



DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Have your cake and eat it too: Analyzing the energy consumption of blockchain protocols

Stefan Vladov





DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

**Have your cake and eat it too: Analyzing the
energy consumption of blockchain protocols**

**Auf zwei Hochzeiten gleichzeitig tanzen:
Energieverbrauchsanalyse von
Blockchain-Protokollen**

Author:	Stefan Vladov
Supervisor:	Prof. Dr.-Ing. Jörg Ott
Advisor:	Dr. Ph.D. Nitinder Mohan
Submission Date:	15.07.2022



I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.07.2022

Stefan Vladov

Acknowledgments

I would like to thank my advisor, Dr. Ph.D. Nitinder Mohan for his continuous support and helpful feedback throughout the entire thesis. I also want to thank the Chair of Connected Mobility at TUM for the testbed and devices which made my experiments possible. Finally, I'd like to thank my friends and family for their support during my studies.

Abstract

The energy costs for blockchain protocols are often the center of heated discussion. Latest figures place the annual energy consumption of Bitcoin and Ethereum as 143 TWh and 62.75 TWh respectively. This is in the annual energy consumption range of countries with inhabitants in the millions, like Bangladesh. In this thesis we review the existing literature on blockchain energy consumption and what are their estimated figures. In addition, we will analyse the causes of the high energy consumption and its impacts on the environment.

Little research has been done on the energy consumption on granular levels of blockchain systems, for example a single network peer or how a small private network scale. Therefore, we will try to fill this gap by performing experiments on small scale networks and measuring their energy consumption. We will focus mainly on Ethereum's Proof-of-Work and Proof-of-Authority consensus algorithms, Ethash and Clique, as well as the permissioned blockchain Hyperledger Fabric.

Finally, we will comment on the future tendencies in blockchain systems, especially on the migration to Ethereum 2.0, which promises to drastically reduce energy costs and improve performance.

Contents

Acknowledgments	iii
Abstract	iv
1. Introduction	1
1.1. Problem statement	3
1.2. Thesis Approach and Outline	3
2. Background and Related Work	4
2.1. Consensus protocols	4
2.2. Proof of Work (PoW)	4
2.2.1. Ethereum	6
2.3. Proof-Of-Stake (PoS)	12
2.3.1. Ethereum 2.0	12
2.4. Proof-Of-Authority (PoA)	14
2.4.1. Clique	14
2.5. BFT and CFT	15
2.5.1. Hyperledger Fabric	15
2.6. Related Work	18
2.6.1. Global energy consumption	18
2.6.2. Non-PoW blockchains	21
2.6.3. Contribution	22
3. Experiment and Testbed Setup	23
3.1. Testbed	23
3.1.1. Software	24
3.2. Ethereum Experiment	25
3.2.1. Test workflow	26
3.2.2. Genesis configuration	27
3.3. Hyperledger Fabric Experiment	27
3.3.1. Test workflow	28
3.3.2. Configuration	29

4. Blockchain Evaluation	30
4.1. Geth PoW network	30
4.1.1. Test parameters	32
4.1.2. Performance	32
4.1.3. System resource consumption	36
4.1.4. Energy consumption	38
4.1.5. Energy waste	40
4.2. Geth PoA network	41
4.2.1. Performance and energy consumption	41
4.3. Hyperledger Fabric	42
5. Conclusion	44
5.1. Summary	44
5.1.1. Global energy consumption	44
5.1.2. Experiment results	45
5.2. Have your cake and eat it too	46
5.3. Future Work	46
A. Reproducibility	47
A.1. Networks	47
A.1.1. Geth network	47
A.1.2. Fabric network	48
A.2. Data Collection	49
A.2.1. pprof	49
A.2.2. Monsoon PowerTool	49
List of Figures	50
List of Tables	51
Bibliography	52

1. Introduction

Since Bitcoin's first inception in 2009 [46], blockchains have become a mainstay of the global market with a market capitalization of 5.92 Billion USD in 2022, which is expected to grow with more than 80% until 2030 [9]. Next to the economical interest there is of course heated discussion regarding the notoriously high energy consumption of blockchain protocols, especially Bitcoin and Ethereum.

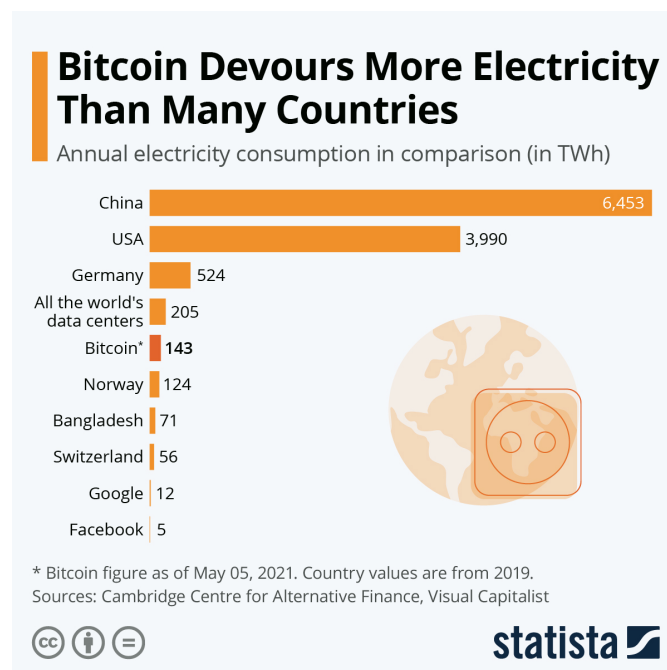


Figure 1.1.: Comparison of bitcoin energy consumption in TWh [5]

Although exact values are unavailable, various estimates can be made. For total annual energy consumption Statista (figure 1.1) estimates 143 TWh for Bitcoin for the year 2021 which is significantly more than the country of Bangladesh, which has a population of roughly 168 million people [3]. The same statistic show also, that bitcoin consumes about 100 times more energy than Google and Facebook. Furthermore according to Digieconomis [27] Ethereum consumes 62.75 TWh, which is similar to

Denmark and even the "joke" blockchain Dogecoin [19] consumes about 3.33 TWh, which is in the range of electricity consumption of Jamaica. In perspective these values can look more extreme, for example a single Bitcoin transaction consumes approximately the same energy as a US household for nearly 2 months [27].

Figures this high naturally create a strong social backlash as the expended CO₂ in the energy generation for blockchain systems can have strong environmental impacts. Moreover, intergovernmental organizations such as the International Monetary Fund(IMF) [17] and European Union(EU) [48] are now actively discussing blockchain protocols and their environmental impacts. Furthermore, China has banned all blockchain transactions since September 2021, although the exact reasons are unknown [72]. In addition, there was a failed petition [52] to ban PoW blockchains in the United Kingdom parliament.

To address this issue, Ethereum has been planning since 2017 a migration that would drastically reduce energy consumption [63], which is planned to fully release in 2023. Moreover, new blockchain systems such as Energy Web [23] are emerging as new carbon-friendly alternatives to Bitcoin and Ethereum.

Nevertheless, the energy consumption of blockchain protocols warrants detailed discussion on how it is caused, how it can be addressed and what are possible alternatives.

1.1. Problem statement

Due to the high energy consumption the profitability and the sustainability of blockchains are questioned not only in the academic field [58], on the political scene [48] and in the environmental area [66]. Especially with the emergence of stricter environmental regulations in the European Union [22] and the drive to use efficient and sustainable systems, blockchain systems are part of a heated debate on their environmental impact. Therefore it is important to determine if this emerging technology is worth investing in despite the high energy costs.

For this thesis we have devised the following objectives and research questions:

1. Investigate current research on blockchain energy consumption. What are the reported values, what is their impact for example on the environment? What is the current direction blockchain systems are heading in terms of energy efficiency?
2. Run tests on small private networks and examine system resource and energy consumption. What are the correlations? What energy is wasted?
3. Discuss the feasibility of PoW. How justified is the usage of PoW? Should we adopt non-PoW blockchains instead?

For the last question, we would like to especially examine the idea of having both a "public permissionless blockchain" that also has "low energy consumption and high throughput", or in the context of the thesis "have your cake and eat it too".

1.2. Thesis Approach and Outline

In this thesis we employ a mixture of both literature review and experiments. Appendixes are added for the more technical parts of the experiments which are mostly related to reproducibility and not results.

Chapter 2 will first provide background on the analyzed blockchain systems, namely what is their structure and function. Then it will conclude with a literature review on energy consumption of blockchain protocols. Chapter 3 will explain the test bed and the experiments we plan to run. Chapter 4 will then contain the result and evaluation of the experiments. Finally, Chapter 5 will summarize the results from the literature review and experiments, and discuss the the topic as a whole.

2. Background and Related Work

This chapter will go through the theoretical background used for the evaluation and conclusion. The first section will go through the different consensus algorithms and their definitions. Moreover, two frameworks: Ethereum and Hyperledger Fabric, will be explained more in-depthly in the evaluation phase. Finally, we'll conclude the chapter with a literature review on the energy consumption of blockchain protocols.

2.1. Consensus protocols

Blockchain consensus algorithms are responsible for reaching an agreement on the state of the blockchain across all the nodes participating in the network[70]. Generally the consensus algorithm in a blockchain depends on the blockchain type: permissionless and permissioned.

Permissionless or open-access constitute systems where there are no requirements for entry and participating nodes can remain anonymous. They are easily scalable, but employ a probabilistic Byzantine agreement, usually reached via solving a cryptographic puzzle with a financial incentive(Proof-of-Work) [70].

Permissioned blockchains, on the other hand, focus on a smaller set of authenticated nodes, which have to be approved to gain access. They usually employ practical Byzantine-Fault-Tolerance(pBFT) or Crash-Fault-Tolerance(CFT) algorithms that reach consensus immediately. Their throughput is usually much higher at the cost of worse scalability due to message overhead [70].

2.2. Proof of Work (PoW)

Proof of Work (PoW) is by far the most notorious consensus algorithm in the context of energy waste. It can be described as an algorithm where the verifier (or miner in PoW blockchains) has to expend a certain amount of computational resources in a specified interval of time to be elected temporarily as the leader in the consensus [41].

The PoW algorithm as described in the original Nakamoto paper consists of three procedures [46]:

- **Chain validation:** This step verifies that the transactions field of the block are valid.
- **Chain comparison and extension:** The peer compares the chain proposed by himself or the one broadcasted by his peers and picks the longest one.
- **Solution searching:** The peer attempts to solve the search puzzle to prove the validity of the proposed block.

The third step is the most computationally intensive as it involves executing the hash function H with input x (containing the blockdata and hashcode) and an incremented *nonce* value for a specific difficulty h until the following condition is fulfilled:

$$th = H(x||nonce) \leq D(h) \quad (2.1)$$

where th is the target header [70].

As per definition, the probability of a node winning the peer election is entirely based on its share of computational power. [41] Formally, it can be expressed with the following formula:

$$Pr_i^{\text{win}} = \frac{w_i}{\sum_{j \in N} w_j} \quad (2.2)$$

where N is the set of nodes participating in the network and w_i is a generalized resource of computational power [70]. Most often the computational power is expended CPU time, but in other variances of the algorithm it can also include memory, disk storage, time elapsed, etc.

Energy waste

In the paper which coined the PoW algorithm, the authors also mentioned the so called "bread pudding" algorithm, where the expended computational work, could be used for the benefit of the network [41]. The PoW algorithm in Bitcoin and Ethereum is most certainly not a "bread pudding" algorithm as the expended computational work is not utilized outside of the verification process. Moreover, as previously mentioned, the PoW algorithm is CPU intensive. Additionally, as microchips (in CPUs, ASICs and GPUs) are one of the most power consuming components, there is a precedent to examine the overall power consumption of PoW networks and their environmental impact.

2.2.1. Ethereum

First introduced in 2015, Ethereum is at the time of writing the blockchain with the second-highest market cap [14]. Its implementation of the PoW algorithm is called Ethash and is further explained later in this section.

In this thesis for the PoW evaluation we will be focusing mainly on Ethereum, as it is currently in an interesting position between Proof-Of-Work and a Proof-Of-Stake migration called Ethereum 2.0[63]. Nevertheless, Ethereum and Bitcoin share many similar structures and concepts, and when possible we will draw parallels between them.

Technical Overview

Ethereum [10] can be imagined as a state machine, replicated across all the network peers. Each valid transaction submitted by a user is a state transition, that gets executed on each of the peer's world state. This state machine is called the EVM [10], and in comparison to Bitcoin, allows the execution of Turing-Complete code.

Accounts are either owned by a user or a smart contract (more on them later) and controlled with a private key. Moreover, the account consists of the following elements [73]:

- *nonce* - the number of transactions sent by the sender
- *balance* - number of Wei owned by the address
- *storageRoot* - hash of the root node of a Merkle Patricia tree
- *codeHash* - the hash of the EVM of this account, which gets executed during a message call. This field is immutable.

The block is in this context the collection of state changes [73] that get appended to the blockchain. Here are some of the key attributes of the Ethereum block:

- *parentHash* - hash of parent block
- *ommersHash* - hash of list of ommers block (more about them in Ethash)
- *beneficiary* - address to payout mine rewards
- *difficulty* - value for difficulty level of block
- *number* - number of ancestor blocks
- *gasLimit* - current gas limit

- *gasUsed* - total gas used in transactions
- *timestamp* - Unix's time() at block inception
- *extraData* - arbitrary data array. Usually used for miner signatures
- *mixHash* - used to prove with nonce that a sufficient computation time has been carried out
- *nonce* - used to prove with mixHash that a sufficient computation time has been carried out

gasLimit and *gasUsed* will be further explained in chapter 4 as they are key to measuring and determining Ethereum performance. The importance of the *ommersHash*, *mixHash* and *nonce* will be examined in the Ethash section. Generally, Bitcoin blocks[46] follow the same structure, but omit the *ommerHash*, gas features and *mixHash*.

Transactions, Smart Contracts and Gas

Now, we'll talk about the functional elements of the EVM, e.g those that lead to state changes, namely the transactions, smart contracts and gas.

Transactions

A transaction is a cryptographically signed instruction sent to the EVM from an external actor [73]. External in this case refers to outside of the blockchain, in the sense of an actual human or software, in comparison to smart contracts which are not allowed to send messages by themselves. The transaction hash has the following general structure [73]:

- *nonce* - number of sent transactions
- *gasPrice* - price per unit of gas
- *gasLimit* - maximum amount gas that can be expended during execution of transaction
- *to* - target address
- *value* - amount of Wei(Ethereum) to be transferred
- *r, s* - contains signature of sender

Smart Contracts

The EVM is a Turing-complete stack-based language, which executes deterministically. EVM code is usually written in a higher level language, the most popular currently being Solidity [60].

There are two sub-types of transactions, ones that result in message calls and ones that create smart contracts.

Initialization message calls include the following field:

- *init* - EVM-code fragment, which contains the actual code of the smart contract, which will get executed when this account receives a message call.

The message call transaction in turn contains the following field:

- *data* - specifies the input data, e.g the arguments which are supplied to the smart contract

Ethash

Ethash is the current PoW algorithm for Ethereum 1.0 [24]. It is memory-hard in the sense that it consumes a significant amount of memory. It was initially devised as an application-specific integrated circuit(ASIC) resistant algorithm. Although Ethhash ASICs are feasible, they are not economical in the sense that, due to memory bandwidth being a limiting factor, they can't outcompete GPUs, which are a commodity hardware and therefore more easily available [43].


#	COIN	CURRENT	2GB	3GB	4GB	5GB	6GB	8GB
1	 ETH Ethereum	4.914 GB 15,051,760	Passed 8,240,000	Passed 7,680,000	Passed 11,520,000	27 Aug 2022 6:52 15,360,000	11 Aug 2024 20:12 19,200,000	12 Jul 2028 22:52 26,880,000

Figure 2.1.: DAG size growth [16]

Memory hardness is achieved with a directed-acyclic-graph(DAG) structure, which grows linearly overtime. A new one is generated every 30000 blocks. This 125 hour window is called an epoch. Initially it was 1 GB 2.1, but for the current mainnet it's 4,91 GB. The size of the DAG therefore constitutes the minimum memory requirement to operate an Ethereum mining node, hence why the current ideal device for mining are GPU cards with more than 4GB of memory.

Ethash has the following general workflow [24]:

1. Compute a seed by scanning through the block headers up until the current epoch.

2. From the seed generate a 16 MB pseudorandom cache. Light clients store it for verification
3. From the cache, generate the DAG dataset, with the property that each item in the dataset depends on only a small number of items from the cache. Full clients and miners store the dataset.
4. Mining involves grabbing random slices of the dataset and hashing them together. Verification can be done with low memory by using the cache to regenerate the specific pieces of the dataset that you need, so you only need to store the cache.

Step 4. can then be further broken down into the following workflow figure 2.2. Generally, it follows the same procedures as the PoW in Bitcoin where it checks the generated hash value against the target difficulty. If the value is above the target, the nonce is incremented. In Ethash, the addition are the mixing steps, where the generated hash is passed into a mixing function, which returns an address of a memory page(128B) in the DAG. This procedure is repeated 64 times until the final digest is produced and compared against the target [24].

As a sidenote, as Ethash development coincided with the SHA3 standard, its hashes are not standard sha3 and are therefore referred to as "Keccak-256" and "Keccak-512" [24].

In conclusion, Ethash is a predominantly a CPU heavy algorithm, which in comparison to Bitcoin's PoW also requires an increasing amount of memory to perform the aforementioned operations. It's unsuitable for ASICs, that's why mining is predominantly done on graphics cards, which due to their higher memory bandwidth are more efficient.

Simplified GHOST

An important aspect of the blockchain consensus is the fork resolution. Generally, this is the heaviest chain, the one that has the most computation expended on it [73]. As a block header contains a difficulty field it is easy for miners to recursively compute the total difficulty of the chain and choose the one with the highest difficulty.

Longest-Chain-Rule(LCR) is the protocol used in Bitcoin [46], where the valid chain is the one with the highest total difficulty.

The GHOST protocol is a proposed alternative to LCR for Bitcoin [61], where in the computation of the total difficulty ommers(also known as uncle blocks) are included. This serves the purpose of decreasing the feasibility of selfish mining and increasing the overall security of the blockchain.

Ethereum uses a modified version of the GHOST protocol [73], where, like Bitcoin, ommers are not included in the total difficulty of a block. Instead, for the winning

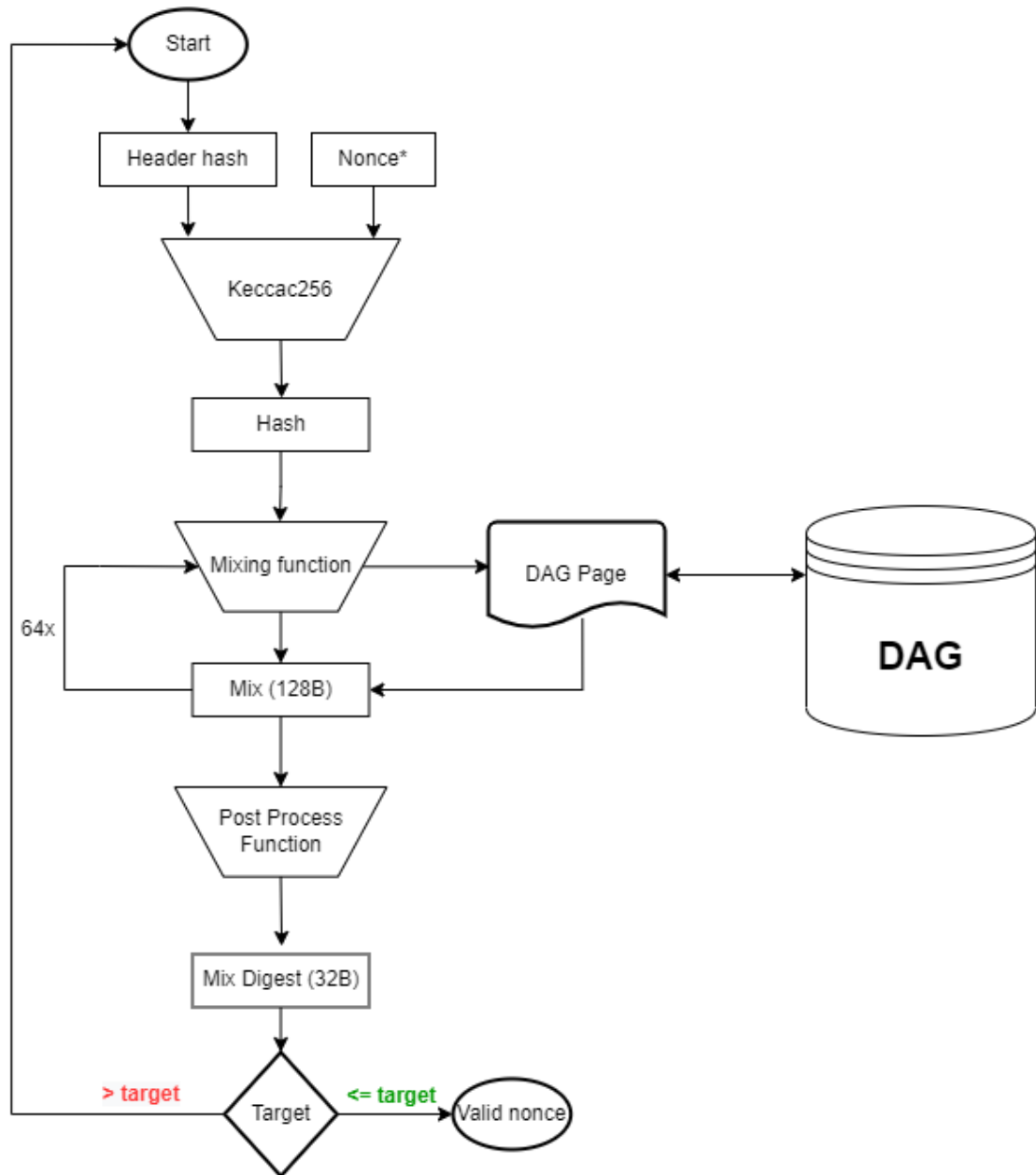


Figure 2.2.: Mining with Ethash

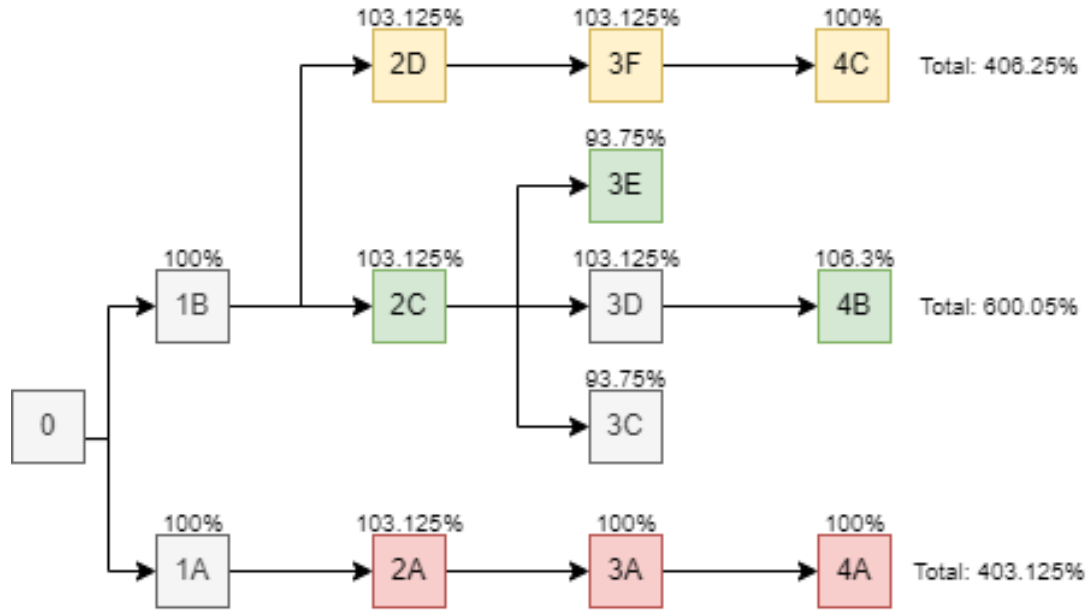


Figure 2.3.: Distribution of mining rewards for modified Ghost. Blocks colours correspond to miners. On the right is the total mining reward for this branch

block, the miner gets an additional reward (approx. 3.125% of coinbase reward) for each valid uncle block, and the miner of every uncle block gets part of the coinbase reward himself (approx. 93.75%). This calculation is done up to 7 generations. Therefore miners are incentivized to pick blocks with a higher amount of uncles to maximize block rewards. As shown in figure 2.3, the tree with the highest amount of mining rewards is the one with the most uncles (given same length). Moreover, miners can still get rewards for their uncle blocks, reference green miner, who has collected more rewards than red, despite only winning 2 blocks.

In terms of energy waste, although there is no significant difference in the mining work miners expend in LCR and modified Ghost, there is in the payout. In LCR, stale blocks are completely wasted energy in the sense that they don't give a benefit to the miner (no block reward), neither do they increase the security of the chain as they can spawn a selfish attack. In modified GHOST focusing on the tree with the most uncles gives some justification to the expended computational power as the reward for the mining the block increases substantially.

2.3. Proof-Of-Stake (PoS)

Peercoin was the first blockchain to utilize the Proof-Of-Stake algorithm as a way to address the excessive computational demands of PoW[50]. Here the concept of coin age was first introduced, which is a product of network tokens and the length of time they are held. Originally, (in Peercoin) expending coin age reduced the difficulty of the search puzzle. In modern iterations of PoS [49], mining is removed altogether in favour of block leaders being chosen solely on their stakes. Generally, a higher stake corresponds to a higher chance of being elected as leader.

Removing the computational demands of PoW will naturally drastically improve the energy efficiency of the algorithm. Additionally, PoS has a higher transaction confirmation speed, which also leads to a higher throughput and overall blockchain efficiency. [49]

2.3.1. Ethereum 2.0

For the PoS evaluation we'll turn our attention towards the Ethereum 2.0 migration, which if successful would turn Ethereum into the largest PoS blockchain. The concept of Ethereum 2.0 was initially conceived [28] to address the high energy requirements of Ethereum 1.0 and Ethash, as well as the limited throughput and increasing storage costs. Particularly, it tries to tackle the following disadvantages of Ethereum 1.0:

- Miners trying to solve the PoW challenge in order to mine a block requires overall a significant amount of energy, which if not being 100% gained by renewable methods, harms the environment and contributes to climate change. According to Digiconomist [27], the current yearly energy consumption of the Ethereum network is around 60 TWh, while the carbon footprint is around 35 MT of CO₂.
- The proof-of-work consensus algorithm implies a higher chance for successfully mining blocks by those miners with high computational power. This is usually achieved using specialized equipment tailored for mining, which is a significant financial expense. Therefore to maximize profits, miners group together in centralized mining pools, which is the dominant way to mine[25].

Consensus

Ethereum 2.0 intends to solve the two aforementioned problems by transitioning from the Ethash to a PoS consensus mechanism. Compared to PoW, the risk-based staked currency of PoS is not an amount of expended computational work, but instead the provided deposit in the form of a staked unit of a cryptocurrency.

Should a node become a validator and therefore gain the skill of creating new blocks on the blockchain, the node has to stake a certain (specification-defined) amount of units of a cryptocurrency as a protective deposit. While being a validator, a node's job is to listen for new incoming blocks from peer-nodes, to execute the code stored in these blocks, to check the blocks' signature and to promote that block across the network. The set of nodes, which are chosen for validating a block is randomly selected [55].

Should a set of nodes detect a fraudulent node on the network, which is trying to commit a 51% attack, the nodes either have the option to ignore the fraudulent node or to blacklist it on the network in combination with deleting the fraudulent node's stake from their network. In the latter case, the node would lose, from the perspective of the (majority) set of nodes, its security deposit, which creates the protective risk against fraudulent behaviour [55].

In this way, not only is the energy consumption reduced (according to Ethereum by "99,95%"[63]) by not using Ethash, but also no special equipment is required, which lowers the barrier for newcomer nodes.

Sharding

An additional feature of Ethereum 2.0 is "Sharding", which is described as "the process of splitting a database horizontally to spread the workload"[59]. Sharding will split up the overall ledger data horizontally into different sections (shards), which are then handled by a subsection of nodes. Therefore, certain traffic can be limited to these subsections, in case the transactions affects only the data stored in the nodes in the subsection. In this way, Sharding reduces the overall network traffic as well as the storage load per node and increases the number of transactions per second as well as the amount of fitting devices for running a validator [59].

In comparison to an unsharded Ethereum blockchain, it is due to the fast data transaction and therefore the lack of necessity not yet intended for a Sharded Ethereum blockchain to store executable code, but only the data relevant to the transaction[59].

The Beacon Chain and the Merge

Currently, the PoS consensus mechanism is not yet in operation on the Ethereum mainnet, but on a side blockchain, called Beacon Chain. This beacon chain "does not process transactions or handle smart contract interactions"[62] but only forms a "ledger of accounts that conducts and coordinates the network of stakers"[62].

The next two main events in the launch of Ethereum 2.0 are:

1. Connecting the yet separated Beacon Chain with the mainnet in order to switch the consensus mechanism from PoW to PoS. This event is called "the Merge" and it is planned for the 3rd or 4th quater of 2022[63][62].
2. The introduction of Sharding onto the mainnet after the Merge. This event is planned for 2023[59][62].

2.4. Proof-Of-Authority (PoA)

Proof-Of-Authority (PoA) is a consensus algorithm, where a subset of pre-authorized nodes called validators manage the consensus in the network. The stake is the identity of each node and nodes who have exhibited malicious behaviour are banned from the network [21][48]. Due to its relatively simple mechanism, it achieves high transaction throughput with a low energy cost.

A notable example of this consensus algorithm is EW Chain [23], which uses a permissioned version of Proof-Of-Authority. Therefore it manages to achieve high scalability with low energy costs, a claimed 6 orders of magnitude less than PoW [48]. For example, in 2021 the entire network energy consumption was approximately 7.5 kilowatts, which is 2.2 million times less than Bitcoin or 1.0 Million for Ethereum.

2.4.1. Clique

As part of the thesis we will evaluate a lesser known PoA algorithm called Clique [21]. It's most widely used in Ethereum private networks, especially the biggest test network Görli [34]. Outside of the consensus algorithm, all other parts of the Ethereum ecosystem function practically the same.

In comparison to the PoW on the mainnet, in Clique the genesis configuration now contains the addresses of the pre-authorized nodes, which are the validators. Only they are allowed to seal blocks, which is the PoA version of mining. Other peers may be later added or removed as a validator based on the consensus. Key features of Clique include:

- Block finality - sealed blocks are final
- Rotation principle - miners cannot seal concurrent blocks
- Voting mechanism for joining or leaving the authorized list of miners

We have deliberately chosen Clique for the experiments as it would allow us to evaluate the function of the Ethereum blockchain with two different types of consensus algorithm.

2.5. BFT and CFT

Crash-Fault-Tolerance(CFT) and practical Byzantine-Fault-Tolerance(BFT) are another two types of permissioned algorithms, which take two different approaches to security.

CFT is designed to sustain up to $\lfloor N/2 \rfloor$ failures where N is the number of nodes. There are two types of nodes: Leaders and Followers, where the former is solely responsible for constructing and adding the blocks to the blockchain. Block propagation is done with a two phase commit protocol(2PC), where the leader first checks that followers are ready to commit (commit-request phase) and then broadcasts the committed or aborted the transaction. An important prerequisite for CFT is that the network has a high level of trust, as the algorithm does not handle malicious actors. A popular example that uses an implementation of CFT called Raft[56] is used in Hyperledger Fabric[64].

On the other hand, Byzantine-Fault-Tolerance(BFT) is designed to handle generally up to $1/3N$ of malicious nodes. Similarly to CFT it uses a three phase commit protocol(3PC), where there is an extra phase in-between(pre-commit), which checks whether enough nodes will commit to the transaction[48]. Popular implementation of BFT is practical Byzantine-Fault-Tolerance[11], versions of which are used in Hyperledger Besu[37] and Hyperledger Indy[39].

In conclusion, both CFT and BFT algorithms generally achieve a performance much higher than permission less algorithms. Moreover, they achieve immediate finality on blocks. However, 2PC and especially 3PC have a much higher message complexity[58]. Therefore they have a more limited scaling, often making them unsuitable for larger networks.

2.5.1. Hyperledger Fabric

Hyperledger Fabric is a distributed permissioned ledger part of the Hyperledger Foundation. Fabric serves the purpose of enabling the development of modular and versatile solutions that simultaneously satisfy the need for trust, transparency and accountability.

Some of its key differentiating characteristics include an easily configurable architecture, support of multiple programming languages for smart contract development and exchangeable consensus protocols. Properties such as high performance, scalability and privacy are also typical for Fabric.

Network architecture

As a permissioned ledger, a Fabric network is initially defined by a set of organisations (often legal entities), which determine a set of policies that will serve as the initial configuration of the network. These organisations are often referred to as the consortium. Afterwards, of course, other organisations may also join the network or network policies can be changed, following an agreement from the existing consortium [8].

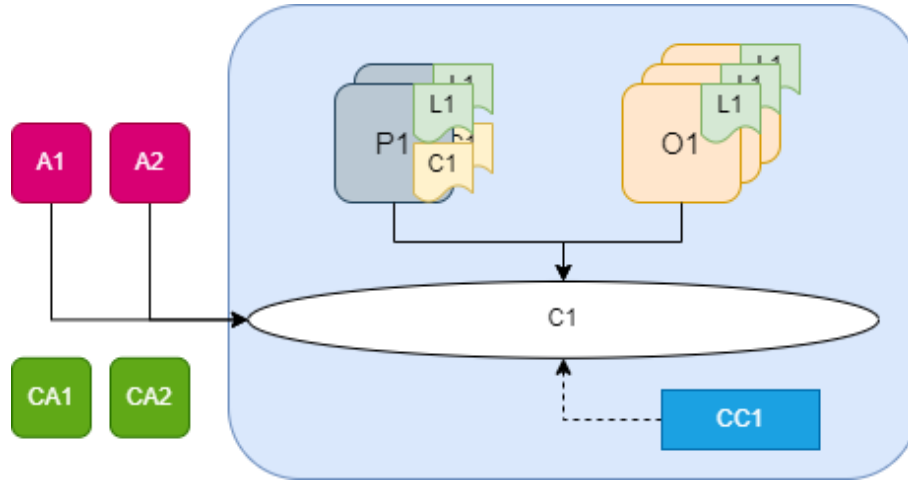


Figure 2.4.: Hyperledger Fabric Network Overview

In Hyperledger Fabric, there are two fundamental types of network nodes - peers and orderers [51][64]. Peers(P1) host ledgers(L1) and chaincode(C1) figure 2.4, the latter being a representation of the business logic, synonymous to smart contracts.

A collection of physical peers providing points for access to organizations willing to transact with each other form the logical structure known as a channel(C1) figure 2.4. Channels provide an efficient sharing of infrastructure and at the same time preserve data and communications privacy. A peer can belong to multiple channels for each of which it keeps its own ledger.

Orderers, on the other side, gather endorsed transactions and order them into transaction blocks. These are eventually distributed to every peer node in a channel, where any local copies of the ledger are updated accordingly. [64] The ordering service further enforces basic access control for channels, regulating who can configure them, as well as who can read and write data. All orderers also contain the blockchain portion of the ledger, but not the state database.

In general, there are several different ways to reach consensus on the ordering of transactions between ordering service nodes, which constitutes the pluggable consensus

algorithm of Hyperledger Fabric. The current recommended option is Raft, with Kafka being deprecated since version 1.2, and with a BFT-Smart algorithm currently being in discussion. [4]

There are several other elements in the Hyperledger Fabric network, these include Certificate Authorities (CA1) and Applications (A1) figure 2.4. The former manage the identity of participating actors via issuing digital certificates and the latter constitute applications developed with the Fabric SDK which communicate with the network.

Execute-Order-Validate

When executing chaincode, a 3-phase process, the so-called execute-order-validate model [64], ensures that all the peers in a blockchain network maintain their ledgers in the same state, which derives the following transaction flow:

- **Execute** In the Fabric network, the application-specific endorsement policy determines how many endorsing peers need to vouch for the correct execution of a given smart contract invocation. After the client has sent the signed transaction to the appropriate number of peers, each of them executes the transaction, confirming its validity and producing a read/write set, which contains the state variables which have been modified during the contract execution. Afterwards the peer returns a signed endorsement to the client.
- **Order** With enough endorsements the client can send the transaction to the ordering service, which will then establish the total order of transactions for the block, based on the chosen consensus algorithm (by default Raft).
- **Validate** The block is then propagated via gossip across the network. After a peer receives a block it will then verify it has a sufficient number of endorsements, otherwise it ignores it. Then based on the previously produced read/write set, it will check the transaction for state modification conflicts. If a transaction has a read input from the state database, which has already been modified by a transaction earlier in the block, it will be ignored.

Raft

Raft is a "leader and follower" distributed consensus model [56]. In the context of Fabric, each Raft node is an Orderer node. The following is a brief rundown of Raft workflow:

"Raft nodes are always in one of three states: follower, candidate, or leader. All nodes initially start out as a follower. In this state, they can accept log entries from a leader

(if one has been elected), or cast votes for leader. If no log entries or heartbeats are received for a set amount of time (for example, five seconds), nodes self-promote to the candidate state. In the candidate state, nodes request votes from other nodes. If a candidate receives a quorum of votes, then it is promoted to a leader. The leader must accept new log entries and replicate them to the followers." [64]

Channel configuration

As previously mentioned, the consortium defines the channel configuration, which contains several parameters which directly influence the transaction workflow [65]:

- **Batch size** defines the size of blocks, based on the *absolute max bytes* or *max message count*. Blocks are cut if they are going to exceed either of those parameters
- **Batch timeout** is the amount of time to wait before cutting a block. Directly influences latency
- **Block validation** defines the signature requirements, e.g how many endorsements a transaction needs to be considered valid

2.6. Related Work

In this chapter we will examine the available literature on the energy consumption of blockchain systems, future research and legislative directions. Finally we'll make conclusions based on what is covered and where our research can fit in.

2.6.1. Global energy consumption

It is well known that PoW blockchains, especially Bitcoin, consume vast amounts of energy [69]. Still getting exact figures on the global energy consumption is hard, if not impossible. This is often the case, because miners withhold information on the hardware they are using and the associated energy costs. In the literature an estimated lower and upper bound are often used, based on the works of Vranken(2017)[68] and Krause and Tolaymat (2018)[42]. These bounds can be expressed as the following for the lower bound:

$$total\ power\ consumption \geq total\ hash\ rate * min\ energy\ per\ hash \quad (2.3)$$

and for the upper bound respectively:

$$total\ power\ consumption \leq \frac{block\ reward * coin\ price + transaction\ fees}{avg.\ blocktime * min.\ electricity\ price} \quad (2.4)$$

The lower bound can be explained as the minimum amount of power, when for example using the most efficient hardware, that will achieve the total hash rate. The total hash rate can be calculated based on the difficulty and block times in the blockchain.

On the other hand, the upper bound refers to the financial feasibility of mining. Generally, miners would feel inclined to mine, when the financial rewards from the block reward and transaction fees compensate operating energy costs. Of course the upper bound is therefore highly volatile, especially in sudden changes in crypto currency value and energy prices [48].

Furthermore, Sedlmeir [58] uses the following system to estimate the total energy consumption of the top 5 PoW blockchains for 2020 in figure 2.5, which for example places the Bitcoin power consumption between 60 TWh and 125 TWh a year.

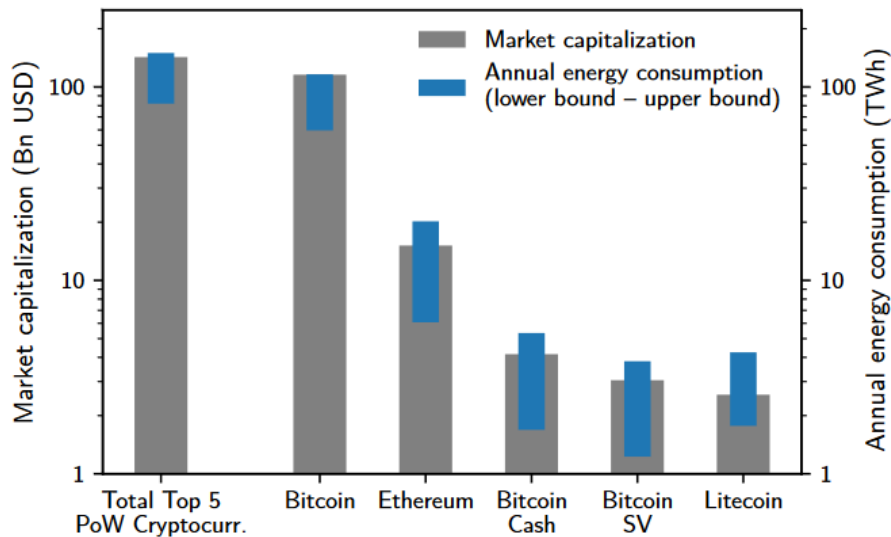


Figure 2.5.: Market capitalization and est. bounds of the top 5 PoW blockchains [58]

This is further backed by a study of Gellersdörfer [31], which accounts 2/3 of the energy demand to Bitcoin. For more recent data, Digieconomist[6] estimated an all time high of 205 TWh in May 2022, a 64% increase for 2 years based on Sedlmeir's estimates[58]. In perspective the current consumption of 131.51 TWh(July 2022)[6] is still an exorbitant amount, somewhere between the annual power consumption of Norway and Argentina [44]. Alternatively, the energy expenditure of a single Bitcoin

2. Background and Related Work

transaction can be equated to the energy requirements for several weeks or months of a German household [58].

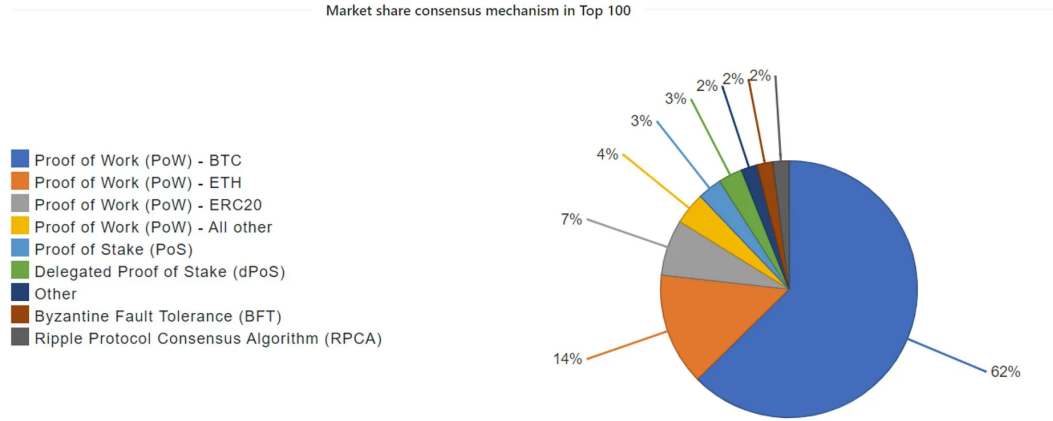


Figure 2.6.: Market share on consensus algorithm[33]

Blockchains that utilize PoW still constitute the majority of the crypto currency market share with 87%[33], moreover there seems to be correlation between the market size and energy consumption of a blockchain [15][58]. A popular is that miners will simply prefer to mine blockchains with a higher block reward.

Hardware

For mining hardware generally Bitcoin uses ASICs, Ethereum - GPUs. In the following table you can find newer devices with their respective hashrate, power supply and price, which is based on Bitmain[7] and Amazon[2] as of July 2022.

Generally there is a 6 magnitude of order difference between the hashrate for PoW and ASIC-resistant PoW blockchains. Although, ASICs are much more energy efficient for hashing than GPUs, there does not seem to be a noticeable difference in power consumption based on the hashrate alone. Researchers agree that energy price is the driving factor in mining [58][42][31], therefore for miners the energy price, and not the upfront cost, is the determining factor on what and how much hardware they operate.

However, both ASICs and GPUs used for mining blockchains contribute significantly to e-waste[6].

Device	Type	Hashrate	Power supply	Price
Antminer S19 Pro	BTC	110 TH/s	3250W	\$5940
Antminer S19	BTC	95 TH/s	3250W	\$3990
Antminer T19 Hydro	BTC	145 TH/s	5438W	\$2175
Antminer ETH/ETC Miner E9	ETH	2400 MH/s	1920W	\$9999
NVIDIA GeForce RTX 3090	ETH	121MH/s	350W	€1,649
NVIDIA GeForce RTX 3060 Ti	ETH	60MH/s	200W	€559.00
AMD Radeon RX 6700 XT	ETH	47MH/s	230W	€675
AMD Radeon RX 580	ETH	28MH/s	185W	€440

Table 2.1.: Mining devices

Environmental impact

Unsurprisingly the environmental impact of PoW blockchains is a heated discussion. Generally, translating the total power consumption to carbon footprint is not a good practice. For example, research from the Cambridge Centre for Alternative Finance (CCAF) reports that 76% of hashers use renewable energy source to some degree, whereas non-renewables still account 39% of the total hashing energy consumption [1]. Moreover, the 2021 report of the European Commission [48] identifies that Bitcoin miners often operate in remote locations where they can make use of cheap excess energy, often from renewable energy sources, e.g solar panels, wind turbines, hydro, etc.

Additionally, the European Commission report gives the following strategies for lower energy consumption and costs [48]:

1. Use of renewable energy sources
2. Operating during hours when the energy price is low (e.g during the night)
3. Using the excess energy from energy producers which cannot be stored

2.6.2. Non-PoW blockchains

The most popular alternative to Proof-Of-Work in terms of energy efficiency is Proof-Of-Stake [58]. Moreover, in paper from the UCL Centre for Blockchain Technologies (UCL CBT) [53] various different PoS blockchains were compared and their energy consumption was predicted based on a mathematical model. The paper then concluded that PoS is 3 orders of magnitude less energy intensive than the Bitcoin PoW, reference figure 2.7.

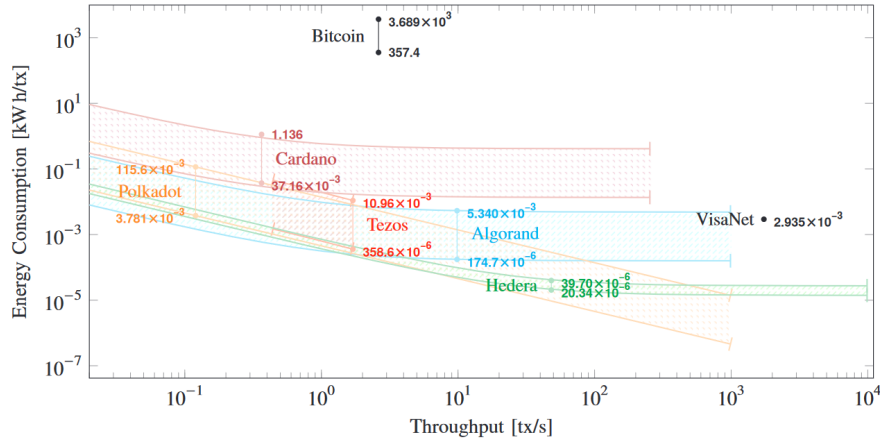


Figure 2.7.: PoS energy consumption comparison [53]

2.6.3. Contribution

A lot of papers estimate the global energy consumption of blockchain networks, its implications and energy impact. Most of them take a black box perspective and only consider the network as a whole. Significantly less research is done on lower levels of the blockchain protocols, for example the energy expenditure of single nodes or how small networks scale. In this thesis we will examine such cases.

Therefore in the following chapters, we will create a small test network for Ethereum and Hyperledger Fabric. On this network we will perform tests to measure the expenditure of computational resources and energy, and especially how they scale with different network settings. Finally, we will try to relate the findings to the mainnet to create more generalized assumptions.

3. Experiment and Testbed Setup

In this chapter we will focus on the experimental setup. As previously mentioned in the thesis outline we will aim to examine performance tendencies in private networks and scale them with the available computational resources. Firstly, we'll describe the used hardware, software and then the experimental setup and orchestration.

3.1. Testbed

For the test setup, we have used a combination of physical devices and VMs. Table 3.1 lists all the used devices and their characteristics. Additionally, for the physical devices the appropriate power supply is included. All devices also run *Ubuntu Server* as an OS.

Device	Count	CPU cores	RAM	Power supply
Raspberry Pi 4b	1	4	4 GB	5V
Udoo Board x86	1	4	4 GB	12 V
Fujitsu Esprimo G558	1	8	32 GB	20 V
VM(M)	1	2	2 GB	-
VM(L)	10	4	4 GB	-
VM(XL)	1	8	8 GB	-

Table 3.1.: Device list

For the power measurements we have utilized the high voltage power monitor by Monsoon Solutions [35]. It outputs up to 13V which makes it only compatible with the Raspberry Pi 4b and Udoo Board x86, from which we will draw our energy measurement data.

3.1.1. Software

In this section we'll briefly go through the different software tools used throughout the evaluation section.

Geth

Go Ethereum(Geth) is the go-lang implementation of the Ethereum protocol[29]. At the time of writing geth is the most popular Ethereum client with over 80% utilization based on the mainnet statistics [12]. Therefore it would be representative candidate for the Ethereum network. Additionally, geth supports CPU mining, which would make it easier to have mining and passive nodes in the same configuration.

Web3j

Web3j [71] is the Java client for Ethereum. It allows us to administer the geth peer, as well as send transactions and call smart contracts. As our test orchestration will be done in Java, we will use it to interface with the different Ethereum nodes.

go tool pprof

pprof package is the profiler tool for the go-lang ecosystem. [54] This tool supports several different performance profiles and can produce visualizations for them. In this thesis we will focus on the CPU and Heap profile, which contain the amount of CPU time per goroutine. Luckily, geth supports pprof [13] out of the boxes, which makes gathering this data relatively simple. For more information on how pprof was set up, refer to the appendix A.2.1.

Glances

glances is a monitor tool for keeping track of system resources [32]. We will use it to keep track of system resource consumption on each of the nodes.

Docker

We use docker [18] for the Hyperledger Fabric network, as it is the go-to tool for managing deployments [67]. As we will be only running a couple of containers per node, the performance overhead of virtualization should be negligible.

Hyperledger Fabric Binaries and SDK

We'll be using the Fabric binaries to setup the HF fabric network[67]. Transactions will be sent with the NodeJS Fabric SDK[38] to one of the peers.

3.2. Ethereum Experiment

For the PoW and PoA algorithm we have devised the following experiment setup shown in figure 3.1. It consists of two sub-networks: the **VM network** and the **Local network**.

The **VM network** contains all of the VM nodes. In the test the **VM L** nodes will be used as miners as they fulfill the hardware requirements of mining with geth[13] and we have a large number of them to scale with. A smaller **VM M** will serve as a Full node, which does not mine, but only syncs with the blockchain. The role of **VM M** will be further explained in the workflow section.

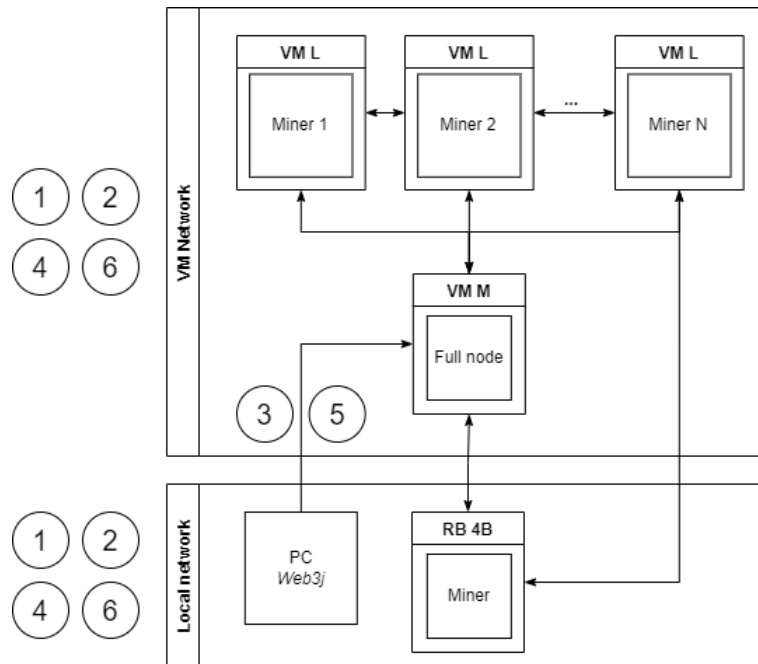


Figure 3.1.: Ethereum network overview

In the **Local network** there are two devices: the PC, which is used as the orchestrator in the test and a physical device which will also play the role of a miner. In the case of the Udoo x86 and Raspberry Pi 4b, there is also an attached energy monitor.

Finally, all nodes are completely interconnected and will therefore propagate all received messages to each other.

3.2.1. Test workflow

For the test we have devised the following general workflow:

1. Start the geth process on all nodes and the physical device. Mining is not enabled yet.
2. Connect all nodes to each other, this ensures full network connectivity.
3. Send transactions to **VM M**. As **VM M** is not a miner, it will store the transactions in its local mempool
4. Start the mining process on the mining nodes
5. Verify that all transactions are successfully mined and part of a block
6. Stop the geth process on all nodes and collect data

For step 1. we ensure that we run the experiment in a clean environment. We launch all nodes firstly as full nodes. For step 2. we connect all nodes explicitly to each other by adding each of their addresses explicitly. This ensures full network connectivity.

During step 3. we send all transactions to the **VM M** node. It will then verify and store them in its mempool (storage for transactions). This way, once step 4. is executed they will be distributed in a fair way to all the mining nodes, removing the advantage a node would get if it received a transaction first.

For step 4. we start the mining process on all **VM L** nodes. This also serves as the start timestamp for the test. For step 5. we keep track of the blocks propagated back to **VM M** until we can confirm that all transactions are accounted for and are part of a mined block.

Finally, at step 5. we stop the mining process and collect all the measurement data that we have. Step 5. also serves as the end timestamp for the test. The difference between the start and end timestamp we will define as the mining duration in the experiment. Steps 1,2,4,6 are executed for all of the nodes, whereas steps 3,5 are done only between the **PC** and the **VM M** node.

Additionally, all transactions use the default gas cost, e.g there are no tips for faster processing of transactions. This would ensure that all blocks are close to their maximum size, which should be close to the theoretical throughput limit for this setup. Moreover, this setup works the same with both PoW and PoA.

3.2.2. Genesis configuration

The experiment is running as part of an Ethereum private network for which we need an initial genesis configuration[13]. An example can be found below. Notable attributes include the unique **chainId** which we use to define our network as well as the **gasLimit** and **difficulty** (or **period** in PoA), which directly affect the performance and are covered in-depth in the next chapter.

Listing 3.1: Genesis config for private network

```
{
  "config": {
    "chainId": 15870,
    # ...
  },
  "nonce": "0x0",
  "timestamp": "0x621f7acc",
  "extraData": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "gasLimit": "0x47b760",
  "difficulty": "0x100000", # or period in PoA
  "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "alloc": [ #...# ]
}
```

As Proof-Of-Authority needs a list of validators, the **extraData** property will hold the public addresses of all miners, so that they are authorized to seal.

3.3. Hyperledger Fabric Experiment

For the Hyperledger Fabric we follow a very similar experiment architecture as presented in figure 3.2. Now on each of the **VM L** we run a Peer, Certificate Authority(CA) and a database docker container. Three **VM M** each run an Orderer docker container, which forms the RAFT consensus group. This is similar to the official Fabric test network[67]. The PC now runs a NodeJS client with the Fabric SDK that is used for sending transactions to the network.

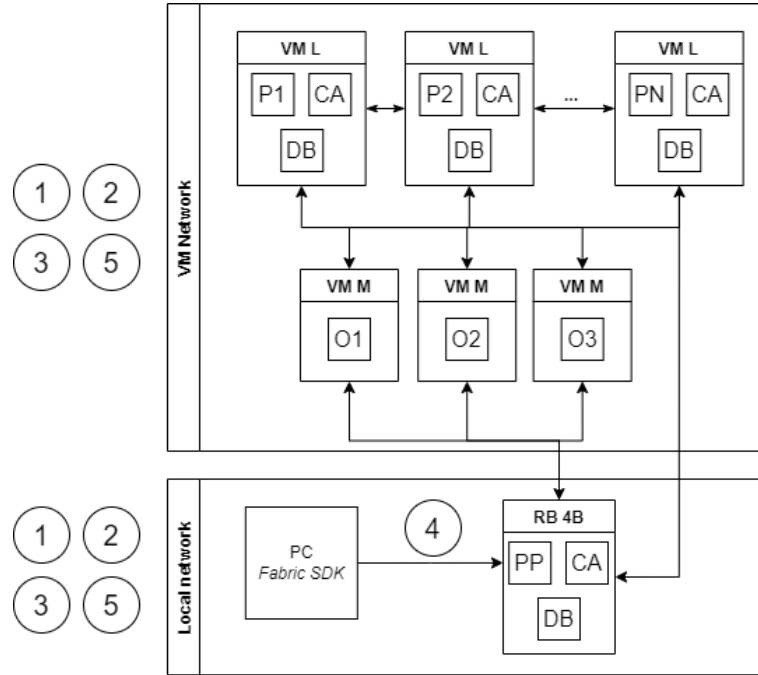


Figure 3.2.: HF network overview

3.3.1. Test workflow

The test on the Fabric instance follows the following steps:

1. Start all the docker containers
2. Create channel and join all peers to channel
3. Deploy chaincode
4. Execute transactions
5. Stop containers and collect measurements

In the case of Fabric it doesn't matter to which peer we send the transaction, as it would also be resent to the other peers in order to gain the required number of endorsements.

3.3.2. Configuration

As Hyperledger Fabric is a permissioned blockchain, it's initial configuration is more in-depth than the previously described genesis. Here need to define all the peers, their certificates and IP addresses explicitly. See appendix A.1.2 for additional information on the setup.

Moreover as Fabric does not have an in-built currency and transaction system, we'll deploy a small chaincode that creates a wallet, which is stored on the blockchain. We'll then used this function as part of the experiment.

Listing 3.2: Chaincode for fabric

```
class CoinContract extends Contract {  
    /**  
     * @param {ctx} context  
     */  
    async createWallet(ctx, walletId) {  
  
        // Check if wallet already exists  
        if(await Wallet.queryWalletByID(ctx, walletId) != null) {  
            throw new Error(`${walletId} already exists`);  
        }  
  
        let cid = new ClientIdentity(ctx.stub);  
  
        return new Wallet(walletId, cid.getMSPID(), 0)  
            .save(ctx);  
    }  
}
```

4. Blockchain Evaluation

4.1. Geth PoW network

In this section we will try to measure the performance and resource consumption of a private PoW network. We will firstly focus on the **throughput** and general performance and how they are influenced in Ethereum. Afterwards we will have a look at the system resource expenditure and total energy consumption to give better context on the energy metrics.

Based on a Ethereum performance research, Schäffer(2019)[57] identified **Block size** and **Block frequency** as the primary bottlenecks in an Ethereum network. Both of these parameters are directly configurable in the genesis file, so we will firstly examine them in-depth.

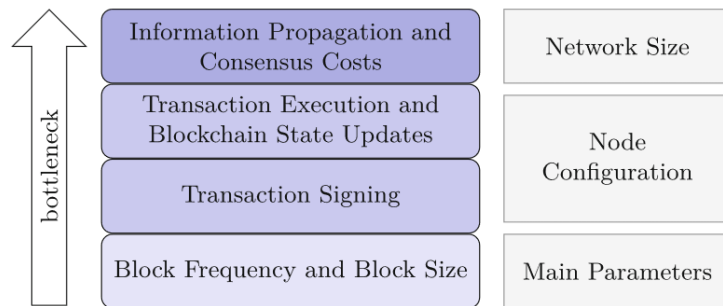


Figure 4.1.: Hierarchy of bottlenecks overview [57]

Block frequency in Ethereum is represented by the difficulty value. As previously mentioned in the Background and Related Work chapter, the difficulty sets the target for the search puzzle [10]. A higher difficulty corresponds to a lower target and subsequently more hashing operations on average to mine a block.

Therefore if the hardware remains unchanged, increasing the difficulty results in a longer execution of the PoW algorithm, longer time between mined blocks and a lower block frequency. A lower difficulty similarly does result in a higher block frequency. Additionally, increasing the hashrate of hardware also increases the block frequency. As the PoW execution is a very resource intensive process, the this parameter should

also relate directly to the energy consumption.

Block size works a little bit differently in Ethereum than in Bitcoin. In Bitcoin there is a fixed block size of 1 MB [46], whereas Ethereum uses the concept of a gas limit [10]. As previously mentioned, each operation executed on the EVM has a fixed gas cost. The gas limit enforces that all operations part of a block have a total gas cost less than or equal to the gas limit. Increasing the gas limit, would therefore increase the number of transactions per block boosting the overall throughput.

To sum up, difficulty impacts the speed at which new blocks are generated and gas limit determines the amount of transactions(information) in each added block. Difficulty is by design self-adjusting and over a long enough period of time, would adjust itself to a target of 15 tps [10]. The general tendency in PoW blockchains is for the difficulty to increase [26] as the market of the blockchain grows.

On the other hand, changing the gas limit seems to be more complicated. For example doubling the gas limit should in theory require half the number of blocks to process the same amount of transactions, practically doubling the throughput. Although, this would substantially improve the expended energy per transaction, it will not change the overall power consumption of the blockchain network. Recall that PoW executes on a block basis and is not influenced by the amount of information in each block. Moreover, substantial increases to the block size would also increase the energy expenditure of the blockchain[58]. Increasing the block size would also increase the speed at which the blockchain grows in terms of storage requirements and would require a higher bandwidth for propagating blocks. The gas limit has been last changed in the London Upgrade [36] and changes to it have been part of heated debate [40].

4.1.1. Test parameters

In table 4.1 you can find the parameters used in the test.

Difficulty	Gas limit	Transactions	Network size	Physical device
0x100000	0x47b760	5000	1..10	RP, Udoo, Fujitsu

Table 4.1.: PoW test parameters

For the **Difficulty** the value $0x100000(1048576_{10})$ was chosen, which on the **VM L** results throughput: 17.78tp/s which is comparable to the mainnet of 15tp/s. The **Gas limit** of $0x47b760(4\,700\,000_{10})$ is the default value for mainnet, so we have kept it the same. 5000 **Transactions** results in approximately 23 blocks based on the default gas limit. **Network size** is the number of nodes in the network and is scaled from 1 up to 10. Finally, the **Physical device** refers to which of the physical devices was used.

4.1.2. Performance

In this section we will go through the general performance metrics. Recall, that we define **Mining duration** as the time since the mining process was started on all nodes until all transactions are verified to be part of the block, which is the end goal of the test. **Mean block frequency** would then be average time in seconds between blocks. For the **throughput** we then derive the following formula:

$$Throughput(tps) = \frac{Transactions}{Number\ of\ Blocks * Block\ frequency} \quad (4.1)$$

Finally, take into consideration the **uncle rate**, e.g the percentage of blocks which are stale.

Device comparison

Firstly, let's have a look at figure 4.2 which shows how the different device types perform on the tests on their own, or in other words as the only miner in the network.

At first glance the Fujitsu is clearly outperforming the other devices with more than 4 times the throughput, but keep in mind that it has a much higher power supply. The Raspberry Pi 4b is the slowest device by a significant margin, although it operates on a 5V power supply.

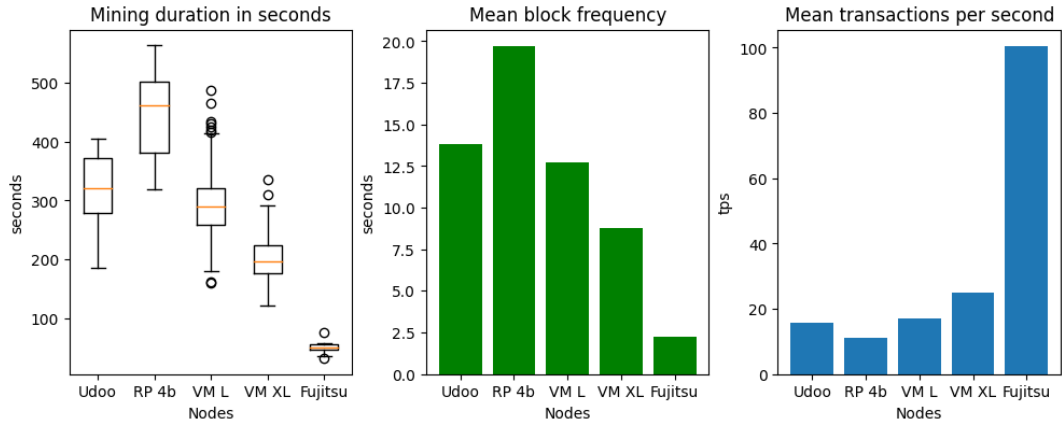


Figure 4.2.: Devices comparison

The VM L and VM XL are part of the VM network, and therefore not directly comparable to other devices in terms of hardware. Moreover, the VM XL has double the amount of cores (8), which is an overall 40% power gain over the VM L.

Network scaling

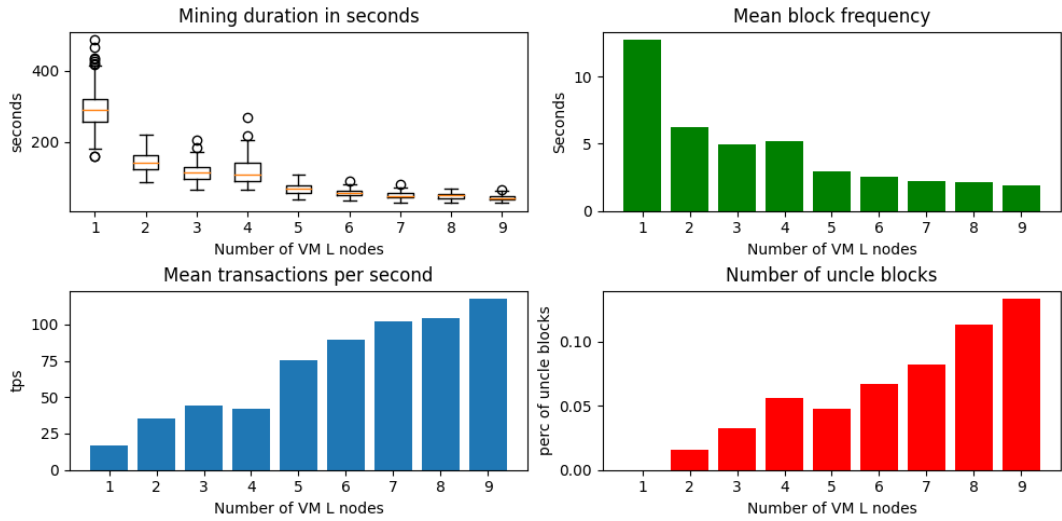


Figure 4.3.: Performance of VM L network

Now let's have a look at how the performance scales with the size of a network. Let's take a homogeneous network made up entirely out of the **VM L** nodes as shown in

figure 4.3. Additionally, keep in mind that this experiment was done entirely on the VM network, where network latency was not a factor.

It seems that scaling the network causes a logarithmic decrease performance, generally converging to 50 seconds for the total mining duration and a 2 second block frequency. A similar tendency was also observed in the work of Schäffer(2019) [57], where the measured throughput was lower than the theoretical throughput.

A possible, explanation for this could be the increasing rate of uncle blocks. Generally, when the propagation delay of blocks is higher than the block frequency, blocks can't be built effectively one after another, causing branches in the blockchain and the occurrence of uncle blocks [20]. This of course represents a waste of computational resources as uncle blocks do not contribute to new information being added to the blockchain.

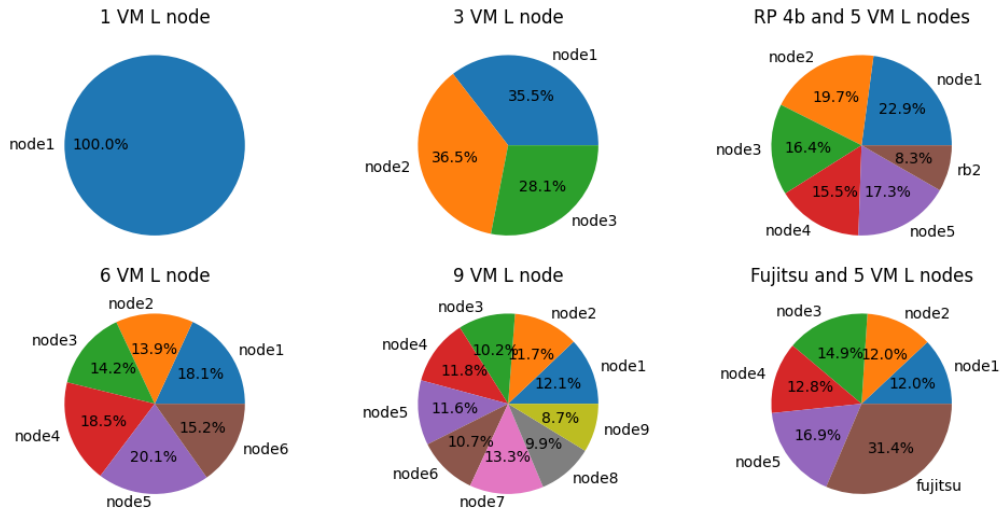


Figure 4.4.: Block distribution

Finally, let's have a look at the block distribution in figure 4.6, e.g how many blocks has each miner successfully mined. Following the formula 2.2 for winning the search puzzle, we should expect equal shares across the nodes. This seems to be roughly the case for the homogenous networks, but there is still some variance. This could be the cause of some internal issues in the configuration of the VM network node, which causes a bottleneck like in figure 4.1. With mixed network there are pronounced differences, generally the Raspberry Pi is slower and therefore achieves a lower share, likewise the fujitsu has the highest share in its respective network.

Mixed networks

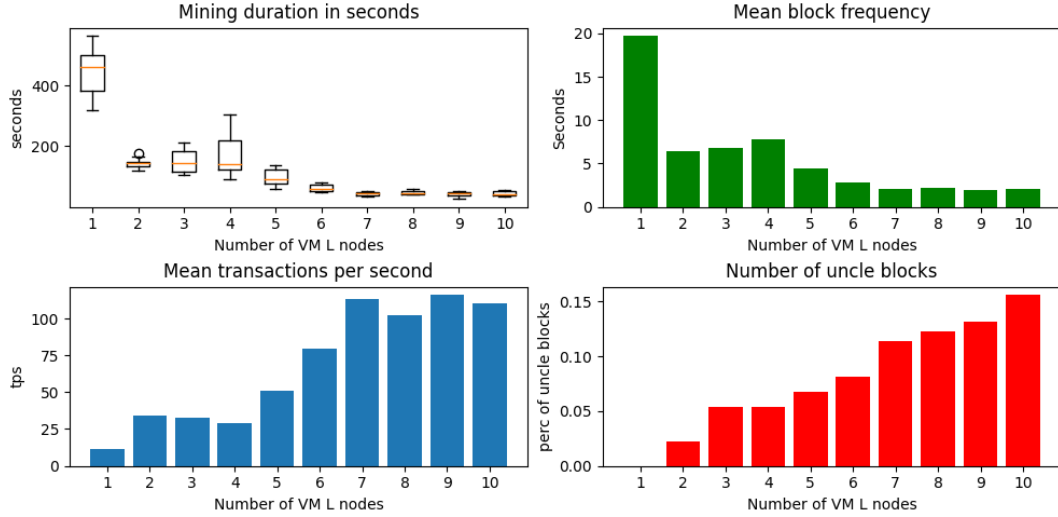


Figure 4.5.: Raspberry Pi 4b network performance

Now let's have a look at figure 4.5, which shows mixed networks consisting of a physical device and a number of VM L nodes from the VM network. Generally, they follow the same tendencies from above, but have a more pronounced plateau on performance gains between network size 7 to 10, where they again converge on 50s and 2s respectively. Additionally, there is a higher rate of uncle blocks, most likely the result of the increased network latency from adding a device from an external network.

4.1.3. System resource consumption

Let's consider the system resource consumption. With the glances tool we can extract the system statistics before the test and during the load of the mining process. The results are in table 4.2 and table 4.3.

	cpu_total	mem_total	mem_percent
count	77.000000	7.700000e+01	77.000000
mean	3.270130	3.974017e+09	9.862338
std	3.420683	0.000000e+00	0.048772
min	1.900000	3.974017e+09	9.800000
25%	2.400000	3.974017e+09	9.800000
50%	3.000000	3.974017e+09	9.900000
75%	3.100000	3.974017e+09	9.900000
max	32.400000	3.974017e+09	9.900000

Table 4.2.: Normal OS stats

	cpu_total	mem_total	mem_percent
count	16.000000	1.600000e+01	16.000000
mean	94.056250	3.974017e+09	17.506250
std	16.088421	0.000000e+00	0.156924
min	35.100000	3.974017e+09	17.300000
25%	98.800000	3.974017e+09	17.375000
50%	99.300000	3.974017e+09	17.500000
75%	99.500000	3.974017e+09	17.625000
max	99.600000	3.974017e+09	17.700000

Table 4.3.: Under load stats

As we can see the node uses almost all the available CPU and approximately 690MB of memory. Generally, this holds true for all of the tested devices, which will consume all the available CPU resource during the mining process.

With the pprof tool we can extract the CPU measures for the entire test. This information includes the cumulative time each goroutine(function in the context of go) spends on CPU:

Listing 4.1: pprof output PoW

```
Showing nodes accounting for 632.20s, 71.14% of 888.64s total
Dropped 1119 nodes (cum <= 4.44s)
Showing top 10 nodes out of 38
      flat flat% sum% cum%
      0 0% 0% 862.09s 97.01% consensus/ethash.(*Ethash).Seal.func1
    0.96s 0.11% 0.11% 862.09s 97.01% consensus/ethash.(*Ethash).mine
    0.28s 0.032% 0.14% 856.02s 96.33% consensus/ethash.hashimotoFull
   54.85s 6.17% 6.31% 855.74s 96.30% consensus/ethash.hashimoto
  566.16s 63.71% 70.02% 566.16s 63.71% runtime.memmove
    1.09s 0.12% 70.15% 77.52s 8.72% crypto/sha3.(*state).Read
    5.70s 0.64% 70.79% 75.67s 8.52% crypto/sha3.(*state).padAndPermute
    0.73s 0.082% 70.87% 72.72s 8.18% crypto.Keccak512
    1.66s 0.19% 71.06% 68.54s 7.71% crypto/sha3.(*state).permute
    0.77s 0.087% 71.14% 66.83s 7.52% crypto.Keccak256
```

From the text output we can see that Ethash constitutes approximately 97% of the total CPU consumption.

Figure 4.6 shows several CPU related metrics for the homogeneous VM L network. These include the aforementioned mining duration, the mean time each node spends on the CPU and finally the total time all nodes spend on the CPU.

As seen on the graph, the mean time spend on the CPU is a factor of the mining duration. This is the case, because the start and end timestamps from the experiment workflow are the only time mining is allowed, which coincidentally is where 97% of the CPU expenditure happens. Additionally, as the VM L nodes are multi-core, each core is executing the algorithm in parallel, hence why the mean CPU time per node is a factor of the mining duration.

Finally, whereas the mining duration sinks logarithmically, so does the cumulative CPU time of all nodes grow as each additional node will dedicate its whole CPU resources. This of course constitutes a substantial increase in total computational resources. A single node will need on average 293s to mine all transactions and consume 820s of CPU time, whereas a 9 VM L will do it in 43s and consume a total of 2076s.

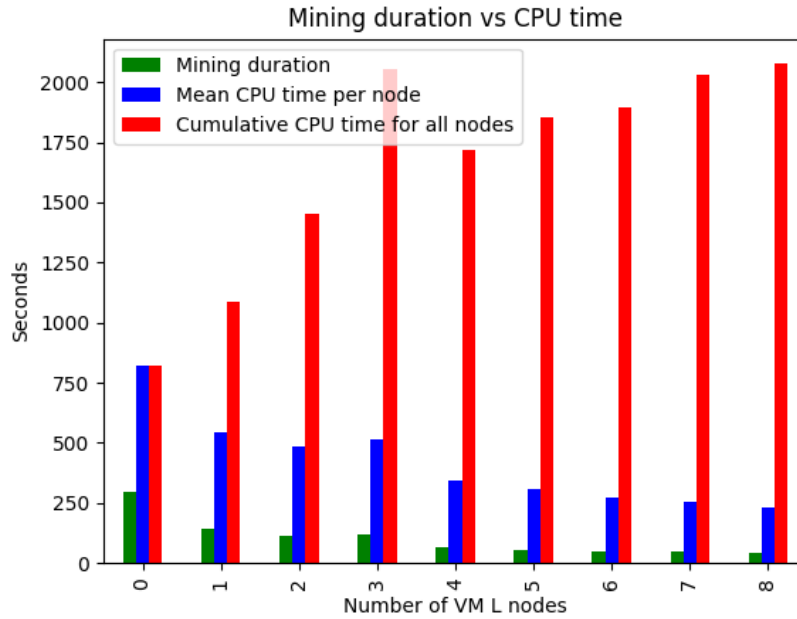


Figure 4.6.: CPU time on VM L network

4.1.4. Energy consumption

Figure 4.7 shows the power consumption during the test runs for both measurement devices: Raspberry Pi 4b and Udoo x86. We have used the total energy expenditure for the device during the experiment run. Similarly to the CPU time graph, the total power consumption seems to be correlated to the mining duration and mean CPU time. This makes sense as adding new miners to the network increases the overall hash power, which decreases the time needed to mine blocks, where the bulk of the resource expenditure is. Overall, both devices seem to have consumed a similar amount of energy.

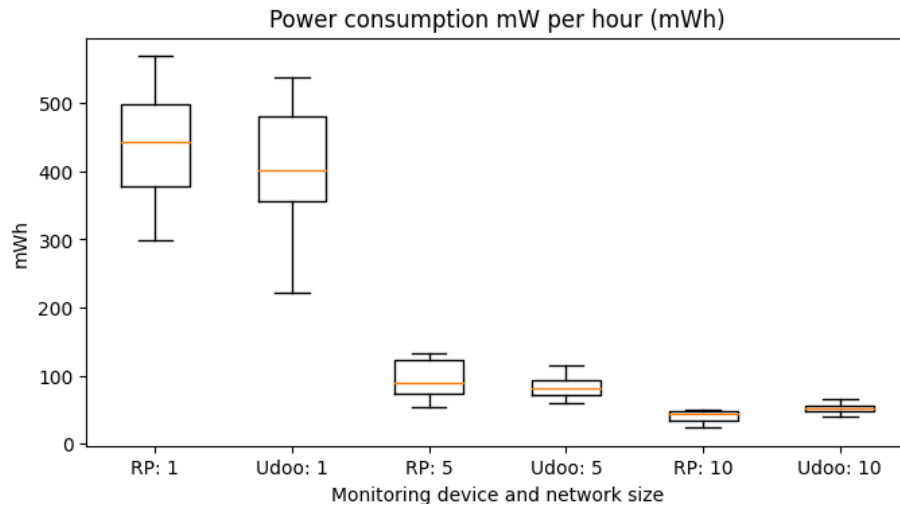


Figure 4.7.: Average power consumption

Additionally, we can take a look at the values when idle and under load to determine the exact overhead of mining:

Device	Idle	Under load	Overhead
Raspberry Pi 4b	5674mW	7616mW	1942 mW
Udoo Board x86	3435mW	9305mW	5807mW

Table 4.4.: Device list

Interestingly, the Udoo has a much lower idle power consumption compared to the RP 4b, which in turn leads to a much larger overhead during mining. Moreover, based on the overhead you can determine the energy expenditure only for the mining process.

4.1.5. Energy waste

We define wasted energy as energy expenditure that either does not bring benefit to the individual miner or the network as a whole.

Figure 4.8 shows two types of energy expenditure: energy expenditure that has resulted in the successful mining of a block, and energy expenditure that has not. In this context, the latter can be seen as a waste as it does not allow the miner to propose a new block, nor gives him the full mining reward. As a side note, in Bitcoin [46] this energy expenditure is a complete waste as stale blocks do not give a reward. In Ethereum [10] this is not entirely true as an uncle block could still give a mining reward. In the graph we only consider winning blocks for simplicity. In conclusion, with increasing the network size, the overall energy expenditure drops, but depending on the relative hash power of the device, only part of this energy will successfully lead to the creation of a block.

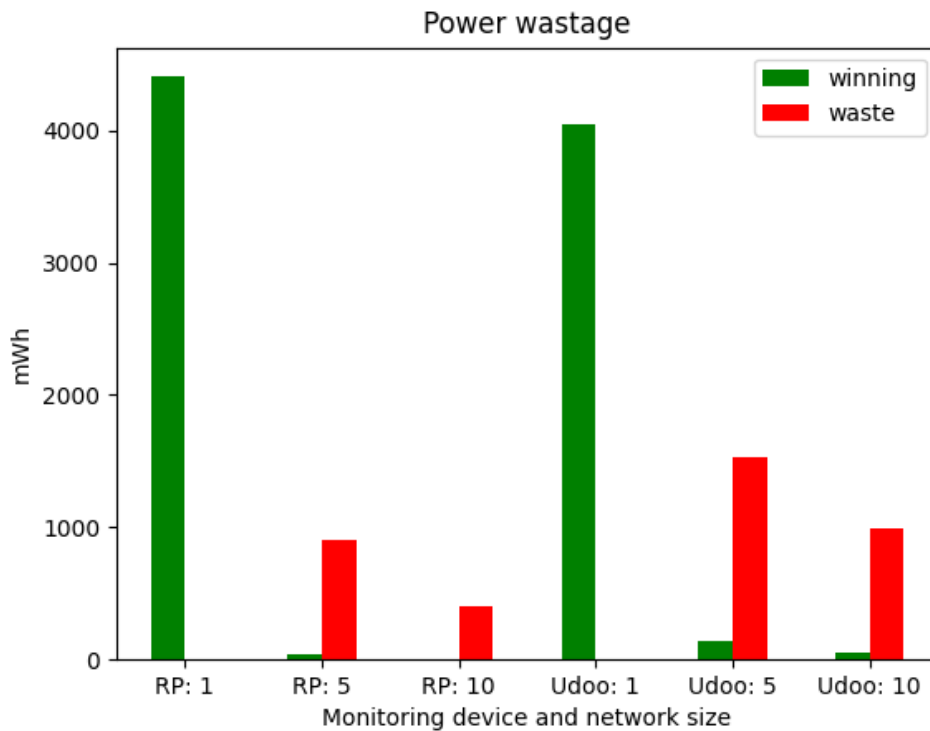


Figure 4.8.: Power wastage

4.2. Geth PoA network

For the Proof-Of-Authority network we can reuse the already existing test setup with slight changes to the test parameters:

Period	Gas limit	Transactions	Network size	Physical device
2s	0x47b760	5000	1..10	RP, Udoo

Table 4.5.: PoA test parameters

As there is not search puzzle to solve there is no difficulty parameter. Instead a period defines the minimum time window between sealed blocks. We have chosen 2s, as this matches our best block frequency from the PoW test.

4.2.1. Performance and energy consumption

Executing the test setup on the PoA network leads to the results in figure 4.9. One can make a couple of observations:

- the mining duration has much less variance compared to PoW
- parameters are much more consistent when scaling the network
- the average CPU time a order of magnitude lower

The pprof tool then produces the following output:

Listing 4.2: pprof output PoA

```
Showing nodes accounting for 0.34s, 0.94% of 36.26s total
flat flat% sum% cum cum%
0.16s 0.44% 0.44% 0.22s 0.61% clique.(*Clique).FinalizeAndAssemble
0.07s 0.19% 0.63% 0.07s 0.19% clique.(*Clique).Finalize
0.03s 0.083% 0.72% 0.04s 0.11% clique.(*Clique).Seal
```

In PoA the consensus algorithm amounts to 0.94% of the total CPU time in comparison to 97% in PoW. This is a reduction from 800s of CPU time in the worst case in the PoW network to approximately 340ms. Therefore, PoA performs not only much more consistently in the limited scaling we have done, but also expends less energy due to the missing mining overhead.

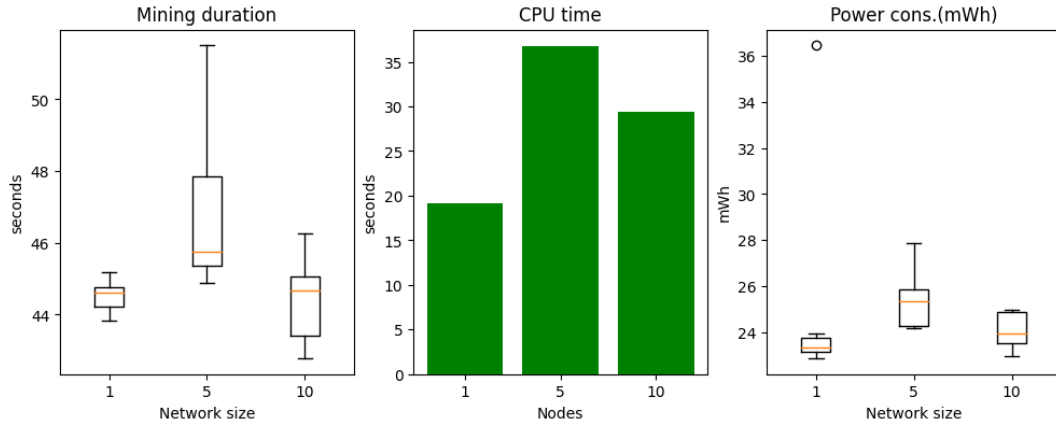


Figure 4.9.: PoA performance

4.3. Hyperledger Fabric

In Hyperledger Fabric the **Block frequency** and **Block size** are also defined in the initial channel configuration as BatchTimeout and BatchSize. Moreover, Fabric parameters are more granular supporting setting the maximum amount of messages, preferred and absolute bytes per batch.

Listing 4.3: Batch settings

```
# Batch Timeout: The amount of time to wait before creating a batch
BatchTimeout: 2s

# Batch Size: Controls the number of messages batched into a block
BatchSize:

# Max Message Count: The maximum number of messages
# to permit in a batch
MaxMessageCount: 10

# Absolute Max Bytes: The absolute maximum number of bytes allowed for
# the serialized messages in a batch.
AbsoluteMaxBytes: 99 MB

# Preferred Max Bytes: The preferred maximum number of bytes
# allowed for the serialized messages in a batch. A message
```

```
# larger than the preferred max bytes will result in a batch
# larger than preferred max bytes.
PreferredMaxBytes: 512 KB
```

On the Hyperledger Fabric network consisting of 5 peers, one of which was running on the Udoo x86, and 3 orderers we achieved the results in table 4.6. For 1500 transactions, which consisted of a wallet creation, we achieved an average 34.6 tps with a power consumption of 33 mWh. Please note, that the achieved throughput is due to our setup limitations which are explained in detail in appendix A.1.2.

Transactions	Duration	Throughput	Power consumption
2000	57.7s	34.6 tps	33mWh

Table 4.6.: HF Performance

Generally, the performance of Hyperledger Fabric varies greatly depending on the hardware, chosen database, transaction sizes as well as network configuration. Papers vary highly with throughput measurements ranging from 150 tps [47] to over 9000 tps[45].

5. Conclusion

In this thesis we analyzed the existing literature on the energy consumption of blockchain protocols and afterwards we ran experiments on small scale private networks. In this chapter we will summarize our findings and draw conclusions.

5.1. Summary

5.1.1. Global energy consumption

In Chapter 2 we examined the current reports on blockchain power consumption. We have summarized the discussed blockchains in table 5.2. Data is based on July 2022 from Coinmarket [15] and Digieconomist [6]. Hyperledger Fabric and Clique are not part of the table as they are used in private networks, so estimates on their global power consumption are not possible.

Blockchain	Consensus	Market size	Est. power consumption
Bitcoin(BTC)	PoW	\$378,019,397,370	131.26 TWh
Ethereum(ETH)	PoW	\$130,365,178,156	55.23 TWh
Energy Web(EWT)	PoA	\$220,112,982	7.5 KW[48]

Table 5.1.: Blockchain protocols summary

Most studies use an estimation approach, first proposed by Vranken(2017) [68] and Krause and Tolaymat (2018) [42]. The formula and further info can be found in Chapter 2. Actual energy figures from mining operations are not publicly available.

PoW blockchains have seen the highest market share of 87% [33]. The figures for Bitcoin and Ethereum point to an annual energy consumption that is greater than the majority of countries [44]. However conclusions on the environmental impact of blockchain systems should be done carefully, as a large set (76%) use renewables in their mining operations [48].

Researchers agree that non-PoW blockchains have orders of magnitude lower energy consumption [58][48]. Additionally, the EU Commission [48] further argues that the energy consumption difference between PoW and non-PoW is so large, that it is questionable whether comparisons between them are useful at all.

5.1.2. Experiment results

Our experiments we have mainly tested the Ethereum PoW network, with side tests on Ethereum PoA and Fabric CFT. Our highlights include:

- Difficulty and gas limit directly impact the performance, CPU utilization and energy consumption of the Proof-Of-Work network
- Decreasing block frequency substantially, either by reducing the difficulty or increasing the hashpower in comparison to the difficulty, drastically increases the rate of uncle (stale) blocks. A high percentage of stale blocks represents a significant throughput decrease, as well as additional energy waste.
- By switching Ethash with Clique we have reduced the amount of CPU time from 97% to 0.94%. This comes more in-line with the expected 99.5% energy decrease in Ethereum 2.0 [30]

Finally, in the following table we have summarized our energy measurements:

Network setup	Network size	Throughput(tps)	Energy consumption
Raspberry Pi 4b(PoW)	1	11.03	440.64 mWh
Raspberry Pi 4b(PoW)	5	51.23	94.00 mWh
Raspberry Pi 4b(PoW)	10	109.94	40.48 mWh
Udoo x86 (PoW)	1	15.73	405.11 mWh
Udoo x86 (PoW)	5	72.30	83.14 mWh
Udoo x86 (PoW)	10	111.05	51.97 mWh
Udoo x86 (PoA)	1	113.64	24.66 mWh
Udoo x86 (PoA)	5	106.09	25.51 mWh
Udoo x86 (PoA)	10	111.51	24.07 mWh
Udoo x86 (Fabric)	5	34.6	33 mWh

Table 5.2.: Blockchain protocols summary

5.2. Have your cake and eat it too

Having already discussed the estimates of global blockchain power consumption and having performed experiments on smaller networks, let's come back to the thesis title. Based on what we have analyzed so far, the operation of a large scale "public permissionless blockchain" that has both a "low energy consumption" and "high throughput" should be technically feasible. Especially, the Ethereum 2.0 [63] has promised to alleviate the high energy costs of PoW as well as drastically improve blockchain performance and scalability.

Hyperledger Fabric and Energy Web provide alternatives, but they do not really fit the aforementioned use case. Fabric is highly performant, but its permissioned consensus makes it unscalable to a global level [70]. Energy Web provides the same functionality as Ethereum for a fraction of the energy costs, but its permissioned PoA does not fully fit the "public permissionless blockchain" image of Bitcoin or Ethereum.

The authors personal opinion on the matter is that the direction, in which blockchains are heading are mostly based on economical and not technical principles. One such example is Bitcoin [46], which is extremely outdated and slow by current blockchain standards. Still, it is the biggest blockchain market by a huge margin [15], and likely to stay like this for the foreseeable future. Therefore we believe, that the first movers advantage largely defines the financial success of a blockchain. For example, Bitcoin was the first decentralized currency and Ethereum [10] was the first to utilize Smart Contracts [10]. Bitcoin and Ethereum are still the de facto biggest players on the market. Therefore, as an answer to the question "Can you have your cake and eat it too?", the author answers - in theory yes, but whether or not this will become the new standard is outside of his competencies.

5.3. Future Work

In this thesis we have focused entirely on Ethereum, PoW and PoA, and Hyperledger Fabric, however, there are various other blockchain protocols that can be examined. Such examples are hybrid-consensus models, DAG-models and pBFT blockchains [70]. Once as Ethereum 2.0 has released, in 2023 or later [63], it should be a prime candidate for in-depth analysis of its energy costs.

Other expenditures on energy like networking, memory and disk usage become negligible once compared with the majority of the energy costs in PoW generated from mining, thus it would make sense to focus on those aspects of blockchain networks, especially non-PoW protocols.

A. Reproducibility

This appendix contains short descriptions on how the different tool are used in order to aid reproducibility. Please refer also to the accompanying repository.

A.1. Networks

In the repository both networks are in different directories and operate completely independently.

A.1.1. Geth network

For detailed information please consult the *readme.md*.

Requirements

- geth version *1.10.16-stable-20356e57* or newer
- go lang version *go1.17.6 linux/amd64* or newer

The test network deployment is generally automated with scripts. You can modify *constants.sh* and *transaction_test.sh* to change the addresses, which peers are enabled and what the test parameters are.

The setup is distributed, e.g each geth process runs on a different VM. Synchronisation is done through bash scripts. The client that sends transactions and manages the miners is a Java Application with the Web3j library.

The general workflow of the test is then the following:

1. Generate certificates for each of the peers
2. Generate a new genesis file or import an existing one (the geth tool *puppeth* can be used to generate one)
3. Initialize each peer with the genesis block
4. Start all peers

5. Connect each of them with the *addPeer()* command
6. Send transactions to the designated passive peer
7. Start mining
8. Verify continuously whether all transactions are part of a block
9. When confirmed stop mining and output test information to csv
10. Stop all the peers, depending on config, a CPU profile is generated on exit
11. Connect to each of the peers and download the CPU profile

A.1.2. Fabric network

For detailed information please consult the *readme.md*.

Requirements

- docker and docker-compose on each VM that runs a Fabric node
- Hyperledger Fabric binaries on the orchestrator PC

The test network deployment is generally automated with scripts. You can modify the *configtx.yaml* and *crypto-config.yaml* for the initial Fabric settings.

scripts/00_enviromental_variables.sh contains the settings for the script automation.

The setup is also distributed. All of the docker containers are defined in the *docker-compose.yaml*, which is then distributed to each of the VMs. Afterwards on each VM the specific containers can be run. The client that sends the transaction is a nodejs application with the Fabric SDK.

The general workflow of the test is then the following:

1. Generate cryptographic configuration for each peer and orderer
2. Create genesis block
3. Start network
4. Create channel and join peers to channel
5. Deploy chaincode for test
6. Execute transactions

7. Measure performance
8. Stop network and remove containers

As mentioned in the main body of the thesis the achieved performance was much lower than the theoretical one. The reason is that transactions are sent synchronously in Fabric. The process starts when a transaction is submitted and ends when the transaction is committed to the state database [38]. There is no async sending or batching like in the Geth network, where all transactions can be asynchronously dumped without problems. Therefore to send a higher load of transactions, parallelism is required on the side of the client. Due to time constraints we were not able to develop a setup that sufficiently loads the HF network.

A.2. Data Collection

The following are the tools used for data collection.

A.2.1. pprof

The pprof tool can be used on one of the download CPU profile to extract the amount of time spent in each goroutine(go function). For example the following command outputs the goroutines descending based on their cumulative time on the CPU:

```
go tool pprof -text -cum cpuprofile_node1-1.profile
```

A.2.2. Monsoon PowerTool

The Monsoon PowerTool is used to operate the High Voltage Monsoon Power Monitor [35]. With this software we can set the input voltage for our physical devices and record the current and power. Generally, we run the power tool during the entire test session, and then based on timestamp extract the relevant timespans that we want to investigate.

List of Figures

1.1. Comparison of bitcoin energy consumption in TWh [5]	1
2.1. DAG size growth [16]	8
2.2. Mining with Ethash	10
2.3. Distribution of mining rewards for modified Ghost. Blocks colours correspond to miners. On the right is the total mining reward for this branch	11
2.4. Hyperledger Fabric Network Overview	16
2.5. Market capitalization and est. bounds of the top 5 PoW blockchains [58]	19
2.6. Market share on consensus algorithm[33]	20
2.7. PoS energy consumption comparison [53]	22
3.1. Ethereum network overview	25
3.2. HF network overview	28
4.1. Hierarchy of bottlenecks overview [57]	30
4.2. Devices comparison	33
4.3. Performance of VM L network	33
4.4. Block distribution	34
4.5. Raspberry Pi 4b network performance	35
4.6. CPU time on VM L network	38
4.7. Average power consumption	39
4.8. Power wastage	40
4.9. PoA performance	42

List of Tables

2.1. Mining devices	21
3.1. Device list	23
4.1. PoW test parameters	32
4.2. Normal OS stats	36
4.3. Under load stats	36
4.4. Device list	39
4.5. PoA test parameters	41
4.6. HF Performance	43
5.1. Blockchain protocols summary	44
5.2. Blockchain protocols summary	45

Bibliography

- [1] *3rd Global Cryptoasset Benchmarking Study with Apolline Blandin*. Newsroom – Compass. Oct. 5, 2020. URL: <https://compassmining.io/education/3rd-global-cryptoasset-benchmarking-study-with-apolline-blandin/> (visited on 07/09/2022).
- [2] *Amazon.de: Low Prices in Electronics, Books, Sports Equipment & more*. URL: https://www.amazon.de/-/en/ref=nav_logo (visited on 07/12/2022).
- [3] *Bangladesh Population*. URL: <https://www.worldometers.info/world-population/bangladesh-population/> (visited on 07/12/2022).
- [4] A. Barger, Y. Manevich, H. Meir, and Y. Tock. *A Byzantine Fault-Tolerant Consensus Library for Hyperledger Fabric*. Number: arXiv:2107.06922. July 14, 2021. arXiv: 2107.06922[cs].
- [5] *Bitcoin Devours More Electricity Than Many Countries*. URL: <https://www.statista.com/chart/18632/estimated-annual-electricity-consumption-of-bitcoin/> (visited on 07/12/2022).
- [6] *Bitcoin Energy Consumption Index*. Digiconomist. URL: <https://digiconomist.net/bitcoin-energy-consumption/> (visited on 07/12/2022).
- [7] *BITMAIN. Cryptocurrency Mining Hardware & Solutions | BITMAIN*. URL: <https://www.facebook.com/Bitmain> (visited on 07/12/2022).
- [8] *Blockchain network — hyperledger-fabricdocs main documentation*. URL: <https://hyperledger-fabric.readthedocs.io/en/release-2.2/network/network.html> (visited on 06/22/2022).
- [9] *Blockchain Technology Market Size Report, 2022-2030*. URL: <https://www.grandviewresearch.com/industry-analysis/blockchain-technology-market> (visited on 07/11/2022).
- [10] V. Buterin. “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.” In: (), p. 36.
- [11] M. Castro and B. Liskov. “Practical Byzantine Fault Tolerance.” In: (), p. 14.
- [12] *Clients - ethernodes.org - The Ethereum Network & Node Explorer*. URL: <https://www.ethernodes.org/> (visited on 07/03/2022).

- [13] *Command-line Options | Go Ethereum*. URL: <https://geth.ethereum.org/docs/interface/command-line-options> (visited on 07/03/2022).
- [14] *Cryptocurrency Market Capitalization*. URL: <https://www.slickcharts.com/currency> (visited on 06/26/2022).
- [15] *Cryptocurrency Prices, Charts And Market Capitalizations*. CoinMarketCap. URL: <https://coinmarketcap.com/> (visited on 07/12/2022).
- [16] *DAG size calculator and calendar*. minerstat. URL: <https://minerstat.com/dag-size-calculator> (visited on 06/30/2022).
- [17] *Digital Currencies and Energy Consumption*. URL: <https://www.imf.org/en/Publications/fintech-notes/Issues/2022/06/07/Digital-Currencies-and-Energy-Consumption-517866> (visited on 07/11/2022).
- [18] *Docker Documentation*. Docker Documentation. July 1, 2022. URL: <https://docs.docker.com/> (visited on 07/03/2022).
- [19] *Dogecoin*. URL: <https://dogecoin.com/> (visited on 07/11/2022).
- [20] L. Eichhorn. "Exploring Latency Boundaries of Blockchains in Edge Computing Networks Using Emulation." Technical University Munich, 2021.
- [21] *EIP-225: Clique proof-of-authority consensus protocol*. Ethereum Improvement Proposals. URL: <https://eips.ethereum.org/EIPS/eip-225> (visited on 07/10/2022).
- [22] *Energy efficiency directive*. URL: https://energy.ec.europa.eu/topics/energy-efficiency/energy-efficiency-targets-directive-and-rules/energy-efficiency-directive_en# (visited on 07/13/2022).
- [23] *Energy Web*. Energy Web. URL: <https://energyweb.org/> (visited on 07/10/2022).
- [24] *Ethash*. ethereum.org. URL: <https://ethereum.org> (visited on 06/30/2022).
- [25] *Ethereum (ETH) Ethash | Mining Pools*. URL: <https://miningpoolstats.stream/ethereum> (visited on 07/11/2022).
- [26] *Ethereum Difficulty Chart - ETH Difficulty*. URL: <https://www.coinwarz.com/mining/ethereum/difficulty-chart> (visited on 07/09/2022).
- [27] *Ethereum Energy Consumption Index*. Digiconomist. URL: <https://digiconomist.net/ethereum-energy-consumption/> (visited on 07/11/2022).
- [28] *Ethereum upgrades (formerly 'Eth2')*. ethereum.org. URL: <https://ethereum.org> (visited on 07/11/2022).
- [29] *ethereum/go-ethereum*. original-date: 2013-12-26T13:05:46Z. July 3, 2022.

- [30] E. Foundation. *Ethereum's energy usage will soon decrease by ~99.95%*. URL: <https://blog.ethereum.org/2021/05/18/country-power-no-more/> (visited on 07/13/2022).
- [31] U. Gellersdörfer, L. Klaaßen, and C. Stoll. "Energy Consumption of Cryptocurrencies Beyond Bitcoin." In: *Joule* 4.9 (Sept. 16, 2020), pp. 1843–1846. ISSN: 2542-4351. DOI: 10.1016/j.joule.2020.07.013.
- [32] *Glances - An Eye on your system*. URL: <https://nicolargo.github.io/glances/> (visited on 07/03/2022).
- [33] *Global ranking of consensus mechanisms | Unblocktalent*. URL: <https://www.unblocktalent.com/topics/building-blocks/consensus/consensus-ranking/> (visited on 07/12/2022).
- [34] *Görli Testnet*. Görli Testnet. URL: <https://goerli.net> (visited on 07/10/2022).
- [35] *High Voltage Power Monitor | Monsoon Solutions | Bellevue*. Monsoon Solutions. URL: <https://www.msoon.com/high-voltage-power-monitor> (visited on 07/02/2022).
- [36] *History and Forks of Ethereum*. ethereum.org. URL: <https://ethereum.org> (visited on 07/04/2022).
- [37] *Hyperledger Besu*. Hyperledger Foundation. URL: <https://www.hyperledger.org/use/besu> (visited on 07/11/2022).
- [38] *Hyperledger Fabric SDK for Node.js*. fabric-sdk-node. URL: <https://hyperledger.github.io/fabric-sdk-node/> (visited on 07/12/2022).
- [39] *Hyperledger Indy*. Hyperledger Foundation. URL: <https://www.hyperledger.org/use/hyperledger-indy> (visited on 07/11/2022).
- [40] *Increasing ETH's Gas Limit: What we can safely do today*. Ethereum Research. Oct. 16, 2020. URL: <https://ethresear.ch/t/increasing-eth-s-gas-limit-what-we-can-safely-do-today/8121> (visited on 07/04/2022).
- [41] M. Jakobsson and A. Juels. "Proofs of Work and Bread Pudding Protocols(Extended Abstract)." In: *Secure Information Networks*. Ed. by B. Preneel. Boston, MA: Springer US, 1999, pp. 258–272. ISBN: 978-1-4757-6487-1 978-0-387-35568-9. DOI: 10.1007/978-0-387-35568-9_18.
- [42] M. J. Krause and T. Tolaymat. "Quantification of energy and carbon costs for mining cryptocurrencies." In: *Nature Sustainability* 1.11 (Nov. 2018). Number: 11 Publisher: Nature Publishing Group, pp. 711–718. ISSN: 2398-9629. DOI: 10.1038/s41893-018-0152-7.
- [43] Linzhi Corp. *2019 ETC Summit, E1400 Ethash Architecture Overview*. Oct. 6, 2019.

- [44] *List of countries by electricity consumption*. In: *Wikipedia*. Page Version ID: 1089576754. May 24, 2022.
- [45] T. Nakaike, Q. Zhang, Y. Ueda, T. Inagaki, and M. Ohara. "Hyperledger Fabric Performance Characterization and Optimization Using GoLevelDB Benchmark." In: *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). May 2020, pp. 1–9. doi: 10.1109/ICBC48266.2020.9169454.
- [46] S. Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System." In: (), p. 9.
- [47] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif. "Performance Analysis of Hyperledger Fabric Platforms." In: *Security and Communication Networks* 2018 (Sept. 9, 2018), pp. 1–14. issn: 1939-0114, 1939-0122. doi: 10.1155/2018/3976093.
- [48] *New Thematic Report: Energy Efficiency of Blockchain Technologies*. EUBlockchain. URL: <https://www.eublockchainforum.eu/news/new-thematic-report-energy-efficiency-blockchain-technologies> (visited on 07/08/2022).
- [49] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz. "Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities." In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 85727–85745. issn: 2169-3536. doi: 10.1109/ACCESS.2019.2925010.
- [50] *Peercoin whitepaper - whitepaper.io*. URL: <https://whitepaper.io/document/139/peercoin-whitepaper> (visited on 07/02/2022).
- [51] *Peers — hyperledger-fabricdocs main documentation*. URL: <https://hyperledger-fabric.readthedocs.io/en/release-2.3/peers/peers.html> (visited on 07/11/2021).
- [52] *Petition: Ban the mining of and trade in "proof of work" cryptocurrencies within the UK*. Petitions - UK Government and Parliament. URL: <https://petition.parliament.uk/petitions/601629> (visited on 07/11/2022).
- [53] M. Platt, J. Sedlmeir, D. Platt, J. Xu, P. Tasca, N. Vadgama, and J. I. Ibañez. "The Energy Footprint of Blockchain Consensus Mechanisms Beyond Proof-of-Work." In: *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C). ISSN: 2693-9371. Dec. 2021, pp. 1135–1144. doi: 10.1109/QRS-C55045.2021.00168.
- [54] *pprof package - net/http/pprof - Go Packages*. URL: <https://pkg.go.dev/net/http/pprof> (visited on 07/03/2022).

- [55] *Proof-of-stake (PoS)*. ethereum.org. URL: <https://ethereum.org> (visited on 07/11/2022).
- [56] *Raft Consensus Algorithm*. URL: <https://raft.github.io/> (visited on 06/22/2022).
- [57] M. Schäffer, M. di Angelo, and G. Salzer. "Performance and Scalability of Private Ethereum Blockchains." In: *Business Process Management: Blockchain and Central and Eastern Europe Forum*. Ed. by C. Di Ciccio, R. Gabryelczyk, L. García-Bañuelos, T. Hernaus, R. Hull, M. Indihar Štemberger, A. Kő, and M. Staples. Vol. 361. Series Title: Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2019, pp. 103–118. ISBN: 978-3-030-30428-7 978-3-030-30429-4. DOI: 10.1007/978-3-030-30429-4_8.
- [58] J. Sedlmeir, H. U. Buhl, G. Fridgen, and R. Keller. "The Energy Consumption of Blockchain Technology: Beyond Myth." In: *Business & Information Systems Engineering* 62.6 (Dec. 1, 2020), pp. 599–608. ISSN: 1867-0202. DOI: 10.1007/s12599-020-00656-x.
- [59] *Sharding*. ethereum.org. URL: <https://ethereum.org> (visited on 07/11/2022).
- [60] *Solidity — Solidity 0.8.15 documentation*. URL: <https://docs.soliditylang.org/en/v0.8.15/> (visited on 07/02/2022).
- [61] Y. Sompolinsky and A. Zohar. "Secure High-Rate Transaction Processing in Bitcoin." In: *Financial Cryptography and Data Security*. Ed. by R. Böhme and T. Okamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 507–527. ISBN: 978-3-662-47854-7.
- [62] *The Beacon Chain*. ethereum.org. URL: <https://ethereum.org> (visited on 07/11/2022).
- [63] *The Merge*. ethereum.org. URL: <https://ethereum.org> (visited on 07/11/2022).
- [64] *The Ordering Service — hyperledger-fabricdocs main documentation*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.3/orderer/ordering_service.html (visited on 07/11/2021).
- [65] *Updating a channel configuration — hyperledger-fabricdocs main documentation*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/config_update.html?highlight=batch%20size (visited on 06/22/2022).
- [66] *Using Blockchain Technology in Environmental Conservation*. URL: <https://earth.org/using-blockchain-technology-in-environmental-conservation/> (visited on 07/13/2022).
- [67] *Using the Fabric test network — hyperledger-fabricdocs main documentation*. URL: https://hyperledger-fabric.readthedocs.io/en/latest/test_network.html (visited on 07/03/2022).

- [68] H. Vranken. "Sustainability of bitcoin and blockchains." In: *Current Opinion in Environmental Sustainability*. Sustainability governance 28 (Oct. 1, 2017), pp. 1–9. ISSN: 1877-3435. DOI: 10.1016/j.cosust.2017.04.011.
- [69] A. de Vries. "Bitcoin's Growing Energy Problem." In: *Joule* 2.5 (May 2018), pp. 801–805. ISSN: 25424351. DOI: 10.1016/j.joule.2018.04.016.
- [70] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim. "A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks." In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 22328–22370. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2896108.
- [71] *Web3j*. URL: <https://docs.web3j.io/4.8.7/> (visited on 07/09/2022).
- [72] *What's behind China's cryptocurrency ban?* World Economic Forum. URL: <https://www.weforum.org/agenda/2022/01/what-s-behind-china-s-cryptocurrency-ban/> (visited on 07/11/2022).
- [73] D. G. Wood. "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER." In: (), p. 41.