

2022

Incentive Analysis of Blockchain Technology

Rahul Reddy Annareddy
ra00010@mix.wvu.edu

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>



Part of the [Computer and Systems Architecture Commons](#), [Digital Communications and Networking Commons](#), [Other Computer Engineering Commons](#), and the [Other Engineering Commons](#)

Recommended Citation

Annareddy, Rahul Reddy, "Incentive Analysis of Blockchain Technology" (2022). *Graduate Theses, Dissertations, and Problem Reports*. 11562.
<https://researchrepository.wvu.edu/etd/11562>

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

Incentive Analysis of Blockchain Technology

by

Rahul Reddy Annareddy

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

Roy S. Nutter Jr., Ph.D.
Y. V. Ramana Reddy, Ph.D.
Matthew C. Valenti, Ph.D., Chair

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2022

Keywords: Blockchains, Cryptocurrency

Copyright 2022 Rahul Reddy Annareddy

Abstract

Incentive Analysis of Blockchain Technology

by

Rahul Reddy Annareddy
Master of Science in Computer Science

West Virginia University

Matthew C. Valenti, Ph.D., Chair

Blockchain technology was invented in the Bitcoin whitepaper released in 2008. Since then, several decentralized cryptocurrencies and applications have become mainstream. There has been an immense amount of engineering effort put into developing blockchain networks. Relatively few projects backed by blockchain technology have succeeded and maintained a large community of developers, users, and customers, while many popular projects with billions of dollars in funding and market capitalizations have turned out to be complete scams.

This thesis discusses the technological innovations introduced in the Bitcoin whitepaper and the following work of the last fifteen years that has enabled blockchain technology. A complete implementation of a blockchain network and cryptocurrency based on first principles is presented in order to illustrate the design and technical choices that have to be made while implementing a blockchain.

To understand the incentives that drive the adoption of blockchain technology, two successful blockchain projects, CryptoKitties, which is a Non-Fungible Token (NFT) project based on the ERC-721 specification and Helium, are analysed as case studies. These case studies first examine how these projects leveraged blockchain technology from a technical standpoint, followed by a discussion of the incentives that were built into the projects, which allowed millions of users to participate in these networks and create value.

The economic incentives created by cryptoassets which are a combination of cryptocurrency and tokens, are explored. The purpose of this thesis is to provide an informational overview of the incentives and technical choices driving the development and adoption of blockchain technology.

Acknowledgements

I would first like to thank my committee chair and advisor, Dr. Matthew C. Valenti, for giving me the opportunity to work with him and his students. This thesis would not be possible without his constant guidance and support.

I would also like to thank Dr. Roy S. Nutter and Dr. Y. V. Ramana Reddy for being on my committee.

Finally, I would like to express my gratitude to my family. I would like to express my thanks to my mother, father, and brother. Their support seemingly has no limit, and has been much appreciated throughout my life.

Contents

Acknowledgements	iii
List of Figures	vii
List of Tables	ix
1 Introduction	1
1.1 Introduction	1
1.2 Thesis Outline	2
2 Blockchains Explained	4
2.1 Components That Enable Blockchain Networks	4
2.1.1 Cryptographic Hash Functions	4
2.1.2 Transactions	6
2.1.3 Asymmetric Key Cryptography	7
2.1.4 Ledger	8
2.1.5 Blocks and Chaining Blocks	8
2.2 Types of Blockchains	9
2.2.1 Permissionless	10
2.2.2 Permissioned	10
2.3 Reaching Consensus	11
2.3.1 Proof of Work (Bitcoin)	12
2.3.2 Proof of Stake (Ethereum)	13
2.3.3 Conflicts and Resolution	14
2.4 Chapter Summary	16
3 Blockchain Implementation	17
3.1 Blockchain Node Components	17
3.1.1 Blocks and Blockchain	18
3.1.2 Peer-To-Peer Messages	19
3.1.3 Transactions and Transaction Pool	20
3.1.4 Mining and Rewards	21
3.2 Blockchain Explorer and Wallet User Interface	23
3.2.1 Blockchain Explorer	23
3.2.2 Wallet	24

3.2.3	Request Coins from Genesis Block	25
3.2.4	Transfer Coins	25
3.2.5	Mine Transactions and Transaction Pool	26
3.3	Summary	26
4	Helium Blockchain Case Study	27
4.1	Principles Behind the Helium Network	27
4.1.1	LoRa and LoRaWAN	27
4.1.2	Helium Blockchain	28
4.1.3	LongFi	28
4.2	Components of the Helium Network	29
4.2.1	Hotspots	29
4.2.2	Routers	29
4.2.3	Validators	30
4.2.4	Helium Console	30
4.2.5	Devices and Device Owners	30
4.3	Proof Of Coverage	31
4.3.1	Consensus Protocol	34
4.3.2	Honeybadger BFT	35
4.3.3	Proof Of Location	35
4.4	Helium Token Economics and Rewards	36
4.4.1	Helium Token (HNT)	36
4.4.2	Data Credits (DC)	36
4.4.3	Burn and Mint Economics	37
4.5	Incentive Analysis	38
4.5.1	Network Growth	38
4.5.2	Validators	39
4.5.3	Token penalty for inaccurate location	39
4.5.4	Helium Improvement Proposals (HIP)	39
4.5.5	Community Owned Hardware	40
4.6	Summary	40
5	Non Fungible Tokens and CryptoKitties	41
5.1	Non Fungible Tokens	41
5.1.1	Standardization	42
5.1.2	Immutability and Provable Scarcity	42
5.1.3	Programmability	43
5.1.4	ERC-721	43
5.2	CryptoKitties	44
5.3	Incentives Analysis of CryptoKitties	45
5.3.1	Limited Supply	45
5.3.2	Speculative Mechanics, Liquidity, and Money Making Economics	45
5.3.3	Viral Story and Marketing	46
5.4	Summary	47

6	Blockchain Incentive Analysis	48
6.1	Game Theory	48
6.1.1	Nash Equilibrium	49
6.1.2	Schelling Point	49
6.2	Incentive Analysis	49
6.2.1	Double Spend Problem	50
6.2.2	Attack Models	51
6.2.3	Checkpoint Proof of Work Consensus	54
6.3	Summary	55
7	Conclusion	56
7.1	Conclusions	56
7.2	Future Research	59
	References	61

List of Figures

2.1	Simple program to compute SHA256 Hash of a File	6
2.2	Verifying a transactions digital signature. Figure is from [3].	7
2.3	Chaining of blocks over time	8
2.4	Permissioned and permissionless blockchain nodes visualized. Figure is from [3].	11
2.5	Simple program to check hashtimes with increasing number of leading zeros .	13
2.6	Blockchain confict visualized where miners adopt different blocks. Figure is from [2].	15
2.7	When the next block is add to block B, the chain containing block A is orphaned. Figure is from [2].	15
3.1	Blockchain Architecture.	18
3.2	Block Class Implementation.	18
3.3	Blockchain Class Implementation.	19
3.4	Genesis Block Creation Code.	19
3.5	Message Class Implementation.	20
3.6	Sending Message to Peer Nodes.	20
3.7	Transaction Class Implementation.	21
3.8	Transaction Pool Class Implementation.	21
3.9	Function that mines for the next block.	22
3.10	Block Hash Computation Implementation.	22
3.11	Blockchain Explorer User Interface.	23
3.12	Wallet User Interface	24
3.13	Request coins from Genesis Block User Interface.	24
3.14	Create Transactions User Interface.	25
3.15	Transaction Pool and Mine Transactions User Interface.	26
4.1	Helium Overview. Figure is from [5].	30
4.2	Challenger and Beaconer. Figure is from [4].	32
4.3	Hotspots that witnessed the challenge and received packets from the beacon. Figure is from [4]	33

4.4	Helium Explorer representation of PoC challenge. The large white dot represents the Beaconer. Each of the yellow dots represents a witness. The background is a hexagonal grid depicting the number of hotspots in each hexagon. Figure is from [4].	34
4.5	The cycle that allows the network to grow while keeping the prices of DC and HNT stable and all components of the network working. Figure is from [8].	38
5.1	Fungible vs Non-Fungible digital assets. Figure is from [11].	42
5.2	ERC-721 Basic Primitives code. Figure is from [12].	43
5.3	CryptoKitties with different attributes visualized. Figure is from [14].	44
6.1	Red chain represents the correct chain that is replaced when the longer blue chain is published by the dishonest miner.	50

List of Tables

2.1	SHA256 hash outputs for different inputs	5
2.2	Increasing hash difficulty by increasing the number of leading zeros the hash must have (Results for M1 ARM Processor)	12
6.1	Payoff matrix to explain Nash Equilibrium.	49
6.2	Payoff matrix showing rewards for publishing with or against the majority of miners in the blockchain network.	53
6.3	Payoff matrix showing the improved reward for being against the majority of miners publishing blocks. The optimal strategy for individual miners is different from the strategy for miners participating in a group and no Nash equilibrium exists.	54
6.4	Payoff matrix showing rewards when the inversion occurs and the minority becomes the majority. Because the minority the attacker promised bribes to is now the majority, the attacker was able to publish their block with zero cost incurred.	54

Chapter 1

Introduction

1.1 Introduction

Blockchains were introduced in 2008 with a white paper, *Bitcoin: A Peer-to-Peer Electronic Cash System* [1] published under the pseudonym Satoshi Nakamoto. A blockchain is a distributed database or ledger that is shared among the nodes of a computer network. Blockchain technology was created to enable cryptocurrency systems like Bitcoin. Over the past decade, new breakthroughs in cryptography and consensus algorithms have enabled blockchains to be used for various applications.

This thesis focuses on two areas of blockchain technology. The first area of focus is understanding the core concepts of blockchains by implementing a blockchain network based on first principles. The second area of focus is understanding the incentives that are built into blockchains as well as the incentives that have to be understood and implemented to create useful and successful blockchain projects.

While blockchain technology has been widely documented in the literature, a deep understanding of blockchain concepts is best achieved by implementing one using first principles. Thus, the first contribution of this thesis is an implementation of a complete blockchain network.

Decentralized per-to-peer file sharing systems like Napster and Bittorrent existed before Bitcoin. These systems never reached large-scale adoption because of poorly designed incentives. For instance with Bittorrent, the network works well if users of the network con-

tinue to seed the files they download. The Bittorrent network does not incentivize the users seeding files with any economic reward. This causes users to stop participating as soon as they download a file. Over a period of time this leads to decreasing network performance until the system is no longer viable. To avoid this fate, cryptocurrencies were designed with incentives that reward honest participants and penalize dishonest participants. Before the invention of Bitcoin and blockchain technology, the creation of a trustless distributed ledger was considered impossible.

Looking at the landscape of blockchain based projects since Bitcoin was introduced, several major projects with billions of dollars in funding like Celcius and more recently FTX have turned out to either be Ponzi schemes or outright scams. Very few blockchain based projects have succeeded and continue to have active communities. Two such projects are the Helium Network and an Ethereum Non-Fungible Token project called CryptoKitties. Exploring the technical details behind the implementations for these projects as well as analyzing the incentives that drove these projects is useful towards understanding the incentives that drive adoption of blockchain projects. The second key contribution of this thesis is a review of several papers, journals, and articles that analyze the incentives driving the adoption of blockchain technology and providing an introductory informative overview of the various aspects of the field.

1.2 Thesis Outline

Blockchain technology is better understood by breaking it up into components and exploring how each component works. Chapter 2 provides a high-level technical overview of blockchain technology by examining the components that constitute blockchains one by one. The differences between permissioned and permissionless blockchains is explained. Popular blockchain consensus protocols like Proof of Work and Proof of Stake are discussed. Chapter 3 presents an implementation of a blockchain network based on the blockchain components presented in Chapter 2. Chapters 4 is a case study of Helium Blockchain network. Chapter 5 is a case study of an Ethereum Non-Fungible Token project called CryptoKitties. These case studies first examine how these projects leveraged blockchain technology from a tech-

nical standpoint, and then discuss the incentives built into these projects that have allowed millions of users to participate in these networks and create value. Chapter 6 explores the incentives that are built into blockchains. Game theoretic and economic concepts are applied to blockchain technology to model how dishonest actors could attack the network and explore how blockchains disincentivize dishonest actors. Finally, in chapter 7 reviews the findings of the thesis and propose avenues for future research.

Chapter 2

Blockchains Explained

This chapter is a high-level technical overview of blockchain technology. The first section provides a brief overview of the ideas and technologies invented, implemented, and adopted since the Bitcoin whitepaper first developed the blockchain concept. The second section looks at the different types of blockchains. The third section discusses the consensus protocols for the two most popular blockchains: Bitcoin and Ethereum. The final section goes into the details of how block and transaction conflicts are resolved in the blockchain. This chapter will give readers insight into the different aspects of blockchain technology before a custom implementation is discussed in the next chapter.

2.1 Components That Enable Blockchain Networks

2.1.1 Cryptographic Hash Functions

The cryptographic hash functions serve a multipurpose role in blockchain networks and are fundamental to make blockchains work. They are used for address derivation, creating unique keys, securing block data, and securing block header data.

Hashing is a method of applying a cryptographic hash function to data which computes a unique output for almost any kind of an input (ex: text, image, video, etc.). These cryptographic hash functions derive the same result only for the exact same input. Even the smallest change to the input (could be 1 bit) results in a completely different and unpredictable output hash value. Table 2.1 shows the *SHA256* hash outputs for different inputs.

Input	SHA256 Hash Output
1000	0x6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
1001	0xd4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35
"Hello, World!"	0xdffd6021bb2bd5b0af676290809ec3a53191dd81c7f70a4b28688a362182986f

Table 2.1: SHA256 hash outputs for different inputs

The outputs for 1000 and 1001 are completely different even though the inputs only differ by a 1 bit.

A cryptographic hash function has to have the following properties:

- Preimage Resistant: It must be computationally infeasible to compute the correct input value given an output hash value i.e. they are one-way [2].
- Second Preimage Resistant: It must be computationally infeasible to compute or find a second input to the function that produces the same hash output i.e given "x", it is infeasible to find "y" such that $\text{hash}(x) = \text{hash}(y)$ [2].
- Collision Resistance: No two inputs can have the same output hash value i.e given two different values "x" and "y", $\text{hash}(x)$ cannot be equal $\text{hash}(y)$ [2].

Bitcoin uses the SHA-256 hashing algorithm while Ethereum uses the Keccak-256 algorithm. The SHA-256 algorithm has an output size of 32 bytes i.e 256 (32 x 8) bits. So, the hash output of the algorithm has approx. 10^{77} (2^{256}) possible output hashes. Since there are infinite possible inputs and a fixed number of outputs to the SHA-256 algorithm, the possibility of collisions; i.e., two inputs having the same hash exists. On average, SHA-256 generates a collision every 5×10^{76} hashes. The hash rate of the entire Bitcoin network is 5×10^{18} hashes per second. At this rate, it still takes 10^{12} years to find a hash collision. Therefore, for all practical purposes SHA-256 is collision resistant [3].

A cryptographic nonce is a random number that is combined with block data while computing hashes i.e $output - hash = hash - function(data + nonce)$. Changing the nonce and observing the computed output hash for validity is a key part of the consensus protocol of several blockchain networks. Figure 2.1 presents a function that can be used to compute the SHA256 hash of any file.

```
○ ○ ○

import hashlib

sha256_hash_generator = hashlib.sha256()
filename = input("Enter the input file name: ")

with open(filename,"rb") as f:
    # Read and update hash string value in blocks of 1K
    for byte_block in iter(lambda: f.read(1024),b""):
        sha256_hash_generator.update(byte_block)
    print(sha256_hash.hexdigest())
```

Figure 2.1: Simple program to compute SHA256 Hash of a File

2.1.2 Transactions

A transaction represents an interaction between two nodes or users of a blockchain network. The transfer of cryptocurrency between two blockchain nodes represents a transaction. Physical (real-world) transactions could also be recorded in a transaction. Each block in a blockchain contains zero or more transactions.

The data that a transaction consists of is different for different blockchain implementations but the key elements in most cases are: sender's address, digital signature, transaction inputs, and outputs. The inputs are the digital property to be transferred and contain a signature from the sender so they can be authenticated. The outputs are the accounts that will receive the digital property and the amount to be received.

Regardless of how a transaction is submitted to a node, determining the validity and authenticity of the transactions is of utmost importance. That is, each transaction must support the protocol, smart contract, and data formatting mentioned in the blockchain specification. Each submitted transaction also needs to be signed by the sender's private key and be verified with their public key.

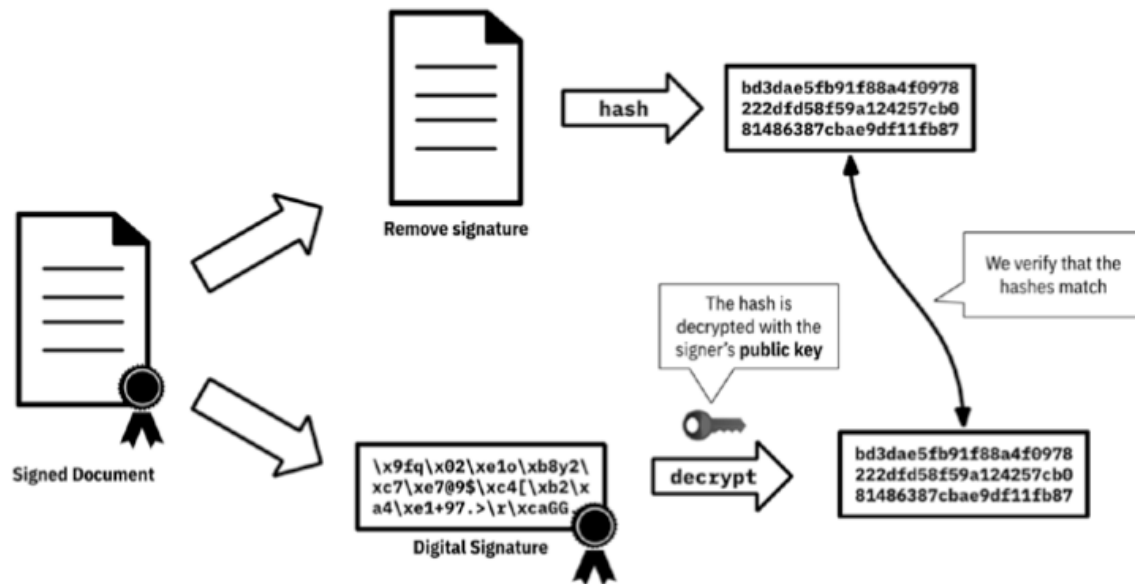


Figure 2.2: Verifying a transactions digital signature. Figure is from [3].

2.1.3 Asymmetric Key Cryptography

Asymmetric-key cryptography (AKC), also known as public-key cryptography, uses a pair of keys, a public key and a private key that are mathematically related. The public key is made public while the private key has to be kept secret to maintain protection. Although the keys are related, it is computationally infeasible to compute the private key from a public key. The private key can be used to encrypt and the private key can decrypt and vice versa [2].

AKC provides a way to verify the integrity and authenticity of transactions while allowing the transactions to be public. This is achieved by using the private key to encrypt and digitally sign a transaction such that anyone with the public key can decrypt it. Since the public key is available to everyone, encrypting the transaction with the private key proves that the transaction's digital signer has access to the private key [3].

Private keys are used to digitally sign transactions. Public keys are used to derive addresses. Public keys are used to verify signatures generated with private keys. AKC provides the ability to verify that the user transferring cryptocurrency to another user owns the private key that digitally signed the transaction. Figure 2.2 visualizes how digital signatures

are verified with asymmetric key cryptography.

2.1.4 Ledger

A ledger is a collection of transactions. Modern day accounting software is an example of a software ledger. These ledgers can be implemented in a distributed way by having distributed ownership as well as distributed architecture, which could be realized through blockchain technology. The major interest in distributed ledgers is due to trust, security, and reliability concerns associated with centralized ledgers.

A decentralized ledger automatically provides redundancy, since a copy of the ledger is stored by every node. The loss of one node or a significant number of nodes does not impact the working of the network. All transactions, rewards, and currency created are transparent to all nodes connected to the network. Each transaction published to the decentralized ledger is cryptographically secure. Due to the nature of the consensus protocols, misbehaving by publishing fake transactions or double spending currency is disincentivized.

2.1.5 Blocks and Chaining Blocks

Users of a blockchain network submit transactions to the blockchain network using software like wallets, crypto exchanges, smartphone applications, and web applications. The software connects to a DNS service that provides a list of blockchain nodes the transaction can be sent to. These nodes propagate the transaction to all the nodes in the network. The nodes which are competing to publish blocks will then include this transaction in a published

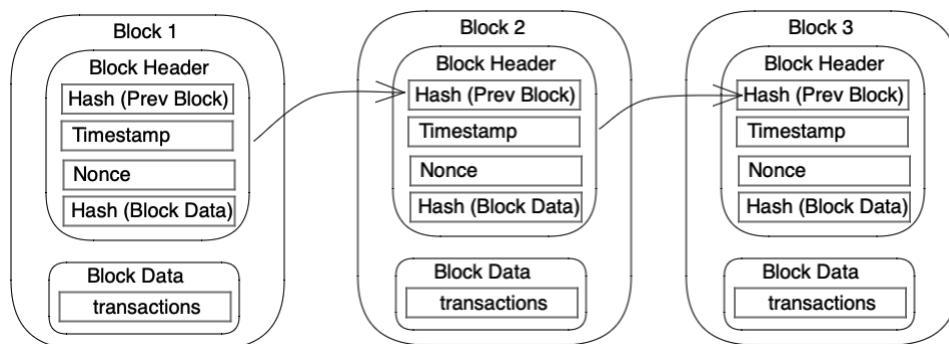


Figure 2.3: Chaining of blocks over time

block. Each block contains a block header and block data. The block header contains the metadata for the block and the block data contains the transactions included in the block.

Every implementation of a blockchain network has its own standard for what data field each block needs to have. The most common data fields in most blockchain implementations are discussed below.

Block Header:

- Block Number
- Previous block's hash value
- Hash of the current block
- Timestamp
- Nonce Value

Block Data:

- List of transactions (bitcoin) and other events (contracts for Ethereum) included within the block.

Blocks are chained together through each block containing the hash of the previous block's header forming the blockchain. If a previously published block were changed, it would have a different hash. This in turn would cause all subsequent blocks to also have different hashes since they include the hash of the previous block. This makes it possible to easily detect and reject altered blocks.

2.2 Types of Blockchains

Blockchains can be categorized based on the way they implement their permission model, which dictates who can or cannot maintain the blockchain. If anyone can create a node for a blockchain and publish a block, it is called a permissionless blockchain. If only particularly authorized nodes/ users can publish blocks, it is called a permissioned blockchain. Simply, a

permissioned blockchain is like a local area network controlled by some company/ entity while a permissionless blockchain is like the open internet, where anybody can participate. The difference between these two different kinds of blockchains is important because it impacts the decisions about blockchain components [2].

2.2.1 Permissionless

Permissionless blockchain networks are decentralized ledger platforms open to anyone publishing blocks, without needing any permission to setup and connect a node or application to the network (ex: Bitcoin). Permissionless blockchain networks are usually open source software, available on Github for anyone to download and run. Since the specifications and source code are publicly available, anyone who has a properly setup blockchain node can read the blockchain and publish transactions to the blockchain (read and write to the ledger).

Since permissionless blockchains are open to all, malicious nodes / users will attempt to publish blocks with fake transactions, double spent coins, and will utilize all kinds of techniques to subvert the system. To prevent this, permissionless blockchains use a consensus protocol (discussed in section 2.3) which acts as a conflict resolution system that allows all nodes on the blockchain network to agree over the true state of the blockchain. This consensus protocol requires nodes on the network to expend resources (proof of work) or collateralize currency (proof of stake) when attempting to publish blocks. The consensus protocols incentivize non-malicious behavior by rewarding protocol executing nodes with the blockchain's native cryptocurrency.

2.2.2 Permissioned

Permissioned blockchain networks only allow authorized nodes/ users to read and publish blocks. Permissioned blockchain networks also use consensus protocols to publish blocks, but they do not require nodes to expend or maintain resources since establishing the identity of a node is required for a node to participate in a permissioned network. The network nodes all trust each other since each node has to be authorized with read and write access and this authorization can be revoked if any of the nodes act against the interests of the network.

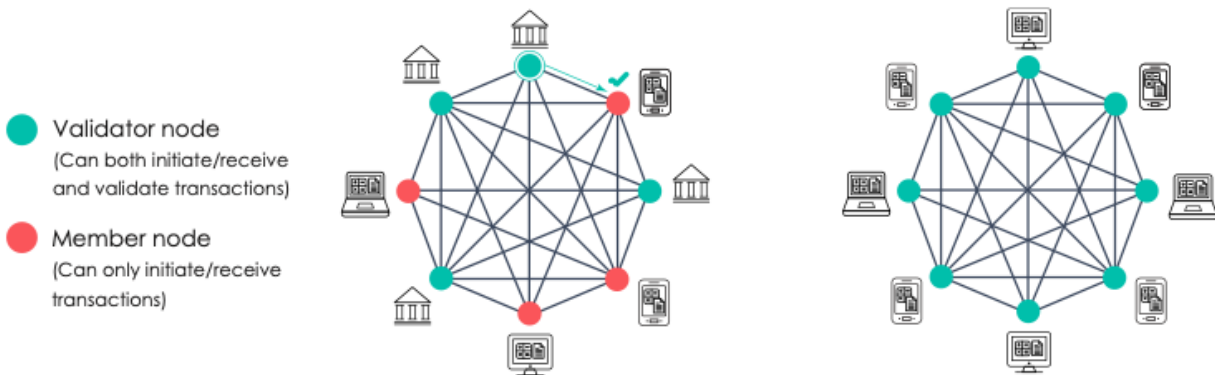


Figure 2.4: Permissioned and permissionless blockchain nodes visualized. Figure is from [3].

The level of transparency can be adjusted in permissioned blockchains. Transaction data in the ledger can be selectively revealed. For example, the blockchain that records a transaction between two parties can reveal that the transaction took place without disclosing the contents. Some permissioned blockchains require all users to be authorized to send and receive transactions (they are not anonymous). This idea is being explored by banks in the form of CBDC's or Central Bank Digital Currencies which is a concept that will likely gain traction.

2.3 Reaching Consensus

An important problem that a blockchain network has to solve is determining which node on the network adds the next block to the blockchain. In the case of permissionless networks, several nodes are competing to generate the hash and publish the next block to obtain the rewards (cryptocurrency award and transaction fees) associated with publishing the next block. All the nodes and users on the network only know each other through public key addresses. All of the nodes and users on the network are motivated by financial gain and naturally distrust all other nodes on the network. In this state of distrust, nodes are not necessarily motivated to allow other nodes to publish to the blockchain or allow transactions that do not benefit themselves. To make this work, blockchains use consensus protocols to enable a collection of mutually distrusting users to work together.

When a node or user joins a blockchain network, the initial state of the system and the

Leading Zeros	Hash Tries
0	1
1	4
2	61
3	6920
4	46,801
5	819,249
6	44,443,227
7	12,969,432,764

Table 2.2: Increasing hash difficulty by increasing the number of leading zeros the hash must have (Results for M1 ARM Processor)

consensus model are agreed upon. Every block is linked to the previous block header's hash. Regardless of the consensus protocol, every user and node on the network can independently check the validity of every block. These properties need to be satisfied by the consensus protocol. In reality, the node is implemented by open source software that handles all of these requirements and handles the connection to the network.

2.3.1 Proof of Work (Bitcoin)

Blockchain networks that implement the proof of work consensus protocols require the node publishing the next block of transactions to solve a computationally intensive puzzle. The solution to the puzzle is the proof that the work was done. The puzzles are designed such that solving the puzzle is difficult but validating the solution is easy.

The most common puzzle is requiring the hash of a block to be lower than a certain target value. Publishing nodes constantly make changes to the block headers like updating the value of a nonce and recomputing the hash until the constraints on the hash are met. Generating this hash is computationally intensive. The constraints and hash target value can adjust difficulty to increase or decrease how often blocks are published to the blockchain.

Bitcoin uses the proof of work consensus protocol. The Bitcoin network adjusts mining difficulty every 2016 blocks to moderate the rate of block publication to approximately 10 minutes per block. The difficulty is adjusted by increasing or decreasing the number of leading zeros that the hash computed by the nodes is required to have. Increasing the number of leading zeros increases difficulty and decreasing the number of leading zeros decreases

```
from timeit import default_timer as timer
import random
import time

def block_miner(text, digits):
    import hashlib
    n = 1
    ntext = text
    while hashlib.sha256(ntext.encode('utf-8')).hexdigest()[0:digits] != "0"*digits:
        n+=1
        ntext = f'{text} {n}'

    return(text,n , hashlib.sha256(ntext.encode("utf-8")).hexdigest())

ZEROS_REQUIRED = 6
for i in range(0, ZEROS_REQUIRED + 1):
    start = time.perf_counter()
    name, iters, hash = block_miner("Hash try #", i)
    end = time.perf_counter()

    print(f'{name} {iters:10,} in {end-start:7.4f} seconds => {hash}')
```

Figure 2.5: Simple program to check hashtimes with increasing number of leading zeros

complexity. Since the computational power of the network keeps increasing, the puzzle difficulty also keeps increasing.

An important idea for the consensus protocol is that the amount of work put into solving a puzzle does not increase or decrease the likelihood of solving the next puzzle i.e. each new puzzle is independent of previous puzzles. This means that when a node receives a completed valid block from another node on the network, it no longer makes sense to continue working on the current hash. The nodes are incentivized to build off of the newly published block instead because other nodes will be building off that block.

2.3.2 Proof of Stake (Ethereum)

The proof of stake consensus protocol is based on the idea that the more stake a node or user has invested in the blockchain network, the more likely it is that they will want the project to succeed. Conceptually, a stake is an amount of native cryptocurrency that a user has invested into the system. This stake could be locked by a special transaction type or held in designated staking wallets. Once staked, the cryptocurrency is no longer available to be spent. The more that a node has staked, the more likely it is that the node will publish the next block. With this consensus protocol, computationally intensive operations do not

need to be performed. Instead, publishing nodes are rewarded with transaction fees for the transactions they include in the published block. There are several ways in which blockchain networks use stakes. Two of the common approaches are random selection of staked users and multiparty voting.

When the random selection of staked users approach is chosen, the blockchain network will look at all the nodes with stakes and choose among them based on the ratio of the node's stake to the overall cryptocurrency staked. If a node has 20 percent of the entire blockchain network's stake, that node's block is published 20 percent of the time and with 1 percent stake the block is published 1 percent of the time. Capping the maximum stake a node can put up incentivizes a lot of nodes to participate in the consensus protocol and adds to the randomness of choosing the published block.

When the multi round voting approach is chosen [2] the staking process is more complex. The blockchain network will select several of the blocks created by staked nodes to create proposed blocks. Then, several rounds of voting occur where all the staked nodes in the network cast votes to decide which proposed block is published. This approach allows all staked users to play a role in securing the blockchain and arriving at a consensus.

Proof of stake systems allows "rich" nodes to easily stake more cryptocurrency than "poor" nodes resulting in the "rich" nodes earning more of cryptocurrency through transaction fees. This can cause accumulation of cryptocurrency with staking "whales". This can be avoided by maintaining an upper limit on the amount of cryptocurrency any node can stake. Also, because of the competitive nature of the consensus protocol, any node trying to "control" the network will be incredibly cost prohibitive.

2.3.3 Conflicts and Resolution

Because of the size of the blockchain networks, it is possible that multiple blocks are published in different parts of the network at the exact same time. Network latency between nodes, physical distance between node clusters, and the number of competing publishing nodes all are factors that contribute to blockchain conflicts. These factors cause several versions of the blockchain network to exist at any given point of time. These conflicts have

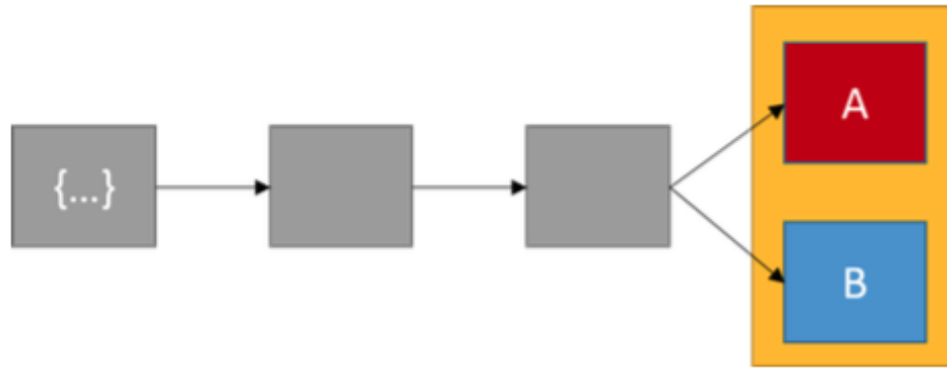


Figure 2.6: Blockchain conflict visualized where miners adopt different blocks. Figure is from [2].

to be resolved and the network has to agree on the true published blocks for the network to remain consistent.

An example to understand how conflicts occur is as follows:

Node A creates a Block A with transactions 1,2, and 3 and distributes the block to nodes on the network. In another part of the network block B creates a block with transactions 1,2,4, and 5 and distributes the block to nodes on the network. This causes a conflict: Block A has transaction 3 but not transactions 4 and 5. Block B has transactions 4 and 5 but not transaction 3.

The conflict created 2 versions of the blockchain. Nodes that receive 1 of these blocks will continue mining transactions based on the block they currently have which could cause double spending of cryptocurrency and other problems.

The conflicts are resolved quickly by waiting for the next block to be published and adopting the longer blockchain to be correct. The transactions that were in the orphaned

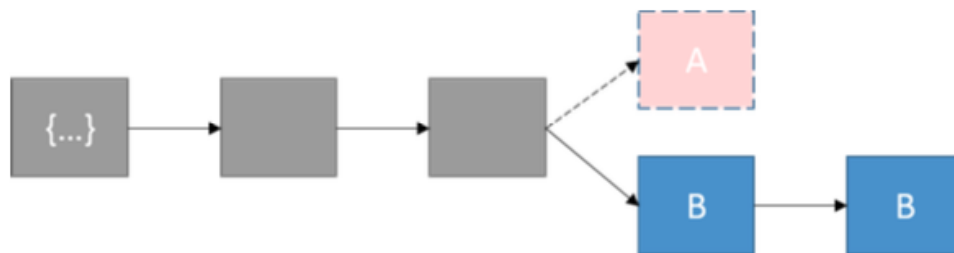


Figure 2.7: When the next block is add to block B, the chain containing block A is orphaned. Figure is from [2].

block and not in the adopted block are added back into the pending transactions pool. Because this possibility of blocks being overwritten exists, a transaction on the blockchain is not accepted until several additional blocks are created on top of the block having the transaction. The more blocks that are added on top of a block, the more likely it is that the block will not be orphaned and overwritten.

2.4 Chapter Summary

In this chapter, the two different kinds of blockchains were discussed. A breakdown of all the major components that go into building a blockchain presented along with a brief discussion of each component. The consensus protocols which are the heart of decentralized ledgers were discussed. The Proof of Work (POW) protocol implemented by Bitcoin and the Proof of Stake (POS) protocol implemented by ethereum were discussed. The conflict resolution mechanism that solves for the true blockchain out of the different versions of a blockchain present at any given time in the blockchain network is discussed at a rudimentary level. This chapter prepares readers to understand the design choices made in the next chapter where a blockchain is implemented from scratch.

Chapter 3

Blockchain Implementation

3.1 Blockchain Node Components

The blockchain implementation's architecture is shown in Figure 3.1. Each blockchain node is a HTTP server. All of the blockchain nodes are connected to each other through peer-to-peer websocket connections. Not included in the architecture diagram is the client websocket server. The client websocket server establishes the peer-to-peer connections when a new node is added to the blockchain network by connecting the new node to all existing nodes and providing the new node with a copy of the blockchain.

The implementation of each blockchain node is the same. The blockchain nodes and all component classes are implemented with *Python3* as the programming language. *Flask* library is used to create the HTTP server and the *socketio* library is used to create the socket server and send peer-to-peer messages. Blockchain nodes consist of a blockchain, wallet, transaction pool, and miner. The logic handling all incoming and outgoing connections (peer-to-peer messages and HTTP requests) are part of the blockchain node. The nodes use separate ports for peer-to-peer and HTTP requests. When a node is initialized, the HTTP server starts listening on a port in the range of 5000 to 5999. To connect to all peer nodes and get the current copy of the blockchain, the node has to connect to the socket server running on port 6000. The socket server creates the peer-to-peer connections and provides the newly connected node with the latest version of the blockchain. This section gives the high-level implementation details for each component in the node. The classes and functions provided

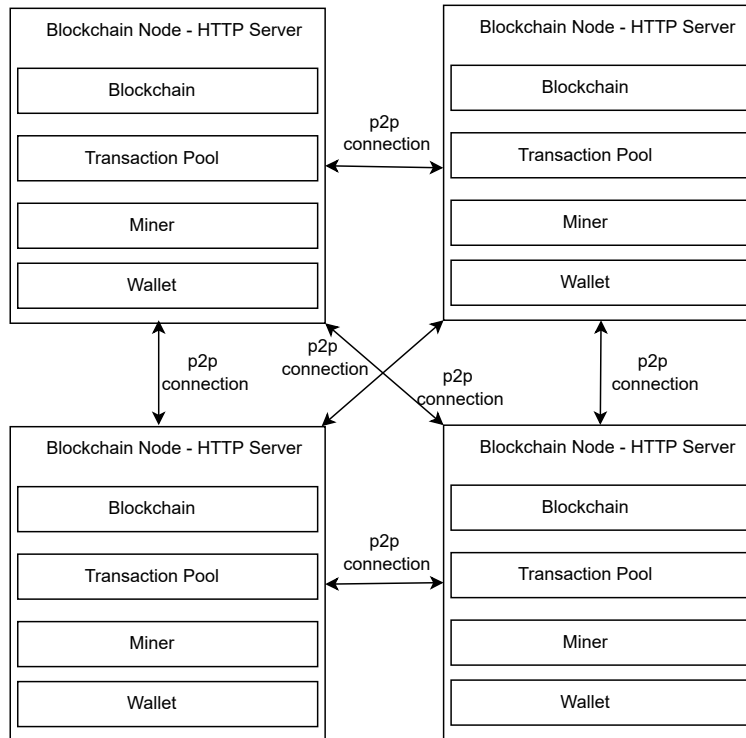


Figure 3.1: Blockchain Architecture.

in the Figures are partial parts of the code. Each of these classes has several methods and wrappers not shown in the Figure. The full implementation for each class can be found on the Github Repo¹.

3.1.1 Blocks and Blockchain

```
class Block:
    def __init__(self, index, timestamp, previous_hash, transactions, nonce, hash=None, mined_by=None):
        self.index = index
        self.timestamp = timestamp
        self.previous_hash = previous_hash
        self.transactions = transactions
        self.nonce = nonce
        self.hash = hash
        self.mined_by = mined_by
```

Figure 3.2: Block Class Implementation.

¹<https://github.com/rahulreddy15/thesis-repo>

Fig 3.2 shows the *Block* class and its initialization function. Each Block contains *index*, *timestamp*, *previous_hash*, *transactions*, *nonce*, *hash*, and *mined_by* as data fields.

```
class Blockchain:
    DIFFICULTY = 5

    def __init__(self, chain=[genesis_block]):
        self.chain = chain
        self.length = len(self.chain)

    def get_last_block(self):
        return self.chain[-1]
```

Figure 3.3: Blockchain Class Implementation.

```
def genesis_block():
    genesis_block = Block(
        index=0,
        timestamp=time.time_ns(),
        previous_hash='0',
        transactions=[],
        nonce=0)
    genesis_block.hash = Block.hash_block(genesis_block)
    return genesis_block
```

Figure 3.4: Genesis Block Creation Code.

Fig 3.3 shows the *Blockchain* class and its initialization function. If the blockchain provided by the socket server is empty, then a new genesis block has to be created. The function that creates the genesis block is shown in Figure 3.4. The blockchain has a variable called *chain* which stores all the Blocks added to the blockchain in sorted order of index.

3.1.2 Peer-To-Peer Messages

Figure 3.5 shows the *Message* class and its initialization function. The message class has the following 3 hard-coded types:

- *ADD_TRANSACTION* - Message type used to send a transaction added by a node to all other nodes on the network to add into their respective transaction pools.
- *START_MINING* - Message type used to ask all nodes on the network to start mining for the next block.
- *BLOCK_FOUNDED* - Message type used to send a mined block to all other nodes on the network to add into their local blockchain copy and stop mining.

Figure 3.6 shows the function used to send the message to all peer nodes.

```
class Message:

    ADD_TRANSACTION = 'add_transaction'
    START_MINING = 'start_mining'
    BLOCK_FOUNDED = 'block_founded'

    def __init__(self, messageType, data):
        self.message = messageType
        self.data = data

    def getMessage(self):
        return json.dumps({'message': self.message, 'data': self.data})
```

Figure 3.5: Message Class Implementation.

```
def send_message(message):
    connected_sockets = session.getConnectedNodes()
    for socket in connected_sockets:
        io.emit(socket, message.getMessage())
```

Figure 3.6: Sending Message to Peer Nodes.

3.1.3 Transactions and Transaction Pool

Figure 3.7 shows the *Transaction* class its initialization function. Whenever a node adds a transaction. The transaction is created as an instance of the *Transaction* class. The transac-

tion is then sent to all other connected nodes on the network with the *ADD_TRANSACTION* message type. Every node that receives this message, adds the transaction into their *Transaction Pool*. Figure 3.8 shows the *Transaction Pool* class and its initialization function. The *transactionPool* data field stores all unmined transactions in an array.

```
class Transaction:
    STATUS_PENDING = 'pending'
    STATUS_SUCCESS = 'success'
    STATUS_FAILED = 'failed'

    def __init__(self, sender, sender_name, receiver, receiver_name, amount, reward=False, status=STATUS_PENDING):
        self.sender = sender
        self.sender_name = sender_name
        self.receiver = receiver
        self.receiver_name = receiver_name
        self.amount = amount
        self.reward = reward
        self.status = status
```

Figure 3.7: Transaction Class Implementation.

```
class TransactionPool:
    def __init__(self):
        self.transactionPool = []

    def addTransactionToPool(self, transaction):
        self.transactionPool.append(self.validateTransaction(transaction))
```

Figure 3.8: Transaction Pool Class Implementation.

3.1.4 Mining and Rewards

This blockchain implements Proof of Work as the consensus protocol. This implementation requires the mining process to be triggered by one of the nodes on the network. When a node triggers the mining process, all connected nodes receive the *START_MINING* message. Once this message is received, all the nodes on the network start competing to find the hash for the new block that is going to be published to the blockchain. Each node takes all the transactions in their respective *Transaction Pool*, verifies them by querying the blockchain

```
def mine_block(self, transactions, mined_by):
    block_json = {
        'index': len(self.chain),
        'timestamp': time.time_ns(),
        'previous_hash': self.get_last_block().hash,
        'transactions': transactions,
        'nonce': 0,
        'hash': None,
        'mined_by': mined_by
    }
    block = Block.from_json(block_json)\

    while Block.hash_block(block)[:Blockchain.DIFFICULTY] != '0' * Blockchain.DIFFICULTY:
        block.nonce = block.nonce + 1

    block.hash = Block.hash_block(block)
    return block
```

Figure 3.9: Function that mines for the next block.

```
def hash_block(block):
    block_json = block.to_json()
    if 'hash' in block_json:
        del block_json['hash']
    encoded_block = json.dumps(block_json, sort_keys=True).encode()
    return hashlib.sha256(encoded_block).hexdigest()
}
```

Figure 3.10: Block Hash Computation Implementation.

and uses all valid transactions to create a new *Block*. The function used to mine for the next block is shown in Figure 3.9. The function used to compute the hash is shown in Figure 3.10. The hash is computed by constantly incrementing the *nonce* until the difficulty criteria is met. The *nonce* is initialized with a random value generator (this makes sure that there is a winning node). Once a node finishes mining, the newly created block is sent to all connected nodes with the *BLOCK_FOUND* message type. All connected nodes that receive the block interrupt their mining process and add the newly received block to their blockchain. This works because this implementation assumes that all nodes participating on the network are honest. Once the new block is published to the blockchain, the node which computed the hash is awarded 100 coins. This *Transaction* is sent to all nodes to include in their *Transaction Pool*. Nodes receive their reward when the next block contain the reward transaction is mined.

3.2 Blockchain Explorer and Wallet User Interface

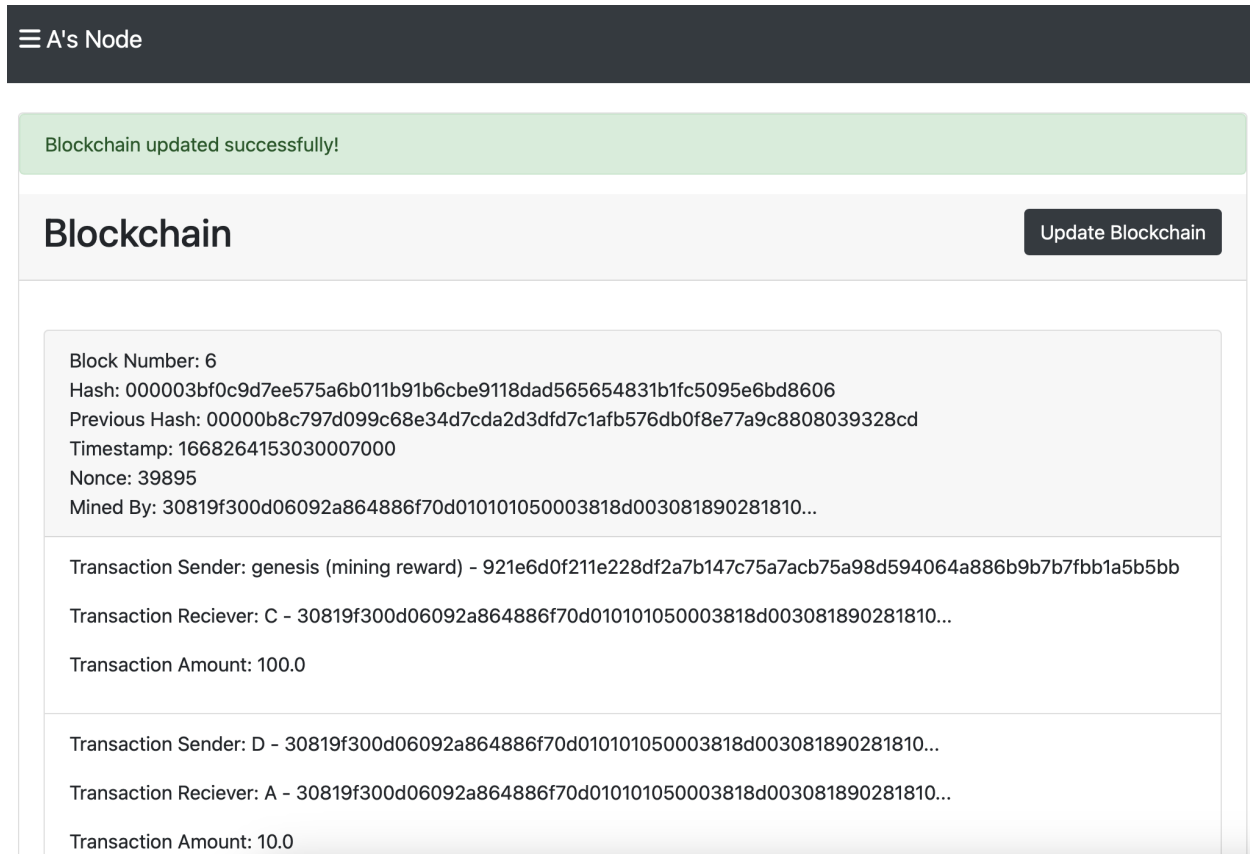


Figure 3.11: Blockchain Explorer User Interface.

3.2.1 Blockchain Explorer

The blockchain explorer is the home page of the user interface. It can be accessed by clicking on the *Dashboard* tab in the navigation menu. The blockchain explorer displays all the blocks of the blockchain arranged by reverse order of block number. When new blocks are mined, clicking the *Update Blockchain* button fetches the latest version of the blockchain from the node. The *Update Blockchain* button had to be implemented because the user interface serves static HTML documents. Fig 3.11 shows a screenshot of the blockchain explorer page.

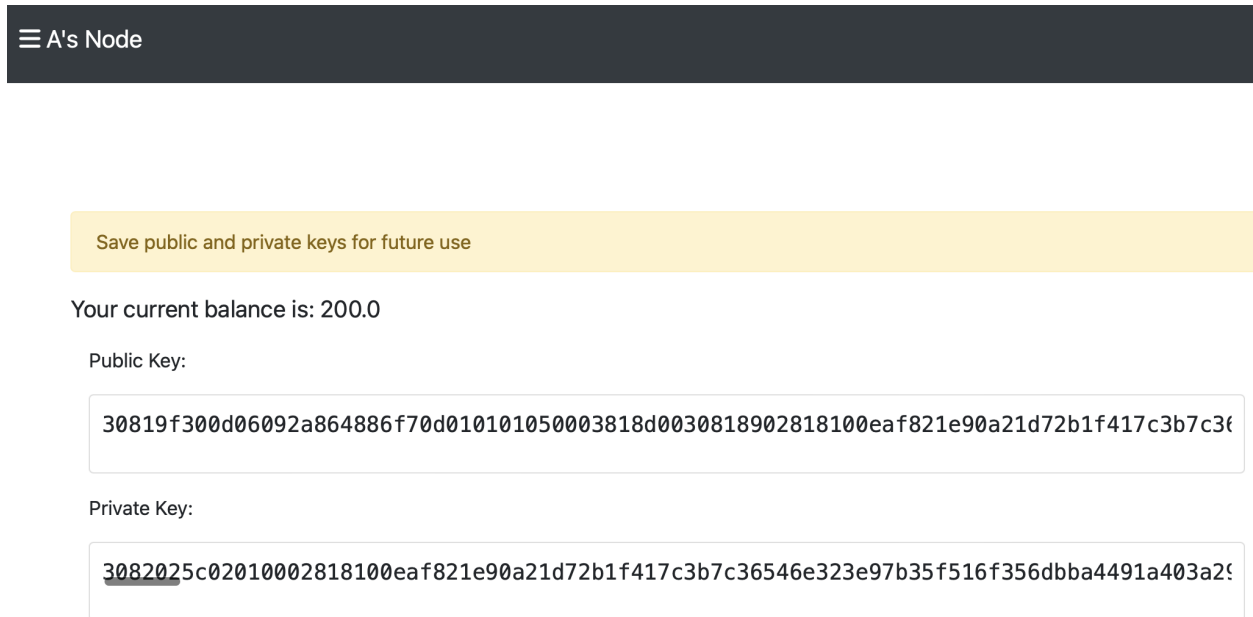


Figure 3.12: Wallet User Interface

3.2.2 Wallet

The wallet can be accessed by clicking the *My Wallet* tab in the navigation menu. The wallet page contains the node's public key, private key, and wallet balance. Since the wallet page is served as static HTML, refreshing the page once a block is mined will reflect the updated wallet balance. Figure 3.12 shows a screenshot of the *My Wallet* page.

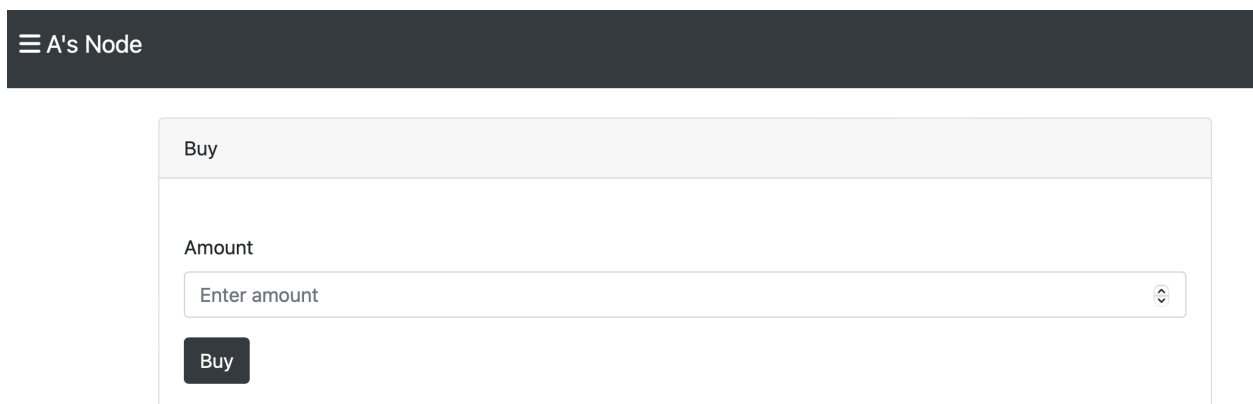
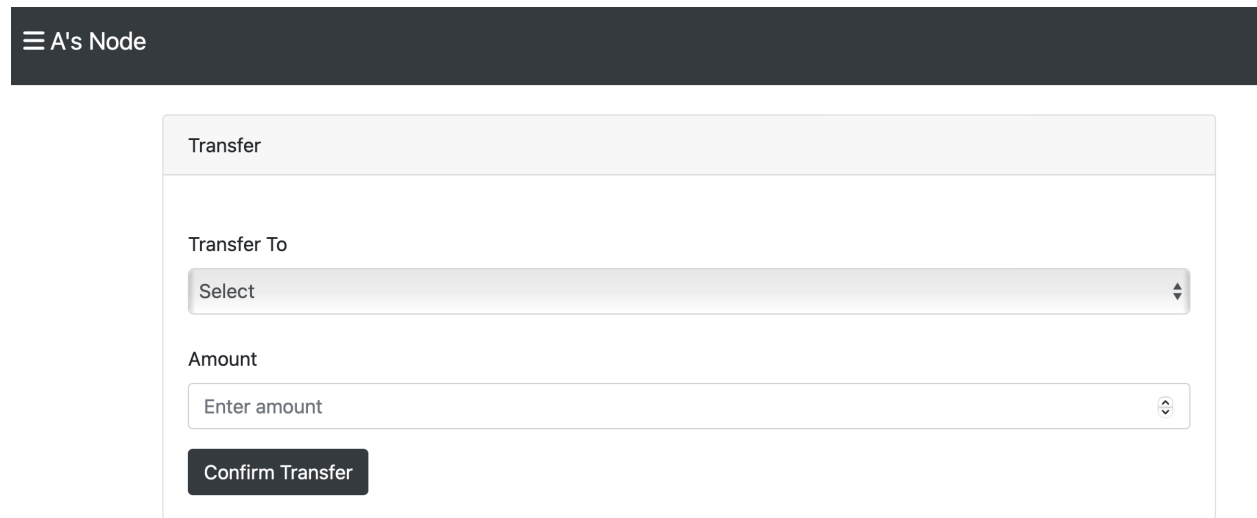


Figure 3.13: Request coins from Genesis Block User Interface.

3.2.3 Request Coins from Genesis Block

The buy page can be accessed by clicking the *Buy* tab in the navigation menu. The buy page allows nodes to request coins from the genesis block. When a node requests x amount of coins, a transaction is added from the genesis block to the node for x coins. Coin request transactions are included into published blocks by the blockchain network 50% of the time. Figure 3.13 shows a screenshot of the *Buy* page.



Transfer

Transfer To

Select

Amount

Enter amount

Confirm Transfer

Transaction History

Sender/Receiver	Amount	Status
genesis (mining reward) - 921e6d0f211e228df2a7b1...	100.0	success

Figure 3.14: Create Transactions User Interface.

3.2.4 Transfer Coins

The transfer functionality can be accessed by clicking the *Transfer* tab in the navigation menu. The first section of the transfer page allows nodes to transfer coins to other connected nodes. The *Transfer To* dropdown gives a list of connected nodes. The *Amount* input field is for entering the transfer amount. Once *Confirm Transfer* is clicked, the transaction is

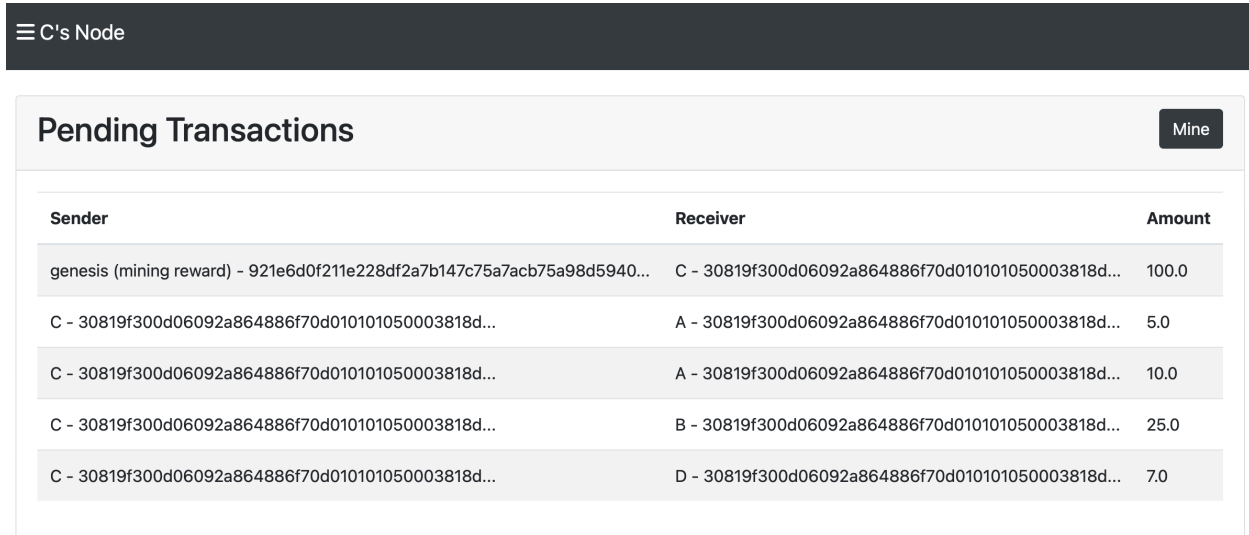


Figure 3.15: Transaction Pool and Mine Transactions User Interface.

sent to all nodes on the network to add into their respective transaction pools. The second section called *Transaction History* shows all past and pending transactions submitted to and from this node's wallet address. Figure 3.14 shows a screenshot of the *Transfer* page.

3.2.5 Mine Transactions and Transaction Pool

The mining functionality can be accessed by clicking the *Mine* tab in the navigation menu. The *Pending Transactions* section shows all of the transactions currently in the node's transaction pool. The *Mine* button sends a message to all connected nodes to start mining for a hash and publish the next block of transactions. Figure 3.15 shows a screenshot of the *Mine* page.

3.3 Summary

The high-level implementation details of the blockchain are presented. Screenshots for each page of the blockchain nodes user interface are presented. The full implementation of each component is available in the Github Repo².

²<https://github.com/rahulreddy15/thesis-repo>

Chapter 4

Helium Blockchain Case Study

The Helium network, also called “The People’s Network” is a decentralized wireless network that allows internet of things devices (IoT), to wirelessly connect to the internet and geolocate. The Helium network is a collection of hotspots hosted by individuals around the world. It provides an alternative to using satellite or cellular technology for IoT devices by providing a low power, low cost, secure peer-to-peer wireless network run on open standards. It allows IoT devices to geolocate and send and receive data to web servers. The Helium network is powered by a novel consensus protocol called *Proof of Coverage (PoC)*. Currently, over 930,000 hotspots are providing coverage, with an additional 11,000 being added per month [4].

4.1 Principles Behind the Helium Network

4.1.1 LoRa and LoRaWAN

LoRa which is an abbreviation for Long Range is a radio frequency transmission technology for low power, wide area networks. Key features of LoRa are:

- LoRa enables long-range data links by encoding information on radio waves using chirp pulses (similar to the way dolphins and bats communicate).
- LoRa radios have a coverage radius up to 3 miles radius in urban areas and 10 miles radius in rural areas.

- LoRa chips require very little power and can last upto 10 years on battery-operated devices.

LoRaWAN which is an abbreviation for Long Range Wide Area Network is a Media Access Control (MAC) layer protocol built on top of LoRa transmission. LoRaWAN is a layer on top of LoRa that allows IoT devices to authenticate and communicate with the internet. When IoT devices connect to an internet gateway, LoRaWAN handles how and where the data packets are forwarded [5][6].

4.1.2 Helium Blockchain

The Helium blockchain is a blockchain built to incentivize the creation of decentralized, public wireless networks. The Helium blockchain is based on the novel Proof of Coverage Algorithm. The blockchain records all transactions that occurs on the Helium network and provides Helium Tokens (HNT) as rewards to all network participants based on the work they do to create and secure the network. Each block records the most recent transactions to the blockchain. Currently, a block is mined every 60 seconds. The Helium blockchain currently supports 36 different kinds of transactions. These include:

- Add Gateway: Adding a new hotspot to the network
- Assert Location: Find the location of a hotspot on the network.
- Proof of Coverage Reciepts: The result of a PoC submitted to the blockchain once a a challenge is complete.
- Rewards: At the end of each epoch (30 blocks), HNT rewards are distributed to all participants based on the reward allocation table.
- 32 other important transaction types

4.1.3 LongFi

Helium created *LongFi* technology. LongFi is a combination of LoRaWAN and the Helium blockchain. The Helium LongFi benefits from all the advantages of LoRaWAN

networks while adding the capacity for gateway owners to earn HNT for providing coverage and securing the network. Helium LongFi supports micro-transactions so customers only pay based on their network usage. The Helium LongFi technology is embedded onto a LoRaWAN gateway to make a Hotspot.

4.2 Components of the Helium Network

4.2.1 Hotspots

Hotspots are the nodes of the Helium network and make up the majority of deployed Helium infrastructure. Hotspots are physical boxes with a packet forwarder and a miner.

- Packet Forwarders are LoRa wireless modules in Hotspots which provide network coverage. They relay radio packets from devices to and from the miner in the Hotspot. They are generally made up of a LoRa concentrator, which is a radio transceiver capable of operating on multiple bands and a supporting processor which runs the packet forwarder.
- Miners transmit data packet payloads between devices and Helium routers, Hotspots also maintain the Helium blockchain. A Miner is usually a small embedded Linux device. Miners receive Helium tokens (HNT) as rewards for data transmission, network coverage validation, and blockchain consensus activities. They send traffic to Helium routers on the internet at large using TCP/IP backhauls like Cable and LTE.

4.2.2 Routers

Helium Routers are cloud servers that contain the entire Helium blockchain and manage the LoRaWAN protocol between devices their corresponding application servers. They are responsible for authenticating and receiving messages from devices. Hotspots find Routers by looking up device owners from packet metadata and a list of Routers that is provided to each Hotspot. Routers add data transfer transactions to the blockchain. Figure 4.1 provides an visual representation of how the different components work on the Helium Network.

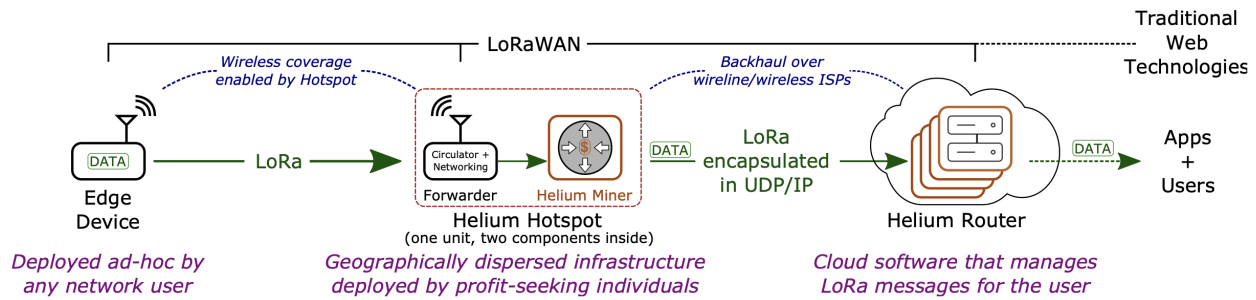


Figure 4.1: Helium Overview. Figure is from [5].

4.2.3 Validators

Validators are a newer component introduced in Helium Improvement Proposal 25. They were added to the Helium blockchain on July 7, 2021. Validators are cloud based nodes responsible for processing blocks and managing the blockchain consensus group. They are also responsible for constructing proof of coverage challenges. Before validators were introduced, miners were responsible for these tasks. To operate as a validator, a Miner must stake 10,000 HNT.

4.2.4 Helium Console

The console is a cloud service provided by helium which acts as a helium router. Instead of buying an Device ID (OUI) and running their own router, users deploying end devices can opt to use the Console. Very few self deployed Routers exist on the Helium Network. Most routing is done through the Helium console.

4.2.5 Devices and Device Owners

Devices are the LoRa-enabled IoT wireless devices. Devices are made with a Device End User Identifier (EUI), an Application EUI, and an App key. These are used during Over the Air Activation (OTAA), part of LoRaWAN, to authenticate to a LoRaWAN Router. Helium differs from LoRaWAN in that EUIs are used to look up Helium Organizationally Unique Identifiers (OUIs). These determine which router to authenticate the devices to. Device owners are the individuals who own and maintain Helium hotspots. In contrast to users or

customers, who send and receive data using the network. It is possible for one to be both a user and a device owner [5].

4.3 Proof Of Coverage

Helium is a physical wireless network that fails or succeeds based on how reliably it can create coverage for users deploying devices that connect to the network. Proof of Coverage (PoC) is Helium's way of verifying the coverage area and dimensions of its network. It is used to test if hotspots are actually providing the coverage they claim to be providing. PoC works by having validators issue challenges to hotspots asking them to prove their coverage. The hotspot being challenged has to solve the challenge and broadcast the solution. Nearby hotspots verify the solution and provide confirmation to the validator that the challenged hotspot is either providing or not providing the claimed coverage. Finally, the entire PoC transaction is written to the blockchain by the validator. PoC relies on the following characteristics:

- LoRa radio frequency has limited coverage distance.
- The strength of the received signal is inversely proportional to the square of the distance from the transmitter.
- Radio frequency signals effectively travel at the speed of light and have little to no latency.

These three facts are used to create a PoC challenge. The PoC challenge involves the following three participants:

- Challenger - The validator that constructs and issues the PoC challenge.
- Beaconer - The target of the POC challenge. Responsible for transmitting challenge packets to be witnessed by nearby hotspots.
- Witnesses - Hotspots that are close to the Beaconer and report receiving challenge packets.

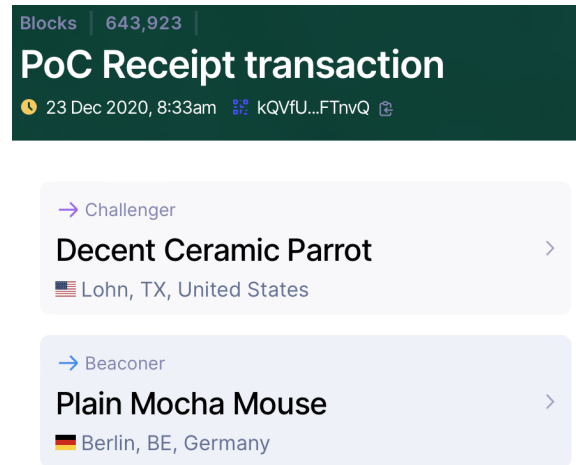
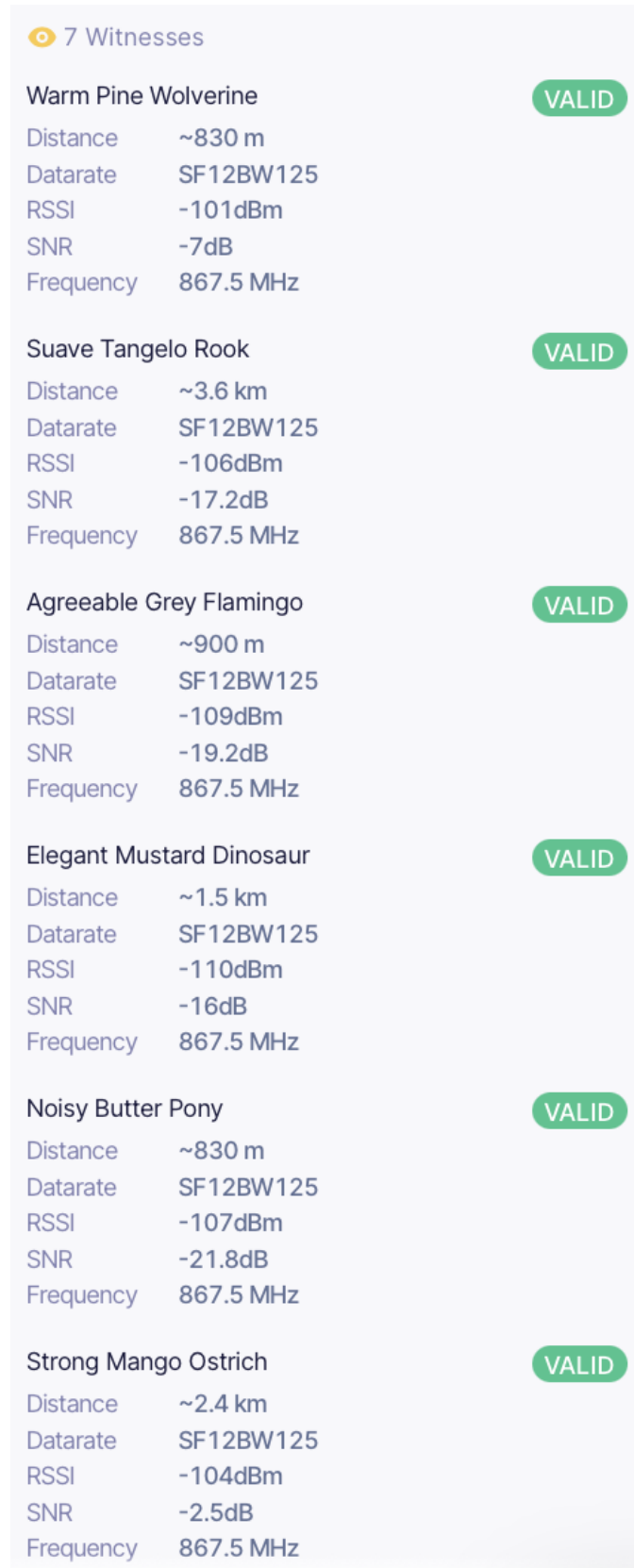


Figure 4.2: Challenger and Beaconer. Figure is from [4].

Figure 4.2 shows the receipt of a PoC challenge. In the image “Decent Ceramic Parrot” is the Challenger asking the Beaconer “Plain Mocha House” to provide proof that the Beaconer is providing network coverage for the range and location the Beaconer has claimed to provide coverage for. The challenge packets once received by the Beaconer are hashed. The hash is then attached with details of the challenge and transmitted by the Beaconer. Figure 4.3 provides a list of Hotspots that received packets from the Beaconer and act as Witnesses in this process. Based on the packets received, the Witnesses validate the location and coverage provided by the Beaconer. This validation be it positive or negative is sent back to the challenger. If more than 4 Witnesses validated the Beaconer’s coverage, the PoC challenge is ended and the receipt for the PoC transaction is added to the Helium blockchain. Figure 4.3 is a screenshot from Helium’s explorer application showing the relative positions of the Beaconers and Witnesses. The frequency at which challenges are issued are decided by the blockchain variable “poc-challenge-rate”. Currently, hotspots are challenged every 500 to 1000 blocks.



7 Witnesses

Warm Pine Wolverine	VALID
Distance	~830 m
Datarate	SF12BW125
RSSI	-101dBm
SNR	-7dB
Frequency	867.5 MHz
Suave Tangelo Rook	VALID
Distance	~3.6 km
Datarate	SF12BW125
RSSI	-106dBm
SNR	-17.2dB
Frequency	867.5 MHz
Agreeable Grey Flamingo	VALID
Distance	~900 m
Datarate	SF12BW125
RSSI	-109dBm
SNR	-19.2dB
Frequency	867.5 MHz
Elegant Mustard Dinosaur	VALID
Distance	~1.5 km
Datarate	SF12BW125
RSSI	-110dBm
SNR	-16dB
Frequency	867.5 MHz
Noisy Butter Pony	VALID
Distance	~830 m
Datarate	SF12BW125
RSSI	-107dBm
SNR	-21.8dB
Frequency	867.5 MHz
Strong Mango Ostrich	VALID
Distance	~2.4 km
Datarate	SF12BW125
RSSI	-104dBm
SNR	-2.5dB
Frequency	867.5 MHz

Figure 4.3: Hotspots that witnessed the challenge and received packets from the beacon. Figure is from [4]



Figure 4.4: Helium Explorer representation of PoC challenge. The large white dot represents the Beaconer. Each of the yellow dots represents a witness. The background is a hexagonal grid depicting the number of hotspots in each hexagon. Figure is from [4].

4.3.1 Consensus Protocol

The Helium blockchain uses a new consensus protocol called the Helium Consensus Protocol. The protocol has the following characteristics:

- **Permissionless:** Any hotspot working according to the consensus rules and network specification can participate in the Helium network.
- **Byzantine Fault Tolerant:** The protocol should be tolerant of Byzantine failures (nodes acting maliciously) such that consensus can still be reached as long as a threshold of nodes are participating honestly. For this reason, the HoneyBadgerBFT consensus protocol was chosen.
- **Based on useful work:** Achieving network consensus should be useful and reusable to the network. In Nakamoto Consensus-based systems like the Bitcoin blockchain, work performed to achieve consensus is only valid for a specific block. By comparison, Helium’s consensus system performs work that is both useful and reusable to the network beyond simply securing the blockchain.
- **High transaction rate:** The protocol should achieve a high number of transactions per second, and once the transaction is seen by the blockchain it should be assumed confirmed. Users sending device data through the Helium Network cannot tolerate long block settlement times typical of other blockchains.

- Censorship resistant transactions: Hotspots should not be able to *censor* (pick and choose which transactions are excluded) the transactions to be included in a block.

4.3.2 Honeybadger BFT

Helium Consensus Protocol is based on a variant of the Honeybadger BFT (BFT is an abbreviation for Byzantine Fault Tolerant) protocol invented by Andrew Miller at the University of Illinois, Urbana-Champaign [7][8].

“Honeybadger BFT is an asynchronous atomic broadcast protocol designed to enable a group of known nodes to achieve consensus over unreliable links” [7]. Honeybadger BFT relies on a scheme known as threshold encryption. Using this scheme, transactions are encrypted using a shared public key, and are only decryptable when the elected consensus group works together to decrypt them [7]. The usage of threshold encryption enables the Helium Consensus Protocol achieve censorship-resistant transactions. Helium implements this by having a consensus group of elected Validators. Each Validator receives encrypted transactions as inputs and proceeds to reach common agreement on the ordering of these transactions before forming a block and adding it to the blockchain. A complete discussion of how the HoneyBadger BFT consensus protocol was created and implemented is beyond the scope of this thesis.

4.3.3 Proof Of Location

Location tracking is one of the most important use case for IoT devices. Currently, GPS is used to triangulate the location of a sensor. This is not a good solution because GPS receivers require high power and can be spoofed. Proof of location is an algorithm proposed in the Helium whitepaper [7] that would allow location triangulation of devices with the Helium network. This algorithm is still in the Helium Improvement Proposal (HIP) phase where it will be discussed further and then moved into the public comments phase where stakeholders can offer suggestions to improve the specification before it is finally implemented.

4.4 Helium Token Economics and Rewards

4.4.1 Helium Token (HNT)

The Helium Token (HNT) is the native cryptocurrency of the Helium blockchain. Hotspots mine HNT while deploying and maintaining network coverage. The Validators, Beacons, and Witnesses participating in PoC earn HNT. Hotspots earn HNT by forwarding packets from IoT devices that are connected to the network. HNT rewards for all stakeholders in the Helium ecosystem are settled at the end of each epoch.

To make sure the supply of HNT does not become inflationary, the max supply of Helium Tokens is capped at 5,000,000 per month. This max supply is halved every 2 years as declared in Helium Improvement Proposal 20 [9] (The current supply of HNT is 2,500,000 HNT per month).

The value of HNT increases or decreases based on the success of the Helium network. HNT can be traded or bought on exchanges like Binance and Coinbase. The value of HNT varied between 9 dollars and 27 cents and 18 dollars and 44 cents over the last year (November 2021 to November 2022). Since HNT has a variable price and is open to speculation, it is not useful for customers of the Helium network to purchase network capacity in the form of HNT because the value of the token could half or double overnight. Variations in the price of HNT will significantly affect network costs for customers. Customers planning IoT deployments over the Helium network need a stable pricing structure. For this reason, Data Credits were created.

4.4.2 Data Credits (DC)

Data credits are a utility token pegged to the US dollar. Data credits are derived by *burning* HNT (burning means the irreversible conversion of HNT to DC and vice versa) and are used to pay all transaction fees on the Helium network. One DC will always cost 0.00001 dollars or 1 dollar buys 100,00 DC. If a customer wants to connect their IoT device to the Helium network, they have to:

- Register their device and generate a device ID on the Helium Console.

- Buy Data Credits at 100,000 DC per dollar and connect the wallet containing DC to the device ID.
- Every time the IoT device data is transported over the network, DC are consumed.
- Each DC will allow 24 bytes to be transferred over the network. So cost per packet is calculated as packet size divided by 24 bytes. If the packet size is 140, then 6 DC ($140/24$) will be consumed every time.
- Failed data transfer will result in no DC being burned.

4.4.3 Burn and Mint Economics

The Data Credits to Helium Token relationship is based on a concept called burn and mint equilibrium. This model allows the supply of HNT to respond to network usage trends. When an equilibrium is found between the amount of HNT that are burned to create DC and vice versa, their supply will remain constant month to month. The amount of Data Credits created by burning HNT is also based on the speculative price of HNT.

The conversion rate of HNT to DC and vice versa is constantly being adjusted. If the conversion rate was 1, then 1 HNT would be burned to create 100,000 DC and burning 100,000 HNT will produce 1 HNT. If a Helium network customer requires 100,000 DCs per month to send and receive data from their IoT sensors, they would have to purchase 1 HNT and burn it into 100,000 DC. These conversion parameters are constantly adjusted to grow the network.

If the network is constantly burning HNT to create DC and given the fixed supply of HNT, eventually the network will run out of HNT. To address this, all the HNT burned to create DC are added back into the monthly supply of HNT in a process called net zero HNT emissions.

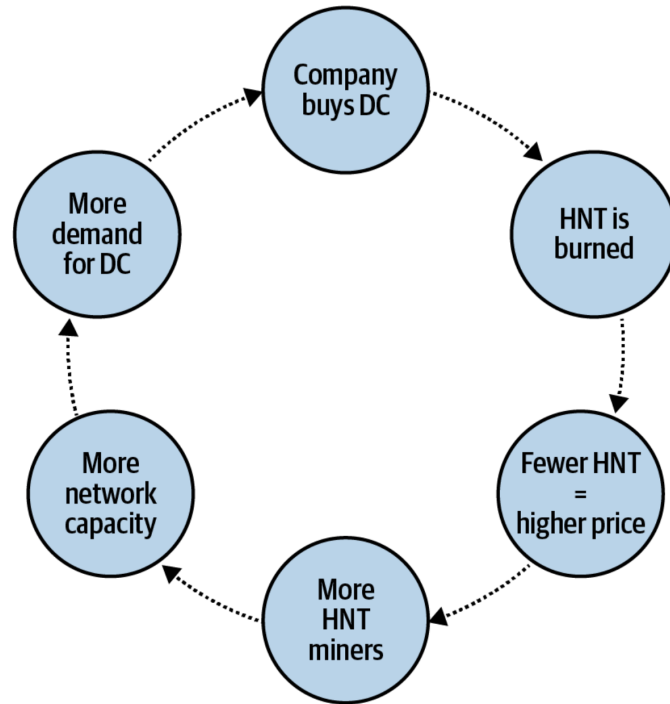


Figure 4.5: The cycle that allows the network to grow while keeping the prices of DC and HNT stable and all components of the network working. Figure is from [8].

4.5 Incentive Analysis

4.5.1 Network Growth

The concept of growing the network is baked into Helium on several different levels. A few ways the Helium network incentivizes network growth are:

- The amount of HNT awarded to a Hotspot is scaled based on the density of the Helium network where the Hotspot is located. The rewards awarded to Hotspots are scaled down rapidly if there are already 4 Hotspots in a 5 kilometer (3.1 mile) radius. This incentivizes Hotspot owners and network operators to deploy Hotspots in areas where there is low network coverage. This causes the network coverage to expand.
- Since HNT is traded on platforms like Coinbase and Binance, HNT can be considered a speculative asset. As long as the network is growing, holding onto HNT is profitable. If the network stops growing, HNT becomes less valuable. This market dynamic incentivizes cooperation amongst Hotspot owners to not stack Hotspots in the same location

and grow the networks coverage.

- The mint and burn economic cycle incentivizes network growth as shown in Figure 4.6.

4.5.2 Validators

Hotspots computing PoCs were causing a drop in network performance. Validators were introduced to solve this problem. To become a Validator, a Hotspot has to stake 10,000 HNT. Initially, Hotspots were hesitant to become Validators due to the large stake required. The reward structure was modified to send 6 percent of all HNT at the end of each epoch to Validators. This change in the reward structure incentivized more Hotspots to participate in the consensus process and generate PoC challenges.

4.5.3 Token penalty for inaccurate location

Hotspots that inaccurately report their location on the network are penalized if their location is revealed to be inaccurate during the PoC process. 40 percent of HNT awarded to the Hotspot is confiscated at the end of an epoch if the location reported by the Hotspot was determined as inaccurate.

4.5.4 Helium Improvement Proposals (HIP)

Any change that the Helium network makes are done through the Helium Improvement Proposal (HIP) process. These proposals are submitted by members of the Helium community, addressing a wide array of topics including token economics, blockchain functionality, and allocation of HNT rewards. Once a proposal reaches maturity, it is put to a vote.

Helium uses the on-chain voting mechanism to decide on HIPs. The voting is open to all token holders via the Helium website helium.com. Votes are weighted by the HNT balance of each voting wallet. 1 HNT represents 1 vote. Votes are submitted by issuing a transaction from the token holder's wallet to a wallet associated with their desired ballot option, typically "For" or "Against" [10].

The implementation stage is where the heavy lifting happens with HIPs. After being approved, the code for each HIP is reviewed by a community of developers, and the rollout

is closely monitored for any complications that may arise. These implementations are not necessarily permanent. Approved HIPs can also be closed or withdrawn at a later time after a new vote. This voting system incentivizes all community members to behave well and work together to decide the future of the network.

4.5.5 Community Owned Hardware

All the Hotspots operating on the Helium network are owned by individuals. Less than 4.7 percent of Hotspot operators own 10 or more Hotspots [8]. This means that the network is truly decentralized. This observation show that network dynamics have disincentivized big companies from buying a large number of Hotspots to centralize authority and influence the future direction of the network.

4.6 Summary

The Helium network is constantly expanding and providing broader connectivity for IoT devices. The network has also begun deploying Hotspots with 5G capacity in dense urban areas (5G will go online in 2023). The rapid growth can be attributed to Helium providing well defined standards for Hotspot hardware, making all software open source, including the community in the decision making process, and aligning the incentives of network stakeholders properly.

The work done on the Helium project will serve as a source of inspiration to all blockchain projects going forward. The Helium network showed that creating complex real-world blockchain projects is possible.

Chapter 5

Non Fungible Tokens and CryptoKitties

5.1 Non Fungible Tokens

Non-fungible tokens (NFT's) are unique digital items with blockchain-managed ownership. An NFT can be proved to be unique and not interchangeable with another digital asset or token. The record which proves the uniqueness of a NFT is a cryptographic transaction on a blockchain [11].

NFTs can be understood better by comparing them with fungible tokens. Real world currency notes are fungible tokens. Any 20 dollar bill can be exchanged for any other 20 dollar bill or two 10 dollar bills. Tickets to a movie and titles of property ownership are unique. Bitcoin and Ethereum tokens (ETH) are fungible. The fungibility of a digital asset is relative as well as subjective. Consider tickets to a sporting event. Tickets within the same section are fungible but tickets to box seats and bleacher seats are non-fungible. A tickets is more valuable to a fan of the sports team compared to a ticket owner who is not a fan. These nuances drive the implementation and incentive structure for NFTs.

Each blockchain implementation has to specify its own standards for NFTs. Currently, Ethereum is the most popular blockchain for NFTs because of the ERC -721 NFT standard that was adopted. This standard describes how NFTs can be managed, traded, and owned in accordance with the smart contract that added these NFTs to the Ethereum blockchain.



Figure 5.1: Fungible vs Non-Fungible digital assets. Figure is from [11].

A few characteristics of NFTs that have emerged and are consistent with all NFT standards are:

5.1.1 Standardization

Digital assets like movie tickets and domain name ownership records do not have a standard format to represent them. The way a video game represents in-game collectibles (gun skins in CS GO) is entirely different to the way movie tickets are represented. By utilizing non-fungible token standards on public blockchains, all digital assets can have common, reusable, and inheritable standards applicable to all NFTs. These standards will include basic properties covering ownership, transfer of ownership, and validation. The standards could also specify image size and format and other attributes that would allow new applications to be built on these standards [11].

5.1.2 Immutability and Provable Scarcity

This is common to all blockchains. The tokens as well as information enclosed in the tokens are highly resistant to tampering. This results in substantial trust and transparency. Smart contracts allow creators to place an upper limit on the number of NFTs the smart contract can supply. Once NFTs are created and issued through a process called *minting*, the content of the NFT can no longer be changed.

5.1.3 Programmability

This is the major differentiating aspect separating NFTs and real world assets. NFTs are natively programmable. The smart contract that mints an NFT, the copyright and possible royalties associated with an NFT, ownership transfer of NFTs, and all other NFT interactions are built into the smart contract. As an example, if a collateral function is built into an NFTs smart contract, then the NFT owner can use their NFT as collateral in another smart contract with the conditions of possession mentioned within the smart contract i.e if certain payment obligations are not met then the ownership of the NFT is automatically transferred. Programmability allows complex mechanics over the ownership of digital assets [11].

5.1.4 ERC-721

```
interface ERC721 {
    function ownerOf(uint256 _tokenId) external view returns (address);
    function transferFrom(address _from, address _to, uint256 _tokenId)
    external payable;
}
```

Figure 5.2: ERC-721 Basic Primitives code. Figure is from [12].

ERC-721 is an NFT standard pioneered by CryptoKitties. It was the first standard implemented for representing digital assets on a blockchain. ERC-721 is an inheritable smart contract standard. This means that developers can create new ERC-721 compliant smart contracts by simply importing the standard contract. The two important primitives of the ERC-721 standard are in Figure 5.2. The *ownerOf* primitive represents the owner of the token by storing the owner 256 bit wallet address. The *transferFrom* primitive allows permissioned transfer of NFTs between owners. Several other important primitives are included in the standard, but explaining all of those is beyond the scope of understanding NFTs as a concept.

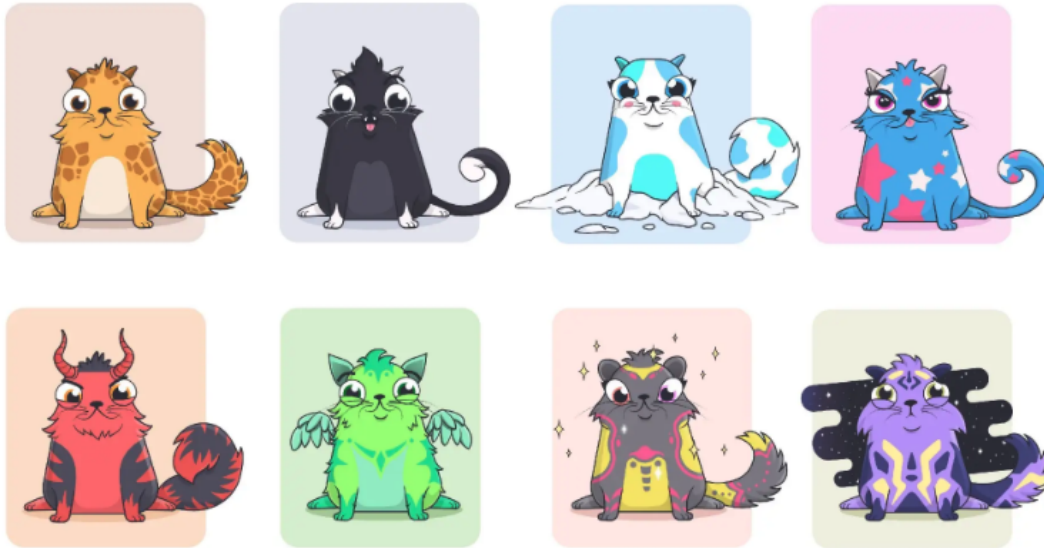


Figure 5.3: CryptoKitties with different cattributes visualized. Figure is from [14].

5.2 CryptoKitties

CryptoKitties is a game where players can collect, breed, and sell virtual cats for actual money. Every single cat in the game is unique and impossible to replicate. The ownership of each cat is recorded to the Ethereum blockchain through the CryptoKitties smart contract [13].

While the Ethereum blockchain is the technology underpinning CryptoKitties, the game itself is run by a “Genetic Algorithm”. This algorithm mimics real life cat genetics. The data that each kitten stores is like the DNA of living creatures. Each cat has “cattributes” which include base, highlight, accent colours, pattern, mouth shape, eye colour and shape, tail, and ear colour and shape.

The game is essentially a trading game with collectible cats. Players can go on the CryptoKitties marketplace and buy a cat either for a fixed price or via an auction. Players can find other players who have cats of the opposite gender and breed a new cat. Cats with certain “cattributes” are more valuable than others in the marketplace. The breeding dynamics of the genetic algorithm were selected to incentivize increased participation in the project. These dynamics are discussed in section 5.3.2.

5.3 Incentives Analysis of CryptoKitties

5.3.1 Limited Supply

The CryptoKitties smart contract had a hard limit on the maximum number of “Generation-0” cats that would be sold. All new cats going forward had to be generated by breeding “Generation-0” cats. The hard cap on the number of cats built into the smart contract is 50000. Developers would continually increase or decrease the supply of cats available until the hard cap was reached based on the public interest in the game. When interest went down more cats were made available to allow people to buy cats cheaply and vice versa when the interest went up [13].

The limited supply also meant that people were worried about missing out on having a CryptoKitty as well as missing out on getting in at the ground floor of a revolutionary new technology (digital asset ownership on the blockchain). These factors incentivized players to join CryptoKitties.

5.3.2 Speculative Mechanics, Liquidity, and Money Making Economics

The breeding and trading mechanics of CryptoKitties led to players called “breeders”. These players would buy a couple of cats, get them to breed until they produced a cat with “rare attributes”, and sell this cat to other breeders for profit. This created a “breeder community” dedicated to breeding and flipping cats. As long as new users keep coming and buying cats and there is a limited supply of cats with certain “rare attributes”, breeders could make good profits. Cats with “rare attributes” have sold for 25 Ethereum (\$110,000) at the project’s peak. This incentivizes breeders to keep breeding because there might be a payday at some point in the future. Attributes inheritance mechanics had a lot of random mutation associated with them. This caused a slot machine like incentive structure to develop where if you bred cats for long enough and were lucky, there was a chance to make big bucks. At the peak of the CryptoKitties project in mid 2018, nearly 5000 Ethereum (\$3,000,000 per day) was being transacted through selling, trading, and breeding per day.

At that time the CryptoKitty marketplace was liquid i.e. people were able to get in make a profit and get out of the market quite easily. This liquidity gave players confidence that their money would not be lost and incentivized players to play the game [13].

CryptoKitties are only valuable as long as someone is willing to buy them at a price profitable to the seller. This created a dynamic where players were constantly marketing the game by word of mouth to increase participation in the game. The game was designed to incentivize players to try and grow the network and make it more valuable.

5.3.3 Viral Story and Marketing

Storytelling played a major role in the success of CryptoKitties. According to how they were marketed, Cryptokitties would do for consumer acceptance of blockchain based digital asset ownership what FarmVille did for the adoption of Facebook and what Angry Birds did for the adoption of smartphones. On top of this, CryptoKitties were cute cats that were easily shareable over social media. Overall CryptoKitties was a fun game for people interested in smart contracts and blockchain technology to jump into the field [11].

At one point in 2018 CryptoKitties were so popular, the entire Ethereum network was choked processing CryptoKitties transactions. The daily number of pending transactions reached 11,000 transactions per day at peak. At that time the Ethereum network could only process 15 transactions per second. This caused Ethereum nodes to start placing high fees to include ERC-721 smart contract transactions. This meant that new cat buyers had to pay extremely high fees and wait several days for their transactions to be settled and added to the blockchain. This caused a “Cryptokitties bubble”. The high transaction fees meant that people could no longer participate in the game, this caused the market for CryptoKitties to fall significantly towards the end of 2018. Today, CryptoKitties averages about 50 Eth of transactions per week, which is not an insignificant amount in the world of smart contract transactions [13].

5.4 Summary

Non-fungible tokens (NFTs) are blockchain based tokens that each represent a unique asset like a piece of art, digital content, titles to property etc.. An NFT is a irrevocable digital certificate of ownership and authenticity for a physical or digital asset. The creation of programmable digital ownership creates a huge field of potential opportunities for application developers.

Cryptokitties is considered a pivotal project in the world of NFTs and programmable digital assets. Cryptokitties was the first project to sell ownership of images as tokens and highlight gaming as a possible use case for blockchains. Cryptokitties shifted focus from rigid protocols like Bitcoin to more flexible and programmable protocols like Ethereum. After the CryptoKitties project, there was a huge amount of interest and development effort that shifted towards creating Ethereum smart contracts. The *transferFrom* and *breeding* functions in the CryptoKitties smart contract are the precursor to all modern decentralized finance applications.

Chapter 6

Blockchain Incentive Analysis

Routledge, Jones, and Ebrahimi (2020) argue that the two important properties that blockchains have to incentivize to work are *consensus* and *permanence* [15]. Everyone participating in a blockchain network has read and write access to the blockchain. Every participant can propose their own version of the blockchain. For a blockchain to work, it is a necessary condition that all participants agree on which proposed blockchain ledger is correct. This is the *consensus* property. Once consensus is reached, for all network participants to be able to trust the network they should no longer be able to change transactions in previous blocks or switch out old blocks with new blocks. This property of being unable to update previous blocks is called *permanence*. The mechanics of how consensus protocols like proof of work inherently incentivize consensus and permanence are discussed with concepts from economics and game theory.

6.1 Game Theory

Game theory is a branch of mathematics that analyzes strategic interactions between rational decision makers. The major components of game theory are the players making decisions, the strategies players deploy, and the resulting payoff. In a zero sum game, some players gain at the expense of other players. In a non-zero sum game, the gains a player makes do not cause losses for others i.e. strategies exist where everyone's outcomes can be balanced. The following game theory concepts are useful to analyze blockchains.

	B takes action	B does not take action
A takes action	(4,4)	(4,0)
A does not take action	(0,4)	(0,0)

Table 6.1: Payoff matrix to explain Nash Equilibrium.

6.1.1 Nash Equilibrium

Nash equilibrium is when a player has no incentive to modify their strategy irrespective of their opponent's strategy. Considering the payoff matrix in Table 6.1:

If A takes action, B has a payoff of 4 if B takes action and a payoff of 0 if B does not take action. If A does not take action, B still has a payoff of 4 if B takes action and a payoff of 0 if B does not take action. Here, the action taken by A has no effect on the optimal strategy for B and vice versa. Therefore, both A and B taking action is the Nash equilibrium [16].

6.1.2 Schelling Point

A Schelling point (or focal point) is a solution that players of a game default to in the absence of communication. If two bikers are heading straight at each other, to avoid collision both bikers (assuming they are used to driving in the US) will swerve right. Drivers used to driving on the left (like they do in India) will both swerve left to avoid the collision. In both cases, the move that both players default to because they have no way of communicating with each other is the Schelling point. As this example showed, Schelling points depend upon culture and having shared context about what is done in a situation [16].

6.2 Incentive Analysis

Extending game theory to the Bitcoin blockchain, there are two major players: the miners and the users. The users set up wallets and send and receive Bitcoins between wallets. The miners take transactions on the blockchain and participate in the Proof of Work consensus protocol to publish blocks. The miners that contributed resources to compute the hash are rewarded with Bitcoin. Since miners decide which transactions are added to the blockchain, they have power in this system. Miners can behave dishonestly by lying about solving the

proof of work problem, including fake transactions to award themselves additional coins, and double spend coins. Miners could choose to behave dishonestly during the consensus process and mess up the entire blockchain. Game theory mechanics can be applied to analyze miner behavior.

6.2.1 Double Spend Problem

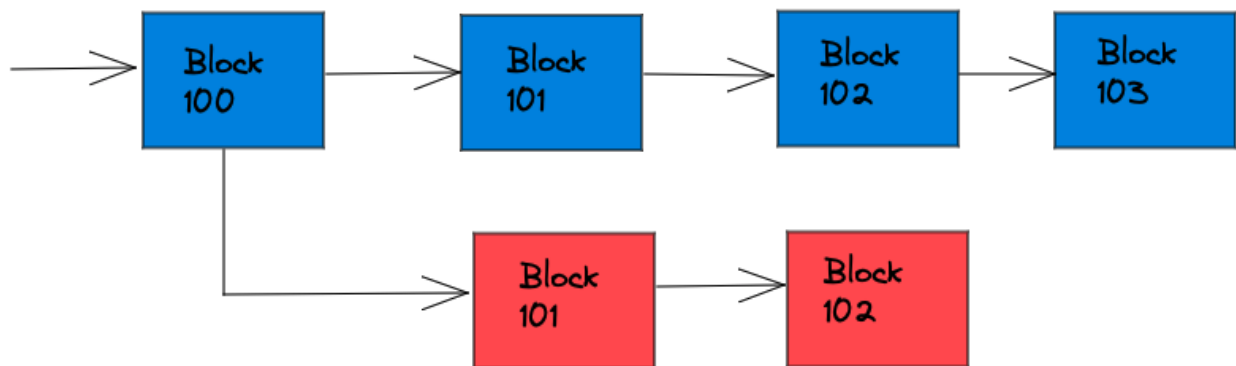


Figure 6.1: Red chain represents the correct chain that is replaced when the longer blue chain is published by the dishonest miner.

Double spending is the major attack available to miners. Double spend attacks are carried out by publishing a real payment to the blockchain network, waiting for that transaction to be added to a published block, and publishing a longer chain of blocks that were mined secretly that do not contain the transaction. The transaction is no longer part of the longest chain and for all practical purposes “never happened”.

Figure 6.1 visualizes the typical double spend scenario. Assume the latest block on the blockchain is Block 100. A miner publishes a transaction to the network. The blocks depicted in red are published blocks that include this transaction. Let’s assume the transaction is included in Block 102. As soon as Block 102 containing the transaction is published, the miner that added the transaction in the first place publishes 3 new blocks none of which contain the transaction. Since the chain with the 3 blocks is longer, Block 103 is transmitted to all miners connected to the blockchain. If the forked blockchain having Block 103 is adopted by a majority of nodes on the blockchain, then the transaction in effect never happened and the coins spent in the transaction can be spent again.

On the current Bitcoin blockchain, double spending is never observed. This is because getting double spend to work requires coordination between a majority of miners on the blockchain network which is very difficult to reach in a decentralized network and miners capable of executing successful double spend attacks have a large stake in Bitcoin's trustworthiness and reputation.

6.2.2 Attack Models

Modeling the ways miners or attackers could behave to subvert the Bitcoin blockchain are discussed. The *honest majority* model assumes that 51 percent of network participants are honest. The honest majority model is not relevant in an adversarial environment.

The *uncoordinated choice* model assumes that due to decentralization miners will not be able to coordinate their dishonest actions. The Schelling point in this case is for all the miners to act honestly because they will miss out on mining rewards if they do not participate towards securing and propagating the blockchain. Bitcoin is based on the uncoordinated choice model.

Coordinated Choice Model

The coordinated choice model assumes that miners are able to communicate and coordinate to pick and choose what transactions go on the blockchain. The coordinated choice model is relevant because Bitcoin's mining process results in a large mining reward with an extremely low (because of the large number of miners) probability of reward for individual miners. This kind of high-risk and high-reward payoff rewards structure disincentivizes individual miners and decentralization. Most miners will prefer to have a small rate of income over a long period of time. The constant income stream can pay for electricity and other charges the mining process incurs. This economic insight incentivizes miners to coordinate into mining pools where miners pool their computational resources together to publish blocks and share rewards amongst members of the pool. Miners working together in pools have a much higher probability of publishing the next block compared to individual miners working alone. Mining pools enable participating miners to earn Bitcoin rewards on a regular basis.

From the perspective of the Bitcoin blockchain, a pool is just a single mining node. Miners in the pool interact with the pool's server. The server sends the transactions and headers the pool is working on solving to all the miners in the pool. All of the miners in a pool have predetermined nonce starting points to avoid duplicating hash computations. Once a miner finds a solution, the miner sends it to the pool server which then publishes the block to the network. The block publishing reward is sent to the pool server which distributes the reward amongst the miners in the pool [17].

Mining pools have to take precautions to make sure that only miners who are contributing work to find the hash are rewarded. So, every miner in the pool has to send partial solutions of Proof of Work to the pool server. These partial solutions show that the miner is actually working and can be used to assess the computational power a miner had provided to the pool. Miners are rewarded proportional to the computational resources they provide to the pool. These mining pools are susceptible to manipulation in the following ways:

- *Pool Hopping* - Miners participating in pools can earn greater rewards by submitting partial solutions to a pool and jumping to another pool where they submit more partial solutions and so on. Whenever one of these pools publishes a block, the miner will be rewarded. Pool-hopping-resistant reward schemes were developed and adopted by mining pools to end pool hopping [17].
- *Block Withholding Attacks* - While miners cannot steal rewards from a mining pool, they can still prevent the mining pool from publishing a solution by discarding full solutions and only submitting partial solutions to the pool server. This causes the pool server to have to allocate rewards to these miners who are doing no useful work. Block withholding attacks are used by mining pools to sabotage other mining pools. The attacker pool registers miners to the target mining pool where these miners run the block withholding attack and earn rewards without providing useful work to the pool. In the long run, the target pool can no longer maintain rewarding participating miners and is abandoned due to cost overruns. Taking out competitive mining pools is profitable. To prevent this, modified mining protocols have been suggested that do

not let miners differentiate between full and partial solutions causing them to submit every solution and in turn making block withholding attacks impossible [17].

Mining pools create the possibility of 51 percent attacks on the blockchain where 51 percent of all mining capacity of the Bitcoin blockchain coordinates and joins a single mining pool. This mining pool cannot rewrite past transactions or steal money from wallets. However, they can prevent all other miners on the network from publishing new block and double spend coins since they have sufficient miners to decide the valid chain. This kind of an attack on the Bitcoin blockchain while possible has never occurred and is unlikely to occur. A 51 percent attack on a blockchain breaks all trust that users have in the blockchain. This along with speculators selling off Bitcoin will cause Bitcoin to devalue and eventually crash to having no value. Maintaining the trust of users and speculators is essential to Bitcoin having any value whatsoever. This dynamic has so far disincentivized miners from forming pools capable of 51 percent attacks [17].

Briber Attacker Model

	Miner disapproves block	Miner approves block
Majority disapproves published block	P	0
Majority approves published block	0	P

Table 6.2: Payoff matrix showing rewards for publishing with or against the majority of miners in the blockchain network.

The briber attacker model was presented by Vitalik Buterin (creator of Ethereum) as a possible non-coordinated attack model. The model assumes there is an attacker who is able to offer bribes to miners and achieve double spending at zero cost. The base games payoff matrix is in Table 6.1. The game works on the assumption that the majority of miners earn rewards as long as they are in the majority, and the miners who are in the minority earn nothing. If a majority of miners approve the publishing of a block, they earn a partial amount of the publishing rewards while all the miners in the minority against publishing a block earn nothing.

In this scenario, an attacker could offer a bribe amount greater than the reward achieved by participating in the majority to all miners in the minority to approve the publishing of

	Miner disapproves block	Miner approves block
Majority disapproves published block	P	$P + \epsilon$
Majority approves published block	0	P

Table 6.3: Payoff matrix showing the improved reward for being against the majority of miners publishing blocks. The optimal strategy for individual miners is different from the strategy for miners participating in a group and no Nash equilibrium exists.

a block. The new payoff matrix is presented in Table 6.3. All miners in the minority will earn a reward of $P + \epsilon$. Given that the reward for being in the majority is fixed, the incentive for participating in the minority is greater because more rewards can be earned. Other miners who realize it is more lucrative to be a part of the minority will switch to the minority since their optimal strategy is independent of other miners in the network. But most miners in the majority who are rational and want to achieve greater rewards will also switch over to the minority. This causes an inversion where the minority becomes the

	Miner disapproves block	Miner approves block
Majority disapproves published block	P	$P + \epsilon$
Majority approves published block	$P + \epsilon$	P

Table 6.4: Payoff matrix showing rewards when the inversion occurs and the minority becomes the majority. Because the minority the attacker promised bribes to is now the majority, the attacker was able to publish their block with zero cost incurred.

majority, reverting the rewards back to the original P. The new payoff matrix when this inversion occurs is in Table 6.4. The attacker achieved the goal of publishing the blocks they wanted at zero cost since the bribe was only offered to miners in the minority. Since the minority becomes the majority, the network automatically rewards participating miners with the predetermined reward. This attack formally called a $P + \epsilon$ attack was presented by Vitalik to show a weakness with Proof of Work and endorse Proof of Stake. Proof of Stake requires a substantial amount of Ethereum deposited to participate in the consensus protocol. Miners are heavily penalized if found to be dishonest [18].

6.2.3 Checkpoint Proof of Work Consensus

Routledge, Jones, and Ebrahimi (2020) suggest an improvement to the Bitcoin Proof of Work consensus protocol by introducing the concept of a checkpoint block to the longest

chain mechanism. They argue that the introduction of a checkpoint block will limit the set of potential predecessor blocks to recently added blocks as all newly published blocks can only be published after the checkpoint block. The checkpoint is implemented to go hand in hand with the Bitcoin blockchains settlement lag. Anyone accepting Bitcoin as a mode of payment should wait until the transaction is in a block behind the checkpoint block before providing their services. The norm with Bitcoin is that a seller receiving Bitcoin will have to wait at least six blocks (about an hour since a block is published every 10 minutes) before providing their services. With a checkpoint introduced, the blocks behind the checkpoint can never be modified and transactions behind the checkpoint are confirmed to be on the Bitcoin blockchain. Routledge, Jones, and Ebrahimi (2020) provide a game theoretic proof for the viability of a checkpoint block. The proof is beyond the scope of the thesis [19][15].

6.3 Summary

Consensus and Permanence were introduced as the concepts any blockchain consensus protocol has to incentivize. Strategies like Nash Equilibrium and Schelling point were introduced to used as tools to analyze incentives. Cryptocurrency rewards are the driving factor behind all the incentives on blockchains. Consensus protocols incentivize miners honestly working to secure the blockchain by rewarding them with cryptocurrency. The miners acting dishonestly and trying to subvert the blockchain via double spending and other malicious attacks have to be disincetivized by either denying them cryptocurrency rewards or penalizing their staked cryptocurrency.

Chapter 7

Conclusion

7.1 Conclusions

Blockchain Implementation

This thesis aimed to understand blockchain networks by implementing one from first principles. A working blockchain network was implemented. The full implementation is available in the Github Repo¹. All design and technical choices were made with the goal of simplifying all the blockchain components and making the blockchain implementation approachable for future research and extensions. Assumptions and design choices made while implementing the blockchain network are:

- The user interface having the blockchain explorer, wallet, buy page, transfer page, and mine page is served to the browser as static HTML. Data is added to these static pages using the *Jinja* templating engine. The pages are served by a HTTP server written with the *Flask* framework.
- To allow for the deployment of blockchain nodes anywhere on the Internet, the *Websockets API* is utilized to create peer-to-peer connections between the nodes. To simplify the implementation, every blockchain node is connected to every other blockchain node through a websocket connection. This is different from the way Bitcoin is implemented. Bitcoin creates a peer-to-peer network similar to the way torrents work by

¹<https://github.com/rahulreddy15/thesis-repo>

utilizing a heartbeat protocol. Socket connections are handled by utilizing the *socket.io* library.

- The mining process is triggered by one of the blockchain nodes and is not automatic the way it is in the Bitcoin blockchain. This simplifies the process of setting the mining difficulty for the blockchain network. The mining difficulty is hardcoded into the *Blockchain* class.
- The blockchain network was designed with the assumption that all blockchain nodes are honest. This assumption greatly simplified the design of the Proof of Work consensus protocol. The logic which verifies the validity of each transaction is run for every transaction by every node on the network. Once a block is published to the network, it is automatically added to the blockchain because all the transactions in the block have already been verified.
- To create a competitive environment where one of the blockchain nodes succeeds by finding a hash and publishing it to the network, the nonce of each blockchain node is initialized to a random value before the mining process begins. So, the starting nonce value is different for each blockchain node.
- The SHA256 hash function used to find hashes for blocks, verify digital signatures of transactions, and generate public and private wallet keys is an API from the *PyCrypto* library.

Incentive Analysis

The Helium blockchain and its novel Proof of Coverage algorithm enabled the creation of a network of community owned Hotspots for IoT devices. The Helium network incentivizes increasing network coverage by adjusting HNT rewards based on Hotspot density, turning HNT into a speculative asset, and creating a mint and burn economic model that balances out the supply of DC and HNT to maximize network coverage. Validators were a new kind of node introduced to reduce the burden of creating Proof of Coverage challenges and publishing new blocks on Hotspots. When Hotspots were hesitant to stake 10,000 HNT to participate

in the network as Validators, the rewards structure was adjusted to incentivize Hotspots to participate as Validators. The Helium network disincentivizes Hotspots from reporting their location and network coverage inaccurately by heavily penalizing dishonest Hotspots. The incentives structure for the Helium network is constantly being adjusted through community voting with Helium Improvement Proposals.

Non-fungible tokens (NFT's) are unique digital or physical items with blockchain managed ownership. NFTs were standardized on the Ethereum blockchain with the ERC-721 standard. The creation of the ERC-721 NFT standard was pioneered by a game called CryptoKitties. CryptoKitties is a game where players can collect, breed, sell, and trade virtual collectible for actual money. The mechanics of breeding cats is powered by a genetic algorithm that occasionally breeds cats with highly valuable "cattributes". Cats with these "rare cattributes" were being sold for large amounts of money. This created a slot machine like incentive structure where, if a player bred cats for long enough and was lucky, they could make a lot of money. The growth of the game was incentivized by controlling the supply of cats available to be purchased. Players of the game understood that their CryptoKitties were only valuable as long as there was demand to buy them. This meant that all players were incentivized to grow the number of people playing the game. In addition to this, viral marketing and the inherent shareability of cat pictures on social media platforms combined with the chance to make a quick buck greatly incentivized people to play the game. While CryptoKitties has lost a lot of its players since the games peak in 2018, the ideas introduced in the CryptoKitties smart contract have inspired modern decentralized finance applications.

Consensus is the property that all miners choose the same block (hence chain) as a predecessor for their new block of transactions. Permanence is the property that miners do not choose "old" blocks as predecessors that would create an alternate chain to eliminate blocks previously on the consensus blockchain. Both these properties have to be incentivized for a blockchain and its consensus protocol to be useful. The Bitcoin blockchain incentivizes consensus by providing miners that participate in the consensus protocol with rewards. This rewards process has incentivized the creation of mining pools where miners pool their resources in order to have a better chance of receiving a share of the block publishing reward. The Bitcoin blockchain incentivizes permanence by tying the value of Bitcoin to the trust and

security of the network. If transactions in the blockchain cannot be trusted, all stakeholders (miners, users, and speculators) stand to lose everything. Coordinated-choice and Briber-attacker models are possible ways to attack the permanence property by adopting a forked version of the blockchain. These kinds of attacks are also disincetivized for the reasons mentioned above.

The analysis of incentives discussed in this thesis is a summarization of literature published in the area of incentive analysis of blockchains. The contribution of this thesis include an informative introduction to blockchain networks with the help of the Helium Network and CryptoKitties case studies. Additionally, Game theory and attack models are used to introduce how blockchains incentivize consensus and permanence.

7.2 Future Research

Blockchain Implementation

Future researchers working in this are could extend the implementation of the blockchain network by adding the following functionality:

- *Peer To Peer Heartbeat Protocol* - The current implementation works by connecting each node on the network to every other node on the network. As the number of nodes in the network increases this becomes infeasible. The *Peer To Peer Heartbeat Protocol* is a way of propagating messages to all peers of the network without being connected to all of them.
- *Standardize Message Data* - The current implementation does not have standards for messages. The messages could be from a couple kilobytes to several megabytes in size. This will cause problems as the size of the network grows. Implementing standards will allow faster propagation and processing of messages.
- *Automatic Mining* - The current implementation requires one of the nodes on the network to start the mining process. Being able to automatically mine transactions and publish block after set periods of time would bring the implementation closer to how real blockchains work.

- *Dynamically Update Mining Difficulty* - Currently, the difficulty for the Proof of Work puzzle is hard-coded into the *Blockchain* class. Being able to dynamically update this difficulty based on the amount of time publishing a new block takes will bring the implementation closer to how real blockchains work.

Incentive Analysis

The current work explores the incentives that made Helium and CryptoKitties successful blockchain projects. The incentives inherent to blockchains that have allowed it to create decentralized applications in trustless environments have been discussed. Distilling the principles behind these incentives into a predictive framework that can be used to analyze the viability of future blockchain projects is a future area of research. Several countries have released preliminary reports on creating Central Bank Digital Currencies (CDBC). Extending the current understanding of incentives to analyze the incentives that would enable the successful adoption of CDBC is another area future researchers can explore.

References

- [1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized Business Review*, pp. 21–30, 2008.
- [2] D. Yaga, P. Mell, N. Roby, and K. Scarfone, “Blockchain technology overview,” *NISTIR 8202*, Oct 2018.
- [3] D. V. Flymen, *Learn Blockchain by Building One: A Concise Path to Understanding Cryptocurrencies*. Apress, 2020.
- [4] Helium-Explorer, “Helium explorer proof of coverage receipt.” <https://explorer.helium.com/txns/kQVfUV88j3s1Glj4fRXFb5MqBeFdTq8ILK2RlaFTnvQ>.
- [5] D. Jagtap, A. Yen, H. Wu, A. Schulman, and P. Pannuto, “Federated Infrastructure: Usage, Patterns, and Insights from “The People’s Network”,” in *Proceedings of the 21st ACM Internet Measurement Conference, IMC ’21*, (New York, NY, USA), p. 22–36, Association for Computing Machinery, 2021.
- [6] Actility, “Lorawan network server – actility: Leader in IOT network connectivity.” <https://www.actility.com/lorawan-network-server/>, Jul 2019.
- [7] A. Haleem, A. Thompson, A. Allen, R. Garg, and M. Nidjam, “Helium: A decentralized wireless network.” <http://whitepaper.helium.com/>, Nov 2018.
- [8] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, “The honey badger of bft protocols,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS ’16*, (New York, NY, USA), p. 31–42, Association for Computing Machinery, 2016.
- [9] Helium-Community, “Hip-20 adjusting helium tokens max-monthly supply.” <https://github.com/helium/HIP/blob/main/0020-hnt-max-supply.md>, Oct 2021.
- [10] Helium-Community, “Helium improvement proposal process and on-chain voting.” <https://github.com/helium/HIP>.
- [11] D. Finzer, “The non-fungible token bible: Everything you need to know about nfts.” <https://opensea.io/blog/guides/non-fungible-token>, Apr 2022.
- [12] “Erc-721 non-fungible token standard.” <https://ethereum.org/en/standards/tokens/erc-721/>, Oct 2017.

- [13] A. Serada, T. Sihvonen, and J. T. Harviainen, “Cryptokitties and the new ludic economy: How blockchain introduces value, ownership, and scarcity in digital gaming,” *Games and Culture*, vol. 16, no. 4, pp. 457–480, 2021.
- [14] CryptoKitties-Team, “Cryptokitties blog: Breed these imaginary cryptokitties.” <https://www.cryptokitties.co/blog/post/will-you-be-the-first-to-breed-these-imaginary-cryptokitties/>, Sep 2018.
- [15] Z. Ebrahimi, B. R. Routledge, and A. Zetlin-Jones, “Getting Blockchain Incentives Right,” Tech Report, Carnegie Mellon University, 2019.
- [16] S. Alexander, “Nash Equilibria and Schelling Points.” <https://www.lesswrong.com/posts/yJfBzcDL9fBHJfZ6P/nash-equilibria-and-schelling-points>, Jun 2012.
- [17] Y. Sompolinsky and A. Zohar, “Bitcoin’s Underlying Incentives,” *Queue*, vol. 15, no. 5, p. 29–52, 2017.
- [18] T. Obasi, *The Economics of Cryptocurrency. Incentivizing Decentralisation*. GRIN Verlag, 2018.
- [19] I. Eyal and E. G. Sirer, “Majority is not enough: Bitcoin mining is vulnerable,” *CoRR*, vol. abs/1311.0243, 2013.