Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Implementing a Blockchain-based Social Network

John Keaney

Supervisor: Dr. Hitesh Tewari

A dissertation submitted to the University of Dublin, Trinity College, in partial fulfilment of the requirements for the degree of Master in Computer Science
MCS (Computer Science)

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: _____ Date: _____

Abstract

Decentralised social networks have seen a rapid increase in popularity as an alternative to the domineering centralised social networks by promising to strip control of the network away from one single entity. Blockchain-based social networks take this advantage a step further, by guaranteeing users the power to decide the future of a network. The consensus mechanism of Proof-of-Work allows users to determine the future of the site, through the updating and appending of Smart Contracts, executable code hosted on the blockchain. While proofs-of-concept for blockchain-based social networks do exist in literature and even some in industry, few reach the implementation phase or go beyond a naïve design.

This research proposes a series of different blockchain-based social network designs, each iteration superior to the last in terms of memory and cost reduction, while still guaranteeing a series of rights to the users uncompromised and remaining secure by the blockchain's smart contracts. First, we propose a naïve design, a culmination of the best aspects of the current State-of-the-Art. Iterating on this, we integrate Sharding into the architecture of the site to improve mining throughput and drastically reduce storage size. Finally, we utilise Non-Interactive Proof of Proof-of-Works to prune unnecessary blocks and further reduce the size of the required ledger.

Our Sharding design is validated by the creation of a working webpage application of the site. An analytical implementation of these three designs is also produced and a dataset consisting of Reddit posts is uploaded to each version. We evaluate the results and find that our Sharding and NiPoPoW designs see a 90%+ reduction in both storage and cost requirements to the mainchain.

Acknowledgements

Thank you to my Academic Supervisor Dr Hitesh Tewari for his continued guidance throughout this project. Your advice has been invaluable to the success of this dissertation!

Also, a thank you to my Mum, and Dad as well as my sisters Lisa and Sarah who have not only supported me countless times throughout these five years at Trinity but also my whole life.

Contents

List of Figures

List of Tables

Abbreviations

BSN	Blockchain-based Social Network
CLI	Command-line Interface
CSN	Centralised Social Network
DApps	Decentralised Applications
DSN	Decentralised Social Network
ETH	Ethereum Currency
GUI	Graphical User Interface
IPFS	InterPlanetary File System
NFT	Non Fungible Token
NIPOPOW	Non-Interactive Proof of Proof of Work
OSN	Online Social Network
POS	Proof-of-Stake
POW	Proof-of-Work
POWPOW	Proof of Proof-of-Work
SOTA	State-of-the-Art

1 Introduction

In under two decades, social media has gone from a novel creation followed by a niche group of people to a fully integrated part of our lives and an extension of our very selves. While there are a plethora of benefits to these online social networks (OSN), there remains room for a sinister, ruthless entity lurking within. Although users may subscribe to social media for an explicit purpose, such as privacy, the centralised nature of these networks means that they have little to no say in how operations are run. As a result, overnight, the defining principles of an OSN can be completely rewritten.

The introduction of Satoshi Nakamoto's pivotal paper *Bitcoin: A Peer-to-Peer Electronic Cash System* [1], has led to a revolution in decentralised technologies, granting users the power to decide the future of a network, with an underlying set of absolute guaranteed first principles. With all this in mind, blockchain, a system which allows a group of individuals to securely maintain a trust-less ledger without relying on a third entity, would be an ideal solution to this existential threat centralised networks have on our existence.

Wouldn't it be nice to have a social network which, rather than being dictated by some supreme entity, was controlled by the users, for the users? A social network which, guaranteed from its inception, abides by well-defined principles. With this dissertation, I pursue this objective and investigate the creation of a blockchain-based Reddit clone, its challenges, and potential solutions.

1.1 Motivation

Open-source decentralised social networks (DSNs) have seen a grand increase in popularity in recent years, but there still exists some issues that the layperson user may fall victim to, as with centralised social Networks (CSNs). Users may upload their content online, but social networks consist of communities and threads that can be easily found, joined and left. Thus, they require a more complex and constantly updating structure than just an immutable database. Should a user connect and receive updates

from a malicious node, how are they to know that their rights have necessarily been violated? Methods exist to provide some protection, but these are easy to circumvent. With a blockchain-based social network (BSN), this is not the case.

Although there have been suggestions for the design of blockchain-based social networks in the recent research literature, only some have reached the implementation phase. Most work outlines its creation, but more literature is needed on analysing or providing the best possible design for the user in matters such as optimal storage space. As of the writing of this dissertation, only a handful have attempted to recreate a simple forum such as Reddit as a blockchain social network.

1.2 Objective

This dissertation aims to determine the viability of constructing a blockchain-based social network. Specifically, an attempt will be made to replicate a blockchain-based version of Reddit.com since no application which attempts to replicate Reddit's community structure exists as of the writing of this dissertation.

The primary objective of this dissertation is to design and implement a BSN Reddit clone, with the successful creation of constituting that the following features are enabled via smart contracts:

1. Guarantee a series of rights to each user, held solid by the site's constitution
2. Create and deploy communities on the blockchain
3. Post content and comments to communities hosted on decentralised storage

The secondary objective of this dissertation is an investigation into how the scalability issues of a BSN might be addressed. Solutions to this problem will be simulated, evaluated and compared against other potential designs.

1.3 Overview

The remaining chapters of this dissertation are structured as follows:

Chapter 2 - Background: Covers the underlying technologies used to create modern decentralised applications, as well as the protocols necessary to create a new scalable yet secure BSN.

Chapter 3 - State-of-the-Art: Discusses State-of-the-Art literature related to the design of blockchain-based social networks. It also discusses the current state of BSNs in the industry.

Chapter 4 - Design: Focuses on the design process behind the creation of three different BSNs, a naive design, a sharding design and a NiPoPoW design.

Chapter 5 - Implementation: Details two different implementations of our designs from section 4, a webpage frontend implementation written in React.js, and an analytical implementation written in Python.

Chapter 6 - Analysis: Gives an overview of the dataset we used to perform our analysis as well as the statistics recorded.

Chapter 7 - Evaluation: Compares and evaluates the results of our analysis.

Chapter 8 - Conclusion: Offers concluding remarks and discusses potential future work.

2 Background

This chapter will cover the necessary technologies required to understand the reasoning behind our design choices for this implementation of a blockchain-based social network.

2.1 Decentralised Networks

Unlike centralised networks, where one node has authority over the network, control is divided among the various peers in a decentralised network. Decentralised networks add increased complexity and are considered more difficult to manage; however, they offer no single point of failure.

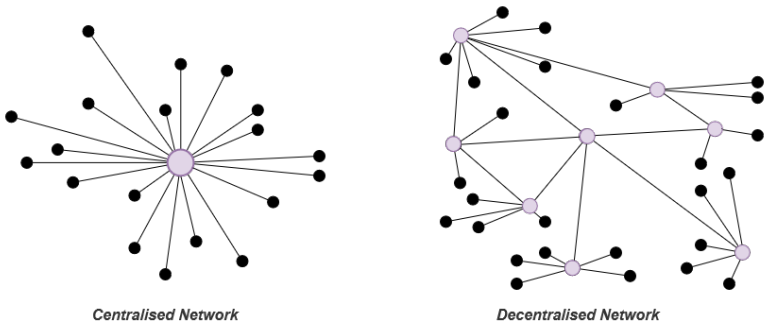

Figure 2.1: Centralised Vs Decentralised Networks

With this in mind, one might see how decentralisation is used to create a social network which does not rely on a single entity. However, this fact is still similar to centralised networks, as there is no network-level guarantee that any node will propagate content to other nodes with their best interests in mind, potentially acting maliciously.

2.2 Foundational Blockchain Concepts

2.2.1 Hashing

Hashing functions, such as those discussed in [2], are deterministic functions which generate a string of characters, known as a *hash* or *digest*, based on a given input. These hashes act as unique identifiers for the input, a secure format from which determining the source content is irreversible unless the recipient has the original input. Any change in input results in an 'avalanche effect', creating a drastic change to the hash. Since the chance of any potential collision of hashes is low, hashes are used as addresses for content, known as hash addresses or occasionally as hash pointers.

As an example, Bitcoin [1] uses the SHA-256 hashing function [3], which generates a 256-long hash no matter the input. This methodology allows us to ensure the data integrity of any content since any change will result in a change in its identifying hash.

2.2.2 Proof of Work (PoW)

Suppose an individual hopes to find a particular hash of their content. Since the resulting hash is essentially uniformly random there is no procedure to find a suitable hash for a secure hashing function quickly. The only method would be to manually hash the content through an iteration of bits until the required hash is found.

Pulling from Adam Back's Hashcash paper [4], Hal Finney expanded on this concept in his Reusable Proof-of-Works paper [5] utilising the property of uniformly random, infeasible to reverse hashes to create a metric to determine a level of work among users, known now as just Proof-of-Work (PoW). Users, referred to in this scenario as *Miners*, can append a nonce of bits to given content and hash this with the goal of creating a hash with at least *T leading zeros* in exchange for some reward. Once found, other nodes can verify the proof of their effort in an instance. This work's difficulty level can be easily adjusted by increasing or decreasing the number of T leading zeros required in our proof.

2.3 Blockchain

Blockchain, composed of the above technologies, along with an algorithm for choosing the safest and most honest node, guarantees data integrity [6] among users through a system of information stored in *chained blocks*. Originally proposed by S. Haber, W.S. Stornetta to timestamp documents to prevent tampering [7], and further perfected and

popularised by S. Nakamoto [1], blockchain acts as a secure distributed immutable ledger which may be appended to by recording new transactions between nodes at some computational cost.

Blockchain's structure consists of a series of linked blocks containing a header, and some decided content in the body, such as a list of currency transactions as seen in Figure 2.2. Blocks are linked to their previous parent blocks via hash addressing by creating a hash of said parent stored within the header. The only exception is the genesis block, the initial block with no parent. A Merkle Root is generated from the body and stored in the header.

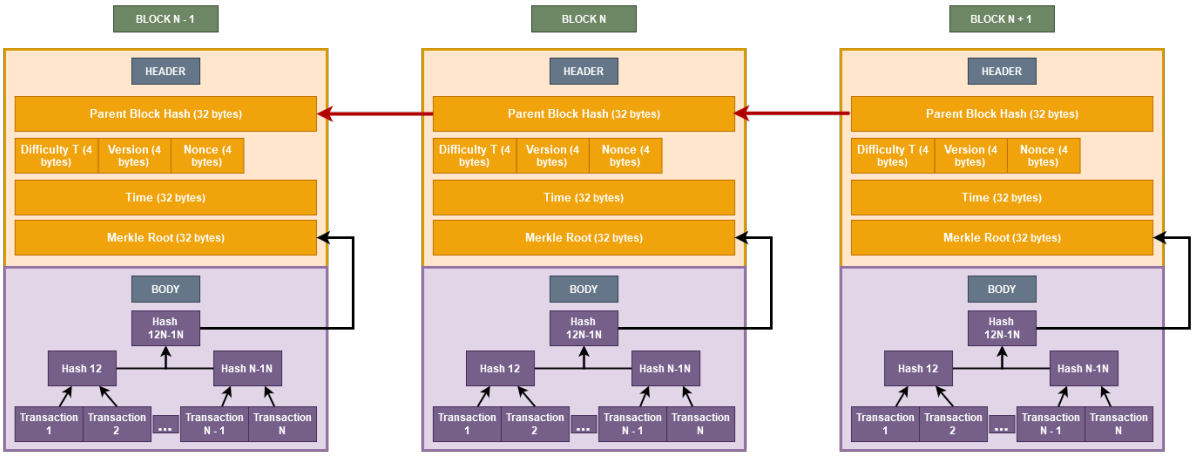

Figure 2.2: Blockchain Structure

For new content to be appended to the chain, first, a PoW must be generated by some miners and stored within the block via the nonce. Once a PoW has been verified as accurate, it gets appended to the chain. For competing chains, nodes accept a chain with the most PoW.

2.3.1 Blockchain Security

Should a malicious attacker attempt to change the content of a past block by the avalanche effect of hashing, the block's hash completely changes, making the proof-of-work effectively null. Malicious entities must attempt to reconstruct a corrupted block's PoW but must do so for every child block due to their PoW changing via the linked hash address. This task becomes infeasible with a long enough ledger. Nakamoto [1] detemined that a depth of 5 blocks is enough to reduce probability of chain corruption to 0.1%, which was further validated as secure in Juan Garay et al. [8]. Figure 2.3 and 2.4 demonstrate the outcome of a compromised chain, which changes all future block hashes.

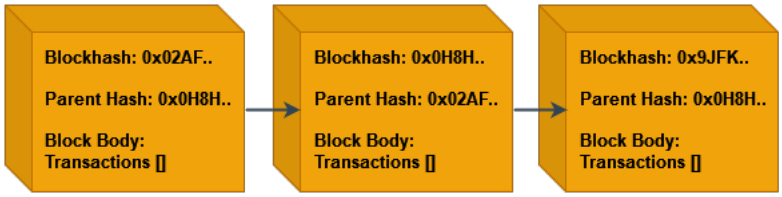

Figure 2.3: Uncompromised Chain

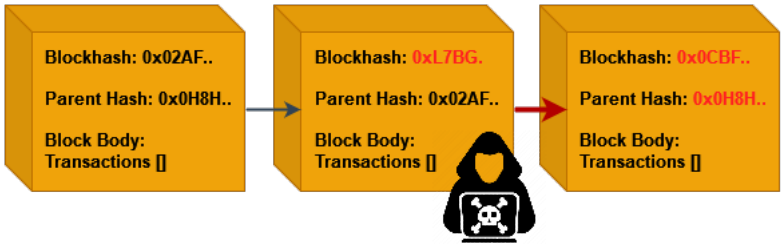

Figure 2.4: Compromised Chain

In order for a malicious node to succeed in corrupting the network through appending new blocks, it must generate PoWs faster than the honest nodes. This attack requires 51% control of the hashing power of the participating miners in the network for a long enough period, which most consider infeasible.

PoW performs the operation of acting as a consensus mechanism between nodes. PoW resolves our earlier issue whereby a layperson user may get infected by a malicious entity. Users now have a metric to determine which nodes are valid. Should we store critical infomation in a blockchain, such as currency or site code, users can determine whether this has been changed by corrupted PoW.

2.4 Smart Contracts

Although initially created as a distributed ledger for electronic currency, blockchain is not limited to just storing a list of currency transactions. Vitalik Buterin proposed his blockchain, Ethereum, in his original Ethereum Whitepaper , which describes how a weak low-level version of scripting is possible via Bitcoin. Named *Smart Contracts*, these programs that are stored within the blockchain automatically execute when some specified conditions are met without the need for a trusted third party.

2.4.1 Solidity

In this same paper original Ethereum Paper [9], Buterin proposed his blockchain programming language, Solidity, to facilitate easier creation and further depth with Smart Contracts. Solidity has since gone through multiple versions, with potential of parallel contracts currently being worked on as mentioned in Shuai Wang et al. [10].

2.4.2 Ethereum Gas

While the currency of Ethereum is *ETH*, a smaller unit exists when referring to the processing of transactions known as gas. One unit of gas represents $1x10^{-9}$ *ETH* or 1 *GWEi*. When interacting with or deploying a smart contract, gas is required to perform any state change on the blockchain. A public and private key pair must sign off each transaction. With gas, we can quickly determine the real-world cost of our application by tallying the total.

2.4.3 Decentralised Applications - DApps

Buterin also foresaw how the creation of these Smart Contracts could lead to an entire ecosystem of decentralised applications, also known as DApps. Through DApps, we can recreate any typical software application but with the added benefit of security, immutability and decentralisation, where *code is law*. With this, one could see the initial steps to creating a BSN with an immutable constitution, where smart contracts contain the social network's code.

2.5 InterPlanetary File System - IPFS

Blocks size are limited to prevent the blockchain ledger from growing too quickly and causing increased centralisation. Block size limits pose a problem for our social network, as posts on social networks such as Reddit often far exceed the size limits of a typical block, currently 1MB for Ethereum.

Instead of using blocks to store content, we might use them as pointers to an immutable distributed database. InterPlanetary File System (IPFS) is one such file system, which consists of a series of peer-to-peer protocols for posting, retrieving and updating content via content-based addressing, where the address of content is based on its hash. IPFS also has a form of version control. When content is updated, instead of uploading an entirely new version, only the edit is needed with a pointer to the original node.

2.6 Sharding

For new blocks to be appended to the chain, they must be validated by the majority of nodes. Block validation leads to a delay in the blockchain, limiting the number of transactions. With Ethereum Classic, this was only 15 transactions per second, completely outclassed by Visa with 24'000 transactions per second as mentioned in

[11]. Several solutions have been suggested to improve the scalability of Ethereum and other blockchains, one of which is sharding.

Sharding [12], not unlike database sharding, involves the dividing of the network into smaller interconnected subnetworks referred to as shards, shard chains or sidechains. Each of these shards works in the same fashion as a typical *unsharded* chain, processing and validating blocks with only a portion of the blockchain storage and transaction processing responsibilities.

A mainchain or beacon chain is used to help maintain consensus and manage shards, allocating work and preventing blockchain issues such as double spending [13]. This structure can be seen in Figure 2.5. Miners are randomly assigned to shards to distribute the hashing power evenly. A compiled transaction record is submitted to the beacon chain at regular intervals through a smart contract, known as a *rollup*. With each node running as a parallel pipeline, transaction throughput is accelerated proportionally to each additional shard. Lower storage requirements and computation cost lead to further decentralisation as less powerful nodes can now participate.

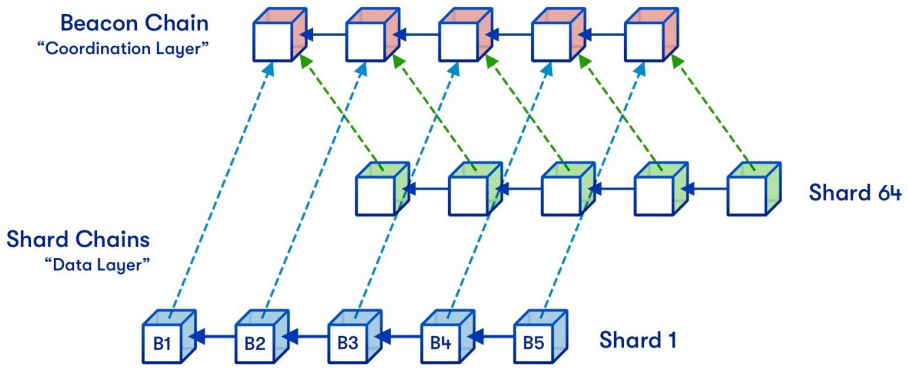

Figure 2.5: Sharding Diagram [14]

For our purposes, we will focus on the storage benefit sharding adds by decreasing the size of the required ledger.

2.6.1 Crosslinks

Sharding uses crosslinks to connect shard chains to our beacon chain. Whereas typical blockchain ledgers only require a pointer to the previous block, with sharding, a pointer is needed to point from the mainchain to the respective sidechain block and vice versa. The property of crosslinks worth mentioning is that the number of shards can be decreased or increased, not affecting the beacon chain from appending new blocks nor interrupting the proof-of-work.

2.7 Non-Interactive Proof of Proof of Work (NiPoPoW)

Even with sharding, the beacon chain is still required to maintain the entire mainchain ledger of all blocks to maintain consensus. This causes a scaling issue when we reach hundreds of thousands of blocks. As of 2023, Reddit.com has over 3 million communities, as mentioned in [15]. Should we have one block for every community with a max block size of 1MB, this could lead to a beacon chain raw ledger size of 2.8TB, entirely out of reach for the average device. This begs the question, is it possible to achieve our goal of maintaining consensus without needing a history of every block to know its PoW?

2.7.1 Revisiting PoW

With PoW, we prove to others that a level of computational effort has been completed via a Hashcash puzzle [4] where we append a nonce to some content to find some hash with T leading zeros. This proof can be verified in an instance with minimal effort.

It should be noted that this requirement only preferences that the first T leading zeros are needed. Occasionally we produce a hash with extra zeros over what is necessary. For example, say we must generate a hash of length 4, with a $T = 1$. The result of these successful hashes can be seen in figure 2.6 to the right. Our extra zeros, e, are highlighted in green. What we might notice about this table is that of all hashes we generate, about $1/2^e$ of all successful hashes will have e extra zeros. The law of concentration around the means guarantees this property and that our hashing function is uniformly random with a hash that cannot be predicted.

Hash	*Extra Zeros*
0000	3
0001	2
0010	1
0011	1
0100	0
0101	0
0110	0
0111	0

Figure 2.6: T=1 Successful Hashes

2.7.2 Approximating PoW

With this knowledge, we can approximate the number of blocks on a chain if we only have a list of all blocks with e extra zeros. For example, should we know that we have four blocks with three extra leading zeros, then we can approximate that we have about thirty-two blocks total. This logic translates then to the PoW of a chain, where

we can now approximate the PoW of a complete chain using only a list of blocks with e extra leading zeros. To formula this approximation, where C is some entire chain, $C\uparrow^e$ is a subset of chain C such that $C\uparrow^e = \{b \mid b \in \text{C where } b \text{ has } e \text{ extra zeros}\}$ and $|\text{C}|$ or $|C\uparrow^e|$ is used to denote the proof of work of any chain C or $C\uparrow^e$ respectively:

$$|C| \approx 2^e * |C\uparrow^e|$$

By utilising this approximation, we can now estimate the PoW of a chain without the need for every block and thus maintain our constitution. Proving our PoW in this way is referred to as a Proof of Proof-of-Work (PoPoW) as in Aggelos Kiayias et al. [16], or with this specific methodology Non-Interactive Proof of Proof-of-Work (NiPoPoW) as in Dionysis Zindros et al. [17].

2.7.3 Interlink Pointers Data Structure

To implement a NiPoPoW protocol, we must update the *linked list-like* structure of our blockchain. This new structure, known as an Interlink, is a skip-list-like structure with minor adjustments. With just PoW, each block only points to the previous parent block. With NiPoPoW, each block now points to the most recent block with e extra zeros for each level of e extra zeros less than the current blocks e value. The exception to this rule is the genesis block which points to no previous block(s) and has an e value of infinity. This structure allows for more efficient traversal when proving our NiPoPoW. This Interlinks structure described can be seen in Figure 2.7.

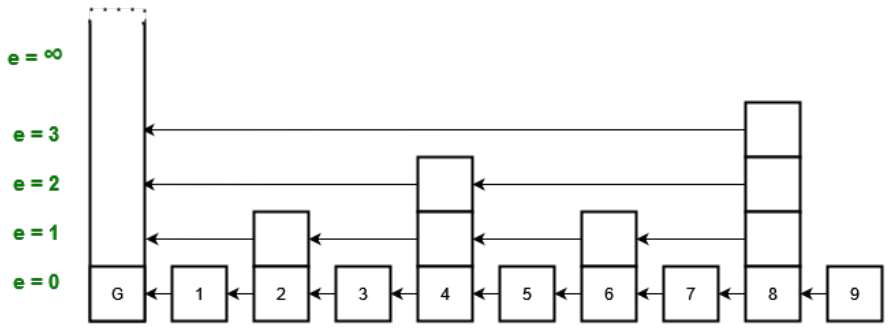

Figure 2.7: Interlinks Data Structure

2.7.4 NiPoPoW Security & Succinctness

With NiPoPoW, we greatly decrease the potential size of our chain. In terms of raw ledger size, with time, the number of blocks at each level will be approximately half of the lower level. Theorem 8 (Number of levels) of the initial NiPoPoW paper [17] found that the number of levels of our chain will be at most log($|\text{C}|$) where $|\text{C}|$ is

the length of some chain. We can say then that the number of blocks at the highest level will be $|C|/2^{log_2(|C|)}$. Using our law of logs where $a^{log_a(b)} = b$, we can simply this to $(|C|/|C|)$ or just 1. This means that, on average, our top level within our interlinks structure will have only one block.

2.7.5 Minimum NiPoPoW Block Count

Unfortunately, this is not enough blocks to create a properly secure blockchain from malicious attackers, but it does show we can create a NiPoPoW of only one block. Satoshi Nakamoto highlighted the dangers of low block counts in the Calculations section of the initial Bitcoin paper [1]. Should a malicious attacker gain 10% of all hashing power on the network, honest nodes would need to have a depth of five blocks worth of PoW finished to reduce the probability of corrupting the blockchain to < 0.1%. Literature often refers to this depth of blocks as suffix parameter k.

In the NiPoPoW paper [17], similar results were found, where the highest level of the interlinks requires at least a depth of six blocks to reduce the probability of an attacker launching a successful attack to < 0.1%. This requirement to reduce malicious attackers from succeeding is referred to as security parameter m in [17]. Theorem 10 (Optimistic succinctness) of [17] goes further, saying that for our chain to be secure, our proof should be $4 * m * log(|C|)$ blocks long, where m is our security parameter and |C| is the length of the chain at the lowest level.

3 State of the Art

This chapter discusses State-of-the-Art literature related to the design of blockchain-based social networks. It also discusses the current state of BSNs in the industry.

3.1 Blockchain-based Social Networks

Blockchain-related literature has seen an explosion in popularity since 2017 and again in 2020 in tandem with the widespread increase in the popularity of cryptocurrency in the news cycle. However, there is relatively little reporting made on the implementation of BSNs, where there exists only a handful of BSN architecture design ideas and even fewer attempts at actual implementations.

Quanqing Xu et al. (2018) [18] is the earliest paper to actualise a BSN as an Ethereum-based social network via a Decentralised Application. A minimal proof-of-concept is presented where the functionality of a typical CSN is mimicked using an Account Manager and User Smart Contract. Site functionalities such as account creation and posting are described and aided by use case diagrams. Distributed storage, specifically the InterPlanetary File System (IPFS) protocol, is used to reduce the cost of the server-side hardware and increase data availability. Future work proposed that an examination into the performance and cost should be explored and that a frontend framework could be utilised to beautify the User Interface.

Tharuka Sarathchandra et al. (2021) [19] is another proof-of-concept which utilised the same technologies as Quanqing Xu et al. (2018) [18] but focused on the integration of modern Web3 libraries and Frontend frameworks into the application. To aid in the deployment of IPFS content, they utilised infura.io[1], which provides a suite of methods to deal with IPFS. They found that with ten concurrent Windows 10 intel core i7 laptops with 16GB memory running the application, the upload of 1GB of data took about 16s, despite using infura as a middleman library. While other libraries such as MetaMask[2] were discussed they offer negligible benefit.

[1] https://www.infura.io/product/ipfs
[2] https://metamask.io/

Ningyuan Chen et al. (2021) [20] suggests using a decentralised autonomous organisation (DAO) to dictate the site's operations. DAO is an established series of smart contracts where the site members or shareholders operate it through autonomous programs. Other sites might establish moderators to manually delete posts; a DAO can do a lot of this work automatically, with agreed-upon principles created by the community. The code of DAO acts like law, where every law is expressed as some executable code, run on the blockchain to maintain consensus to be executed without hindrance. Utilising a DAO can guarantee a series of rights to each user in line with our primary objectives of section 1.2.

Shuai Zeng et al. (2019) [21] is the first paper to describe using sharding for a BSN to improve block mining throughput and thus the system's scalability. Miners are randomly assigned to shards to help improve the block mining throughput. Shuai Zeng et al. (2019) [21] also discuss a reputation-based system to improve security. Users are categorised into different user types, including malicious, suspicious, normal or credible. Based on their reputation type, they are given different rights, including browsing, posting or voting. The paper proposes the refinement of this reputation-based system in future work, but they have yet to publish an update. While a reputation-based system may help to reduce illegal content from being posted, it is difficult to define and could be easily manipulated. It could potentially be abused by a select group of individuals, removing power from users and going against our initial objectives outlined in section 1.2.

Outside of research, there exist few projects which are entirely hosted on a blockchain ledger. Most projects have been abandoned, or are stuck in development, such as Peepeth[3] or Sapien[4]. Steemit[5] is the only existing blockchain-based social network which still remains active. It is arguable if it truly abides by the underlying principles of blockchain though, as it uses an ill-defined reputation-based system called Proof-of-Brain [22], which rewards those who hold wealth with more currency. Today most sites which utilise blockchain technology only incorporate a small form of it, such as Reddit's recent release of its Non-Fungible Token (NFT) project[6].

[3]https://peepeth.com/
[4]https://app.sapien.network/
[5]https://steemit.com/
[6]https://nft.reddit.com/

4 Design

This chapter will provide an overview of our design objectives and the three iterations of my implementation of a blockchain-based social network. Beginning with a naive approach, we describe an implementation based on the current State-of-the-Art acting as a baseline. Then we segment our communities into multiple sidechains defined as a sharding approach. Finally, we prune the mainchain while maintaining chain security, known as the NiPoPoW Approach. Additional features that were designed but ultimately not included in the implementations will also be discussed.

4.1 Design Objectives

As highlighted in our introductory objective section 1.2, our core requirements for this application are that they guarantee a series of rights to each user, that any user can create communities on the blockchain and that any user can post content to these communities. All implementations should abide by these principles. We aim for a tangible level of decrease in storage requirement with each iteration.

4.2 Naive Approach

As can be observed from our State-of-the-Art discussion in section 3, these architectures all centre on essentially the same simplistic structure, an Ethereum blockchain DApp, written in Solidity where posts are content-addressed to IPFS nodes. All new content is appended to the chain in order of its block creation.

We refer to this structure as a naive approach as it does not consider the issues that may arise at scale, particularly concerning storage. Figure 4.1 takes the best of these papers and showcases the culmination of these approaches into one architecture.

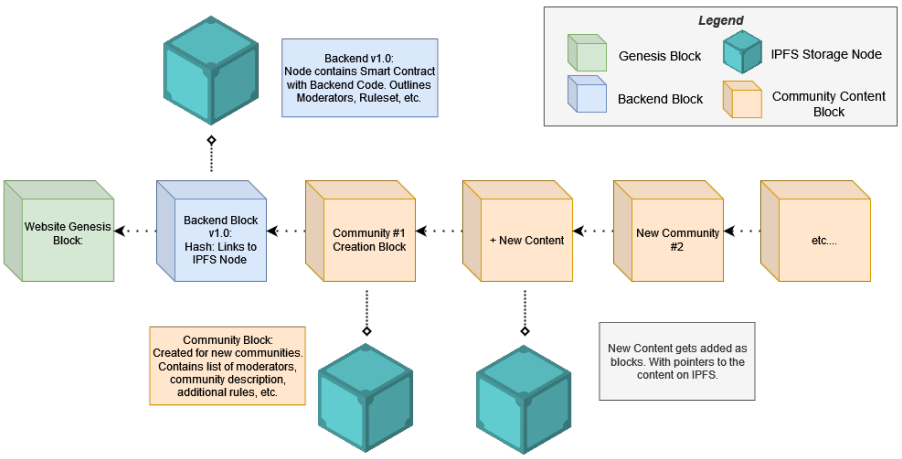

Figure 4.1: Naive Design

This naive implementation will be the basis for which we compare our improvements for each iteration in the later evaluation section 7 of the dissertation. In this implementation, I have chosen classical PoW as our consensus mechanism. Proof of Stake (PoS) [23], where control is largely determined by wealth, is also an option. PoS's impact on chain creation alone could be the basis for future work, as its integration would drastically affect cost and security.

4.2.1 Posting Protocol

Posting content with this naive approach works mostly the same way as described in our State-of-the-Art section. Users request a make_post contract method hosted on the chain via a user-friendly environment, such as through the React.js library[1]. Content is uploaded to IPFS, with the content addressable hash returned and appended to the chain. Figure 4.2 is a use case of the protocol for uploading content to the application. The process for creating communities is much the same, only that the payload type is different.

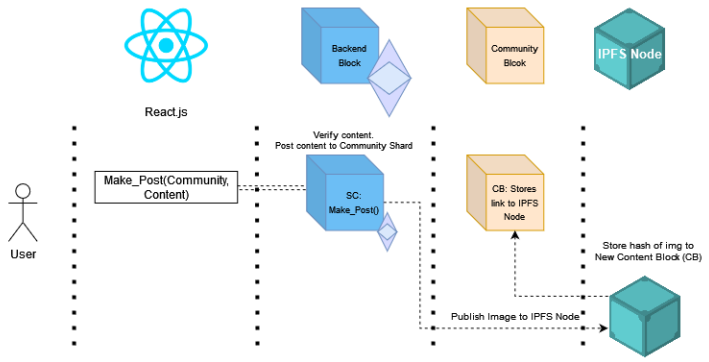

Figure 4.2: Posting Content

[1]https://legacy.reactjs.org/

With this implementation, we achieve our objectives originally discussed in 1.2 at the cost of several disadvantages compared to traditional DSNs, including storage increase due to block headers, potential cost increase, and post throughput decreases, among other things. With this in mind, we shift our research focus to improving the storage of a BSN while maintaining our objectives from section 1.2.

4.3 Sharding Approach

Sharding has also been suggested for use in BSNs in Shuai Zeng et al. [21], although here, it is only used as a method of improving block mining throughput. Our implementation differs in that the construction of the site is moulded by sharding rather than random blocks assigned to any shard. We refer to this as horizontal sharding.

4.3.1 Horizontal Sharding

With horizontal sharding, a new shard is created each time a community is created, rather than a set specific amount of shards that each user is randomly assigned to. This shard crosslinks a community from the mainchain to one in the respective shard as in Figure 4.3. Horizontal sharding allows us to forgo shards for communities we wish not to be a part of and discard shards that we no longer use. Users still need to maintain the mainchain in order to participate and maintain the site's constitution.

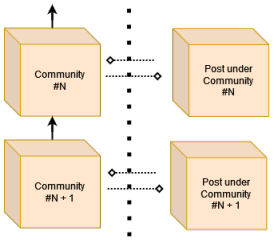

Figure 4.3: Horizontal Sharding

Depending on the site or a community's configuration, we can prune a community's shard should it get long enough, only taking the best content and replacing it with a fresh shard preventing large storage requirements.

Horizontal sharding can also be utilised within shards themselves for the purposes of constantly updating blocks, as shown when we create a ban list shard in Figure 4.4. Since these shards are optional, we can discard a long enough chain and restart only with the most recent block, reducing storage requirement.

4.3.2 Sharding Moderation

With the mainchain, control is determined by the hashing power makeup of the miners. On the community level, governance is configurable based on our backend. One might determine control of the community in a first-come-first-serve fashion, similar to Reddit, where the founder is the head moderator and can choose to elect other others. Should the moderator abandon the community, control is passed to the next miner

in the subscriber list with a full chain history. An alternative, should we deviate from our Reddit-like approach, would be to base control on the hashing power makeup of that shard, similar to the mainchain.

4.3.3 Sharding Cost Reduction

Miners will want to receive the most rewards for the smallest cost. By splitting the mining of blocks into various communities, there is less wasted energy used by the mining pool, all focusing on one block. Mining in smaller communities is encouraged, as there is less competition for mining rewards, aiding in preventing "echo-chamber" communities.

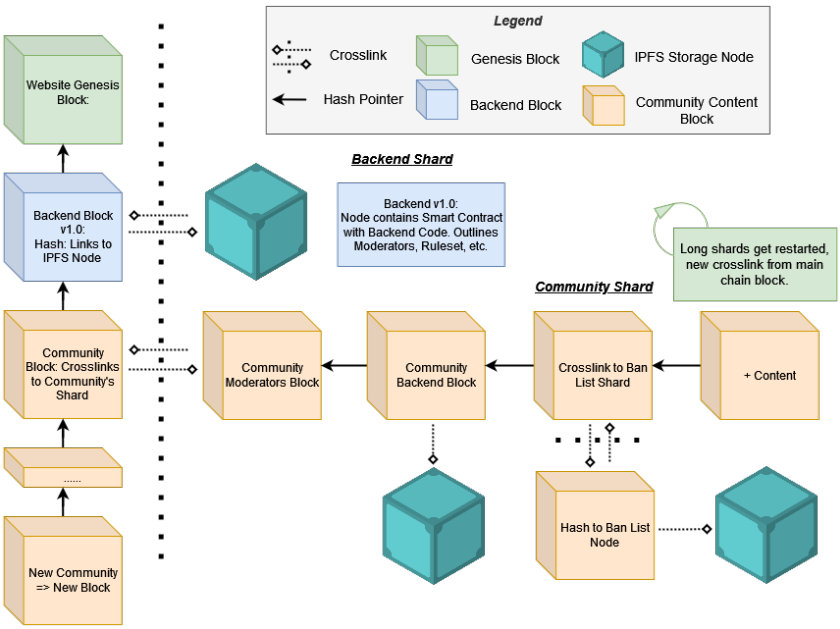

Figure 4.4: Sharding Design

4.4 NiPoPoW Approach

While our sharding design will greatly reduce the size of our total chain length relative to our naive design, it still suffers from scalability issues at the cost of maintaining our objective of the users controlling the site. In order to maintain this *'constitution'* we require the PoW of the entire mainchain. Horizontal sharding will allow us to no longer meet the requirement of storing every community block and its posts but will still require us to store the mainchain. Should a new crosslink block be created for every community, with time, this will become infeasible for the average user, let alone the average mobile device, calculated to require 2.8TB of memory as discussed in section 2.7.

4.4.1 NiPoPoW Reduction

As part of our consensus mechanism, we require the history of the PoW of every block to select a secure mainchain. By utilising NiPoPoWs, we can reduce this requirement to only polylog mainchain blocks as discussed in Dionysis Zindros et al. [17] and section 2.7.4. Instead of requiring each block in the mainchain, we only require the blocks of the highest secure layer, the backend blocks and the genesis block.

Taking our example from section 2.7, and assuming that our NiPoPoW chain must meet the security requirement discussed in 2.7.4, we would expect a block amount reduction from about 2.8 million to $4 * 6 * log_2(2800000)$ or 528 blocks. Converting this to bytes, with a max Ethereum block size of 1MB, reduces 2.8TB to 0.52GB of required minimum storage space.

4.4.2 Hybrid Nodes

Should we utilise NiPoPoW in our application, it would require the pruning of many blocks and as such, removing the connection to shards. We propose that the NiPoPoW design requires two versions of the application. Similar to the hybrid nodes approach seen in Parikshit Hegde et al. [24], a small minority of nodes would be required to store the entire mainchain history, as in the sharding design. These users would be known as *full nodes*. The vast majority of users, say those with mobile phones would store the NiPoPoW pruned chain and would fetch their wanted blocks from full nodes. These users would be known as *hybrid nodes*.

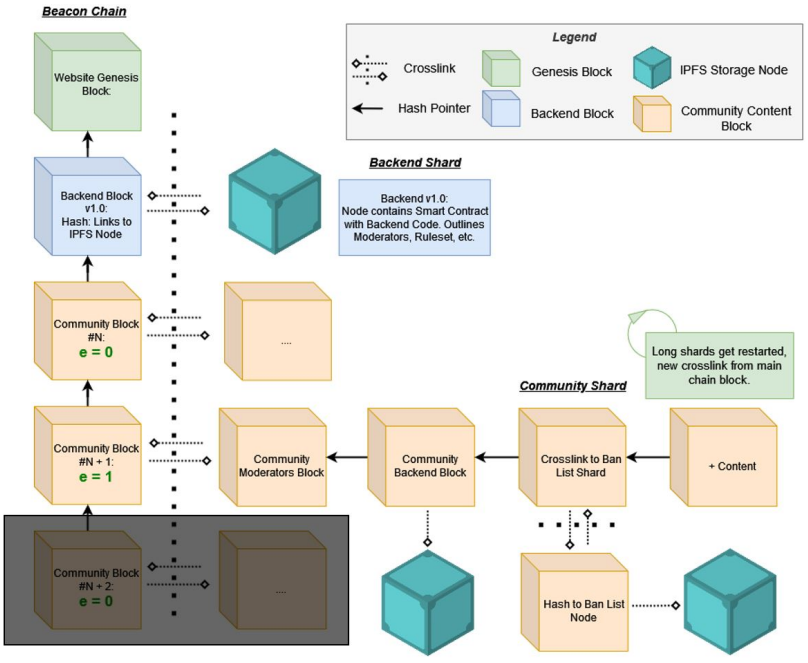

Figure 4.5: NiPoPoW Design

In Figure 4.5, we demonstrate this improvement by only requiring blocks on the main-chain with hashes of extra leading zeros 1, our backend blocks and our genesis block. As such, Community N + 2 is *shadowed out* and not required by the network. This user has requested to join Community N from a full node and has a pointer allowing it access.

4.5 Additional Design Considerations

Along with these features, additional design considerations were made to flesh out the site's functionality. Due to time constraints, not all of these were implemented, but they will be discussed here.

4.5.1 User Banning

Unfortunately, due to the immutable nature of blockchain, making simple edits to the honest blocks is practically impossible. Each time we wish to ban a user, their name must be added to a ban list, and a new block must be appended to the chain. We cannot just simply link this ban list to an IPFS node since it too is immutable, and using a different mutable storage source presents a security risk and compromises our applications objectives.

One solution is to use a financial incentive to prevent users from malicious activity. Like the original HashCash paper [4], to prevent excess spam from plaguing the site's users who want to join a community are required to complete some form of computational puzzle such as Proof-of-Stake [23]. This PoS is then verified by a veteran user, such as a moderator. This feature can also be used to prevent spam attacks such as low-fee flooding as discussed in [25] by toggling an increase in mining difficulty till the barrage of malicious attackers settles.

Another solution would be to append a separate ban shard as in Figure 4.5 and in Figure 4.6. With each community shard, a ban shard would be generated, crosslinked to a block in the community shard, containing a list of all verified new users. Freshly verified users are appended to the ban shard. Once a shard gets long enough, the list of names gets re-initialised as a new shard to prevent lengthy chains, and only the most recently generated ban shard is required to post.

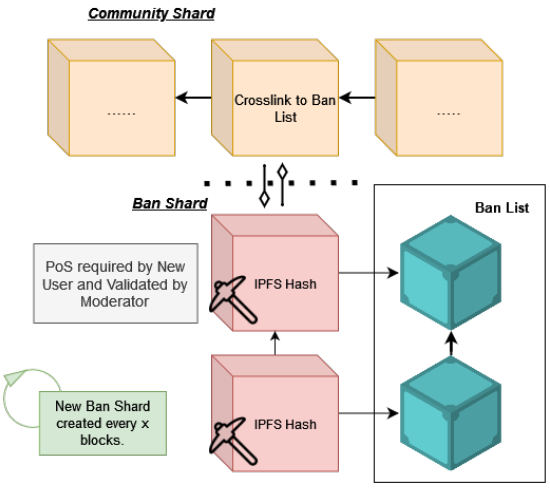

Figure 4.6: Banning Protocol

4.5.2 Voting Protocol

As the site continues to grow, unexpected challenges will arise that will require the backend to be updated for both the mainchain and shard communities. With communities specifically that will have to deal with potentially illegal or dangerous content being posted, a moderation team may be required to update the backend to prevent new technologies from exploiting the site. Malicious users could find ways to circumvent the backend from detecting illegal content and triggering a ban. In such instances, it could be ideal to select a moderation team to prevent these one-off offensives.

Figure 4.7, inspired by Shuai Zeng et al. [20], demonstrates a potential voting architecture used to elect moderators and propose changes to the site's constitution. PoS is required to propose a vote presenting a financial hurdle for spammers attacking the site. Should a constitutional change be required, it must first be approved by the moderators and then by the users. Ideally, elected moderators should be proficient in Solidity and can prevent malicious code from being voted in by chance. Moderators have term limits, and re-elections are scheduled on a regular basis.

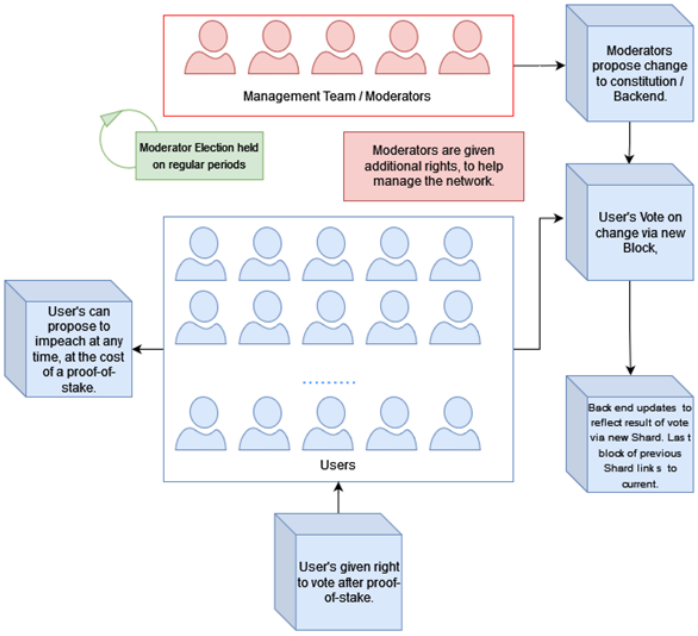

Figure 4.7: Voting Architecture

5 Implementation

This chapter discusses the real-world implementation of our designs from chapter 4 created for the purposes of viability and analysis. Two different implementations were created, those being:

- Web App Implementation (5.4): Written in React.js and using the Web3.js library[1], this implementation was created to showcase the viability of our sharding approach via a user-friendly application.
- Analysis Implementation (5.6): Written in Python and with the web3.py library[2], this implementation was created for analytical purposes to compare performance between the naive, sharding and NiPoPoW approaches.

5.1 Smart Contracts

To allow all necessary functionalities, from creating a site to creating a community to posting content, only three smart contracts written in Solidity were needed. These smart contracts are:

1. Backend.sol Contract (A1.1.1): Root contract from which all others are created. Responsible for deploying our site and listing its founder and moderators. Contains a list of all communities addressable by port, contract transaction address or name.
2. Community.sol Contract (A1.1.2): Constructs our communities. Maintains a list of its moderators. Contains the communities description and links to its posting contract.
3. Posting.sol Contract (A1.1.3): Lists all posts and determines posting permissions and execution. Contains Post struct, which lists author, title, content, origin, etc. Outputs indexed events to users for easy searching on the blockchain.

[1] https://web3js.readthedocs.io/en/v1.8.2/
[2] https://web3py.readthedocs.io/en/stable/

5.2 Additional Tools

With the current price of 1 Ether at around € 1'500.00 as of the writing of this document, it is monetarily infeasible to develop and test our site to even a minuscule degree on the actual mainnet Ethereum blockchain. As such, the Ganache Testing suite[3] has been utilised as a testnet to mimic the Ethereum blockchain. In addition, infura.io[4] is a third-party hosting service used to aid in the hosting of IPFS content, as current implementations of the standard IPFS protocol are lacking in necessary quality API libraries and plagued by slow upload times.

5.2.1 Chain Configuration

The Ganache GUI and Ganache CLI were utilised to initialise the mainchain and sidechain shards, respectively. Ganache GUI was chosen as it provides a friendly interface to investigate the contents of our mainchain, while Ganache CLI is a lightweight application with quick start-up times for creating multiple shards. Both applications produce the same size blockchain. Depending on the choices of miners, the chains configurations can be altered to improve the efficiency and security of the network. With our goals in mind, below is the configuration of our chains and the reason for these choices.

- *chain.allowUnlimitedContractSize*=[false :: boolean]: Prevents large contracts from being stored per block. Thus contracts must be split over multiple blocks. The impact of a contract call for Miners increases depending on the contract size. Malicious attackers require few resources to place more work for miners when contract sizes are large, creating an avenue for Denial-of-Service (DOS) attacks [26]. The consequence of this is a larger chain due to more headers. Large contracts should not be an issue, as changes to the backend are infrequent and limited to the size of our IPFS node content hash address.
- *database.dbPath*=[shard[i].path :: string]: Creates a unique directory for each community shard. More structured shard storage for minuscule memory increase.
- *logging.verbose*=[true :: boolean]: Provides detailed logs of Remote Procedure Call (RPC) requests, or in order words, a log of all of the chain's operations. Required to generate NiPoPoW proof and allows for faster site searching.
- *miner.blockTime*=[0 :: uint256]: Begins mining as soon as a transaction comes in. Speeds up posting time.

[3] https://trufflesuite.com/ganache/
[4] https://www.infura.io/product/ipfs

- *miner.difficulty*=[0x1 :: string]: Default required leading zeros to solve for. Speeds up initial community creation time. Mining difficulty is still adjusted based on the size of the network's hash power.
- *miner.instamine*=["strict":: string]: Returns a transaction's hash before the block is mined. Speeds up some operations, e.g. generating a community and crosslinking it to the mainchain, can happen in parallel.
- *wallet.accounts*=[mainchain.accounts :: (uint256, uint256) []]: Starting account balances and details of a new shard. Allows users to spend their mainchain balance on shards. Prevents unnecessary mining.
- *server.port*=[shard[i].port :: uint16]: Port which shard listens on. Each shard has to have a unique port with our implementation to avoid block-mining conflicts.
- *detach or -D*: Runs background instance via the CLI of the shard in the form of a daemon thread. Returns a unique name for the instance which can be stopped via *ganache instances stop <name>*.

With our chain configured, we can have what is needed to implement our naive design. To actualise the sharding design we move on to discuss how the process of horizontal sharding was implemented.

5.3 Implementing Horizontal Sharding

As mentioned in section 4.3, rather than exclusively using sharding to improve block mining throughput by randomly assigning nodes to shards, our implementation directly incorporates sharding into the structure of the site itself. With our implementation, each community created requires a four step process:

1. Run an instance of an new empty shard on an unassigned port.
2. Deploy a posting contract to the newly created shard.
3. Deploy a community contact to the newly created shard, linking the posting contract.
4. Crosslink the community contract to the mainchain.

For the purposes of demonstration I've used Ganache GUI for both the mainchain and shards to visualise horizontal sharding in Figure 5.1. Zoom in may be required to view all details on a 1920x1080p resolution screen. In Table 5.1, with the current price of Ethereum hovering around €1'500.00 as of the writing of this document, I've also provided an estimate for the cost of this process in fiat currency.

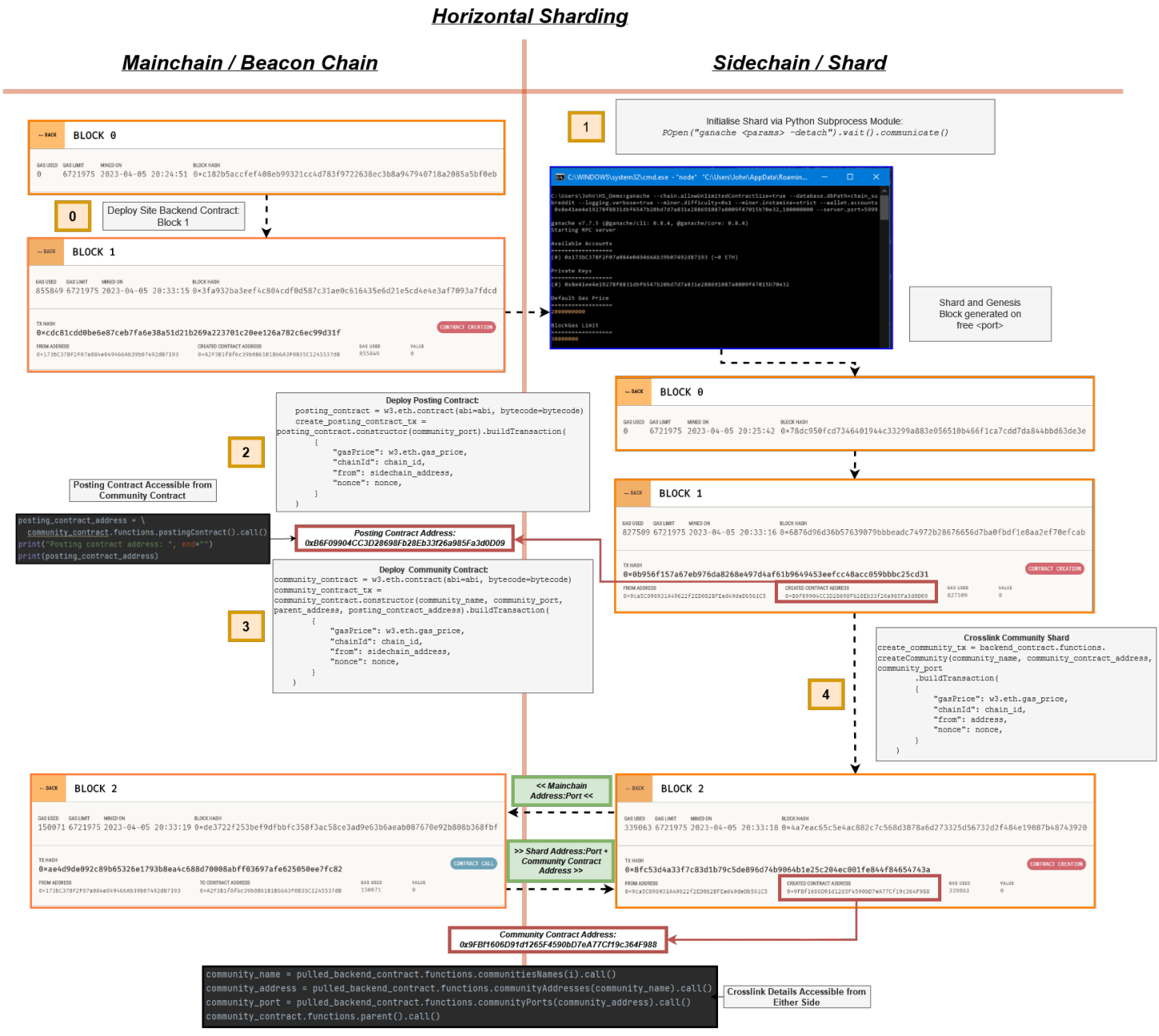

Figure 5.1: Horizontal Sharding - Implementation

	Mainchain	Sidechain
Block 0 Gas	0	0
Block 1 Gas	855849	827509
Block 2 Gas	150071	339063
Total Gas Per Chain	1005920	1166572
Total Gas	2172492	
Euro Cost Per Chain	€1.50888	€1.749858
Total Euro Cost	€3.258738	

Table 5.1: New Community Cost Calculation

The cost to implement horizontal sharding is 0.002172492 *ETH* or about €3.25 at €1500.00 for 1 *ETH* as seen in Table 5.1. With the process of horizontal sharding implemented we can now actualise and validate our sharding design by deploying our contracts and creating a frontend.

5.4 Web App Frontend Implementation

Below are a series of screenshots from the completed BSN frontend implemented using React.js and Web3.js. Material UI or MUI[5] is a frontend react component library used to supplement the creation of the site with a uniform user interface.

5.4.1 Home Page

The application starts on the Home Page, which is seen in Figure 5.2. User's are required to set their public and private key pair so that they might sign off on transactions. They can choose to connect to another already initialised site, through a deployed backend contract's address, or create their own. A navigation bar is available at the top to view the other pages.

[5]https://mui.com/

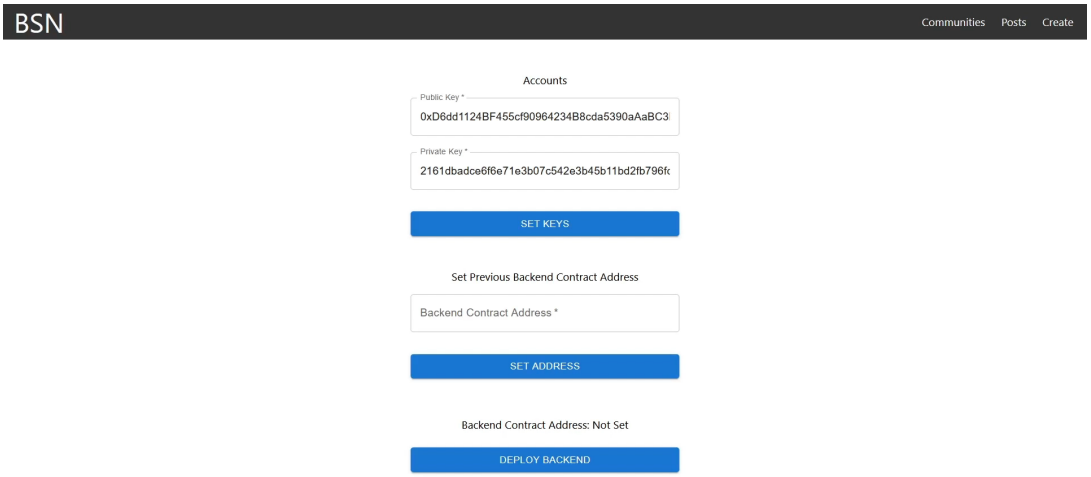

Figure 5.2: Home Page

5.4.2 Create Page

On the Create Page, seen in Figure 5.3, we are able to create a new community on a fresh shard and post content to that community. For new communities, we are required to manually deploy the shard on a free port beforehand. Clicking *Create Community* here deploys the posting contract and community contract connecting the two. Clicking *Post* adds a post to our shard's posting contract.

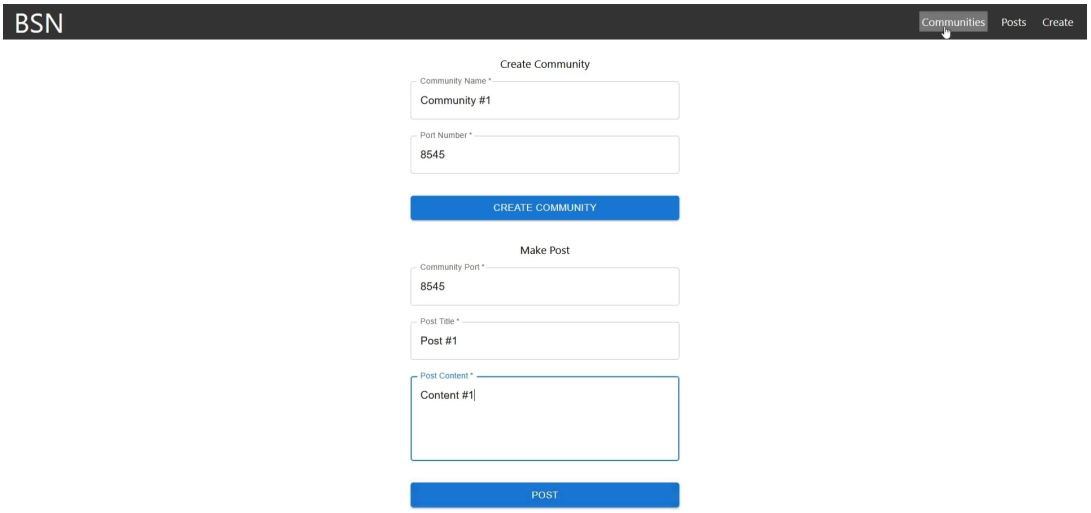

Figure 5.3: Create Page

5.4.3 Community Page

When we click on the Communities Tab, we've presented with the Unloaded Communities Page as seen in Figure 5.4. Pressing the load communities button uses the backend contract to fetch a list of all community shards and returns their name, port and contract address as a table, as seen in Figure 5.5.

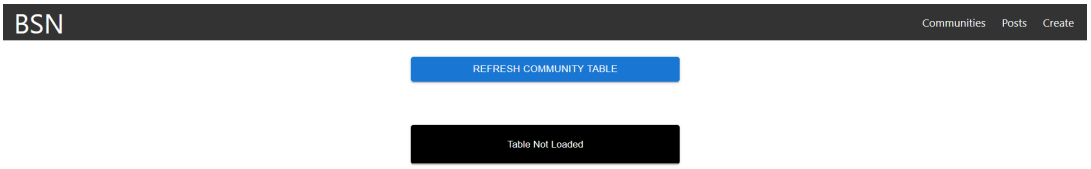

Figure 5.4: Unloaded Community Page

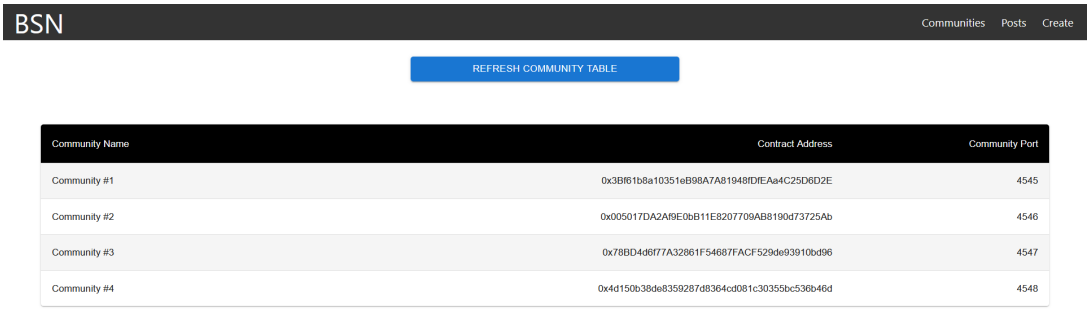

Figure 5.5: Community Page

5.4.4 Posts Page

Much like the Communities Page, once we click on the Posts Tab, we've presented with the Unloaded Posts Page. Inputting a Community Shard's port number, fetch the posting contract at that port and returns a table containing a list of all posts. Each entry of the table contains information on the author, the post's title and the post's content which can be seen in Figure 5.7.

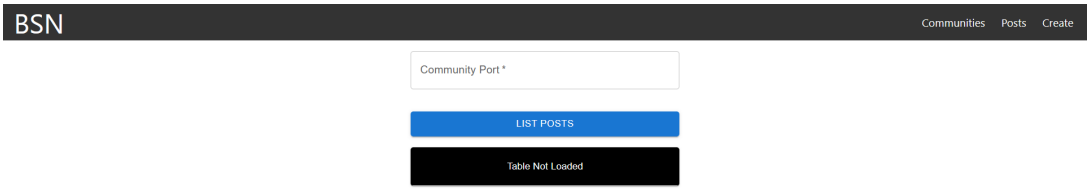

Figure 5.6: Unloaded Posts Page

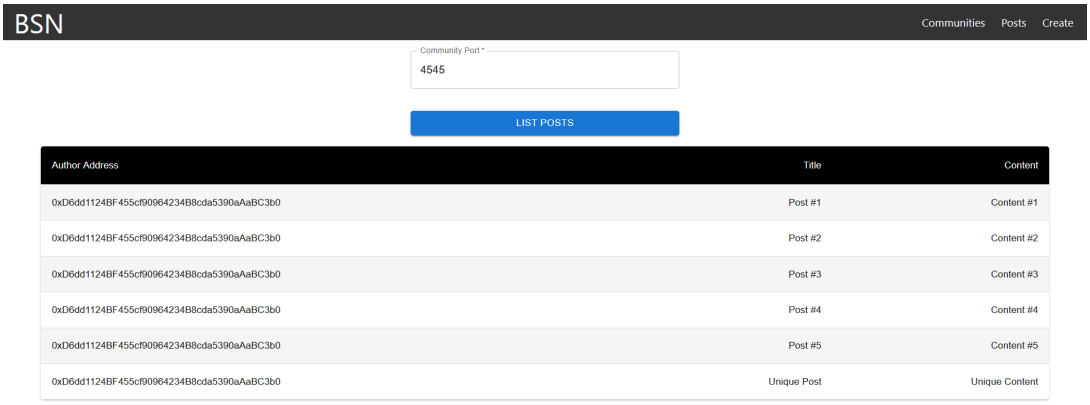

Figure 5.7: Posts Page

From this implementation, we've validated our primary objective of creating a blockchain-based social network, which guarantees a series of rights to each user held solid by the site's constitution and allows the creation of communities and posting hosted on a secure decentralised network. We've also validated that creating this application is possible while achieving an more optimised storage size. This storage improvement is more optimal than what is currently written in State-of-the-Art work for BSNs.

5.5 Web App Implementation Issues

This web app implementation only uses sharding to see its storage reduction, not NiPoPoW as we originally proposed in the NiPoPoW design in section 2.7. Attempting to utilise NiPoPow for our web app implementation would require us to redesign the Ethereum blockchain and all the associated libraries with it. This is not feasible considering the time constraints of this dissertation, and as such NiPoPoW was not used in the web app implementation.

Uploading large datasets to the web app implementation in JavaScript also presented an issue. Currently, there does not exist adequate tooling that allows the upload of a large amount of data to the blockchain in JavaScript, as this process in industry is usually written in a different programming language which is better suited.

Instead, another analytical implementation written in Python and web3.py has been created to utilise our NiPoPoW storage improvement by simulating its construction. This also deals with the uploading and deployment of large datasets of Reddit posts to the chain and compares and evaluates the performance of our three designs.

5.6 Analytical Implementation

After the web app implementation was completed, work began on another version of the tool specifically created for the purposes of analysis to evaluate the performance of our three designs, naive, sharding and NiPoPoW. This implementation was written to deal with the upload of a 0.01MB Reddit corpus of posts from 2016, which we go into more detail on in section 6.

5.6.1 Interlinks Structure

For each block mined in the mainchain, our NiPoPoW interlinks structure, discussed in 2.7, must also be updated. We can represent this structure using a 2D array, where each row represents the number of extra *e* leading zeros, and each column is the index of a block ordered by time mined.

5.6.2 Uploading Dataset Scripts

The Python script used to upload our 0.01MB Reddit Corpus performs as follows:

For all approaches:
Initialise mainchain and prepare dataset:

- Run beacon chain node, set to default network id of 1337, at 127.0.0.1:7545
- Return private key addresses of mainchain wallets
- Filter dataset content to a series of dictionaries containing posts ordered by subreddit name

For the naive approach:

- Iterate through posts
- Validate chain's PoW
- Upload posting contract if the post's community doesn't already exist. Link posting contract to the community.
- Upload post to IPFS, return content addressable hash, store in posting contract
- Repeat for all posts

For the Sharding and NiPoPoW approaches
Allocate a new shard node for each community:

- Generate a new shard for the community if it does not exist, with incremented port and the same mainchain wallets

- Validate chain's PoW, or NiPoPoW for NiPoPoW approach
- Upload posting contract to the new shard
- Upload community contract to new shard, link posting contract
- Crosslink community contract to mainchain.

Then for each new community:

- Upload post to IPFS
- Validate chain's PoW, or NiPoPoW for NiPoPoW approach
- Create content address hash and append to sidechain
- Repeat for all posts
- Stop node once complete

6 Analysis

A series of datasets were uploaded to three implementations of the BSN, the naive approach, the sharding approach and the NiPoPoW approach. Statistics were recorded to analyse the properties of each approach as the dataset of posts were uploaded and stored in text files. This chapter covers the contents and preparation of the dataset, as well as the statistics recorded.

6.1 Dataset

6.1.1 Dataset Origin

To populate our chains with various posts and communities, a dataset containing a collection of Reddit posts supplied was utilised. This corpus of posts was taken from Cornell University's Convokit toolkit [27], a toolkit with tools to extract conversation features which contains our datasets. These corpora themselves were built using Pushshift.io [28], a big-data storage and analytics project started and maintained by Jason Baumgartner.

Of this collection of corpora, I choose the 2016-January corpus. Over the course of its lifetime, Reddit has altered the availability of the metadata behind its posts, and for the first time in 2016, it contained the data we require to create a fully-fledged version of our application. The earliest dataset which followed this standard was in January and was chosen as it was the smallest due to storage limitations on the author's laptop. Originally this dataset was reduced to 10MB dataset of the first 200'000 posts, but this took over 40 hours to upload to one chain, and so was again further reduced to 0.01MB containing only 202 posts.

6.1.2 Datapoint Structure

Each datapoint of the dataset is separated by a newline and represents the metadata of one unique Reddit post sorted by time. The datapoint takes the form of a JSON String, with the following important tags listed in Table 6.1:

Attribute	Description
id	Unique identification string of post
parent_id	Unique identification string of parent post with which post is a reply to
author	Username of the author who made the post
body	Content of the post
subreddit	Name of subreddit where the post was made
subreddit_id	Unique ID of subreddit, as some subreddit have conflicting names
created_utc	Time and Date of post creation in UTC format
score	Overall score of post
ups	Total likes of post
controversiality	Controversy score of a post

Table 6.1: Dataset Datapoint Tags

Extra metadata, including whether the post was edited, pinned, had a post flair, etc., within each datapoint was not used as it was not required to populate a fully-fledged version of the application. Further documentation about these datapoints is available at the official convokit.cornell.edu site. Should an author have deleted their post or account before it is retrieved by pushshift, the author and body tags are altered to only reference "deleted".

6.1.3 Organising Dataset

To improve the efficiency of the upload process, we reorganised this dataset to improve upload times. The required start and stop time of a shard node takes about 2-5 seconds. The dataset's posts were ordered first by subreddit and then by time. This meant that our sharding and NiPoPoW implementations only required shard node startup and stopping for every community rather than every post, reducing this operation from 202 to 146 instances.

6.2 Dataset Makeup

After pruning our 2016-January dataset to 0.01MB, it contained a total of 202 datapoints and thus 202 posts total. Of these datapoints, there were a total of 149 communities. The vast majority of communities in this dataset only have one post. We should then expect from our evaluation of the sharding approach that the majority of shards only require a few hundred kilobytes, with a histogram of the size of each shard similar in look to Figure 6.1.

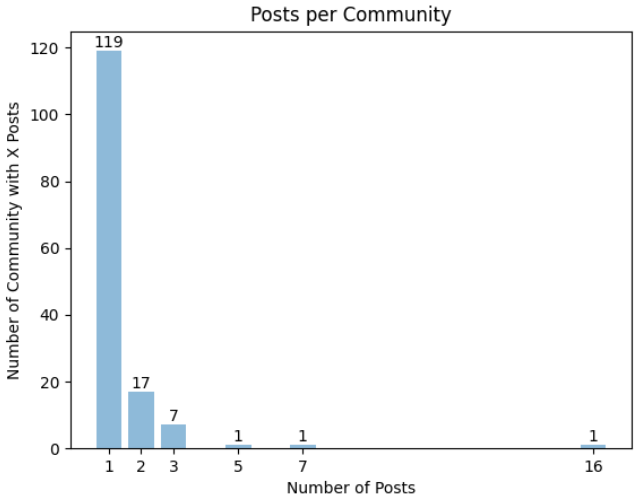

Figure 6.1: Posts Per Community

6.3 Recorded Statistics

Relevant statistics about each approach were recorded at the launch of the application and subsequent to the mining and appending of each new block on the mainchain. Since the objectives of this dissertation are focused on the creation of a BSN with optimised storage, the majority of recorded statistics are relevant to the mainchain. The contents of each shard and the mainchain were also stored in a separate directory. For each block, the statistics recorded are included in Table 6.2.

Attribute	**Description**
most_recent_block_index	Index of the most recently mined block
most_recent_block_size	Size in bytes of the most recently mined block
post_count	Number of total posts uploaded so far with this approach
elapsed_time	Elapsed time since the initialising of the chain
gas_used	Gas used to mine the most recent block.

Table 6.2: Recorded Chain Statistics

From these statistics, we can calculate other relevant information, such as the total size of the chain, the cost and time taken to generate it, etc., which will be discussed in our Evaluation section 7. Along with these statistics, at the mining and appending of each new block on the NiPoPoW approach. Table 6.3 contains relevant information to the creation and validation of our proofs.

Attribute	Description
most_recent_block_index	Index of the most recently mined block
proof_time	Time required to validate the Proof-of-Work of the whole chain
nipopow_proof_time	Time required to validate the Non-Interactive Proof of Proof-of-Work of the most upper valid chain of our interlinks structure
bottom_chain_length	Length of the bottom chain of the interlinks structure
top_chain_length	Length of the uppermost chain with of a valid length, meeting the requirements discussed in 2.7.4

Table 6.3: Recorded NiPoPoW Statistics

Finally, once the upload is completed, we store these statistics as a text representation of the populated interlinks structure in its own file.

6.4 Updating NiPoPoW Requirements

In section 2.7, we covered that security parameter m ideally should be 6. This requires a chain of length 420 blocks before we can utilise a higher level for our Proof of Proof of Work. With only 149 communities, then with our current parameters, we will not reach an upper chain in our analysis. To witness the effect of NiPoPoW then, we will reduce our security parameter m to 1.5, where we should be able to see a switch to an upper chain around block 80. It is at this point where our proof at least be less than the second layer of the interlinks structure, with $4 * 1.5 * log_2(80) = 37.9$ and our second layer having 40 blocks on average. We will also require this upper chain to be at a minimum length 5 so that we can at least abide by the suffix parameter discussed by Nakamoto in [1] and Juan Garay et al. in [8].

7 Evaluation

After uploading the contents of the Reddit dataset to each of our approaches, a collection of graphs were generated based on the various statistics collected during the analysis process. This chapter covers the expectations of our analysis and evaluates the results of each graph, specifically comparing the differences between the three approaches and reasons why their properties differ.

7.1 Expectations

Before evaluating our data, it was expected that our naive approach would be far larger than in size that the other approaches, due to the overloading of posts by sharding. The sharding approach is expected to be only slightly larger than the NiPoPoW approach and the reduction of the mainchain by NiPoPoW. During 2022 reporting, Reddit claimed to have 2.8 million subreddits [29], with a total of 430 million posts [30] so we might expect that our naive approach would be about 150 times larger our sharding/NiPoPoW approach as it stores posts. For the same reason, we would expect that the cost would follow a similar ratio between approaches.

7.2 Results

A total of nine graphs were generated. These graphs serve the purpose of examining the storage, cost, chain makeup and proof validation times of each chain.

7.2.1 Storage Evaluation

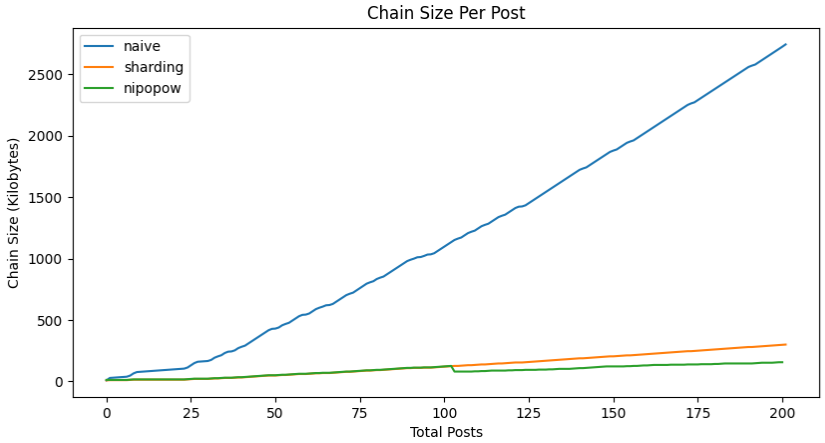

Figure 7.1: Chain Size Per Post

The raw ledger sizes of the approaches generated from our dataset per post are shown in Figure 7.1. As expected, the naive approach has a far larger size sitting at 2.57MB, while the sharding approach only required 0.3MB, and the NiPoPoW approach only required 0.15MB. This reduction is due to the offloading of posts from the mainchain to its shards and is further reduced by the smaller chain due to NiPoPoW. Unlike how we predicted in 7.1, our naive approach is not 150 times larger than our sharding approach. This is likely due to our dataset, which consists mostly of unpopulated communities with only one post, which is not representative of Reddit's ratio of communities to posts ratio.

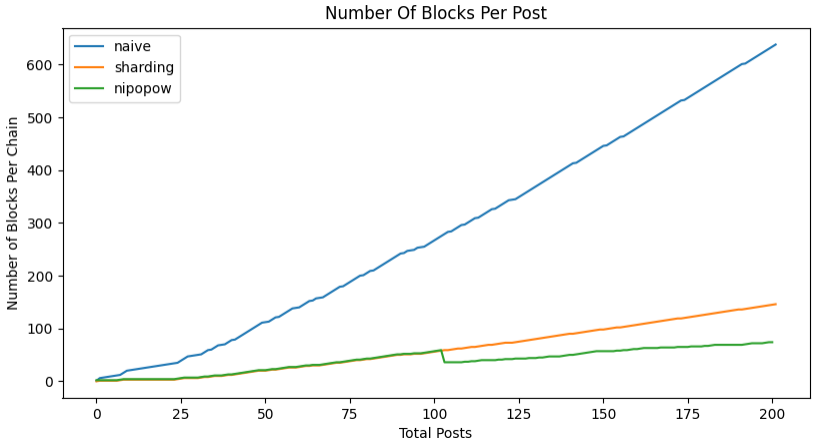

Figure 7.2: Number of Blocks Per Post

Our number of blocks per post graph, Figure 7.2 is similar to Figure 7.1, exhibiting the same behaviour. While all plots in Figure 7.2 are practically straight lines, unlike the sharding plot, the naive and NiPoPoW plot has some slight bumps due to the additional posts the naive chain takes on, and the occasional smaller upperchain produced

by the NiPoPoW approach. As with Figure 7.1, we see a *drop* on the NiPoPoW plot around the 100 total posts mark. The reason for this drop is that at this point, our proof meets our security requirements of an upperchain of length $k + (m * log(|C|))$, as discussed in 2.7.4, allowing us to use the smaller upperchain. This drop allows our chain's length to decrease from 149 blocks to 74 blocks by the end of the run, reducing our chain block length by 50%.

7.2.2 Timing Evaluation

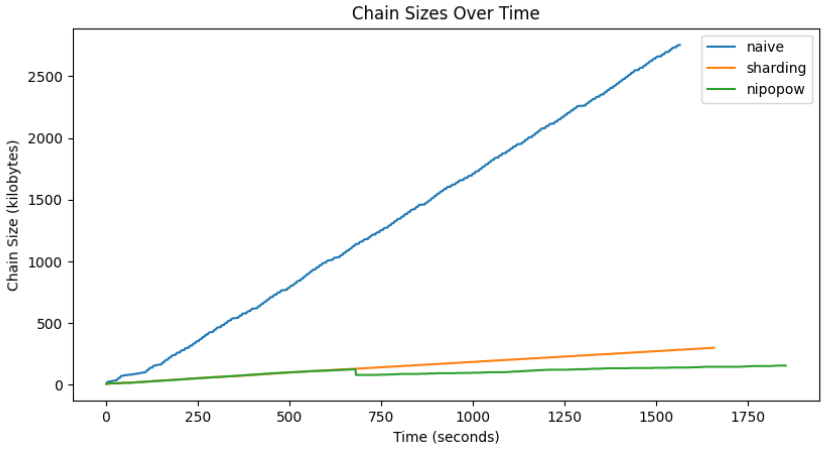

Figure 7.3: Chain Size Over Time

The raw ledger sizes of the approaches generated from our dataset over time are shown in Figure 7.3. Surprisingly, we see only a slight difference in time to generate chains. We would expect that the sharding (27.6 mins) and NiPoPoW (30.9 mins) approaches would take considerably longer than the naive approach (26.1 mins) as they have the added time sink of starting and stopping shard nodes. I suspect the reason for this is due to the efficiency of our horizontal sharding, which begins mining the crosslink block on the mainchain in parallel, saving time. We can see the NiPoPoW approach does indeed take slightly longer than the sharding approach at 30.9 minutes due to the generation of the Interlinks structure and validation of the PoW and NiPoPoW.

7.2.3 Cost Evaluation

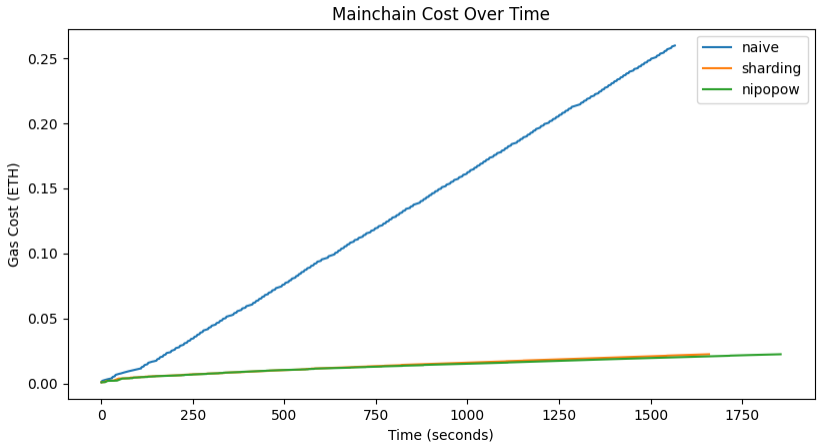

Figure 7.4: Total Mainchain Cost Over Time

Figure 7.4 compares the total cost the mainchain accrues over time. This cost is far higher in the naive approach (0.2598 ETH) compared to the sharding and NiPoPoW approaches (0.0224 ETH), a 91% cost reduction. This reduction is simply due to the offloading of mining costs to the shards. Should we add the mining cost accrued by these shards to the mainchain, all three approaches plots would roughly overlap, resulting in the near the same cost.

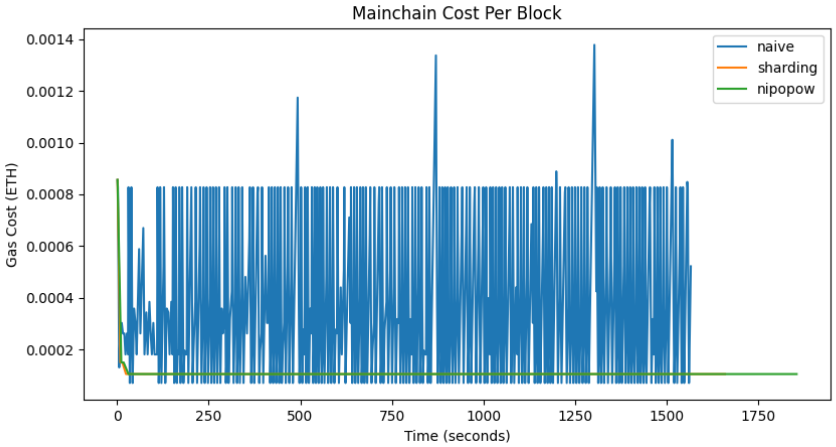

Figure 7.5: Cost Per Block

Figure 7.5 show the cost the mainchain accrues per block. While the naive approach constantly oscillates between 0.0001 and 0.0008 ETH, our sharding and NiPoPoW approaches settle near 0.001 after the backend is deployed. This suggests that deploying contracts which construct communities and create new posts is the more expensive of the operations. A closer look at this fact is viewable in Figure 7.6.

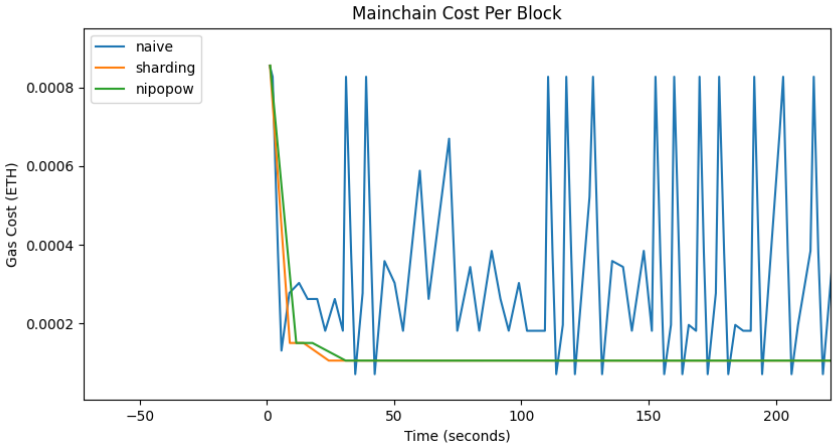

Figure 7.6: Cost Per Block - Zoom in

7.2.4 Sharding Evaluation

In section 6.2, we predicted that a histogram of the size of our shards would be similar in form to the bar chart of communities, Figure 6.1. We see that this is the case in Figure 7.7, where the vast majority of shards are less than 0.2MB in size. We see this as a positive, as having many small shards rather than one set of large shards will better divide the work for miners. Miners will have less competition to mine the blocks of smaller shards, reducing cost and energy waste. A small shard size implies an inactive community. Sharding allows users to drop blocks from inactive communities they would have been forced to keep in our naive design.

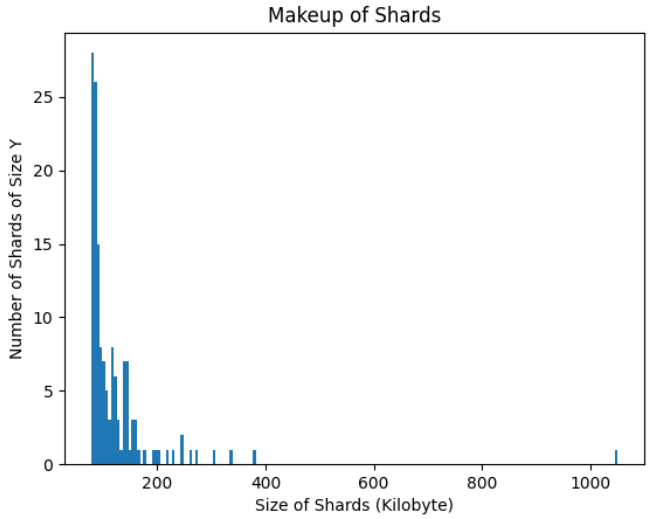

Figure 7.7: Markup of Shards

7.2.5 NiPoPoW Evaluation

7.2.5.1 Distribution of *e* Extra Leading Zeros

Figure 7.8 was created to validate our equation from section 2.7.1, where approximately $1/2^e$ of all blocks with have e extra zeros. While not an exact relationship, we see that each successive bar is about half the height of the previous bar. We can suspect then that we should be able to generate a secure, Non-Interactive Proof of Proof of Work. Figure 7.8 confirms that we can approximate the PoW of a chain based on the length of some level e of our interlinks structure.

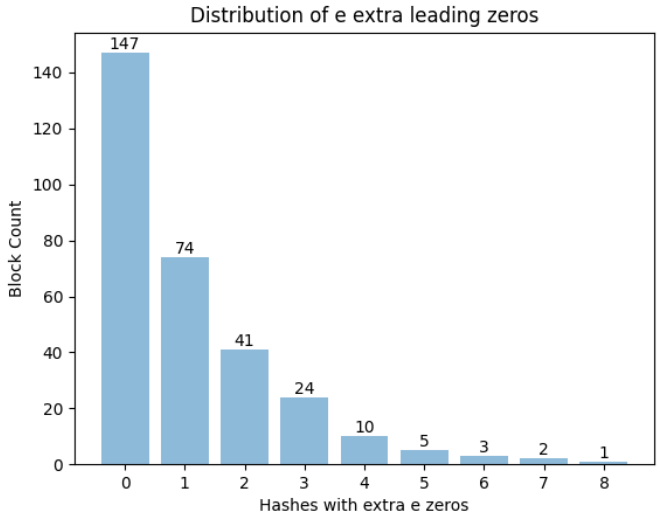

Figure 7.8: Number of Hashes with e Extra Leading Zeros

7.2.5.2 Comparing Proof Validation Times

A comparison of proof validation times is seen in Figure 7.9. You can diagnose from this graph that we began proving the validity of our chain using the upperchain via NiPoPoW around the 100th iteration or 100th block. It's at this point that NiPoPoW has a faster validation time as it requires fewer blocks to be proven. While time to proof PoW continues to rise throughout the uploading, NiPoPoW does not have as steep a rise as PoW. At upperchain e=1, NiPoPoW only adds a block to validate every other block or so causing a more gentle slope.

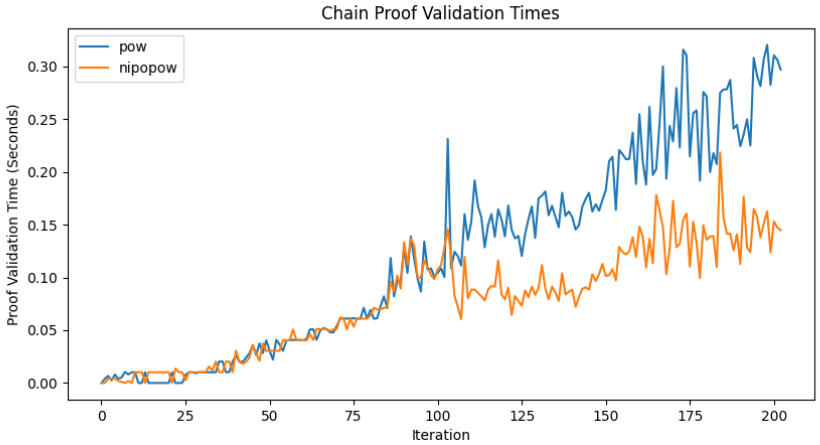

Figure 7.9: Comparing Proof Times

7.3 Summarised results

Considering the summary of our results as seen in Table 7.1, and those discussed in the previous section, it's clear that NiPoPoW is the superior of all three choices at the cost of extra time. However, considering how the size of the upperchain should decrease with time, we might expect that at some point that as the chain and network size grow, the time to mine a new block will eventually be faster than our naive and sharding implementations. The NiPoPoW and sharding chains were also cheaper, and at the current price of Ethereum hovering around €1'500.00, we can expect the NiPoPoW and sharding mainchains to cost €33.6 to mine, while the naive mainchain would cost €389.7 to mine.

	Naive	Sharding	NiPoPoW
Chain Size (MB)	2.57MB	0.3MB	0.15MB
Total Blocks	638	146	74
Timing (min)	26.1m	27.6m	30.9m
Cost Per Chain (ETH)	0.2598	0.0224	0.0224
Euro Cost (€)	€389.7	€33.6	€33.6

Table 7.1: Results Summary Table

8 Conclusion

The primary objective of this dissertation was to determine the viability of implementing a Reddit-like blockchain-based social network and to investigate and address the scalability issues that such an application would have. A specific focus was placed on addressing storage optimisations while maintaining the application's security and principles. This goal was achieved by:

- Researching the necessary background material and literature to understand such a project
- Creating a user-friendly blockchain social network site that guaranteed a set of rights to its users through a constitution and allowed users to create communities and post content.
- Designing and implementing storage optimisations through the use of Sharding and Non-Interactive Proof of Proof-of-Works while maintaining chain security.
- Evaluating and comparing the overall storage, block efficiency, mining efficiency and cost of different designs.

This research has shown that a blockchain social network which utilises Sharding and Non-Interactive Proof of Works is more storage and cost-efficient to the mainchain, at the cost of being slightly slower to generate a chain for. Integrating Sharding and NiPoPoW into the architecture of a blockchain-based social network and evaluating the results is something not yet written about in literature. For a dataset of 202 posts with 149 communities, we saw a storage decrease from 2.57MB to 0.15MB, a 94.17% reduction.

8.1 Future Work

Considering that this dissertation dealt with the creation of a blockchain-based social network, a new and emerging topic with little previous literature, there is significant potential and multiple directions for possible future work.

Our NiPoPoW implementation described in this dissertation is only a proof-of-concept. Before considering its use in industry its performance should be tested using larger datasets and with real users. The evaluation we performed was done using a 0.01MB Reddit Dataset from January 2016, which is minuscule compared to the true scale of Reddit. One avenue is to attempt to upload all 3 million Reddit communities or at least a large enough proportion and validate whether we see a size reduction from 3TB to 500MB as outlined in 2.7.

The features mentioned in section 4.5, such as the voting protocol or ban sharding, were ultimately not implemented in the final version. Currently, this implementation only deals with the upload of text-based content, not images or video, which most sites allow. These features, as well as the DAO or reputation-based system described in Ningyuan Chen et al. [20] could be implemented and their feasibility evaluated when compared to CSN.

During the writing of this dissertation, the majority of Ethereum miners switched to a new fork of Ethereum, known as *The Merge* [31], which uses Proof-of-Stake to mine new blocks. Proof-of-Stake has the potential to reduce mining costs, improving decentralisation by encouraging miners with lower hashing power and increasing block throughput. Proof-of-Stake can also be combined with NiPoPoW. These factors would be of great benefit to a mobile implementation of a BSN as it could allow mining to be operable on smartphones even with larger communities. Future work could compare the benefits and disadvantages of using PoW or PoS with NiPoPoW when creating a BSN.

In our analytical implementation, we simulated a NiPoPoW blockchain based on the Ethereum blockchain. With an ideal implementation, we would create an entirely new blockchain that uses NiPoPoW and the associated libraries. Our current implementation acts only as a simulation giving us a general idea of the results we'd expect. Future research could involve the development of a NiPoPoW chain which utilises smart contracts rather than just a simulation.

Bibliography

[1] Satashi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Aug. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[2] Tom Scott and Computerphile. *Hashing Algorithms and Security*. Youtube. URL: https://www.youtube.com/watch?v=b4b8ktEV4Bg. Accessed on: 20.08.2022.

[3] Wouter Penard and Tim van Werkhoven. *On the Secure Hash Algorithm family*. 2001.

[4] Adam Back. *Hashcash - A Denial of Service Counter-Measure*. Sept. 2002. URL: http://www.hashcash.org/hashcash.pdf.

[5] Hal Finney. *RPOW - Reusable Proofs of Work*. https://nakamotoinstitute.org/finney/rpow/theory.html. Aug. 2004.

[6] Demetrius Rocha and Rafael Almeida. *Blockchain and Data Integrity in Computer System Validation*. URL: https://fivevalidation.com/blockchain-and-data-integrity-in-computer-system-validation/. [Online] Accessed on: 11.01.2023.

[7] W.S. Stornetta S. Haber. "How to time-stamp a digital document". In: *Journal of Cryptology* 3.2 (1991), pp. 99–11.

[8] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. "The Bitcoin Backbone Protocol: Analysis and Applications". In: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Springer Berlin Heidelberg, 2015, pp. 281–310. ISBN: 978-3-662-46803-6.

[9] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. Nov. 2013. URL: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf. Accessed on: 21.11.2022.

[10] Shuai Wang et al. "An Overview of Smart Contract: Architecture, Applications, and Future Trends". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 108–113. DOI: 10.1109/IVS.2018.8500488.

[11] Alchemy Team Web3 University. *Ethereum Sharding: An Introduction to Blockchain Sharding*. 2022. URL: %5Curl%7Bhttps://www.web3.university/article/ethereum-sharding-an-introduction-to-blockchain-sharding%7D. [Online] Accessed on: 12.01.2023.

[12] Gang Wang et al. "Sok: Sharding on blockchain". In: *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 2019, pp. 41–61.

[13] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. "Double-Spending Fast Payments in Bitcoin". In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2012, pp. 906–917. ISBN: 9781450316514. DOI: 10.1145/2382196.2382292. URL: https://doi.org/10.1145/2382196.2382292.

[14] Vitalik Buterin. *Why sharding is great: demystifying the technical properties*. https://vitalik.ca/general/2021/04/07/sharding.html. 2021.

[15] *10 Reddit Statistics Every Marketer Should Know in 2023 [Online]*. URL: https://www.oberlo.com/blog/reddit-statistics. Accessed on: 04.2023.

[16] Aggelos Kiayias, Nikolaos Lamprou, and Aikaterini-Panagiota Stouka. "Proofs of Proofs of Work with Sublinear Complexity". In: *Financial Cryptography and Data Security*. Ed. by Jeremy Clark et al. Springer Berlin Heidelberg, 2016, pp. 61–78. ISBN: 978-3-662-53357-4.

[17] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. "Non-interactive Proofs of Proof-of-Work". In: *Financial Cryptography and Data Security*. Ed. by Joseph Bonneau and Nadia Heninger. Springer International Publishing, 2020, pp. 505–522. ISBN: 978-3-030-51280-4.

[18] Quanqing Xu et al. "Building an Ethereum and IPFS-Based Decentralized Social Network System". In: *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. 2018, pp. 1–6. DOI: 10.1109/PADSW.2018.8645058.

[19] Tharuka Sarathchandra and Damith Jayawikrama. "A decentralized social network architecture". In: *2021 International Research Conference on Smart Computing and Systems Engineering (SCSE)*. Vol. 4. 2021, pp. 251–257. DOI: 10.1109/SCSE53661.2021.9568334.

[20] Ningyuan Chen and David Siu-Yeung Cho. "A Blockchain based Autonomous Decentralized Online Social Network". In: *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. 2021, pp. 186–190. DOI: 10.1109/ICCECE51280.2021.9342564.

[21] Shuai Zeng, Yong Yuan, and Fei-Yue Wang. "A decentralized social networking architecture enhanced by blockchain". In: *2019 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*. 2019, pp. 269–273. DOI: 10.1109/SOLI48380.2019.8955104.

[22] Ned Scott and Dan Larimer. *A protocol for enabling smart, social currency for publishers and content businesses across the internet*. Aug. 2017. URL: https://hive.io/bluepaper.pdf.

[23] Cong Nguyen et al. "Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities". In: *IEEE Access* PP (June 2019), pp. 1–1. DOI: 10.1109/ACCESS.2019.2925010.

[24] Parikshit Hegde et al. "Achieving Almost All Blockchain Functionalities with Polylogarithmic Storage". In: *Financial Cryptography and Data Security*. Ed. by Ittay Eyal and Juan Garay. Springer International Publishing, 2022, pp. 642–660. ISBN: 978-3-031-18283-9.

[25] *Web 3.0 challenges and solutions: Transaction spamming in blockchain*. URL: https://nkn.org/community/blog/web-3-0-challenges-and-solutions-transaction-spamming-in-blockchain/. Accessed on: 04.04.2023.

[26] Vitalik Buterin. *EIP-170: Contract code size limit. Ethereum Improvement Proposals, no. 170*. [Online] Accessed on: 05.04.2023. Nov. 2016. URL: https://eips.ethereum.org/EIPS/eip-170.

[27] *Convokit - Cornell Conversational Analysis Toolkit*. [Online] Accessed on: 02.2023. URL: https://convokit.cornell.edu/.

[28] *Pushshift.io - Learn about Big Data and Social Media Ingest and Analysis*. [Online] Accessed on: 02.2023. URL: https://pushshift.io/.

[29] *20+ Riveting Reddit Statistics [2023]*. [Online] Accessed on: 04.2023. URL: https://www.zippia.com/advice/reddit-statistics/.

[30] *Reddit Revenue and Growth Statistics (2023)*. [Online] Accessed on: 04.2023. URL: https://www.usesignhouse.com/blog/reddit-stats.

[31] Mikhail Kalinin, Danny Ryan, and Vitalik Buterin. *EIP-3675: Upgrade consensus to Proof-of-Stake. Ethereum Improvement Proposals, no. 3675*. July 2021. URL: https://eips.ethereum.org/EIPS/eip-3675. [Online] Accessed on: 02.2023.

A1 Appendix

A1.1 Smart Contracts

A1.1.1 Backend.sol

```
pragma solidity ^0.8.13;

// SPDX-License-Identifier: MIT

contract Backend {

    address[] public moderators;
    address public founder;

    mapping(string => address) public communityAddresses;
    mapping(address => uint16) public communityPorts;
    mapping(uint16 => address) public communityPortToAddress;
    string [] public communitiesNames;
    int public communitiesNumber = 0;


    constructor() {
        founder = tx.origin;
        moderators.push(tx.origin);
    }

    function createCommunity(string memory _communityName,
        address _communityAddress, uint16 _communityPort) public {
        communityAddresses[_communityName] = _communityAddress;
        communityPorts[_communityAddress] = _communityPort;
        communityPortToAddress[_communityPort] = _communityAddress;
```

```
        communitiesNames.push(_communityName);
        communitiesNumber++;
    }

}
```

A1.1.2 Community.sol

```
pragma solidity ^0.8.13;

// SPDX-License-Identifier: MIT

contract Community {

    string public name;
    uint16 public port;
    address public parent;

    address[] public moderators;
    address public founder;

    address public postingContract;

    constructor(string memory _name, uint16 _port,
        address _parent, address _postingContract) {
        founder = tx.origin;
        moderators.push(tx.origin);

        name = _name;
        port = _port;
        parent = _parent;

        postingContract = _postingContract;
    }

}
```

A1.1.3 Posting.sol

```
pragma solidity ^0.8.13;

// SPDX-License-Identifier: MIT

contract Posting {

    struct Post {
        uint32 postCount;
        address _author;
        string _title;
        string _content;
        uint16 _communityPort;
    }

    uint16 public communityPort;
    uint32 public postCount;
    mapping(uint32 => Post) postsByID;
    Post[] public posts;

    event NewPost(uint32 indexed _postID, string indexed _title,
        string indexed _content);
    event NewPostAuthor(address indexed _postAuthor,
        uint16 indexed _communityPort, uint32 indexed _postID);

    constructor(uint16 _communityPort) {
        communityPort = _communityPort;
        postCount = 0;
    }

    function makePost(string memory _title, string memory _content)
        public {
        Post memory _post = Post(postCount, tx.origin,
            _title, _content, communityPort);
        postsByID[postCount] = _post;
        posts.push(_post);

        emit NewPost(postCount, _title, _content);
        emit NewPostAuthor(tx.origin, communityPort, postCount);
```

```
            postCount++;
        }

}
```

A1.2 Project Source Code

The completed project source code can be found at https://github.com/Keaneyjo/bsn.