University</td></tr><tr><td>Faculty of Fundamental Sciences</td></tr><tr><td>Department of Information Systems</td></tr></table>		Vilnius Gediminas Technical University	Faculty of Fundamental Sciences	Department of Information Systems	<table><tr><td>ISBN</td><td>ISSN</td></tr><tr><td>Copies No.</td><td></td></tr><tr><td>Date-.....-.....</td><td></td></tr></table>	ISBN	ISSN	Copies No.		Date-.....-.....	
Vilnius Gediminas Technical University											
Faculty of Fundamental Sciences											
Department of Information Systems											
ISBN	ISSN										
Copies No.											
Date-.....-.....											
Master Degree Studies Information and Information Technologies Security study programme Master Graduation Thesis											
Title	Evaluating the Interoperability of Proof of Burn and other Consensus Algorithms Through Simulations										
Author	Aurimas Šakalys										
Academic supervisor	Antanas Čenys										
<table><tr><td></td><td>Thesis language: English</td></tr></table>			Thesis language: English								
	Thesis language: English										
Annotation The Master's Thesis investigates improving the energy efficiency of blockchain consensus protocols without reducing their decentralization or security guarantees. The study focuses on Proof of Work (PoW) and Proof of Burn (PoB), a lesser-adopted but potentially impactful alternative. Using a custom-built, event-driven simulation framework, the work evaluates the interoperability, scalability, and energy consumption of various consensus configurations. A novel variant of Proof of Burn is introduced, leveraging price control and pseudo-random number generation to maintain block production without fallback mechanisms. The simulation results show that energy-efficient blockchain operation is feasible with Proof of Burn, offering a compelling alternative to Proof of Work. This thesis contributes original findings and tools that may inform future blockchain protocol design and implementation.											
Keywords: Blockchain, Consensus Algorithms, Proof of Work, Proof of Stake, Proof of Burn, Energy Efficiency, Decentralization, Blockchain Simulation, Distributed Systems, Simulator Design, Hybrid Consensus, Environmental Impact, Pricing Mechanisms.											

Vilniaus Gedimino technikos universitetas		ISBN	ISSN
Fundamentinių mokslų fakultetas		Egz. sk.	
Informacinių sistemų katedra		Data-.....-.....	

Antrosios pakopos studijų Informacijos ir informacinių technologijų saugos programos magistro baigiamasis darbas	
Pavadinimas	"Proof of Burn" konsensuso algoritmo sąveikumo su kitais konsensuso algoritmais tyrimas pasitelkiant simuliacijas
Autorius	Aurimas Šakalys
Vadovas	Antanas Čenys

	Kalba: anglų
--	---------------------

Anotacija
Šiame magistro baigiamajame darbe nagrinėjami būdai skirti pagerinti blokų grandinių technologijų konsensuso algoritmų energinį efektyvumą, neprarandant centralizacijos ir saugumo garantijų. Tyrime daugiausia dėmesio skiriama „Proof of Work“ (PoW) ir mažiau taikomo, tačiau perspektyvaus „Proof of Burn“, algoritmų analizei. Naudojant šiam tyrimui sukurtą įvykių eilės pagrindu veikiantį simulatorių, buvo įvertintas šių algoritmų tarpusavio veikimas ir energijos sąnaudos. Darbe pristatoma nauja „Proof of Burn“ variacija, grįsta kainos kontrole ir atsitiktinių skaičių generavimu, leidžianti išlaikyti blokų kūrimą be papildomų konsensuso mechanizmų. Simuliacijos rezultatai rodo, kad energiniu požiūriu, blokų grandinės tinklai gali veikti efektyviai ir saugiai naudojant „Proof of Burn“. Šis darbas pateikia naujus įrankius ir rezultatus, kurie gali būti naudinti kuriant ir tobulinant blokų grandinių protokolus.

Prasminiai žodžiai: blokų grandinės, konsensuso algoritmai, Proof of Work, Proof of Burn, energinis efektyvumas, decentralizacija, blokų grandinės simuliacija, simulatoriaus kūrimas, hibridiniai konsensusai, aplinkos poveikis, kainų mechanizmai.
--

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Aurimas Šakalys, 20185388

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Fundamentinių mokslų fakultetas

(Fakultetas)

Informacijos ir informacinių technologijų sauga, ITSfm-23

(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO)

SĄŽININGUMO DEKLARACIJA

2025 m. gegužės 26 d.

Patvirtinu, kad mano baigiamasis darbas tema „"Proof of Burn" konsensuso algoritmo sąveikumo su kitais konsensuso algoritmais tyrimas pasitelkiant simuliacijas“ yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Mano darbo vadovas profesorius habilituotas daktaras Antanas Čenys.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

(Parašas)

Aurimas Šakalys

(Vardas ir pavardė)

CONTENTS

1. Introduction	13
1.1. Novelty and Relevance	14
2. Analysis of Blockchain and its constituent parts	15
2.1. Consensus Algorithms	15
2.2. Bitcoin and Blockchain	18
2.3. Ethereum, Blockchain and Proof-of-Stake	24
2.4. Proof of Work in Detail	27
2.5. Proof of Stake in Detail	28
2.6. Proof of Burn in Detail	31
2.7. Summary	35
3. Investigation Methodology	35
3.1. Simulator	36
3.1.1. Core Principles of the Simulator	37
3.1.2. Problems in Slimcoin Proof Of Burn implementation	38
3.1.3. Simulator Events	41
3.2. Consensus Simulation Scenarios and Hypotheses	44
3.2.1. Bitcoin Scenario	45
3.2.2. Slimcoin Scenario	47
3.2.3. Unbounded Slimcoin Scenario	49
3.2.4. Pricing Controlled Proof of Burn Scenario	52
3.3. Scenario Power Usage	56
3.4. Results Review	61
4. Conclusion	62
4.1. Limitations	63

LIST OF FIGURES

Figure 1: Visualization of Raft Consensus Algorithm with five nodes in the network exchanging messages (Ongaro & Ousterhout, 2025)	17
Figure 2: High-Level Blockchain diagram illustrating how the blocks are linked	19
Figure 3: Divergence in ledgers between Node A and Node B when transactions arrive at different times.	20
Figure 4: Prediction for Bitcoin (PoW) energy consumption in 2023-2026 (Bublyk et al., 2023)	22
Figure 5: RT, developed by Narayanan et al. (2020), is an IoT OS that Total active mining equipment and electronic waste generation in the Bitcoin (PoW) network over time (De Vries & Stoll, 2021)	23
Figure 6: Ethereum Epoch and Slots (Joseph C., 2020)	31
Figure 7: A model for value decay of a burn transaction of 50 coins.	34
Figure 8: An empty event queue	37
Figure 9: Initialized event queue with a few events in the future	38
Figure 10: Single step simulation advancement - an event is removed from the queue, the simulation time advanced to the time of the event.	38
Figure 11: Rolling average block production time in Bitcoin	46
Figure 12: Rolling average block production time in Bitcoin (reduced reorganization threshold)	46
Figure 13: Average transactions per block – Bitcoin	47
Figure 14: Average production time by block type – Slimcoin	48
Figure 15: Rolling average production time comparison – Slimcoin	48
Figure 16: Average transactions per block – Slimcoin	49
Figure 17: Typed block production statistics – Raze	50
Figure 18: Average transactions per block – Raze	51
Figure 19: Average transactions per block – Raze Only	51
Figure 20: Average production time by block type – Pricing (no randomization)	53
Figure 21: Average production time by block type – Pricing	54
Figure 22: Average transactions per block – Pricing	55

Figure 23: Average production time by block type - Pricing Only	56
Figure 24: Difference between Block Mining Time and Block Production Time	57
Figure 25: Cumulative power usage – across all blocks	58
Figure 26: Cumulative power usage – across all forks	59
Figure 27: Power usage per transaction	60
Figure 28: Power usage per transaction (reduced scenarios)	60

LIST OF TABLES

Table 1: Fork counts based on simulation scenario 59

APPENDICES

Appendix 1: Participation Certificate for Young Scientists Conference	69
---	----

LISTINGS

Listing 1: Ethereum Transaction RPC Call	29
Listing 2: Ethereum Transaction RPC Response	29
Listing 3: An instance of a block mined event.	42
Listing 4: An instance of pseudo random number emitted event.	42
Listing 5: An instance of a block mined event.	43
Listing 6: An instance of pseudo random number receive event.	44
Listing 7: Scenario settings common across all consensus implementations	44
Listing 8: Scenario settings for Bitcoin implementation simulation	45
Listing 9: Scenario settings for Slimcoin 3:1 implementation simulation	47
Listing 10: Scenario settings for unbounded Slimcoin implementation simulation	50
Listing 11: Scenario settings for pure Proof of Burn implementation simulation	50
Listing 12: Scenario settings for Pricing controlled implementation simulation	52
Listing 13: Scenario settings for Pricing Only implementation simulation	55

ABBREVIATIONS

- PoW - Proof of Work
- PoS - Proof of Stake
- PoB - Proof of Burn
- MTBF - Mean Time Between Failures
- AFR - Annualized Failure Rate
- TTS - Trusted Timestamping Service
- DAG - Directed Acyclic Graph
- ASIC - Application-Specific Integrated Circuit
- GPU - Graphics Processing Unit
- FPGA - Field-Programmable Gate Array
- ETH - Ethereum (cryptocurrency)
- JSON-RPC - JavaScript Object Notation - Remote Procedure Call
- DeFi - Decentralized Finance
- NFT - Non-Fungible Token
- DApp - Decentralized Application
- RANDAO - Random Number as a DAO
- EOA - Externally Owned Account
- XCP - Counterparty Token
- MTTF - Mean Time to Failure

1. INTRODUCTION

With the introduction of Bitcoin (Nakamoto & others, 2008), blockchain emerged as a technology that could be utilised instead of only being a theoretical pursuit. While the PoW algorithm pioneered the field, ensuring cryptographic security amongst the nodes, the impact of the algorithm on the environment via vast electrical energy consumption and electronic waste has left a lot to be improved. These issues pushed other, less wasteful consensus algorithm variants to crop up in the blockchain field, namely PoS, which promised to reduce the significant adverse environmental impact that PoW had. With Ethereum completing The Merge in 2022 (Ethereum Foundation, 2024a), we could see that they achieved the improvement that PoS algorithms promised. These improvements have introduced a new risk of reducing decentralisation across the network, which could compromise the core blockchain principle - the ability for potentially selfish or adversarial nodes to reach consensus without the involvement of a central authority.

In this thesis, we explore the potential for the interoperability between multiple consensus algorithms within blockchains, see whether the algorithms can maintain their original performance without much degradation, and perform resource usage analysis. Our analysis is backed by a simulation engine capable of simulating multiple nodes within a blockchain network. We want to show that consensus algorithms that might not have as wide of adoption can still perform as well as more popular ones, providing the required security guarantees and improving the environmental impact.

- **Investigation object:** Blockchain Consensus Protocols;
- **Investigation objective:** Improve the energy requirements for Blockchain consensus algorithms by interoperating PoW and PoB algorithms, without reducing algorithm security;
- **Investigation tasks:**
 1. Perform analysis of PoW, PoS and PoB algorithms, providing detailed reasoning to the need for improvement for the algorithms and providing context for the simulator implementation.
 2. Create the investigation methodology that will be used in the analysis of the simulated consensus algorithms;
 3. Design and implement a simulator capable of providing the results for the suggested investigation methodology;

4. Experimentally investigate the resource usage and performance of consensus protocols by simulating different scenarios; Analyse the gathered results, evaluate the viability of the consensus algorithm improvement scenarios.

1.1. Novelty and Relevance

As blockchain, at least in the mainstream, is a relatively new field, it is rapidly evolving. This evolution is mainly driven by the demands for consensus algorithms to be highly scalable as more and more nodes join the network with rising popularity, energy-efficient, decentralised and secure to ensure trust in the network by participating nodes. Both algorithms are well-documented and widely adopted, both being adopted in a wide variety of blockchain implementations (Noh et al., 2023; Rani & Bhambay, 2022). PoW algorithms usually have high operational costs, as the devices perform large computations requiring large amounts of electrical energy. While PoS addresses these issues, it does so in a way where some sacrifice to decentralisation happens, which shows the relevance for the thesis - we want to see whether other, less adopted algorithms could be used to overcome these issues and to see if there maybe exists significant flaws within the algorithms, which results in low adaptability.

An investigative opportunity exists in the fact that the PoB consensus algorithm remains underutilised, with only a couple of practical implementations within the field. Due to the limited adaptability, we can raise questions regarding its actual viability - whether it is probable to use in a real environment, and if that is the case, whether the algorithm can provide an improvement over the issues faced by other consensus algorithms without sacrificing too essentially in other requirements of a consensus algorithm.

These points will be reiterated in the analytical part of the thesis, as we will explore the causes of the problems within these algorithms in depth, and we will see how PoB should provide improvements over these issues theoretically, eventually culminating in a simulation scenario where we will be able to perform real deployment scenarios, such as simulated ones.

2. ANALYSIS OF BLOCKCHAIN AND ITS CONSTITUENT PARTS

2.1. Consensus Algorithms

Consensus algorithms are critical in distributed systems, as these systems must work without a centralised control mechanism, thus requiring the maintenance of synchronised state information across multiple nodes that most (not necessarily all) of these nodes would agree on. Systems that require communication with or through a centralised point have a single-point failure, where the failure of the centralised point would disable the entire system. Additionally, some systems cannot be designed intelligently with a centralised point, thus requiring consensus algorithms.

The consensus algorithms must also provide robust mechanisms to allow the failure of some percentage of the nodes or devices within the network. To illustrate this necessity, we will look at a simple case study showing the likelihood of hardware failure within a network of devices as the number of devices increases. Even if a single device's failure rate is typically relatively low, making it a non-issue for an individual user, the situation changes drastically when the number of devices grows to hundreds or thousands. Even in scaled hobbyist setups, deploying just up to ten devices, anecdotal evidence suggests failures may occur much earlier than one could anticipate (Yang & Sun, 1999).

We will look at a concrete example of the failure of single and multiple hard drives, as it provides excellent insight into this situation. According to a datasheet of a *Seagate IronWolf Pro* hard drive (Seagate, 2017), they claim a Mean Time Between Failures (MTBF) of 1200000 hours. At first glance, this metric paints a picture of a highly reliable device, which it is, but it might lead one to assume that it would also stay reliable in a scaled deployment.

To better see the actual failure rate, we must convert the reported MTBF into an Annualized Failure Rate (AFR), which is a metric that provides the likelihood that a given device would encounter a failure within a single year. The relationship between MTBF and AFR is described by Equation 1, which in itself was derived by David Szabados (David Szabados, 2010) - a Seagate engineer at the time.

$$\text{AFR} = 1 - \exp\left(-\frac{8766}{\text{MTBF}}\right) = 1 - \exp\left(-\frac{8766}{1200000}\right) \approx 1 - 0.993 \approx 0.007 \quad (1)$$

Equation 1 shows us a 0.7% chance for a single hard drive to fail, showing that the device is highly reliable. This number likely increases as the device ages, but for our purposes, assuming a 0.7% chance across the device's entire lifetime until failure is acceptable.

$$P(\text{Failure with } n \text{ devices}) = 1 - (1 - \text{AFR})^n \quad (2)$$

However, even a 0.7% annual failure rate compounds quickly when deploying multiple devices. For a system containing n hard drives, the overall failure probability can be calculated using Equation 2. This relationship shows that the likelihood of failure increases with the number of devices deployed. As mentioned, external factors influence a higher AFR, especially in a more extensive deployment, like vibrations via spinning disks, heat, and others. However, they are not as important for illustration.

$$1 - (1 - 0.007)^{10} \approx 0.068, \quad n = 10 \quad (3)$$

$$1 - (1 - 0.007)^{100} \approx 0.505, \quad n = 100 \quad (4)$$

$$1 - (1 - 0.007)^{1000} \approx 0.999, \quad n = 1000 \quad (5)$$

Applying Equation 2 for deployments of 10 (Equation 3), 100 (Equation 4) and 1000 (Equation 5) devices, we can see that as the number of devices increases, so does the probability of a failure of one of the deployed devices. Because the probability approaches one as the device count increases, which means that with sufficiently large deployments, it becomes certain that at least one device will fail within a given year, regardless of how reliable an individual component is.

This case study illustrates potential failure rates for a singular device within a node within a distributed system. A single node in a distributed system would have many more individual devices, that might fail, and as hardware failures are a more prominent contributor to the failure of program systems than software failures (Schroeder & Gibson, 2009), this underscores the need for robust consensus algorithms, that would be resistant to failures.

A consensus algorithm allows a group of independent nodes within a network to agree upon a single, truthful state, regardless of the system size or complexity. The algorithms ensure that failures in individual nodes do not lead to the collapse of the entire system. Additionally, consensus algorithms provide resilience against malicious actors who may attempt to introduce incorrect or harmful data into the system, where the actions of such actors can be interpreted simply as a misbehaving/failing node.

At a high level, consensus algorithms typically work by allowing entities within the system to propose changes to the current state. Other entities then validate these proposals based on a predefined rule set (Figure 1). If a majority, or in some cases, a supermajority, agrees on the proposed state, it is committed to a shared log, thereby ensuring consistency. If a majority agreement cannot be reached, these algorithms generally include conflict resolution mechanisms to ensure that only one truthful state is ultimately accepted (Ongaro & Ousterhout, 2014).

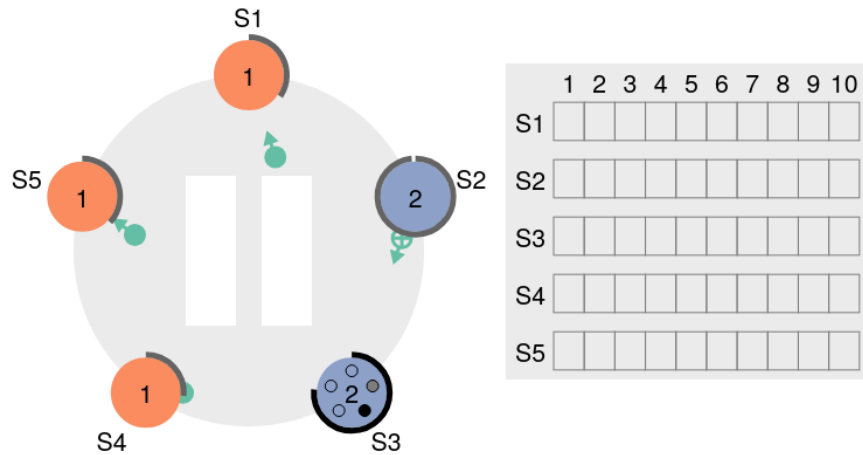


Figure 1: Visualization of Raft Consensus Algorithm with five nodes in the network exchanging messages (Ongaro & Ousterhout, 2025)

Different consensus algorithms achieve these goals in cryptocurrencies and blockchain through various approaches. Proof-based algorithms rely on the use of measurable resources to secure consensus. For example, the Proof-of-Work (PoW) (Jakobsson & Juels, 1999) algorithm, used in Bitcoin (Nakamoto & others, 2008), requires nodes to solve computationally intensive problems to validate transactions within a blockchain network. Similarly, Proof-of-Stake (PoS) (QuantumMechanic, 2011) algorithms select validators based on the resources they are willing to stake, which can be forfeited if they act maliciously. These and other blockchain-specific consensus algorithms will be explored in greater detail later in this thesis, as they form the core focus of this work.

While consensus algorithms are crucial for distributed systems' stability, they have their challenges. Strangely, scalability can become a significant concern as the system grows. If the algorithm is poorly designed, it might lead to significant increases in resource usage and higher latencies as the message exchanges would inevitably have to traverse more considerable distances and be coordinated across an enormous number of nodes. Proof-based algorithms, in particular, face criticism for their

inefficiency in energy consumption; for example, PoW algorithms require nodes to calculate hash values to validate message exchanges repeatedly (Nakamoto & others, 2008). In these cases, while verifying the incoming message is quick and does not require significant computational resources, the validation process demands enormous computational resources, and the required amount grows rapidly as more and more nodes join the network.

2.2. Bitcoin and Blockchain

Conceptually, blockchain origins could be traced back to 1982, when cryptographer David Chaum proposed a protocol for secure systems among mutually distrustful groups in his dissertation (Chaum, 1979). In the dissertation, he discusses systems that build decentralised trust using cryptographic protocols, such as multi-party computation and digital signatures.

The first practical implementation of a cryptographically secured chain of blocks emerged in 1991, developed by S. Haber and W. S. Stornetta (Haber & Stornetta, 1991). They had thought of a system with two participants - a document owner who wanted to get their digital document timestamped in a tamperproof way and a trusted timestamping service (TTS), which would be the one providing the timestamps for the documents. In this scenario, the owner of document x hashes with a hashing function H producing $y = H(x)$. Once this is done, they send the hash y to the TTS, which then creates timestamp t for hash y and produces a digital signature combining both. While this system works, it has issues with the potential for TTS to be incompetent or malicious and issue an invalid timestamp for the given digital document. Haber and Stornetta propose that TTS include a “link” to the previous timestamping request, thus making a chain of requests. In this way, they claim that data would have much better integrity, even if TTS were to act maliciously. In 1993, they improved upon this by introducing Merkle Tree structures and the idea of distributing the chain across multiple devices (Bayer et al., 1993).

Work by Haber and Stornetta laid the initial groundwork for the modern implementations of blockchain. This work culminated with the introduction of a decentralised blockchain technology behind Bitcoin. Satoshi Nakamoto’s seminal 2008 paper (Nakamoto & others, 2008) combined the previously described system of distributed linked blocks with an introduction to a ledger that allowed

them to create a system that could record peer-to-peer transactions without relying on a trusted central authority. This event marked the beginning of blockchain as we know it today.

At the core of current blockchain technology, we have a decentralised and distributed ledger of transactional records across a network of interconnected nodes. Each node within this network maintains its copy of the ledger, which, with the help of consensus algorithms, ensures consistency and transparency within the system. A block within this chain contains transactional data and metadata like the timestamp of when the block was created and the link (in this case, a cryptographic hash) to the previous block, as seen in Figure 2. The creator of the block is responsible for picking which transactions broadcasted over the network are added to the block. In subsequent discussions, we will discuss the mechanism enabling this functionality in greater detail.

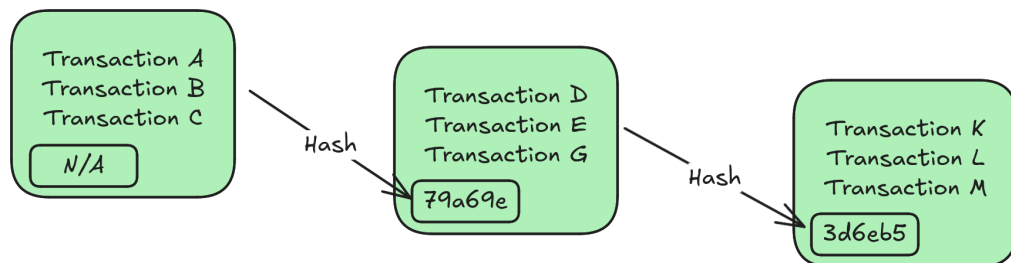


Figure 2: High-Level Blockchain diagram illustrating how the blocks are linked

The recursive references within the block to the previous blocks form the blockchain's fundamental structure and provide some immutability to the chain, although this is not true immutability. If an adversary wanted to modify a particular block, the hash for that block would become different due to this modification, thus breaking the chain towards all the other blocks already mined. If the adversary did this, they would have to advertise multiple valid blocks, meaning they would have to calculate the PoW proof for the number of blocks behind from the point of divergence. If they managed to do this, they could change the chain, as internal mechanisms of fork resolutions would kick in. A situation like this is improbable, as they would have to have 51% of the computing power within the blockchain network. Of course, luck could play a factor in allowing an adversary to perform the attack without the required computing power.

Initially, modern blockchain technology was mainly only used for Bitcoin, just as with the trusted document timestamping system by Haber and Stornetta, blockchain has evolved, and the use of the technology has reached various industries. For example, it is used in supply chain management

(Oriekhoe et al., 2024), where it is used to track shipments, verify the authenticity of high-value merchandise, and ensure chain-of-custody. It has use cases in ticketing systems (Aldweesh, 2023), ownership of digital goods (Alshater et al., 2024) and many others.

Bitcoin emerged partly as a solution to the double-spending problem within a decentralised transactional system. In traditional transactional systems, a centralised third party assures whether a monetary unit has been spent. While such an arrangement works well, it has two significant drawbacks. First, the third party within this system must be trusted by both parties participating in the transaction, both in the sense of competence and non-adversarial action. If the third party is compromised or acts maliciously towards any of the other two participating parties, the trust could be broken irreparably, and the system, at least in the context of these participating parties, would cease to work. Secondly, the existence of a centralised third party means a centralised point of failure as well, where if the third party fails to meet the availability requirements, the system will cease to work as well.

Despite these concerns, there is one overwhelming advantage for centralised systems, so much so that for most use cases, any other alternative has very little reason to be used: centralised systems are significantly more straightforward to implement. The most stripped-down example of how easy it is to deploy such systems is that the third party would need to maintain a ledger (which could be just a pen and paper ledger), which the other participants trust and agree to.

The lack of a central authority in a distributed system introduces a complex challenge. Every node participating in the network must maintain its copy of the transactional ledger. Double-spending in these systems happens when two conflicting transactions (or, more accurately, blocks) are broadcast into the network; the ledgers of different nodes might diverge, as seen in Figure 3.

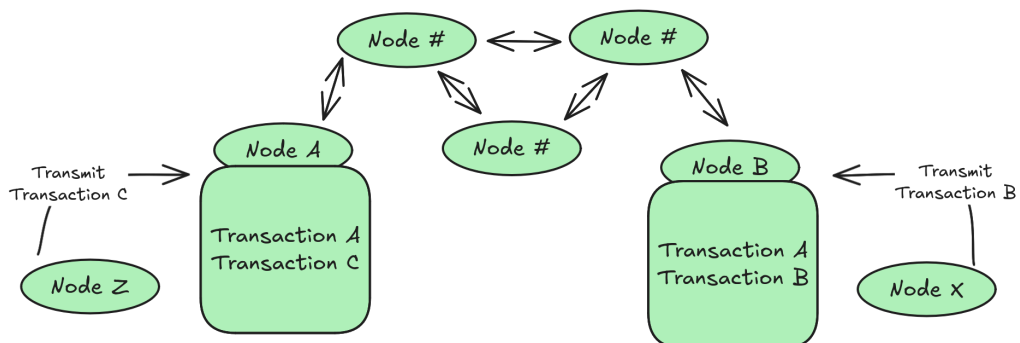


Figure 3: Divergence in ledgers between Node A and Node B when transactions arrive at different times.

This problem is resolved by using the consensus algorithms we have discussed previously. Consensus algorithms ensure that all nodes eventually agree upon a single version of the ledger, even if conflicting transactions are broadcast over the network. Distributed systems that work with transactional ledgers cannot feasibly use consensus algorithms that elect a leader node within the network, as that leader node would have the same problems as a central authority, in addition to the fact that absolutely anyone within the distributed network could be elected as the leader node, increasing the risk of adversarial actions. As such, the algorithms used for transactional ledger blockchains use proof-based algorithms, which do not require a leader node to be elected. Currently, Bitcoin uses the PoW consensus algorithm that requires nodes to perform computationally intensive tasks to validate transactions before they can be added to the blockchain. Ethereum, which initially used PoW, has since transitioned to the more power-efficient PoS algorithm. These algorithms will be explored in greater detail later in the thesis.

The PoW algorithm used in Bitcoin has challenges, particularly regarding scalability and energy consumption. Scalability issues are primarily tied to the configuration parameters of the PoW algorithm. The algorithm adjusts the difficulty of the hash challenge served to the nodes based on how quickly the previous blocks were mined, and it is adjusted in such a way that the mining speed for a block would be approximately 10 minutes. Additionally, the Bitcoin blockchain imposes a limit of 1MB per block, which caps the amount of transactions validated within one block, with the previous requirement for block mining time. Given that the average transaction size in Bitcoin is around 250 (Walker, 2024), we can calculate how many transactions Bitcoin can do per second.

$$T = \frac{1000000B}{250B} * \left(1 \frac{t}{600} s\right) = 6.67 \text{ t/s} \quad (6)$$

With 6.67 transactions per second (Equation 6), Bitcoin compares poorly to other non-centralised transaction processors; for example, Visa claims that it can handle up to 65,000 transactions per second (Visa Inc., 2023). This limitation in Bitcoin's processing capability leads to notable slowdowns in transaction processing, especially if the network experiences a surge in demand; certain transactions would not be included in the currently mined block, resulting in delays until the next mining iteration. This issue compounds as more and more nodes enter the network and broadcast their transactions.

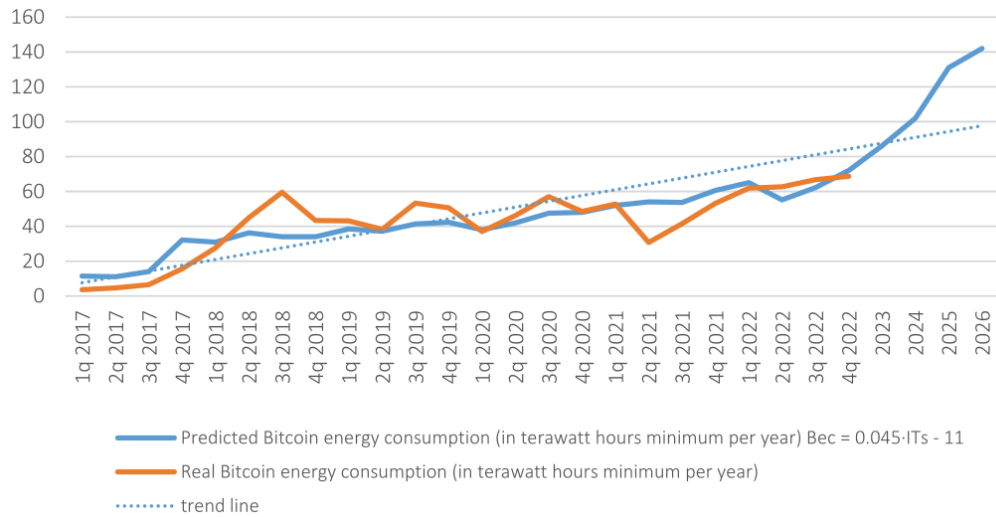


Figure 4: Prediction for Bitcoin (PoW) energy consumption in 2023-2026 (Bublyk et al., 2023)

Beyond scalability concerns, PoW suffers from a significant issue related to energy consumption due to adjustable difficulty and new nodes joining the network (Figure 4). To illustrate, let us consider an initial scenario where only ten nodes join the network. Based on their combined computational capabilities, the algorithm adjusts the difficulty stabilizing block production at the desired 10-minute interval. However, as the network grows in popularity, more nodes join, increasing the power usage across the network. Because new nodes have joined the network, the time taken to mine a single block would likely have reduced below the 10-minute interval, increasing the difficulty and consequently increasing the power usage within the network.

The growth of difficulty due to the network's growing popularity creates another issue: hardware obsolescence. As mining becomes more complex, older hardware, once sufficient to contribute to the network, becomes less viable due to component degradation or return on investment. As such, miners are incentivized to replace outdated equipment with more recent or more efficient devices, leading to increased electronic waste, as shown in Figure 5.

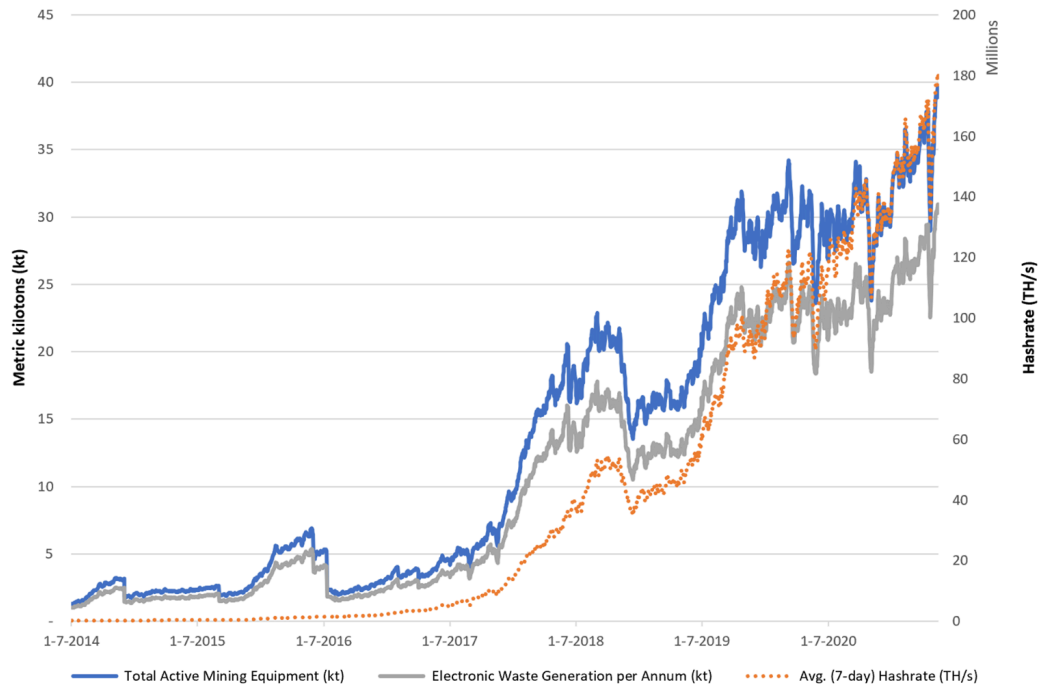


Figure 5: RT, developed by Narayanan et al. (2020), is an IoT OS that Total active mining equipment and electronic waste generation in the Bitcoin (PoW) network over time (De Vries & Stoll, 2021)

In the early stages of Bitcoin mining, the most commonly used hardware type was the Graphics Processing Unit (GPU). As the mining of the block meant that a node would have to find a hash value that is lower than the current target difficulty of the network, usage of GPUs was preferable as GPUs were commodity hardware with capacity to perform highly parallelized hashing operations. As GPUs have other uses, like video game rendering, video editing, transcoding and, more recently, machine learning; once the GPU is considered unusable for mining, it could be repurposed for any of the other uses.

In 2012, Butterfly Labs released the first Application-Specific Integrated Circuit (ASIC) purpose-built for Bitcoin mining (BITMAIN Technologies, 2025). Instead of general-purpose matrix processors in GPUs, ASICs can be designed directly in hardware to perform the calculations required for Bitcoin mining. The device released by Butterfly Labs vastly outperformed mining by use of CPUs and GPUs, so much so that it has now become the standard for Bitcoin mining, as most of the mining is now performed in dedicated ASIC mining arrays. Unfortunately, the use of ASIC has a significant drawback - as they are explicitly designed to perform the Bitcoin mining task, they have no other use case. As the lifetime of ASIC devices comes closer to the end, these devices cannot be repurposed for any other activity and immediately become electronic waste.

There is another alternative, namely Field-Programmable Gate Arrays (FPGA), which are reprogrammable hardware devices where individual transistors within the chip can be flipped to create a hardware-level optimized chip, which could be changed in the future simply by reprogramming it. While initial attempts to use FPGAs for Bitcoin mining were several times slower than mining using GPUs (Oliveira et al., 2012), recent advancements in FPGA technology have shown that they can be programmed in a very efficient way, getting very close to the efficiency shown by the dedicated ASIC hardware (Pandya et al., 2022). In the electronic waste context, FPGAs are significantly better, as they can provide competitive mining speeds and reduce electronic waste, as the FPGAs could be reprogrammed. Unfortunately, while FPGAs can be cheap to use in low capacities, when increasing manufacturing scales, ASICs eventually beat FPGAs out, as a single FPGA unit has a relatively high cost. The most significant cost in ASIC development is the development of the lithographic masks (Ho et al., 2012), which defines the structure of the chip. Once the mask is prepared, the manufacturing costs per single ASIC chip are very low. As such, FPGAs have not been widely adopted, as the return on investment on these chips is very low.

The environmental impact of this dynamic cannot be ignored, as the increase of monetary incentives in cryptocurrency and the growth of the use of dedicated mining devices cause a vicious cycle where energy consumption and electronic waste numbers continue to rise. Efforts to repurpose the mining hardware are insufficient to offset the ongoing environmental damage, as only a tiny part of the devices can be recycled. We can claim that searching for an alternative or improvement to the PoW algorithm is required.

2.3. Ethereum, Blockchain and Proof-of-Stake

Ethereum, introduced in 2015 by Vitalik Buterin (Buterin & others, 2013), marked a groundbreaking step in the evolution of blockchain technology, as it was designed from the ground up to be more than simple blockchain used for financial ledger purposes. The creators of Ethereum had a broader vision - they wanted the blockchain network to serve as a potential platform providing distributed and cryptographically secure computing resources to the network participants. This vision was built upon the fact that the nodes within the network were already equipped with computational resources to calculate proofs for PoW algorithms which could be utilized for additional purposes.

Ethereum was built on the original P2P ledger introduced by Bitcoin by incorporating an additional layer that could be programmed and executed by the participants in the network. This layer came to be known as Smart Contracts (Ethereum Foundation, 2025a). The first time smart contracts were introduced was back in 1994 by cryptographer Nick Szabo (Szabo, 1994), who envisioned them as cryptographically secure computations that could facilitate certain transactions without the requirement of a central authority. While the use of Smart Contracts remained theoretical for a long time, Ethereum turned them into a functioning system, providing the framework for the Smart Contracts to be autonomous and cryptographically secure.

In the Ethereum ecosystem, Smart Contracts operate as an Ethereum account. However, unlike externally owned accounts (EOAs) controlled by individuals or organizations holding private keys, Smart Contracts are controlled by the contracts' code. This distinction makes Smart Contracts autonomous, executing their operations automatically based on the conditions set within their code. Programmable contracts unlocked various use cases for different types of applications in many different industries:

- Decentralized Finance (DeFi):
 - Stablecoins - cryptocurrencies whose value is directly tied to a physical asset (e.g. gold) or fiat currency (Arner et al., 2020).
 - Automated Liquidity Provisioning - mechanisms that enable seamless token swaps or conversions between cryptocurrencies and liquid assets (Bhaskara et al., 2023).
- Supply Chain: Chain of Custody Information: Smart contracts can provide verified and transparent records of product ownership throughout the supply chain. (Santamar\ia et al., 2023).
- Non-Fungible Tokens (NFTs):
 - Ownership and Royalties - NFTs represent ownership over unique attributes, such as verifying ticket ownership or automating royalty payments to artists (Arora et al., 2022).
- Decentralized Applications (DApps):
 - Blockchain-Based Games - games where the state and logic are stored on the blockchain. For example, an entire chess engine, albeit with a smaller board, has been implemented entirely on the blockchain (fiveoutofnine, 2022).

One of the problems the initial PoW algorithm used in Ethereum sought to address was the dominance of ASIC devices in a cryptocurrency mining environment. As discussed before, ASIC devices become obsolete over time and cannot be recycled for other uses. To mitigate this issue, Ethereum uses a bandwidth hard PoW algorithm called Ethash (Han et al., 2019), which uses very large Directed Acyclic Graphs (DAG), which meant that solving for the proof required significant memory capacity, as opposed to SHA256 used in Bitcoin, which only required 256 bits for one hashing operation. Recent research suggests that modern algorithms remain challenging to optimize for ASICs, thereby reducing their usage (Oreder et al., 2022). These problems are not as difficult for commodity hardware like GPUs, encouraging their use and reducing electronic waste, as these devices can be recycled after mining use.

Despite the presence of helpful work within Ethereum, due to the reliance on the PoW, just like Bitcoin, the blockchain network struggled with scalability and issues with quickly rising energy demand. These issues prompted the Ethereum community to search for a more sustainable and scalable alternative, and they slowly moved towards a PoS model. The transition was executed in multiple stages. The first stage came in 2020 with the launch of the Beacon Chain (Ethereum Foundation, 2024b), which ran parallel to the existing Ethereum PoW chain. At this stage, the chain mainly produced empty blocks, which served as testing grounds for the blockchain developers to ensure that the implemented algorithm was reliable without risking any transactional data, allowing the developers to identify issues early on. By 2022, the Beacon Chain began accepting transactional data using the new PoS consensus algorithm. This event was called the Merge (Ethereum Foundation, 2024a) and marked where Ethereum officially transitioned away from the PoW algorithm. During the Merge, the execution clients of the original chain ceased PoW block mining but were kept there as an execution layer.

The Merge was a significant success, reducing Ethereum's energy consumption by an impressive 95.5% (Kapengut & Mizrach, 2023). Beyond its immediate benefits, this transition laid the groundwork for future upgrades to the network, such as sharding, but the full implementation and impact of such improvements remain to be seen.

While the switch to PoS brought significant benefits, it has also increased the risk of centralization within the network. One of the factors increasing the risk is the high barrier of entry to become a validator - a participant within the network must stake a minimum amount of 32 of ETH (Ethereum Foundation, 2024c). With the price of 3238.32€ per ETH (Investing.com, 2025) at the start of 2025, the entry cost to participate in block validation is 103626.24€, which creates a steep financial requirement for newcomers coming into the network, meaning block validation becomes more accessible to entities or individuals with significant financial resources.

Another factor increasing risk is that the Ethereum PoS model inherently ties the validator influence to the amount of ETH staked, allowing wealthy participants who can afford to stake large amounts of ETH to gain a disproportionate advantage in influence over the network operations. This dynamic increases Ethereum's risk of slowly shifting towards a more centralized structure, where mostly highly wealthy validators dominate the network, undermining the foundational principle of equality between network participants. Additionally, the PoS architecture favours nodes with the capacity to maintain high uptimes, as missing validation rounds typically incur penalties. These requirements introduce another barrier for small-scale validators who may lack the resources to invest in the robust and reliable hardware required to ensure continuous operation. Such requirements further concentrate the power of validation in the hands of larger entities that have the infrastructure and technical capabilities to meet such validation demands.

2.4. Proof of Work in Detail

In the previous sections, we have discussed the theoretical implications of consensus algorithms. There are certain aspects of the PoW algorithm used in Bitcoin that is going to play a role in the simulator's design; as such, we shall explore them now.

Bitcoin's PoW algorithm is based on the hashing function of SHA256. As discussed previously, a block would contain transactional information and some metadata, which would be used when attempting to create a valid hash. For a block hash to be considered valid, the hashed value must be lower than some specific value defined by the network's difficulty. If the algorithm only used the information within the block to perform a hash, only one valid value for the hash would be created. However, it is improbable that the created hash would be lower than the target difficulty, even across

the entire network. As such, a nonce value is included with the block information. With the capacity to variate the nonce value, a node can perform multiple hashing operations, hoping to create a hash value lower than the network difficulty.

The difficulty of the network is not constant, and it varies. Bitcoin aims to set the difficulty such that a new block would be mined only once every 10 minutes across all nodes within the network. Due to inherent randomness in hashing, the hash beating the difficulty might be found within 5 minutes, or it might be found within 20. While these are uncommon, they illustrate the potential variability in mining times. Despite this variability, with the network difficulty adjustment, the average block mining time is expected to converge to 10 minutes over some time.

The difficulty adjustment is not continuous; instead, the difficulty changes within a difficulty adjustment interval, which spans 2016 blocks. Given an average block mining time is 10 minutes we can calculate the interval time in weeks (Equation 7).

$$\text{Difficulty Adjustment Interval} = \frac{2016 \text{ blocks} \times 10 \text{ min/block}}{10080 \text{ min/week}} = 2 \text{ weeks} \quad (7)$$

After 2016th block is mined, the difficulty adjustment algorithm kicks in across all nodes and is adjusted as per Equation 8.

$$\text{New Difficulty} = \text{Current Difficulty} \times \left(\frac{\text{Actual Time Taken}}{\text{Expected Time}} \right) \quad (8)$$

Since all nodes within the network have their copy of the blockchain, they calculate the Actual Time Taken across the last 2016 blocks, use Equation 8. Moreover, it can independently arrive at the same value for New Difficulty. To prevent extreme fluctuations in network difficulty, the New Difficulty calculation is bounded from above and below, where the difficulty cannot increase or decrease by a factor of 4 or greater (Equation 9).

$$\frac{1}{4} \times \text{Current Difficulty} < \text{New Difficulty} < 4 \times \text{Current Difficulty} \quad (9)$$

2.5. Proof of Stake in Detail

As the algorithm's name suggests, PoS operates on the principle that block validators must commit a portion of their cryptocurrency holdings as a stake. Once the participant stakes 32, they are

placed into an activation queue. This queue functions as a mechanism that controls the rate at which new validators enter the network. Once the participant exists in the queue, they will have the capacity to participate in the validation of PoS blocks.

The validation process unfolds as follows - first, a transaction is created by an account within the Ethereum network by making a JSON-RPC (Ethereum Foundation, 2025b) call for the `eth_sendTransaction` procedure, as detailed in Listing 1. This procedure involves signing the transaction using the account's private key before submitting it to the Ethereum network. The response (Listing 2) contains the hash of the transaction - a unique identifier for the transaction within the blockchain network.

```
1  {
2    "id":1,
3    "jsonrpc":"2.0",
4    "method":"eth_sendtransaction",
5    "params":[
6      0: {
7        "from":"0xb60e8dd61c5d32be8058bb8 ...",
8        "to":"0xd46e8dd67c5d32be8058bb8 ...",
9        "gas":"0x76c0",
10       "gasprice":"0x9184e72a000",
11       "value":"0x9184e72a",
12       "input":"0xd46e8dd67c5d32be8d46e8d ...",
13     },
14   ],
15 }
```

Listing 1: Ethereum Transaction RPC Call

```
1  {
2    "id":1,
3    "jsonrpc":"2.0",
4    "result":"0xe670ec64341771606e55d6b ..."
5  }
```

Listing 2: Ethereum Transaction RPC Response

The submitted transaction is broadcast over the network, where it will reach other nodes, which will accept it and perform certain validation checks. The validations are performed by the Execution Client, the leftover client from the original PoW implementation. During the execution process, the client verifies two conditions about the incoming transaction: first, it ensures that the sender has

sufficient ETH to cover the transaction; second, it checks that the transaction was signed correctly using the private key of the owner of the transaction. If any of these validations fail, the transaction is rejected and does not proceed to further processing stages.

If the transaction is deemed valid, the execution client adds it to a local pool of pending transactions and broadcasts it over the gossip network. The gossip layer is a layer designed for a fast exchange of critical information within the blockchain, like transactions, blocks, and slashing events, ensuring that this information spreads quickly across the network.

Here, we must talk about the RANDAO algorithm (randao, 2023), which is the core algorithm used in generating random numbers when selecting block proposers and validators from existing validators within the network. This algorithm provides randomization within a distributed system, which is difficult to achieve. RANDAO works through several phases. Initially, each of the n participants in the network generates a secret value, denoted as s_i , which they store. Next, they compute a hash using this secret value $h_i = H(s_i)$, where $i \in [0, n)$. Here, the RANDAO algorithm employs Smart Contracts, which we have discussed before. Each node sends the h_i value that they generated to the Smart Contract, and once all h_i values are sent to this Smart Contract, the contract transitions to the next stage, where it expects the nodes to now send the original s_i values they generated. When receiving s_i values, the Smart Contract verifies each incoming s_i value by checking that they match the h_i values sent previously. If a mismatch is detected (or a node fails to send the required data), the Smart Contract takes corrective action that is entirely dependent on how the Smart Contract is implemented. For instance, the entire process may be restarted, or if the issue is attributed to data corruption rather than malicious action, only part of the process might be restarted, or malicious participants may face penalties. If all the s_i values sent match the corresponding h_i values received, the Smart Contract can compute the final random number by executing the operation in Equation 10.

$$r = s_0 \oplus s_1 \oplus s_2 \oplus \dots \oplus s_{n-2} \oplus s_{n-1} \quad (10)$$

Finally, the process reaches the consensus layer, introduced when Ethereum switched to the PoS algorithm. The validation process happens in set intervals called epochs. Each epoch is split into 32 slots, as shown in Figure 6, each slot being distributed in time, lasting 12 seconds. This results in the total duration of an epoch being 6.4 minutes. The slot duration and epoch size are configurable within

the Ethereum network, and while these parameters have not changed yet, these numbers might be different in the future.



Figure 6: Ethereum Epoch and Slots (Joseph C., 2020)

The random choices from the validator pool by use of RANDAO are all weighed by the amount of currency each validator is staking, meaning the higher the stake, the validator has a higher chance of being chosen to be proposers or attestators.

Firstly, a block proposer is chosen for the current slot. Their responsibility is to take transactions from the local pending transaction pool, package them into a beacon block, and send them out to the network. Next, the attestator committees are chosen. A committee must include a minimum of 128 validators, a requirement designed to significantly reduce the likelihood of a malicious actor gaining control over a supermajority of the committee, effectively reducing the probability of such an attack to approximately 1 in a trillion (Liang et al., 2019). New random committees are also chosen for each slot within the epoch, reducing the risk of sustained attacks. The role of the committee members is to attest to the location of the blockchain's head, including the proposed new blocks. Suppose a supermajority of committee members agrees on the head of the blockchain within the current slot. In that case, the process advances to the next slot, where a new proposer and committee are chosen, a new block is proposed, and this mechanism continues. Once slot 32 of the epoch is reached and the head of the blockchain has been attested, a checkpoint is created, which serves as a formal commit of the previous epoch.

2.6. Proof of Burn in Detail

Lastly, we must analyze how the PoB algorithm works to ensure we understand how to simulate it using the simulator. In essence, the algorithm is very similar to the PoS algorithm, as the validator/minter must use a certain amount of currency that they own, except in the case of PoB, instead of

staking the currency where it disappears from the network only in the cases of adversarial or neglectful actions, the currency disappears as soon as the node participates in the PoB. At a high level, this is achieved by sending out some amount of currency that the node holds into a wallet that no one has control over - a wallet that is impossible to control by anyone, even the algorithm's designers. Once the node burns its currency, it is given a receipt, allowing it to attempt to mint the new block.

Initially, PoB was proposed by Iain Stewart (Stewart, 2012), whose proposal was motivated by the desire to create an alternative to the PoW algorithm that would improve the network's power consumption without sacrificing decentralization. He viewed the algorithm as a form of virtual mining. In the analogy, a miner attempting to mine a PoW block accrues costs via hardware rigs, electrical energy and system upkeep, expecting that the transaction fees and block rewards would outweigh the accrued costs. In PoB, the situation is similar, where the minter of PoB block accrues costs by burning the currency and expects the transaction fees and minting rewards to outweigh their initial investment. In this sense, we could view currency burning in PoB as buying virtual mining hardware.

Compared to PoS, PoB allows for more decentralization because if wealthy participants want to exert more power over the network, they would necessarily have to burn large amounts of currency.

Unfortunately, PoB has not been widely adopted. The most well-known implementations are Slimcoin (Slimcoin Project, 2013) and Counterparty (Krellenstein, 2024).

The counterparty uses PoB to convert Bitcoin into XCP tokens. Without going into great detail, using PoB to convert one cryptocurrency into another is significantly more straightforward, as the person converting the currency only needs to provide the proof that they have performed the burn to get awarded the the cryptocurrency of the other blockchain (Stewart, 2012).

Slimcoin, on the other hand, implements PoB as a mechanism for minting blocks within the blockchain. The entire implementation of Slimcoin is a bit of a Frankenstein. Initially forked from PPCoin (King & Nadal, 2012), the first cryptocurrency to implement a PoS algorithm launched alongside a PoW algorithm. Due to the fork, initially, Slimcoin was a hybrid algorithm employing both PoW and PoS algorithms. However, introducing the PoB algorithm into the mix could now validate the blockchain using all three discussed algorithms. As of 2025, the cryptocurrency has not been in active development for the last 4 years, with the last commit pushed to the master branch occurring in

late 2020 (Slimcoin Project, 2020), and there are no active forks of the cryptocurrency either (Slimcoin Project, 2025).

Despite not being in active development, analyzing how Slimcoin implements its PoB algorithm would provide us with insights into the potential design requirements for the simulator and allow us to recognize any potential pitfalls of this algorithm.

The algorithm implements a concept of burning coins. Once a node creates a transaction that sends a portion of the currency towards an address no one controls, the currency is considered burned. The node could participate in minting a new block by providing the transaction hash as a verification that they burned the currency. The “receipt” cannot be used immediately after performing the burn, as the burn transaction must be some block deep within the blockchain to ensure that the burn transaction cannot be removed easily by a blockchain fork.

The process of minting a PoB block reflects the process of PoW. In PoW, a miner performs hashing operations until they find a hash value lower than the network’s current difficulty or if they receive a newly mined block and validate it. Given $H = \{h_1, h_2, \dots, h_n\}$ is a set of n hashes created during the process of mining a block; there exists an $h_{\min} \in H$ such that $h_{\min} = \min(H)$ from all values in H , assuming no collisions occurred during the mining process.

This is reflected in the Slimcoins implementation of PoB, except that a maximum of 1 hash can be calculated per singular burn transaction while minting a block. This is analogous to PoB, where $n = 1$, meaning that for a single burn transaction, the minter either succeeds in minting a block or does not. The hashing algorithm in PoB works as described in Equation 11.

$$\begin{aligned} \text{Burn Hash} = \\ M(\text{Blockchain}, \text{Burn Transaction}) * \\ H(\text{Blockchain}) \end{aligned} \tag{11}$$

Here, M is a function that returns a multiplier that depends on the age of the Burn Transaction in blocks and the amount of currency burned, and H is a hash function returning a hash for the current state of the blockchain. Here, we notice that if the blockchain does not change, both M and H will always return the same values. In this case, the Burn Hash would only change if a new block

is appended to the blockchain, where now we have one minting attempt per burn transaction per blockchain block.

The two dependencies control two different aspects of the multiplier function M . The amount of currency burned in the burn transaction is inversely proportional to the multiplier, meaning - the more significant amount of currency burned, the lower the multiplier gets, which in turn allows for a higher chance to mint a block based on the network's current difficulty. The age of the burn transaction provides a way to model a decay of the currency burned as time passes, meaning that initially, the multiplier calculated by M could be low. However, it would keep increasing as time passes and more blocks are minted. This decay (and the initial grace period) can be modelled in Figure 7.

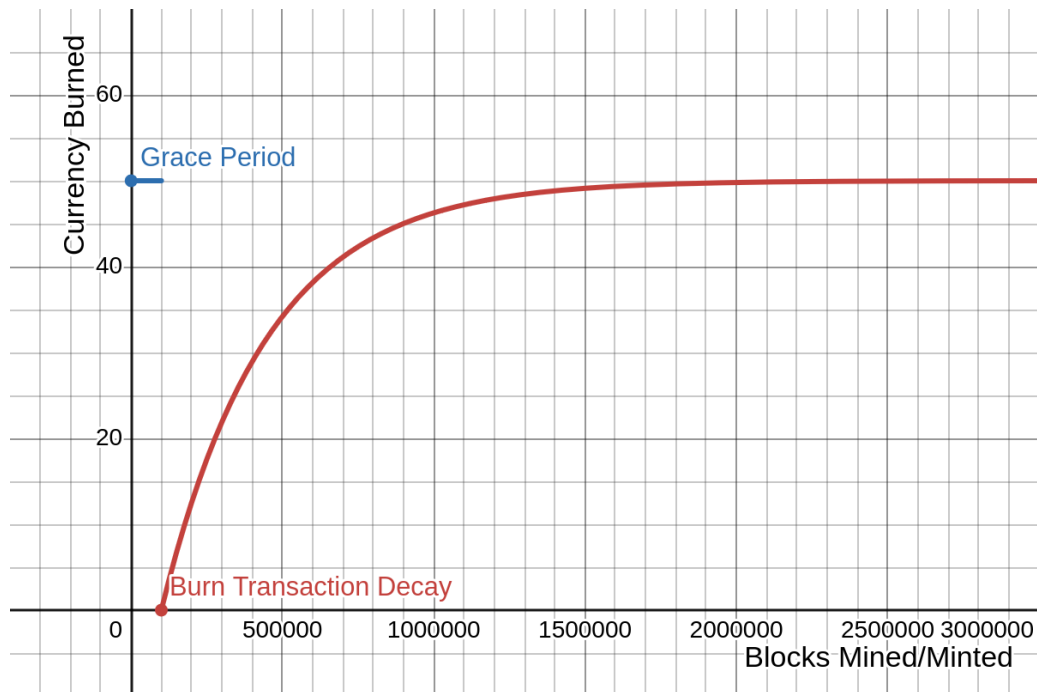


Figure 7: A model for value decay of a burn transaction of 50 coins.

As such, for each minting generation in the blockchain, every minter gets n attempts to beat the network's difficulty, where n is the number of burn transactions they have made. In this scenario, we can raise an issue that there is a high likelihood for the blockchain to essentially deadlock, where every single minter fails to mint a block, as only a single value of H is provided per state of the blockchain. At that point, no more blocks would have the capacity to be mined, and the blockchain would be deadlocked.

As Slimcoin uses a triple-hybrid PoW, PoS and PoB scheme, this allows for the network to continue with mining of PoW or minting of PoS blocks, which in turn would get added into the blockchain, changing the H value, allowing for PoB minters to attempt minting this generation.

2.7. Summary

We have examined three consensus algorithms central to this thesis - Proof of Work, Proof of Stake and Proof of Burn. PoW, currently used in Bitcoin, uses computational power for the security of the blockchain network, however, it faces issues with scalability, high energy consumption and environmental impact from resource-intensive mining tasks and electronic waste. PoS, adopted by Ethereum, addressed the issues plaguing Bitcoin by requiring the participants in the network to stake some of the currency they have to participate in the validation of the next block, where the currency staked could be removed for malicious or neglectful action. While this switch has solved the problem of high energy usage and has improved scalability, it introduced concerns about centralization and high entry barriers as wealthy participants gain prominent influences over the network.

PoB provides an alternative approach requiring participants to burn currency to gain minting rights. This method significantly lowers environmental impact and avoids the hardware dependency of PoW while maintaining some decentralization advantages over PoS. However, PoB has not been adopted widely, with only a few implementations, like Slimcoin incorporating it into hybrid systems. Each algorithm has its trade-offs between energy efficiency, decentralization and scalability, which will be the basis for the simulation exploration in the thesis.

3. INVESTIGATION METHODOLOGY

In this thesis, we will employ a custom-built simulator to model certain blockchain network behaviours using different consensus algorithms. To validate the simulators' accuracy, we will replicate scenarios for which we have real-world data, such as Bitcoin, for the Proof of Work consensus layer, and Slimcoin is a hybrid consensus layer of Proof of Work and Burn. The data generated by the simulator will then be compared with the known data to determine the simulator's accuracy.

After validation, we will introduce scenarios that we are interested in simulating, looking for potential increases in processing speeds and decreases in power consumption when processing, and see whether such improvements are feasible at all.

3.1. Simulator

While the simulator is custom-built, it is partly based on a simulator created by M. Alharby for their Master's Thesis (Alharby & Moorsel, 2020). The simulator is written in Python and is available publicly on GitHub (Alharby & Moorsel, 2022). Their simulator works in three distinct layers: network, consensus and incentives.

The network layer simulates peer-to-peer communication between nodes. While the layer has a few more possible interactions, like a node asking for blockchain data from another node, this is not modelled in the simulator. We have adapted some functionality for this layer in our simulator implementation and will expand on it in detail later.

The consensus layer concerns the simulation of the consensus algorithms. In the original Blocksims paper (Alharby & Moorsel, 2020), they implemented Proof of Work simulating Bitcoin and Proof of Stake simulating Ethereum. The Ethereum layer simulation has been left out of our simulator, but could be introduced without much of an issue. We have partially reimplemented the Bitcoin and additional consensus algorithms and will show how they work in detail. Additionally, this layer provides a detailed transaction execution and transmission implementation, which we have opted not to include in our implementation, opting to simplify the layer, but still having the capacity to reach results that interest us.

Lastly, the incentives layer rewards the nodes within the network for participating, providing rewards based on the consensus layer used. Nodes get rewards for finishing mining blocks, transaction fees, and other rewards defined within the blocks. Our simulation implementation has skipped this layer entirely, as it either does not provide informative results, or the analysis of the results would require analysis in a separate thesis¹.

¹The thesis would have to define and analyse exact pricing for Proof of Burn transactions. For our investigation, we assume some virtual currency that has no ties to real monetary value within the simulator.

For the initial implementation of the simulator and understanding how it worked, we attempted to extend the original codebase (Šakalys, 2023a). The Python implementation has performed poorly when scaled up to extensive simulation scenarios, like simulating a year of actual Bitcoin network with 65,000 nodes (Bitnodes.io, 2025). A rewrite in C# was done (Šakalys, 2023b), which has shown some improvement regarding the performance, but it had issues with extensibility and portability. Lastly, the entire project was rewritten from the ground up using GoLang (Šakalys, 2025), and this implementation will be used to play through the simulation scenarios.

3.1.1. Core Principles of the Simulator

Although the simulator is loosely based on the original Blocksim (Alharby & Moorsel, 2022) simulator, the simulator we have implemented differs in many ways in how network and consensus layers are handled and which data within the simulator we deem crucial.

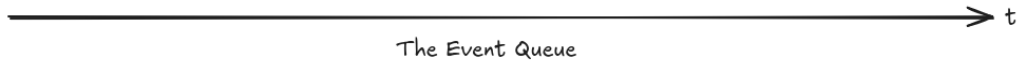


Figure 8: An empty event queue

The binding between all interacting parts of the simulation is the Event Queue (Figure 8). The event queue is implemented using a heap data structure. The event queue has to store events and order them in ascending order by the execution time t of the event. It has to do this ordering quickly, as the number of events grows $O(2n)$ where n is the number of nodes participating in the simulation. As such, usage of a heap allows us to both push and pop elements from the queue at a speed of $O(\log(e))$ where e is the number of events in the queue. If we were to simulate the Bitcoin network with the current capacities of 65,000 nodes (Bitnodes.io, 2025) within the network, on average, around 130,000 events would be scheduled at any one time, but would only take 17 comparisons to sort a newly added or a removed event.

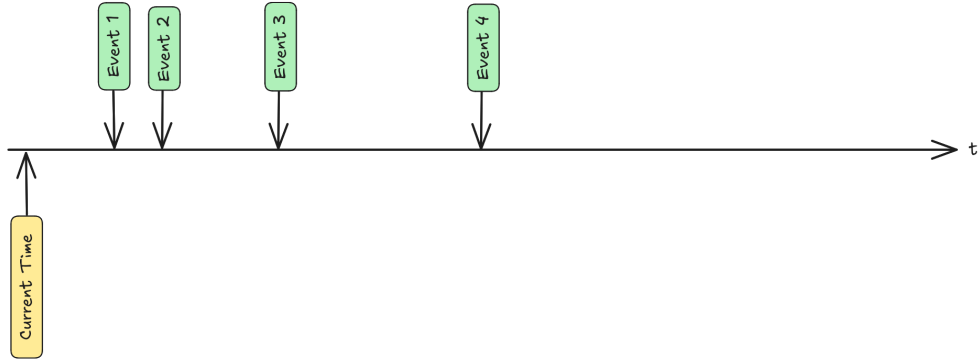


Figure 9: Initialized event queue with a few events in the future

Regardless of the event types that are scheduled (Figure 9) within the event queue, the simulator pops out the earliest event in the queue, updates the simulation time to the time that the event was to be executed at and performs handling operations depending on the event type (Figure 10).

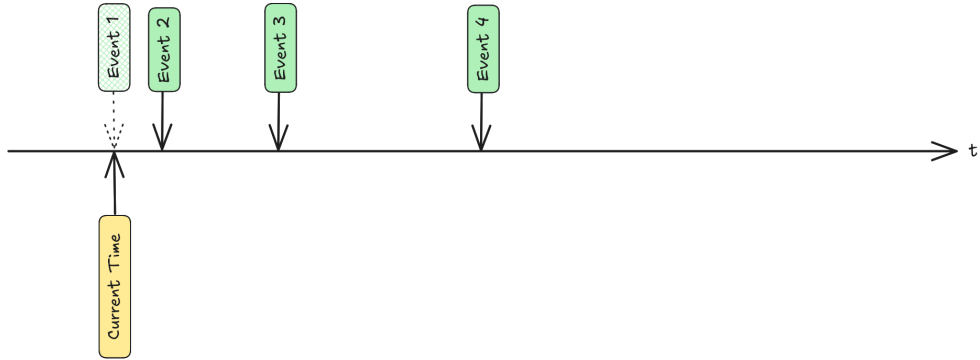


Figure 10: Single step simulation advancement - an event is removed from the queue, the simulation time advanced to the time of the event.

3.1.2. Problems in Slimcoin Proof Of Burn implementation

Before defining the events and how they are handled, we must address the several issues we encountered when evolving the Slimcoin Proof of Burn simulation to address other scenarios. These issues are not problematic within the context of the Slimcoin blockchain, as a Proof of Work scheme backs it and enforces Proof of Work blocks being mined between Proof of Burn blocks. As we attempt to ease these restrictions, we encounter the following issues.

Considering how Slimcoin defines their Proof of Burn hashing, which can be seen in Equation 11, we can notice two issues with the implementation. The Burn Hash is generated by multiplying the result of the hashing function H by the result of the multiplier function M . The first problem arises because the hashing function H only depends on the state of the blockchain at that specific time -

the output of H will be precisely the same for all nodes attempting to mine a new block in the same fork. It will be different across forks, but all nodes will generate the same hash within the fork. As the multiplier function M can only increase (or decrease) the value generated by H , not change it outright, nodes with burn transactions that were burned near in time would manage to create blocks, which would create as many forks as well.

$$\begin{aligned} \text{Burn Hash} = \\ M(\text{Blockchain}, \text{Burn Transaction}) * \\ H(\text{Blockchain}, \text{Burn Transaction}) \end{aligned} \tag{12}$$

To alleviate this problem, we adjust the H function to take in the burn transaction in addition to the state of the blockchain, salting the function and allowing each hash generated by each burn transaction to be unique (Equation 12). This allows the hashing success to be more evenly distributed across the nodes participating in the network.

The second issue is that an H function will only produce a singular value for a single blockchain state, even if salting is enabled. This means none might produce a valid Proof of Burn block across all nodes and burn transactions. Without salting, the issue is even worse, as if H generates a hash that is lower than the difficulty with $M(\text{Blockchain}, \text{Burn Transaction}) = 1$, all nodes automatically fail to mine a Proof of Burn block.

For Slimcoin, this is not a problematic situation, as if no one finds a hash for a Proof of Burn block, they can all switch to the Proof of Work layer and mine a new block that way. With the new block mined, the state of the blockchain changes and all nodes can reattempt mining of a Proof of Burn block. We are blocked as we are trying to see whether to move to a consensus layer that only relies on Proof of Burn. If we do not have Proof of Work blocks to back us up, the likelihood of mining a block across all nodes and all burn transactions is < 1 ; the blockchain will get stuck eventually. We will demonstrate this in effect with one of the simulation scenarios, where the blockchain network gets stuck.

There are a few ways that we could consider alleviating the issue. If we were to make the burn transaction price adjustment into a separate layer, we would not have to have new blocks mined in the main chain to update; we could keep lowering the price, allowing new people to join the network

cheaply, with new burn transactions. This solution would not work, as burn transactions must pass some settlement time before the network accepts Proof of Burn blocks mined using those burn transactions. Although there could be a possibility where an exception to the rule could be made, actually implementing it in a cryptographically safe way is not very likely.

Another way, and we have chosen to simulate it, is to extend the hashing function to accept a nonce (Equation 13).

$$\begin{aligned} \text{Burn Hash} = \\ M(\text{Blockchain}, \text{Burn Transaction}) * \\ H(\text{Blockchain}, \text{Burn Transaction}, \text{Nonce}) \end{aligned} \quad (13)$$

If we allow the node to generate the nonce and do not restrict how often a nonce can be regenerated, we arrive back at Proof of Work implementation. This shows us that, fundamentally, Proof of Burn and Proof of Work are closely related in their working principles. The idea that Proof of Burn uses “virtual” hardware accurately describes its working principles. Unfortunately, we cannot allow a nonce to be generated in the same manner as Proof of Work by the node itself, as any potential wins from a Proof of Burn system suddenly are lost, and we would need to employ a different approach.

For this simulation and testing scenario, we will assume we can generate pseudo-random numbers across the network securely and that we will generate this random nonce every minute.

While we are simulating the scenario, we can assume that this is possible, but we would like to justify that there is a viable way to do this. Additionally, we must do this, as it will allow us to measure any potential power consumption aspects, based on which solution we choose.

We could look at the RANDAO implementation that Ethereum uses (randao, 2023). There would be two ways of approaching this: either hook up our blockchain integration into the Ethereum Smart Contracts to provide us with the random numbers across the entire blockchain network, or implement it into the blockchain protocol itself without reliance on any external systems. Excluding the option for external blockchain dependency, we are left with internal implementation. Sadly, this would not work directly on our blockchain, as RANDAO would not be able to generate pseudo-random numbers in between blocks unless we launched another parallel blockchain that would only be focused on

generating pseudo-random numbers. While this can be done, RANDAO cannot guarantee that a pseudo-random number will be generated each minute, so a different method must be used.

Drand (Randamu, 2025), as described in the specification, provides a promising basis for our pseudo-random number generation requirements. The system works similarly to a parallel blockchain, as the specification describes - a delegated subset of nodes is selected to generate the random numbers. The solution allows for configuring an emittance period - how often should the network generate a new pseudo-random number? While drand does not require all the nodes within the network to be delegated (although there are requirements for the majority of the delegated nodes to be participating), in our simulation, we will assume that every node within our network participates in the pseudo-random number generation, thus introducing power consumption overhead. This also introduces another security assumption - a sufficient number of nodes remain honest and well-synchronised. This leads us to introduce additional event types into the simulation, which will be described in the next section.

3.1.3. Simulator Events

The simulator is capable of simulating four types of events within the simulation:

- Block Mined
- Block Received
- Pseudo Random Number Emitted
- Pseudo Random Number Received

Both block mined (Listing 3) and pseudo-random number emitted (Listing 4) events are straightforward.

The block mined event (Listing 3) simulates a node finishing the work with the block. The event is handled by sending a signal to the consensus layer to adjust required data, like counting the time spent mining the blocks or the number of blocks mined. Then, block received events with the network layer propagation delays are scheduled for all other nodes within the network.

```

1  {
2    "minedBy": 1,
3    "scheduledAt": 1.3494,
4    "dispatchAt": 503.3459,
5    "block": {
6      "id": 1
7      "minedBy": 1,
8      "depth": 1,
9      "transactions": 1093,
10     "startedAt": 1.4394,
11     "finishedAt": 503.3459,
12     "consensus": "Proof of Work",
13   },
14   "previousBlock": {
15     "id": 0
16     "minedBy": -1,
17     "depth": 0,
18     "transactions": 0,
19     "startedAt": 0.0000,
20     "finishedAt": 0.0000,
21     "consensus": "Genesis",
22   },
23 }

```

Listing 3: An instance of a block mined event.

Pseudo-random number emitted events (Listing 4) are even simpler, as they simulate the entire network finishing the generation of pseudo-random numbers using drand (Randamu, 2025). During handling, it schedules a new emittance event based on the configured delay and schedules pseudo-random number received events for all the nodes within the network.

```

1  {
2    "tick": 243,
3    "delay": 1.0000,
4    "dispatchAt": 244.0000
5  }

```

Listing 4: An instance of pseudo random number emitted event.

```

1  {
2    "receivedBy": 1,
3    "scheduledAt": 503.3459,
4    "dispatchAt": 504.0230,
5    "block": {
6      "id": 1
7      "minedBy": 1,
8      "depth": 1,
9      "transactions": 1093,
10     "startedAt": 1.4394,
11     "finishedAt": 503.3459,
12     "consensus": "Proof of Work",
13   },
14   "previousBlock": {
15     "id": 0
16     "minedBy": -1,
17     "depth": 0,
18     "transactions": 0,
19     "startedAt": 0.0000,
20     "finishedAt": 0.0000,
21     "consensus": "Genesis",
22   },
23 }

```

Listing 5: An instance of a block mined event.

The block received event (Listing 5) is handled more flexibly. We check if receiving a block would make the node reorganise its local blockchain. Receiving a block with the same last block as we do automatically triggers a reorganisation. As the network layer in the simulation allows for block reception delays, blockchain forks are an emergent property of the simulation, as two blocks mined events might be handled close in time, before any of the block reception events scheduled from these block mined events are ever dispatched. Because of this, the simulator must implement a method for resolving forks. Our simulator uses a fork resolution threshold, which defines how far back from the received block we must be before accepting the incoming fork.

The reorganisation procedure is relatively simple; nodes synchronise consensus information between nodes, including total processing time, epoch indexes, and other properties, depending on the consensus layer implementation. Then, the currently mined block is marked as abandoned, the block mined event is removed from the event queue, and a new block mined event is scheduled, with the

received block specified as the previous block. The abandoned block is written into the database, as the work was performed for that particular block, and we will have to take abandoned blocks into account as well.

```
1 {  
2   "tick": 243,  
3   "receivedBy": 2,  
4   "dispatchAt": 246.2431  
5 }
```



Listing 6: An instance of pseudo random number receive event.

Lastly, the pseudo-random number receive event (Listing 6) is handled and only pertains to the improved Proof of Burn implementation we are testing. It simply checks whether a node can mine a block. If so, it schedules a new block mined event into the queue.

3.2. Consensus Simulation Scenarios and Hypotheses

While the network and event queue layers stay roughly the same across different consensus implementations, the consensus implementations differ significantly. Each scenario has some configuration level, while specific settings are standard to whichever consensus algorithm is used in a scenario (Listing 7).

```
1 node_count: 65000  
2 simulation_time_in_seconds: 31622400  
3  
4 average_network_latency_in_seconds: 6  
5 chain_reorganization_threshold: 6  
6  
7 average_transactions_per_second: 6  
8 maximum_transaction_per_block: 3000
```



Listing 7: Scenario settings common across all consensus implementations

For a simulation, we try to simulate a year, with the currently existing Bitcoin node count (Bitnodes.io, 2025). While not impossible, it is improbable to run specific scenarios to the very end of the simulation using these defaults due to some emergent properties within the simulation space, which make the simulation advance in tiny time steps or explode in computational environments' memory consumption. As a result, some scenarios run for days or even months without providing

interesting data. We will note any deviations from the standard settings within the scenarios if they appear.

We have chosen an average network latency of around 6 seconds, which was identified as the average latency within the Bitcoin network in 2020 by (Nagayama et al., 2020).

The Chain Reorganisation Threshold is set to six, following the security recommendations for the Bitcoin network (Guo & Ren, 2022). This configuration permits forks to persist longer before being discarded by nodes.

The average transaction per second and maximum transactions per block settings are based on current Bitcoin transaction data and block size limits (Blockchain.com, 2025). In a regular Bitcoin simulation scenario, the blockchain should not become saturated by transactions.

3.2.1. Bitcoin Scenario

The Bitcoin simulation scenario has the settings (Listing 8) based on the actual settings in the Bitcoin blockchain, where the average block frequency is set to be around 10 minutes, and the epoch time for difficulty adjustment is set to be around two weeks, if the average frequency is around 10 minutes.

```
1 proof_of_work:
2   enabled: true
3   epoch_length: 2016
4   average_block_frequency_in_seconds: 600
```

YAML

Listing 8: Scenario settings for Bitcoin implementation simulation

The difficulty adjustment for the consensus layer is done by aggregating the total time spent on mining within the fork. If the average is lower than the expected average, the difficulty increases; if the average is higher - difficulty decreases. The change in difficulty is bounded - it cannot change by a larger multiplier or divisor than 4.

In this scenario, we expect to see the difficulty stabilise over a couple of weeks and not change much across time. As we are not simulating nodes entering or exiting the network or existing nodes changing their hardware, we do not expect much deviation from the results.

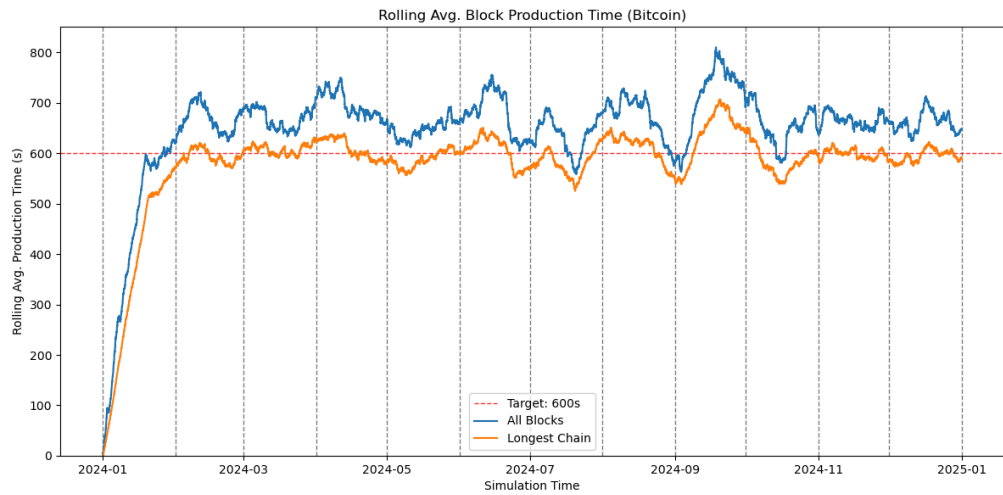


Figure 11: Rolling average block production time in Bitcoin

As we can see in Figure 11, the network starts with a difficulty of 1, which means that anyone attempting to mine a block will be able to do so. As such, the first epochs are really quick, and the difficulty gets adjusted rather quickly as well, eventually stabilizing around 600 seconds per block average.

The Figure 11 also shows the difference between all blocks and only blocks that were part of the longest chain. On average, this indicates that the presence of forks increases the adequate processing time for blocks within the main chain. This outcome is expected, as nodes may continue mining their blocks even after another node has already found a valid block in the typical expected time.

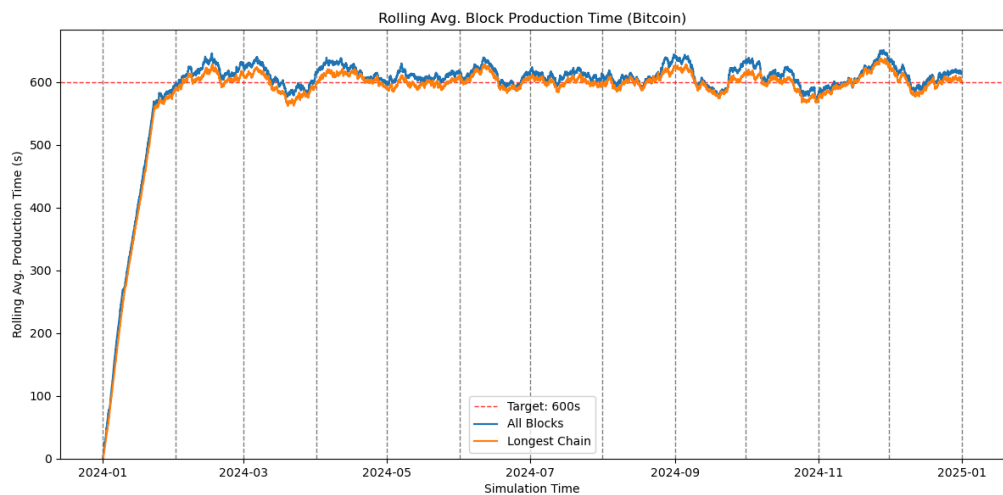


Figure 12: Rolling average block production time in Bitcoin (reduced reorganization threshold)

By reducing the reorganisation threshold from 6 to 2, we can see (Figure 12) that the average block production time between the longest branch and across all branches gets closer.

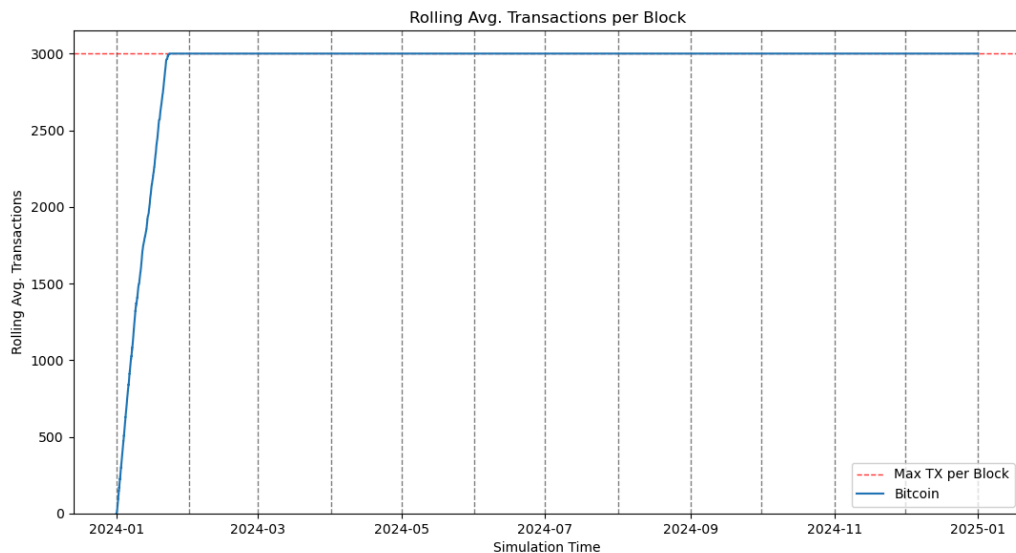


Figure 13: Average transactions per block – Bitcoin

Figure 13 shows a parameter choice we have made, to have the capacity to illustrate changes with other scenarios - we have chosen an average transaction per second count in such a way that it would saturate the capabilities within the Bitcoin blockchain. As it currently stands, the Bitcoin network does not get saturated by transactions to such an extent that there is a transaction backlog, but this will allow us to demonstrate any potential improvements to this metric in other scenarios.

3.2.2. Slimcoin Scenario

The Slimcoin simulation scenario's settings (Listing 8) are mostly based on the actual settings used in the Slimcoin blockchain. The original Proof of Work settings stay, and the Slimcoin implementation runs both the Proof of Work and Proof of Burn consensus layers.

```

1 proof_of_work:
2   enabled: true
3   epoch_length: 2016
4   average_block_frequency_in_seconds: 600
5
6 slimcoin_proof_of_burn:
7   enabled: true
8   interval: 3

```

YAML

Listing 9: Scenario settings for Slimcoin 3:1 implementation simulation

The interesting part about Slimcoin is that while the difficulty can change, the frequency of Proof of Burn blocks is based on the interval - how many Proof of Work blocks must be mined before a Proof of Burn block can be mined. As a result, the minimum possible interval allows for one Proof of Burn

block for every Proof of Work block, representing the fastest achievable ratio. In our scenarios, we will simulate a ratio of 1:1, 1:2, 1:3 and 1:100 as an outlier, to see if with blocks as rare as that, it will provide any insights.

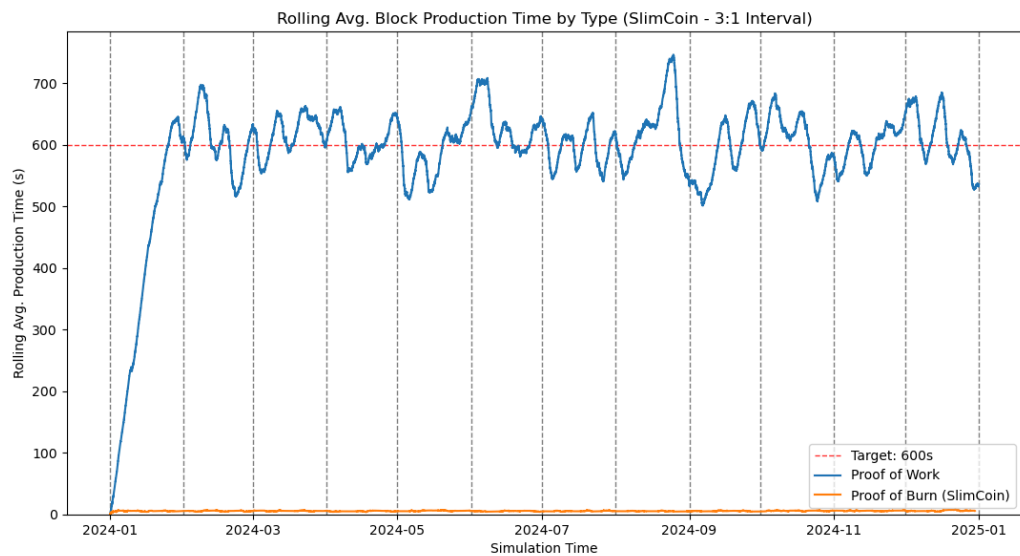


Figure 14: Average production time by block type – Slimcoin

Looking at the production times for Slimcoin in a 3:1 scenario (Figure 14) we can see that the Proof of Work layer still stabilises at around 600 seconds per block, while the Proof of Burn layer shows very short production time - calculating a single hash does not take a large amount of time.

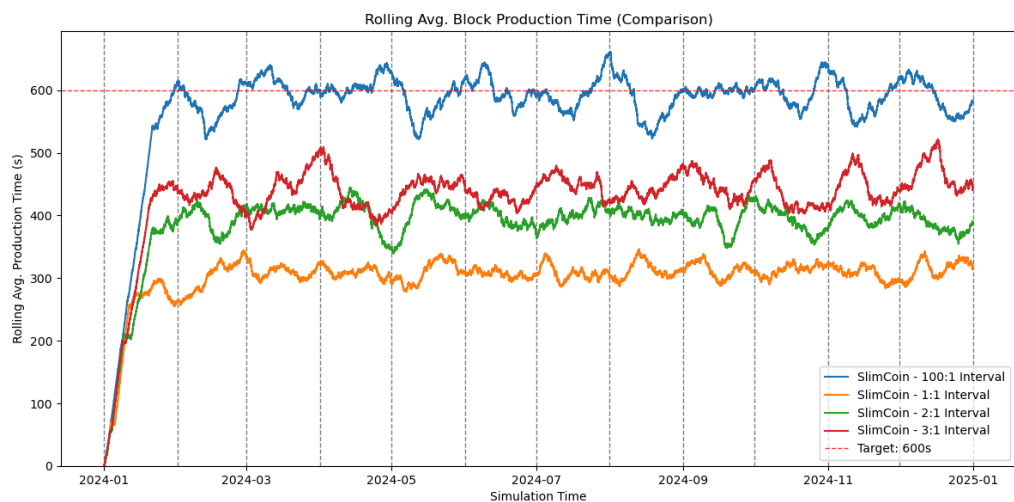


Figure 15: Rolling average production time comparison – Slimcoin

When looking at the collective production times, where we do not separate the time to mine Proof of Work blocks against Proof of Burn blocks, we can see that, on average, the times are reduced. In a 1:1 configuration, we can see the average going to around 300, which is half the expected production time. This outcome is logical - if we mine a block for 600 seconds on average, and the next block is

mined instantly, we expect the production time to be lowered to around half. The same can be seen for the other configurations, where the 2:1 configuration reduces to around 400, the 3:1 configuration to around 450, and the 100 to 1 does not see a significant change in processing time.

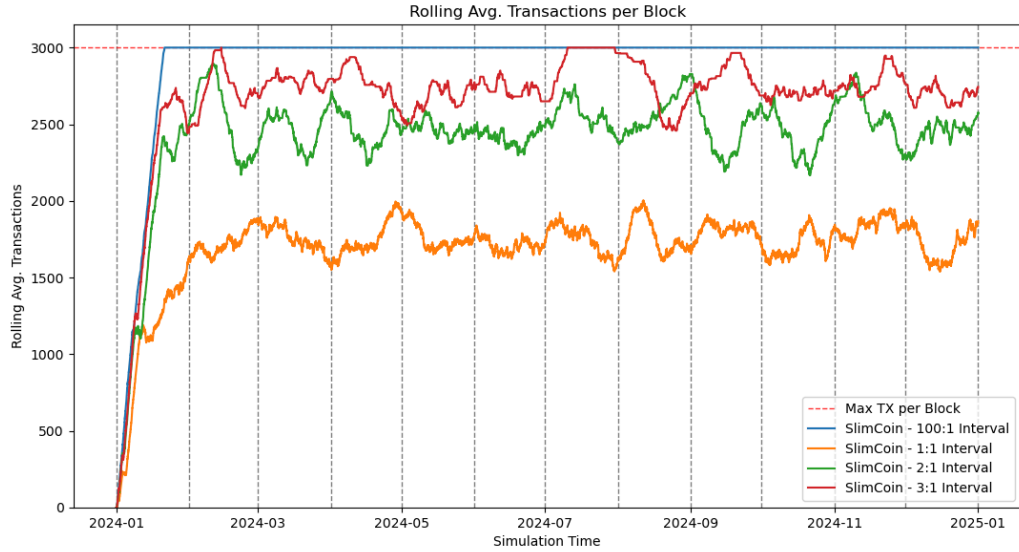


Figure 16: Average transactions per block – Slimcoin

Looking at the transactional averages per block (Figure 16), we can see that introducing a small change allows us to increase our transactional output per unit time. While a 3 to 1 configuration still gets saturated occasionally, the 1 to 1 configuration drops the average to higher than half. This is also expected, as the average transaction per second setting was chosen to saturate the transactions per block barely.

3.2.3. Unbounded Slimcoin Scenario

In the previous scenario, we were bound by Slimcoin’s ruleset. We had to have at least a single Proof of Work block before we could mine a Proof of Burn block. In this case, we have modified the consensus layer to allow for Proof of Burn blocks to be mined, even if no Proof of Work block was mined before (Listing 10).

```

1 proof_of_work:
2   enabled: true
3   epoch_length: 2016
4   average_block_frequency_in_seconds: 600
5
6 razer_proof_of_burn:
7   enabled: true
8   interval: 0.2

```

YAML

Listing 10: Scenario settings for unbounded Slimcoin implementation simulation

This also allows us to attempt a configuration where the Proof of Work layer is fully disabled and only the Proof of Burn layer works (Listing 11). In this scenario, we have chosen settings so that all nodes across the network would have a minimal likelihood of failing, but it would not be 0 for that likelihood.

```

1 proof_of_work:
2   enabled: false
3
4 razer_proof_of_burn:
5   enabled: true
6   interval: 0.7

```

YAML

Listing 11: Scenario settings for pure Proof of Burn implementation simulation

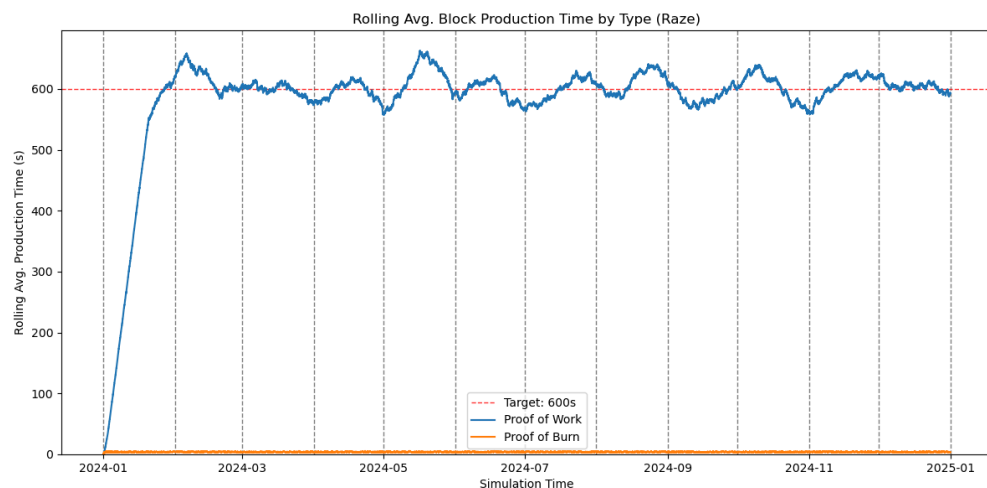


Figure 17: Typed block production statistics – Raze

Unsurprisingly, the results are similar (Figure 17) to regular SlimCoin intervals - we can see the Proof of Work layer stabilising at around 600 seconds per block, while the Proof of Burn layer stays very low, at seconds per block mined.

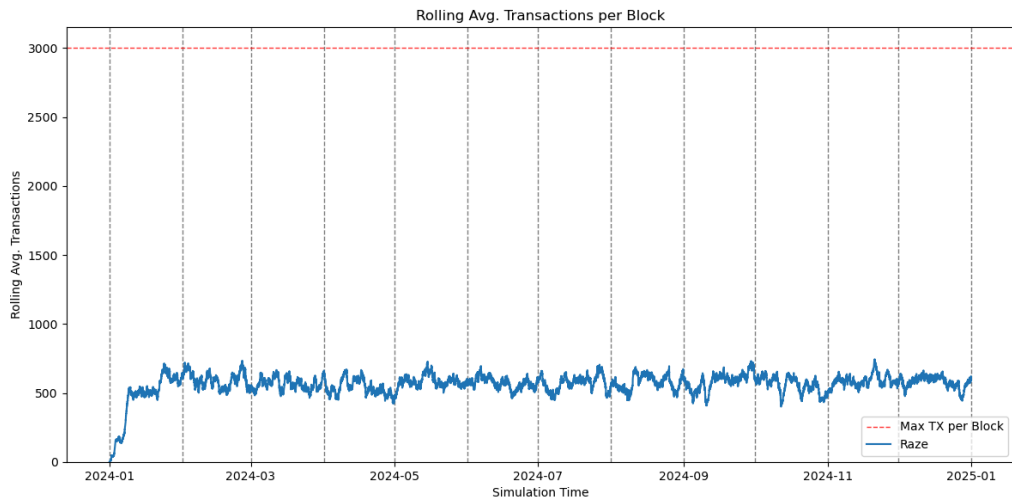


Figure 18: Average transactions per block – Raze

We can see that the average transactions per block drop, as observed with Slimcoin implementation (Figure 18).

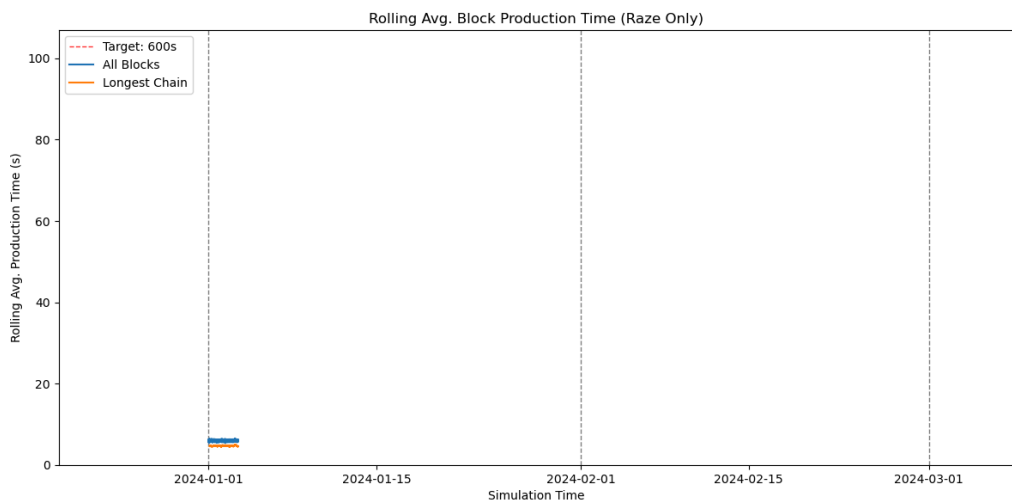


Figure 19: Average transactions per block – Raze Only

We want to see whether Proof of Burn can operate well without the support of other consensus layers. As the hard limitation for Proof of Burn to be after a Proof of Work block is removed, we can launch a scenario where the Proof of Work layer is disabled. We can see the results in Figure 19, and while not extensive, they are rather descriptive. As we can see, the block mining time, just like in the cases before, hovers around 1 second. Moreover, the simulation itself did not go past several days before we reached a situation where no single node within the network managed to mine a block. Furthermore, as nodes within the network only have one attempt per burn transaction until the blockchain state changes, we reach a stalemate and the simulation stops.

These results indicate that a different mechanism must be introduced, as discussed in the previous section (no randomization).

3.2.4. Pricing Controlled Proof of Burn Scenario

The idea behind this scenario is to lock the difficulty of the blockchain network to some static value and attempt to adjust the difficulty by using the price of a Burn Transaction. While previous scenarios simulated these at a high level, in this scenario, we simulate participation in Burn Transactions, budgets and choices in detail.

```
1  proof_of_work:
2    enabled: true
3    epoch_length: 2016
4    average_block_frequency_in_seconds: 600
5
6  pricing_proof_of_burn:
7    enabled: true
8    epoch_length: 1008
9
10   average_block_frequency_in_seconds: 600
11
12   average_price: 100
13
14   settled_period: 10
15   working_period: 1200
```

Listing 12: Scenario settings for Pricing controlled implementation simulation

As such, comparatively, the settings for Pricing simulation (Listing 12) are more complex. The average price does not correspond to any currency and represents some monetary value within the system. We have chosen to forgo the rewards layer from the original BlockSim (Alharby & Moorsel, 2020) as providing actual priced rewards, reusing those rewards in the burn transaction simulation, finding the right balance between burn transaction prices and the rewards is a vastly complex task that deserves its own dedication in a thesis work. For our purposes, we have chosen a more straightforward approach, to have the capacity to illustrate that purely a Proof of Burn layer could work within the restrictions that we have set up.

Each burn transaction has a settlement period, which is how deep the next mined block must be after the block where the burn transaction was created for it to be able to mine Proof of Burn blocks.

We have set this period to be ten, larger than the expected fork resolution threshold of 6. Generally, settlement periods should be around or larger than the expected fork resolution threshold to ensure that nodes cannot abuse the provided power.

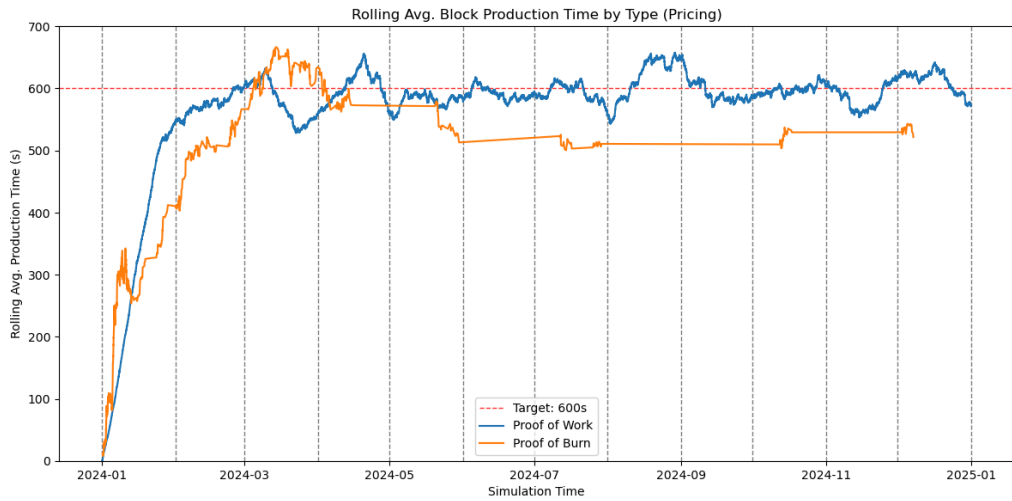


Figure 20: Average production time by block type – Pricing (no randomization)

Our simulation environment schedules all nodes initially and, with this consensus layer enabled, creates burn transactions based on the allocated budget for each node. As the working period would be the same for everyone, the pricing difficulty would stagnate for those working periods, which is what we can see in Figure 20. The block production times have long periods that are highly stable, because all nodes have created burn transactions around the same time.

We fix this by adding some randomisation to the working period. This adjustment simulates more realistic node behaviour - for example, individual nodes might abandon their burn transactions early in favour of submitting cheaper (or more expensive) ones. It also accounts for the likelihood that burn transaction purchases would naturally be more staggered across the network, rather than coinciding.

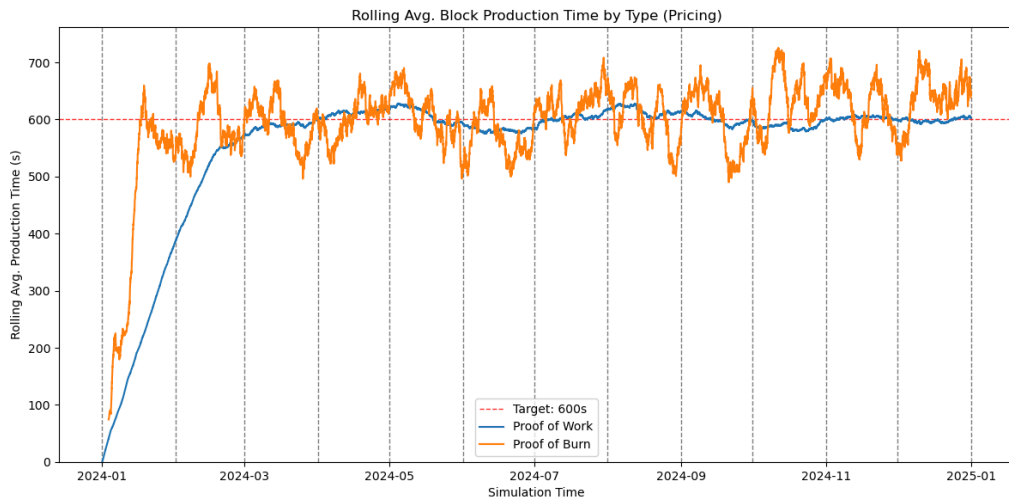


Figure 21: Average production time by block type – Pricing

To maintain decentralization, each node agrees upon the price based on a set of procedures.

The core adjustment process happens across the epoch. Once the configured number of blocks has been mined, the price is adjusted. The adjustment is limited by the same ruleset as Proof of Work implementation - the multiplier is bounded to four sizes, increase or decrease.

The additional price adjustment is present if the working period of the burn transactions is lower than the epoch length. While this is not advisable, it produces a situation where the price might be too large for the nodes to fit within the budget, and because the working period for burn transactions is lower than the epoch size, the price might not be adjusted, and the consensus layer stalls. This adjustment happens if an epoch worth of Proof of Work blocks have been mined, without seeing a Proof of Burn block. This situation indicates that no nodes are willing to buy burn transactions at the current price, prompting a price reduction. This works only if there is a Proof of Work layer backing up the blockchain. If we only leave the Proof of Burn layer running, we cannot allow a lower working period than the epoch.

With the simulation set up like that, we can see in Figure 21 that it. It achieves a semi-stable block mining time, even though it spikes up and down significantly more than Bitcoin (Figure 11).

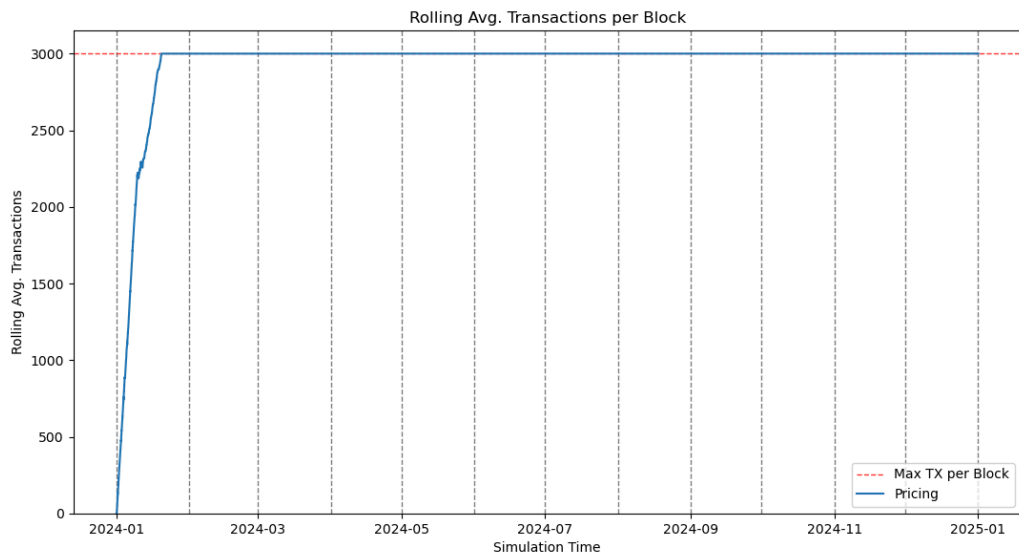


Figure 22: Average transactions per block – Pricing

When looking at the SlimCoin implementation, we might assume that the transaction capacity of the Pricing Proof of Burn implementation would be higher, since additional blocks are introduced. However, that is not the case, as shown in Figure 22. This outcome is expected, considering that the Pricing Proof of Burn consensus layer does not accept blocks mined within 1 second, even if the difficulty is satisfied. In this setup, blocks can technically be mined within 1 second, because only one hashing function is required per existing burn transaction; they are only accepted at a fixed interval, which in our case is 600 seconds. As a result, the transaction capacity of this implementation remains comparable to Bitcoin's, as observed in Figure 13 and Figure 22.

```

1  proof_of_work:
2    enabled: false
3
4  pricing_proof_of_burn:
5    enabled: true
6    epoch_length: 1008
7
8    average_block_frequency_in_seconds: 600
9
10   average_price: 100
11
12   settled_period: 10
13   working_period: 1200

```

YAML

Listing 13: Scenario settings for Pricing Only implementation simulation

If we disable the Proof of Work consensus layer (Listing 13), we can see that the average block production time manages to settle around the expected 600 second interval configured (Figure 23), although it is more noisy, which shows that just the Pricing Only consensus layer can work stably.

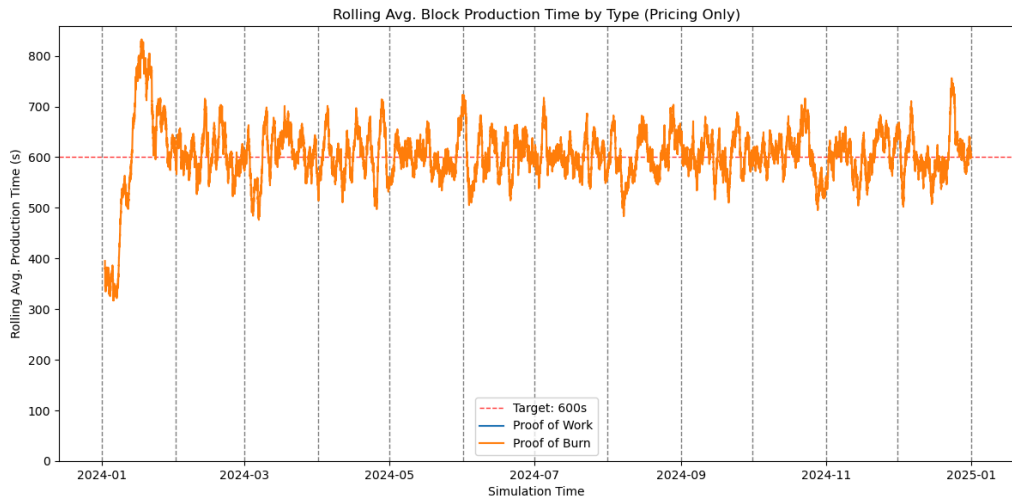


Figure 23: Average production time by block type - Pricing Only

3.3. Scenario Power Usage

For power usage calculation, we use the average Wattage for specific devices. Each node gets two parameters that determine the device's power usage: the idle power draw and the power draw during workload (e.g., mining Proof of Work block). We have chosen the value of around 3000 Watts, as if the nodes used high-power machines. As these nodes are in high-capacity loads most of the time, the idle power draw for these devices are not a high priority factor, as such, we have chosen a value of around 200 Watts, which is on higher end of idle power usage on such devices, but this allows us to provide a more level competition field between pure Proof of Work and pure Proof of Burn implementations.

To introduce variability in the simulated power, we sample each value from a distribution centred around the predefined mean. This simulates the nodes having varied power within their devices. Additionally, this power does play a role in the Proof of Work consensus layer, where it is used to determine when the block will be mined by the selected node.

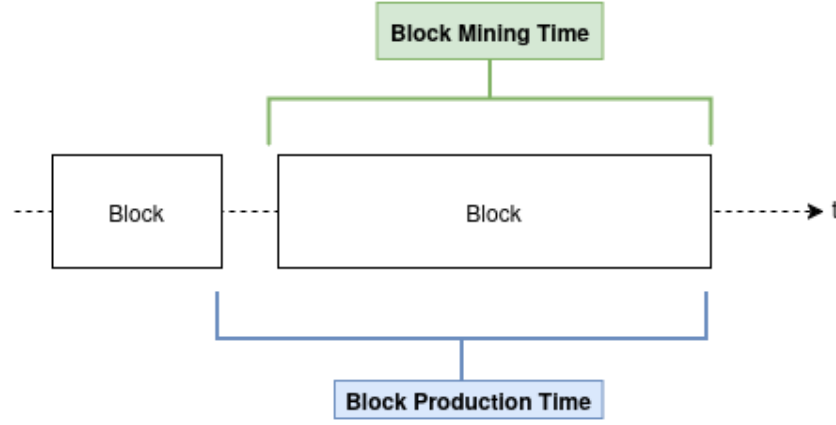


Figure 24: Difference between Block Mining Time and Block Production Time

We calculate the power usage based on the consensus layer that the block was mined in and the time that the block took to be mined and produced. A difference between block mining time and block production time must be noted. For a mined block, the time is calculated from the second that the block started to be mined, and the block was either successfully mined or abandoned due to another mined block coming in over the network. Block production time takes the absolute time from when the mined block was produced to when the previous block was produced (Figure 24). We must establish these times as for the Pricing Proof of Burn consensus layer, the block mining times are within a second range, while the production times are expected to hover around 600 seconds. In these cases, while the nodes are not actively mining for blocks, they are participating in “sub-chain” generating the random numbers to the chain; as such, we must still calculate this “idle” time into our power consumption calculations.

As we have established, the possibility of the Proof of Burn consensus layer running without the necessity of generating Proof of Work blocks once in a while in Section 3.2, we will analyse the power usage implications of using these consensus layers and will see what kinds of improvements or detriments each type of consensus layer show in power usage.

Using the numbers reported by CCAF (Cambridge Centre for Alternative Finance, 2025), we will attempt to estimate the expected power usage in our simulation. The estimated mining hardware efficiency is 22 J/TH. Given that our simulation has a power demand of 3000 Watts, we can calculate the average hashing rate using Equation 14, to get approximately 136 TH/s.

$$\text{Hashrate (TH/s)} = \frac{\text{Power (W)}}{\text{Efficiency (J/TH)}} \quad (14)$$

Having this, we can estimate the amount of hardware devices being used, using roughly 800 EH/s hashrate being reported by CoinWarz (coinwarz.com, 2025), we estimate $\frac{800 \text{ EH/s}}{136 \text{ TH/s}} \approx 5.8$ million devices used.

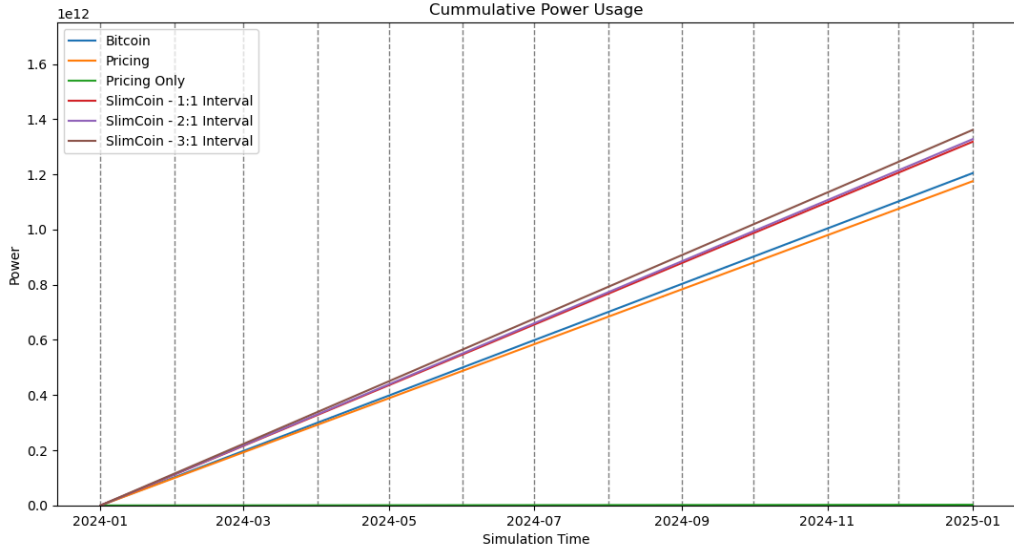


Figure 25: Cumulative power usage – across all blocks

With the annual power consumption estimated to be around 180 TWh, we can estimate the per-node consumption to be around $180 \text{ TWh} / 5.8 \approx 31 \text{ MWh}$. As can be seen in Figure 25, all scenarios where the Proof of Work consensus layer is enabled continue to consume a large amount of energy. For Bitcoin, it settles at around $1.3 * 10^{12} J$, and we can use that to estimate the average power consumption per device within our simulator to see whether it is accurate.

$$\frac{1.3 * 10^{12} J}{3600} \approx 3.6 * 10^8 \text{ Wh} = 360 \text{ MWh} \quad (15)$$

Dividing 360 MWh by 20, we get 18 MWh, which is a bit lower than the estimate across the real Bitcoin network, but it is close enough. With this, we can draw conclusions about the functionality of the other consensus layers and how they perform in terms of power demand and consumption.

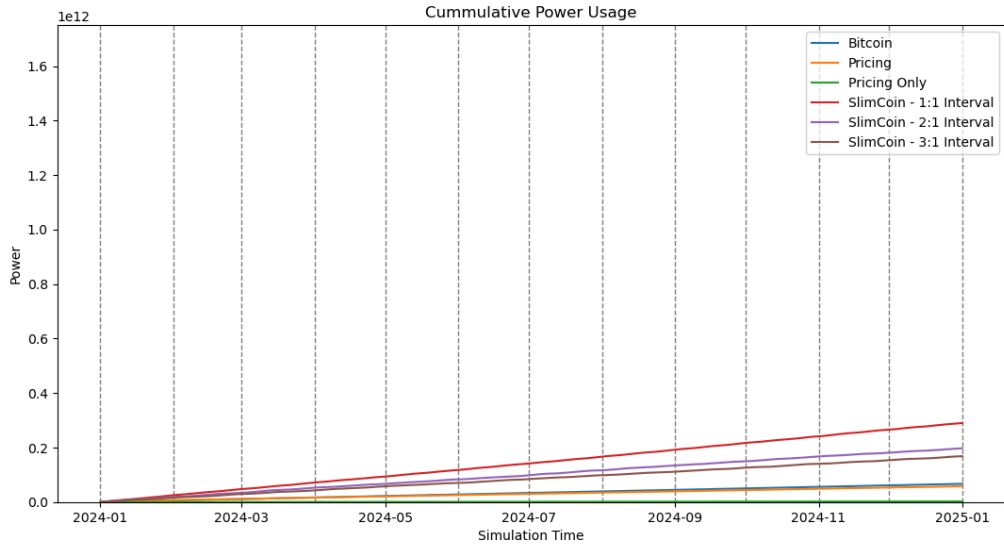


Figure 26: Cumulative power usage – across all forks

SlimCoin’s total power usage is rather interesting, as in Figure 25, we can see that a 1:1 interval configuration consumes more power than Bitcoin. The increased consumption is caused by the behaviour of the 1:1 interval, which attempts to produce a Proof of Burn block immediately after receiving a Proof of Work. If a node fails to do this, it will start with mining a Proof of Work block, but that mining time is unlikely to yield results, as another node is likely to mine a Proof of Burn block, and the time will be spent mining during that time. As seen in the fork counts in Table 1, the Slimcoin implementations with low intervals tend to accrue larger amounts of forks, resulting in a slightly larger power usage.

Table 1: Fork counts based on simulation scenario

Simulation	Forks
Bitcoin	677
Proof of Burn - Pricing	4906
Proof of Burn - Pricing Only	92
SlimCoin - 1:1 Interval	61153
SlimCoin - 2:1 Interval	30236
SlimCoin - 3:1 Interval	19633

The results for Pricing Proof of Burn look very promising. Just introducing these types of Proof of Burn blocks to the Proof of Work does not significantly change the power consumption, and it does not introduce a large number of forks in the network, as can be seen in Table 1.

Significant improvement can be seen in power consumption when only the Pricing Proof of Burn consensus algorithm runs. The total power usage is tiny, amounting to roughly the network's idle power usage.

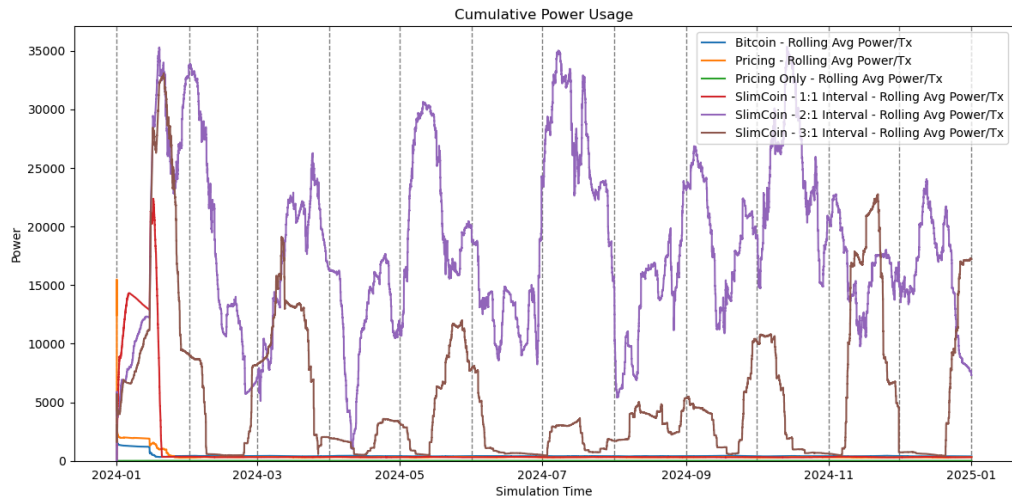


Figure 27: Power usage per transaction

Another metric we look at is the power usage per single transaction. In our results in Figure 27, we can see large spikes in the SlimCoin implementations. This behaviour aligns with our expectations, as the configuration we have chosen, we oversaturate the capacities of the Proof of Work consensus layer to handle transactions. With the introduction of SlimCoin Proof of Work into that environment, the transaction counts are no longer saturated - the network can handle all transactions appearing in the network. As a result, blocks are not filled with transactions, which increases the power usage per transaction.

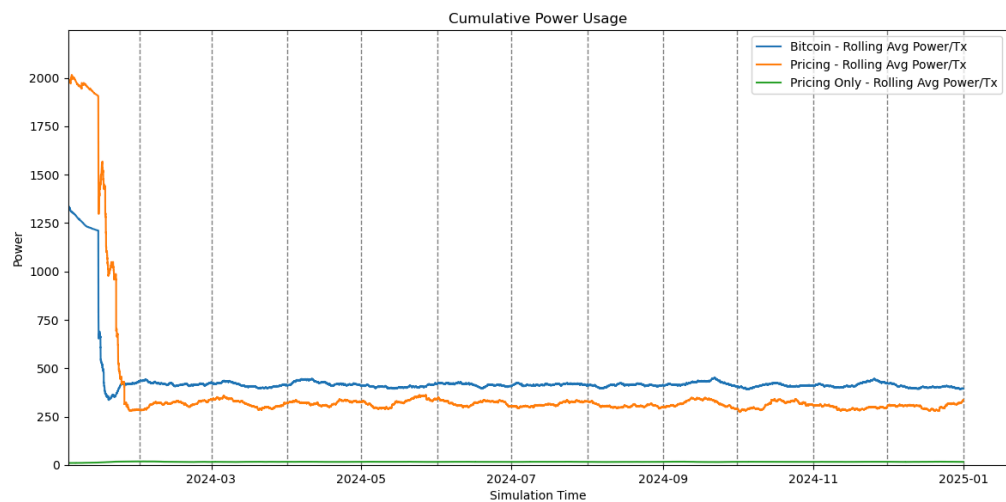


Figure 28: Power usage per transaction (reduced scenarios)

Removing the outliers, we are left with a more reasonable view of power usage per transaction in Figure 28. We can see that introducing Pricing Proof of Burn does not significantly affect power per transaction metrics, and it stays around the same, but disabling the Proof of Work layer produces the largest improvements.

3.4. Results Review

The simulator results have shown us that switching to a pure Proof of Burn implementation from Proof of Work is not a simple process.

Correlating the results of the simulator with real-world data, we have shown that the simulator works close to the expected parameters within Proof of Work. Proof of Work stabilises around 600 seconds per block and consumes roughly the expected power yearly.

Introducing the SlimCoin Proof of Burn layer shows us that we have capabilities to increase transactional throughput by introducing blocks that can be mined almost instantly, with some luck, but without providing any significant effect to the power usage across the network.

Removing the limitation that a Proof of Work block must precede any Proof of Burn blocks mined from the SlimCoin implementation showed us that the backing of Proof of Work layer is strictly necessary, as without it, the blockchain might stall and stop producing blocks.

The insight gained from removing limits in the SlimCoin implementation pushed us to improve the Proof of Burn layer, and we have introduced an inter-block pseudo random number generator across the blockchain network. While there are potential candidates for this, which covers all of our requirements, we had to make an assumption that a safe implementation is possible, as we are unable to verify this within the confines of the simulation.

Lastly, we have switched out an expected difficulty adjustment mechanism, where the difficulty is adjusted based on time spent on the epoch, with a mechanism of controlling it using the price of a burn transaction. Simulating rudimentary economic decision model of the nodes, by assigning a budget and allowing the nodes to participate in transaction burning, but penalising them for going overboard.

The results of the simulator show us that this model has potential and, just like Bitcoin layer, stabilises at producing a block every 600 seconds. As only a minuscule amount of hashing is simulated,

the network's computational power requirements and usage drop significantly. The nodes within the network still have to be powerful enough to run the sub-layer pseudo-random number generation solution. This means that the devices could be downscaled to devices that consume a minuscule amount of power while idling and under load.

4. CONCLUSION

In this thesis, we set out to investigate whether we could use interoperability between less common consensus algorithms, specifically Proof of Burn, to reduce energy demands of blockchain consensus mechanisms without compromising decentralisation.

1. We analysed Proof of Work, Proof of Stake and Proof of Burn algorithms in real-world systems, namely Bitcoin, Ethereum and Slimcoin, focusing on environmental impact, scalability and decentralisation. This analysis helped identify potential improvements and informed the simulator design.

2. A scalable, event-driven blockchain simulator was developed, supporting multiple consensus algorithms, pseudo-random number generation and detailed data export. Over 40GB of simulation data was generated across all simulator runs.

3. The simulation methodology and scenarios were clearly defined, with expected outcomes grounded in real-world behaviour. Aligning Proof of Work-based simulation with real Bitcoin data confirmed the simulator's validity.

4. Several scenarios were tested, including:

- Pure Proof of Work (Bitcoin style)
- Hybrid Proof of Work / Proof of Burn (SlimCoin style)
- Unbounded Proof of Burn, without Proof of Work fallback
- A novel price-controlled Proof of Burn mechanism

The results have shown that the price-controlled Proof of Burn mechanism is feasible and energy-efficient alternative to Proof of Work, with a possible gradual adoption path.

4.1. Limitations

The following points are critical to the interpretation of this research. Omitting them may lead to conclusions that overlook key assumptions made during the study:

1. While we assume a secure and decentralised pseudo-random number generation system to be available (e.g. drand), a thorough verification of its implementation in the context of Proof of Burn has not been performed and remains a future research area.
2. Full economic modelling of the burn transaction pricing was outside the scope of this thesis. However, as it is an essential component of Proof of Burn systems, further research is necessary to determine whether burn pricing mechanisms can reliably balance network incentives. Any implementation of Proof of Burn would have to have some semblance of balance between these aspects, as burn transactions are inherently tied with the pricing — if power is fixed, the price is controlled by the consensus layer; if the market determines the price, the power is controlled by the amount spent.

BIBLIOGRAPHY

- Aldweesh, A. (2023). BlockTicket: A framework for electronic tickets based on smart contract. *Plos One*, 18(4), e284166.
- Alharby, M., & Moorsel, A. van. (2020). Blocksime: An extensible simulation tool for blockchain systems. *Frontiers in Blockchain*, 3, 28.
- Alharby, M., & Moorsel, A. van. (2022, March). *BlockSim: blockchain simulator*. <https://github.com/maher243/BlockSim>
- Alshater, M. M., Nasrallah, N., Khoury, R., & Joshipura, M. (2024). Deciphering the world of NFTs: a scholarly review of trends, challenges, and opportunities. *Electronic Commerce Research*, 1–57.
- Arner, D. W., Auer, R., & Frost, J. (2020). *Stablecoins: risks, potential and regulation*.
- Arora, A., Kanisk, & Kumar, S. (2022). Smart contracts and NFTs: non-fungible tokens as a core component of blockchain to be used as collectibles. *Cyber Security and Digital Forensics: Proceedings of ICCSDF 2021*, 401–422.

- Bayer, D., Haber, S., & Stornetta, W. S. (1993). Improving the efficiency and reliability of digital time-stamping. *Sequences II: Methods in Communication, Security, And Computer Science*, 329–334.
- Bhaskara, A., Frongillo, R., & Papireddygar, M. (2023). A General Theory of Liquidity Provisioning for Automated Market Makers. *Arxiv Preprint Arxiv:2311.08725*.
- BITMAIN Technologies. (2025, January). *Understanding ASIC Miners: The Driving Force Behind Cryptocurrency Mining*. <https://bitmain.com.vc/understanding-asic-miners-the-driving-force-behind-cryptocurrency-mining>
- Bitnodes.io. (2025, May). *Bitnodes*. <https://bitnodes.io/nodes/all/>
- Blockchain.com. (2025,). *Transaction Rate Per Second*. <https://www.blockchain.com/charts/transactions-per-second>
- Bublyk, Y., Borzenko, O., & Hlazova, A. (2023). Cryptocurrency energy consumption: Analysis, global trends and interaction. *Environmental Economics*, 14(2), 49.
- Buterin, V., & others. (2013). Ethereum white paper. *Github Repository*, 1, 22–23.
- Cambridge Centre for Alternative Finance. (2025,). *Cambridge Bitcoin Electricity Consumption Index (CBECI)*. <https://ccaf.io/cbnsi/cbeci>
- Chaum, D. L. (1979). *Computer Systems established, maintained and trusted by mutually suspicious groups*. Electronics Research Laboratory, University of California Riverside, CA, USA.
- coinwarz.com. (2025,). *Bitcoin Hashrate Chart*. <https://www.coinwarz.com/mining/bitcoin/hashrate-chart>
- David Szabados, S. (2010, May). *Diving into MTBF and AFR: Storage Reliability Specs Explained*. <https://web.archive.org/web/20100501151901/http://enterprise.media.seagate.com/2010/04/inside-it-storage/diving-into-mtbf-and-afr-storage-reliability-specs-explained>
- De Vries, A., & Stoll, C. (2021). Bitcoin's growing e-waste problem. *Resources, Conservation and Recycling*, 175, 105901.
- Ethereum Foundation. (2024c, July). *How to Stake Ethereum*. <https://ethereum.org/en/staking/>
- Ethereum Foundation. (2024b, July). *The Beacon Chain*. <https://ethereum.org/en/roadmap/beacon-chain/>

- Ethereum Foundation. (2024a, July). *The Merge*. <https://ethereum.org/en/roadmap/merge/>
- Ethereum Foundation. (2025b, January). *Ethereum JSON-RPC Specification*. <https://ethereum.github.io/execution-apis/api-documentation/>
- Ethereum Foundation. (2025a, January). *Smart Contracts Documentation*. <https://ethereum.org/en/developers/docs/smart-contracts/>
- fiveoutofnine. (2022, February 18). *fiveoutofnine*. <https://www.fiveoutofnine.com/>
- Guo, D., & Ren, L. (2022). Bitcoin's latency–security analysis made simple. *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, 244–253.
- Haber, S., & Stornetta, W. S. (1991). *How to time-stamp a digital document*. Springer.
- Han, R., Foutris, N., & Kotselidis, C. (2019). Demystifying crypto-mining: Analysis and optimizations of memory-hard pow algorithms. *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 22–33.
- Ho, M.-H., Ai, Y.-Q., Chau, T. C.-P., Yuen, S. C., Choy, C.-S., Leong, P. H., & Pun, K.-P. (2012). Architecture and design flow for a highly efficient structured ASIC. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(3), 424–433.
- Investing.com. (2025, January). *Ethereum to Euro Historical Data*. <https://www.investing.com/crypto/ethereum/eth-eur-historical-data>
- Jakobsson, M., & Juels, A. (1999). Proofs of work and bread pudding protocols. *Secure Information Networks: Communications and Multimedia Security IFIP Tc6/tc11 Joint Working Conference on Communications and Multimedia Security (Cms'99) September 20–21, 1999, Leuven, Belgium*, 258–272.
- Joseph C. (2020, April). *The Beacon Chain Ethereum 2.0 explainer you need to read first*. <https://ethos.dev/beacon-chain>
- Kapengut, E., & Mizrach, B. (2023). An event study of the ethereum transition to proof-of-stake. *Commodities*, 2(2), 96–110.
- King, S., & Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *Self-Published Paper, August*, 19(1).

- Krellenstein, A. (2024, April). *Counterparty: An Extension to Bitcoin by State-Machine Replication*.
https://krellenstein.com/adam/get/counterparty-whitepaper_2024-04-25.pdf
- Liang, H., Dong, Y., Ding, Z., Ureña, R., Chiclana, F., & Herrera-Viedma, E. (2019). Consensus reaching with time constraints and minimum adjustments in group with bounded confidence effects. *IEEE Transactions on Fuzzy Systems*, 28(10), 2466–2479.
- Nagayama, R., Banno, R., & Shudo, K. (2020). Identifying impacts of protocol and internet development on the bitcoin network. *2020 IEEE Symposium on Computers and Communications (ISCC)*, 1–6.
- Nakamoto, S., & others. (2008). Bitcoin. *A Peer-to-Peer Electronic Cash System*, 21260.
- Noh, J., Kwon, D., Cho, S., & Yiu, N. C. (2023). Network Participation and Accessibility of Proof-of-Stake (PoS) Blockchains: A Cross-platform Comparative Analysis. *Arxiv Preprint Arxiv:2305.13259*.
- Oliveira, S., Soares, F., Flach, G., Johann, M., & Reis, R. (2012). Building a bitcoin miner on an FPGA. *South Symposium on Microelectronics*, 15.
- Ongaro, D., & Ousterhout, J. (2014). In search of an understandable consensus algorithm. *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, 305–319.
- Ongaro, D., & Ousterhout, J. (2025, January). *Raft Consensus Algorithm Visualizer*. <https://raft.github.io/>
- Orender, J., Mukkamala, R., & Zubair, M. (2022). Is Ethereum's ProgPoW ASIC Resistant?. *Proceedings of the 6th International Conference on Information Systems Security and Privacy*, 310–316. <https://doi.org/10.5220/0008909203100316>
- Oriekhoe, O. I., Ashiwaju, B. I., Ihemereze, K. C., Ikwue, U., & Udeh, C. A. (2024). Blockchain technology in supply chain management: a comprehensive review. *International Journal of Management & Entrepreneurship Research*, 6(1), 150–166.
- Pandya, S. B., Sanghvi, H. A., Patel, R. H., & Pandya, A. S. (2022). GPU and FPGA Based Deployment of Blockchain for Cryptocurrency–A Systematic Review. *2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*, 18–25.

- QuantumMechanic. (2011, July 11). *Proof of stake instead of proof of work*. <https://bitcointalk.org/index.php?topic=27787.0>
- Randamu, I. (2025,). *Group Configuration – drand Specification*. <https://docs.drand.lol/docs/specification/#group-configuration>
- randao. (2023, June). *randao*. <https://github.com/randao/randao>
- Rani, P., & Bhambay, R. (2022). A comparative survey of consensus algorithms based on proof of work. In *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2022, Volume 1: Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2022, Volume 1* (pp. 261–268). Springer.
- Santamaría, P., Tobarra, L., Pastor-Vargas, R., & Robles-Gómez, A. (2023). Smart contracts for managing the chain-of-custody of digital evidence: A practical case of study. *Smart Cities*, 6(2), 709–727.
- Schroeder, B., & Gibson, G. A. (2009). A large-scale study of failures in high-performance computing systems. *IEEE Transactions on Dependable and Secure Computing*, 7(4), 337–350.
- Seagate. (2017, January). *IronWolf Pro Datasheet*. <https://www.seagate.com/www-content/product-content/ironwolf/files/ironwolf-pro-ds1914-3-1701gb.pdf>
- Slimcoin Project. (2013, December 28). *Slimcoin: SLIMCoin Official Repository*. <https://github.com/slimcoin-project/Slimcoin>
- Slimcoin Project. (2020, December). *Slimcoin: Commit 03914e38ab9724d3d1a0d18afff65293a7142e43*. <https://github.com/slimcoin-project/Slimcoin/commit/03914e38ab9724d3d1a0d18afff65293a7142e43>
- Slimcoin Project. (2025, January 15). *Slimcoin Forks Overview*. https://github.com/slimcoin-project/Slimcoin/forks?include=active&page=1&period=&sort_by=last_updated
- Stewart, I. (2012, December 16). *Proof of burn*. https://en.bitcoin.it/wiki/Proof_of_burn
- Szabo, N. (1994). *Smart Contracts*. <https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>
- Visa Inc. (2023, July). *Deep Dive on Solana*. <https://usa.visa.com/solutions/crypto/deep-dive-on-solana.html>

- Walker, G. (2024, December 18). *Transaction Size*. <https://learnmeabitcoin.com/technical/transaction/size>
- Yang, J., & Sun, F.-B. (1999). A comprehensive review of hard-disk drive reliability. *Annual Reliability and Maintainability Symposium. 1999 Proceedings (Cat. No. 99ch36283)*, 403–409.
- Šakalys, A. (2023a, July). *BlockSim-Improved*. <https://github.com/MarkCDavid/BlockSim-Improved>
- Šakalys, A. (2023b, August). *BlockSimSharp*. <https://github.com/MarkCDavid/BlockSimSharp>
- Šakalys, A. (2025, May). *burner*. <https://github.com/MarkCDavid/Burner>

APPENDICES



Appendix 1: Participation Certificate for Young Scientists Conference