Comparing Rewarding Mechanisms of Bitcoin and Fruitchain

Master Thesis

École Polytechnique Fédérale de Lausanne
in collaboration with CEA-LIST

Author
Mustafa Safa Özdayı (mustafa.ozdayi@epfl.ch)

Supervisors
Prof. Rachid Guerraoui (rachid.guerraoui@epfl.ch)
Önder Gürcan (onder.gurcan@cea.fr)
Sara Tucci (sara.tucci@cea.fr)

Lausanne, 2018

So Thou be sweet, let life run bitterly;
So Thou be pleased, let men be wroth with me;
So all things flourish between me and Thee,
Let all between me and the world in ruins be.

A Sufi poem.

Contents

Abstract

Bitcoin is a decentralized payment protocol that operates on a peer-to-peer network. Its functionality is maintained by certain nodes called *miners* who use their computing powers to validate and record transactions. In return, miners are rewarded by the protocol.

Reward allocation in Bitcoin is a random process with a large variance. To gain their rewards in a steady, profitable manner, miners tend to combine their computing powers and form *mining pools*. This leads to centralization of the computing power in the Bitcoin network. Consequences of this might be severe, in an extreme case, the entire network can be controlled by a single entity. The Fruitchain protocol aims to alleviate this threat by modifying the mining process of Bitcoin and introducing a new rewarding mechanism.

In this work, we attempt to capture the differences between rewarding mechanisms of Fruitchain and Bitcoin in a formal manner. Concretely, we compare the protocols against two properties that affect miners' earnings and behaviors: *fairness* and *profitability*. Broadly, in this context, fairness is protocol's ability to distribute rewards in proportion to miners' computing power. On the other hand, profitability reflects a miner's gain over a given period of time.

To compare the protocols, we first explicitly define fairness and profitability under an execution model. We then show Bitcoin and Fruitchain are equivalent in terms of fairness and Fruitchain is more profitable than Bitcoin. Finally, we validate our results and illustrate their implications for miners through simulations.

Chapter 1

Introduction

Bitcoin [1] is an online payment protocol that operates on a peer-to-peer network without a trusted central authority. Transactions are exchanged directly between the users and are recorded in a public, append-only, decentralized ledger called *the blockchain*.

Recording transactions is only possible through a process called *mining* which requires computational effort. There exist certain nodes called *miners* who devote their computational resources to this task. As compensation, the protocol rewards miners in bitcoins, the currency of the protocol[1].

The rewarding mechanism of Bitcoin suffers from few problems. In particular, expected time and variance of receiving payouts can be quite large, especially for miners with relatively low computing power. The Fruitchain protocol [2] attempts to solve this by modifying the mining process and the rewarding mechanism of Bitcoin.

In this work, we attempt to capture and illustrate the differences between Fruitchain and Bitcoin in terms of their rewarding mechanisms.

1.1 Motivations

The difficulty of the mining process increases as the total computing power increases over the Bitcoin network. This is necessary to maintain the se-

[1]Bitcoin with a capital 'B' refers to the protocol and bitcoin with a lowercase 'b' refers to its currency. We shorten the currency by BTC in the rest of this paper.

curity of Bitcoin, however, as an undesired consequence, it has become unprofitable to participate in the mining process as a solo miner.

To participate in the mining process in a profitable manner, miners tend to combine their computing powers and form mining pools [3]. These pools are usually governed by managers who can use computing powers of pool's participants at their will. This introduces a degree of centralization to the network and makes it more susceptible to certain attacks such as the "51% Attack" [4] in which a collaborating majority can rewrite the entire history of transactions. As of February 2018, the total computing power of the network is distributed over only about 14 pools [5].

The mining process and the rewarding mechanism in Fruitchain are designed in a way such that miners can make steady profits without needing mining pools. A detailed study of the protocol seems to be lacking in the literature (other than its authors'), we aim to fill this gap with our work by highlighting its differences from Bitcoin. Concretely, we show the differences between Fruitchain and Bitcoin in terms of their rewarding mechanisms. To this end, we compare the protocols against two properties: *fairness* and *profitability*, that we believe are important to miners. We explicitly define these properties under an execution model. We then establish our results in terms of protocols' abilities to satisfy these properties. Finally, we validate our results and illustrate their effects by doing simulations.

1.2 Related work

The Fruitchain protocol is first introduced in [2] and details regarding its rewarding mechanism are elaborated in [6][2]. It draws ideas from the decentralized mining pool P2Pool [7].

It is shown that the existence of mining pools has an effect on the fair distribution of rewards in the Bitcoin network [8].

Eyal et al. [9] show rewarding mechanism of Bitcoin is not incentive compatible by introducing and analyzing a deviant mining strategy called *selfish-mining*. Work of Sapirshtein et al. [10] further analyzes and optimizes this strategy.

[2]This work, as of February 2018, is unpublished. We obtained the paper by communicating with its authors.

Finally, Carlsten et al. [11] illustrate that Bitcoin instabilizes once the fixed reward of the mining process runs out.

1.3 Structure of the thesis

The rest of the paper is structured as follows. In Chapter 2, we give the necessary background on Bitcoin and Fruitchain. In Chapter 3, we present the execution model that we use in our analysis and define *fairness* and *profitability* formally. We then compare Fruitchain and Bitcoin according to these properties and show each protocol's performance in terms of satisfying them. In Chapter 4, we validate our results and illustrate their implications on miners through simulations. We also do an experimental comparison of Bitcoin and Fruitchain under *the selfish-mining attack* in this Chapter. Finally, in Chapter 5, we provide some concluding remarks.

Chapter 2

Background

In this chapter, we provide the necessary background on Bitcoin and Fruitchain.

2.1 Bitcoin

We explain the Bitcoin protocol by first giving a brief overview. We then explain the related structures in details.

2.1.1 Overview

The nodes exchange currency over the Bitcoin network by broadcasting *transactions.* Certain nodes validate and include transactions in structures called *blocks* and blocks are stored in a public, append-only, decentralized ledger called *the blockchain*. The blockchain is essentially a sequence of blocks and a transaction is considered to be recorded once the block containing it is stored in the blockchain.

Each node maintains a local copy of the blockchain[1]. By collectively maintaining the blockchain, nodes agree on the order of transactions. This makes calculating the balance of every participant and verifying the validity of a transaction possible by querying the data on the blockchain. In

[1]In reality, some nodes only store a part of the chain. We assume every node contains the entire chain to simplify our explanations.

particular, this makes double-spend attacks [12] detectable under a decentralized setting which is, in fact, the main novelty of the Bitcoin protocol.

A node must engage in a process called *mining* to record transactions. Such nodes are called *miners* and every node in the network can be a miner at his will. Briefly, miners first validate and store transactions in blocks and then they attempt to solve a computational puzzle called "Proof-of-Work" (PoW) [13]. When a miner solves the PoW, he gets the right to extend the blockchain with his block. Concretely, upon solving the puzzle, a miner appends his block to his chain and then broadcast it to the network. Other nodes then deliver the block, validate it and append it to their own chains. Through this process, the network reaches to an agreement on the order of blocks and hence, on the order of transactions. To encourage participation in mining, the protocol rewards miners in BTC.

It is possible to have different miners solving the PoW and broadcasting their blocks simultaneously. This causes disagreement on the order of blocks. However, the protocol is designed in a way such that those disagreements are eventually solved as we explain in Section 2.1.3.

2.1.2 Transactions

Transactions are means to exchange currency over the Bitcoin network. The most basic transaction contains a single input and a single output. Conceptually, the input spends some amount from a previous transaction and the output specifies the new owner of this amount

An input refers to the output of a previous transaction to claim the amount denoted by it, i.e. input "spends" the output it references. It is then possible to send this amount to an address by creating a new output.

Technically, an input has a signature and a reference field and an output has a value and a public key field. The value field of an output denotes the amount it contains in BTC. Public keys essentially play the role of addresses in this setting and thus, also referred as addresses.

Only the valid transactions should be recorded in the blockchain. Bitcoin dictates the following validation rules for a transaction[2].

[2]Real validation rules of Bitcoin are slightly more complex [14]. We simplify them in our explanation to highlight the related parts.

(i) Its input references to an unspent transaction output (UTXO).

(ii) Its input's signature is valid under the public key of the output it references.

(iii) The value field of its output is equal or less than the amount its input spends.

By maintaining a set of unspent transactions, the UTXO pool [15], miners ensure these conditions are met and discard the invalid transactions. As an optimization, transactions usually keep a list of inputs and a list of outputs.

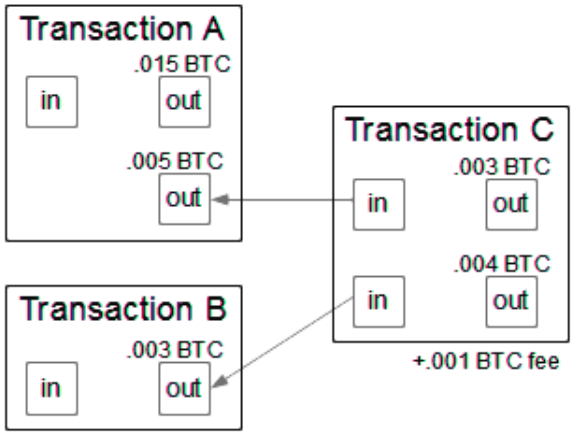


Figure 2.1: A simplified view of Bitcoin transactions [16].

In Figure 2.1, Transaction C claims a total of 0.008 BTC by referring to outputs from transactions A and B. It then sends 0.007 BTC to some addresses with two new outputs. The remaining 0.001 BTC is called *transaction fee* and left as a reward to miner who records this transaction in the blockchain.

2.1.3 Blocks, the Mining Process and the Blockchain

As mentioned before, transactions are not recorded directly in the blockchain but rather, they are included in structures called blocks. Blocks are then

chained together, creating the blockchain. By agreeing on the order blocks in the blockchain, nodes reach an agreement on the order of transactions.

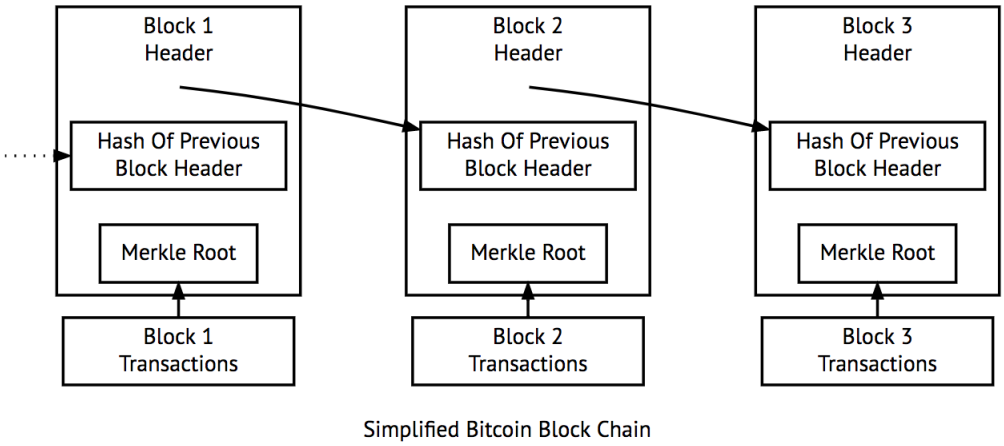


Figure 2.2: A simplified view of the blockchain. Every block contains a reference to the previous block [17].

A block consists of two parts: *header* and *body*. In its body, the block contains its set of transactions as a Merkle Hash Tree [18]. In its header, it contains the root of the tree, a reference to the previous block in the chain and a nonce value.

If miners arbitrarily create blocks, maintaining agreement on the blockchain would not be possible. Thus, the protocol needs to select a miner to extend the blockchain, one at a time. This is done through a process called (block) *mining*.

We first give the validation rules for a block and then explain how (valid) blocks are created by miners. Basically, a block is considered valid if it meets the following rules[3].

(i) It contains only valid transactions.

(ii) The digest of its header is lower than a certain difficulty target[4] when double hashed under SHA-256 cryptographic hash function.

[3] Again, see [14] for the complete list of the validation rules.

[4] This is just a number in the range of the hash function. The higher its value, the easier to create blocks.

To create blocks, miners validate and store transactions in a block structure and set its reference to the last block[5] in the blockchain. Then, they increment block's nonce and hash block's header until it meets the validity rules. As this process requires computational effort, the protocol rewards miners who create blocks as compensation. This is done through a special transaction called the *coinbase transaction* which is contained in every block by convention. Coinbase transaction rewards the creator of the block by a fixed amount plus the sum of transaction fees in the block[6]

It is possible for miners to create blocks simultaneously. In this case, we have a fork on the blockchain. This temporarily disrupts the agreement on the order blocks. Protocol rules dictate that miners should try to extend the branch they first heard. Eventually, a branch becomes longer than the others and when this happens, miners agree on this branch.

For a miner, the probability of creating a block is roughly proportional to his hashrate (number of hash calculations per second). Protocol adjusts the difficulty target such that a block is created roughly in every 10 minutes in the network, regardless of the total hashrate in it. The reason is to keep the probability of having forks small and consequently, maintaining the agreement on the blockchain.

Finally, the longest branch of the blockchain is called as the *main chain* and only the transactions contained in it are added to the balance of users. Blocks that do not reside in the main are called *orphan blocks*. The very first block is called as the *genesis block*.

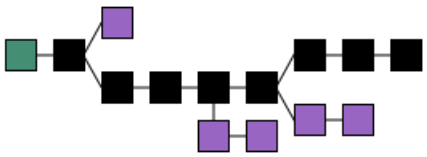


Figure 2.3: A complete view of the blockchain. *Genesis block* is in green. Nodes that are in the *main chain* are highlighted in black. *Orphan blocks* are in purple [19].

[5]Technically, they can set it to any previous block but that is the default behaviour.

[6]The "fixed" reward halves roughly in every 4 years. As of 2018, it is 12.5 BTC. When it runs out, miners will be rewarded only by transaction fees.

2.2 Fruitchain

Fruitchain slightly modifies Bitcoin by adding a by-product to the mining process called *fruit*. The main idea is to have two different difficulty targets for blocks and fruits such that fruits are created more often. By designing a new reward scheme that takes fruits into account, Fruitchain aims to reduce the expected time and variance of payouts and thus, reducing the incentive to join mining pools. In particular, through fruits, it aims to reward miners with low hashrates in a profitable manner.

2.2.1 Overview

Fruitchain mainly functions like Bitcoin. In this section, we explicitly mention the parts in which Fruitchain differs.

As in Bitcoin, miners try to find a solution to the PoW to create blocks. However, the mining process in Fruitchain might yield a by-product called *fruit*. A miner attempts to create a block and a fruit simultaneously by utilizing the 2-for-1 trick [20]. This means that, for example, the prefix of digest determines whether fruit mining is successful and the suffix of digest determines whether block mining is successful. Protocol fixes the ratio between the fruit difficulty target and the block difficulty target to a parameter $c_0 > 1$[7]. So, whenever the block difficulty target is adjusted, the fruit difficulty target is adjusted also to keep their ratio constant.

When a fruit is created, it is broadcast to the network. Just as transactions, they are included in blocks by miners.

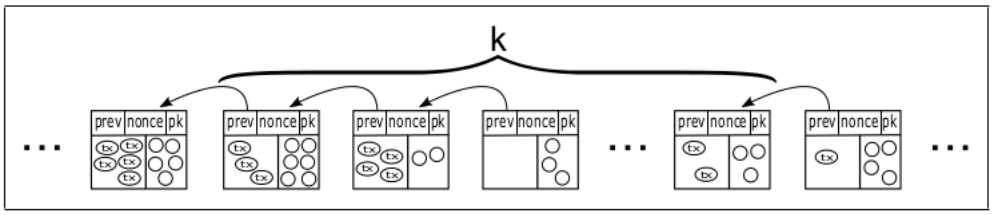


Figure 2.4: The Fruitchain. Empty circles denote fruits [6].

[7]Remember, the difficulty of mining is inversely proportional to difficulty target.

Fruits keep a block reference in their headers, like blocks, and the protocol dictates the following validation rules for a fruit.

(i) The number of blocks between the block containing the fruit and the one that fruit refers to is at most $k - 1$.

(ii) The digest of its header is lower than the fruit difficulty target when double hashed under SHA-256 cryptographic hash function.

The reason for the first point is to prevent fruit withholding for long intervals. A miner can hold his fruits until transaction fees rise above a certain level. Consequently, he might earn more reward per fruit than another miner who broadcasts his fruits often. This might affect the fair distribution of rewards (see Section 2.2.2 for the exact details of rewarding mechanism).

Further, the validation rules of a block (in addition to Bitcoin's validation rules) is as follows.

(i) The fruits that it contains are valid.

(ii) It does not contain any fruit that resides in previous blocks.

Header data of fruits and blocks contain an address field. This is necessary to reward miners under Fruitchain's reward scheme as we explain in Section 2.2.2.1. The details of the headers are given in Figure 2.5.

fruit fields	**bytes**
nonce	4
reward address	20
freshness	16
fruit tax	1
extended header	32
total	73

extended fields	**bytes**
version	4
timestamp	4
difficulty target	4
previous block	32
UTXO set	32
transaction tree	32
fruit hashchain	32
total	140

Figure 2.5: Headers in Fruitchain. Freshness field of a valid fruit refers to one of the k preceding blocks from the block containing the fruit. When a fruit is created, only the fruit header is broadcast. When a block is created, fruit header is serialized with the extended header and the combined data is broadcast [6].

2.2.2 Reward Scheme

The main idea of Fruitchain's reward scheme is to reward block creators in proportion to the number of fruits they collect in their blocks whilst also rewarding fruit creators. Technically, fruit creators get rewards from their easier PoW solutions. Thus, in a sense, fruits play the role of "shares" in mining pools under a decentralized setting [21].

Basically, protocol keeps a k-length sliding window over blocks $B_i, B_{i+1}, \ldots, B_{i+k-1}$. It then distributes the amount B_{i+k} has accumulated (in terms of transaction fees [8]) to the window after giving a portion of it directly to the creator of B_{i+k}. The exact details of the reward scheme is given in Figure 2.6. For clarity, consider $c_1 = \frac{1}{100}, c_2 = \frac{1}{10}, c_3 = \frac{1}{100}$ (see Section 2.2.2.6 for an explanation).

- Let x be the amount of transaction fees collected by the block B_{k+1}.
- Creator of B_{k+1} is directly rewarded by $c_1 x$. Reward pool of k-length window is initialized to $R = (1 - c_1)x$.
- Let t_i denote the number of fruits contained in block B_i. Every block contains an "implicit" fruit that rewards the block creator. (so, $t_i \geq 1$).
- Let $t = \sum_{i=1}^{k} t_i$ denote the total number of fruits in the window.
- Let $n_0 = R/t$ be the normal reward per fruit.
- Let $d(\ell) = c_3 \cdot (1 - \frac{l}{k-1})$ and let $\hat{c_2}(f)$ denote the tax fruit f specifies.
- For each $i \in \{1, 2, \ldots, k\}$:
  - For each fruit out of t_i fruits that B_i collected:
    - Let $l(\ell)$ be the amount of blocks between B_i and the block referenced by the fruit f, i.e. $0 \leq \ell(f) \leq k - 1$.
    - Creator of fruit f is rewarded with $n_0 \cdot (1 - \hat{c_2}(f) + d(\ell(f)))$
    - Creator of block B_i is rewarded with $n_0 \cdot (\hat{c_2}(f) - d(\ell(f)))$

Figure 2.6: Reward scheme of Fruitchain [6].

[8] Fruitchain considers a setting with no fixed block reward.

The parameters k, c_1, c_2, c_3 are respectively called *window length*, *direct reward proportion*, *fruit tax* and *fruit freshness bonus*. We now briefly analyze them and explain how coinbase transactions are generated.

2.2.2.1 Coinbase Transactions

The total reward of a fruit/block cannot be determined at the time of its creation as it depends on the window it resides. Due to this, coinbase transactions cannot be kept in fruits/blocks directly. Instead, when block B_{i+k} is created, nodes calculate rewards over the window according to the reward scheme. Then, coinbase transactions are generated and stored directly in the UTXO pool. The address of miners are derived from the header data of their blocks/fruits (see Figure 2.5).

2.2.2.2 Window Length

Window length parameter k effects how long a miner has to wait to get his total reward from blocks/fruits. Simply, the sliding window has to pass k times over a block for miners to receive the total reward from a block and the fruits contained in it. It also affects how should fruit tax be set. As authors show in [6] and we explain in Section 2.2.2.4, for small k values, fruit tax should be higher. Due to these two reasons, k should not be too high or too small.

2.2.2.3 Direct Reward Proportion

Direct reward proportion c_1 aims to encourage miners to validate and include transactions in their blocks. If miners do not receive a direct reward from their blocks, they might simply avoid the work of validating transactions and create fruits and empty blocks. Under high c_1 values, miners with low hashrates might not be able to sustain their business due to low fruit reward and under low c_1 values, miners might be discouraged to validate transactions.

2.2.2.4 Fruit Tax

Fruit tax c_2 aims encourage miners to put fruits of others in their blocks. Depending on its value, miners might include or exclude fruits of others. We illustrate this with a simple analysis originally given in [6].

Proposition 1. *Let total reward distributed over a window be R and let there be kc_0 fruits in the window. Let M be a miner who has a block in this window with c_0 fruits in it, mined by himself. Let there be x free fruits that were not included in the window. M would earn less reward by including these fruits in his block given that they pay the tax $c_2 < 1/k$ even if they do not claim any freshness bonus, i.e. $d(\ell(.)) = 0$.*

Proof. When M does not include these x fruits, we have $n_0 = R/kc_0$. When he includes them, we have $n_{0,x} = R/(kc_0 + x)$. We see that the reward M earns is given by $v = n_0 c_0$ in the first case. In the second case, this is $v_x = n_{0,x}(kc_0 + c_2 x)$. Setting $v = v_x$ and solving for c_2 gives us $c_2 = 1/k$ which implies for $c_2 < 1/k$ we have $v > v_x$. So, M gains more reward by excluding these fruits. ■

Miners are free to set their own fruit tax for their fruits by adjusting the tax field in the fruit header (see Figure 2.5).

2.2.2.5 Fruit Freshness

Validity rules on fruits do not allow prolonged fruit withholding however, a miner can still hold a fruit he creates for k blocks. If the miner can create one of these k blocks, he can avoid paying fruit tax to others by including his fruits in his block. This can cause reward distribution to be skewed in favor of miners with high hashrates.

To discourage this, fruits are given a freshness bonus, adjusted by parameter c_3. The closer the distance between the block containing the fruit and the block that fruit refers to, the higher the bonus. Thus, it can be seen that this encourages miners to broadcast their fruits as soon as possible to obtain more freshness bonus.

2.2.2.6 Adjusting the parameters

Adjusting parameters should be done with care as briefly explained in the previous sections. In their work [6], authors are particularly concerned with the case where rewards are earned solely through transaction fees. As shown in [11], Bitcoin instablizes in this setting due to an attack called "fee sniping". Authors tune the parameters such that Fruitchain is sufficiently resistant to this attack. We omit the details regarding the analysis as we are not particularly concerned with this attack in our work and just share the default values they suggest: $c_1 = \frac{1}{100}, c_2 = \frac{1}{10}, c_3 = \frac{1}{100}, k = 16$.

Chapter 3

Theoretical Analysis of the Reward Schemes

In this chapter, we analyze Fruitchain and Bitcoin in terms of their reward schemes.

We first introduce a simplified model of Fruitchain in Section 3.1. Using this model, we analyze earnings of miners according to both Bitcoin and Fruitchain. Simply, we ignore the fruits in blocks when calculating earnings under Bitcoin and allocate the whole amount contained in a block to its creator. For Fruitchain, we distribute rewards according to its reward scheme (see Figure 2.6).

After introducing our model, we provide a brief analysis of it in Section 3.2 and then analyze reward properties of Fruitchain, i.e. how much a miner earns from his fruits/blocks, in Section 3.3.

Finally, we explicitly define our properties, *fairness* and *profitability* and analyze Fruitchain's and Bitcoin's performance in terms of satisfying them in Section 3.4.

3.1 Execution Model

We consider a round based execution model[1] as in Garay et al. [20]. An execution is denoted by $\varepsilon(p, p_f, r, N)$ where p and p_f are two probability values such that $p_f > p$ (see 3.1 for their purpose), r is an integer that denotes the length of execution and N is the set of miners. We refer to N as the network and denote the ratio of p_f to p by c_0, i.e. $c_0 = \frac{p_f}{p}$.

We model a miner as $M = (ID, h)$ where ID $\in \{1, 2, \ldots, |N|\}$ denotes his unique identifier and $0 < h \leq 1$ denotes his hashrate fraction. We note that a miner cannot leave the network or change his hashrate during an execution. All miners share two data structures[2]: a list of blocks BC and a set of fruits FS.

We model the blocks as $b = (ID, FS_b)$ where ID is the unique identifier of the miner who created b and FS_b is the set of fruits that are contained in b. The height of a block is its position in BC (indexed by 1) and the length of BC is the number of blocks in it, denoted by $len(BC)$. The last block in BC is called as the head of BC.

Also, fruits are modeled as $f = (ID, B_h)$ where ID denotes the same as in blocks and B_h is the height of the block that f refers to.

At each round, miners are selected by a leader-selection entity S. Briefly, the flow of a round is as follows.

(i) A round starts by S running the algorithm given in Figure 3.1. S broadcasts the output of the algorithm to the network.

(ii) The miner identified by $FruitLeaderID \neq \perp$ creates a fruit $f = (FruitLeaderID, len(BC))$ and sets $FS = \{f\} \cup FS$.

(iii) The miner identified by $BlockLeaderID \neq \perp$ creates a block $b = (BlockLeaderID, FS)$ and sets $FS = \{\}$.

(iv) If neither a block nor a fruit is created, i.e. $BlockLeaderID = FruitLeaderID = \perp$ or just a fruit is created, round terminates.

(v) Otherwise, rewards are distributed according to both Bitcoin and Fruitchain reward schemes. That is, for Bitcoin, block creators get

[1] Technically, we consider a "round-based, synchronous, no-delay, reliable" model.

[2] Note that we do not mean replication here. The blockchain is really a shared data structure in our model which every miner can read and write.

the whole amount in their blocks and for Fruitchain, rewards are distributed according to its reward scheme. Each miner keeps track of how much reward he earns under both schemes simultaneously by maintaining two local variables, one for Bitcoin and one for Fruitchain. After distribution of rewards, round terminates.

We note that if both a fruit and a block is created at a round, first fruit set is updated with the newly created fruit and then the block is created.

We assume rewards are earned only through transaction fees and each block accumulates a constant amount x in terms fees. We further assume that every fruit pays the same tax c_2. Finally, note that freshness bonus of every fruit is c_3 as they always point to the head of BC (see Section 2.2.2 to recall parameters of Fruitchain).

procedure SELECT MINERS
  $BlockLeaderID \leftarrow \perp$
  $FruitLeaderID \leftarrow \perp$
  $r \leftarrow$ **Uniform**$[0,1), r_f \leftarrow$ **Uniform**$[0,1)$
  if $r < p$ **then**
    $BlockLeaderID \leftarrow$ **WeightedRandomSelect(N)**
  if $r_f < p_f$ **then**
    $FruitLeaderID \leftarrow$ **WeightedRandomSelect(N)**
  Return *BlockLeaderID, FruitLeaderID*

Figure 3.1: Miner selection algorithm of S. Algorithm uses two subprecudures, **Uniform** and **WeightedRandomSelect**. **Uniform** simply returns a number within the specified range with uniform probability. **WeightedRandomSelect** takes a miner set as its input, does a weighted random selection over it where weights are the hashrate fractions of miners and returns a miner ID.

3.2 Brief Analysis of the Model

As can be seen, mining is modeled as a Bernoulli process for each round. Informally, the network has a probability p of creating a block and has a probability p_f of creating a fruit at each round. When the block/fruit creation event ticks, its miner is selected by a weighted selection on the set

of miners where weights are the hashrate fractions. Also, note that p and p_f are independent of miners individual hashrates. This is to capture the difficulty adjustment of the mining process, i.e. blocks are created roughly in every 10 minutes in the Bitcoin network regardless of the total hashrate. Similarly, in our model, blocks are created roughly in every $\frac{1}{p}$ rounds and the ratio of $\frac{p_f}{p}$ stays constant as this is how mining in Fruitchain is designed (see Section 2.2.1).

Let M be a miner with hashrate fraction h_M. Then, his probability of mining a block at a round is given by ph_M and his probability of mining a fruit is given by $p_f h_M$. Further, over the rounds, mining is a Binomial process. For an execution that takes r rounds, the expected number of blocks and fruits created by M is simply given by $ph_M r$ and $p_f h_M r$, respectively. For example, under Bitcoin, this means M's expected gain is $ph_M rx$ for this execution as we assume every block contains an amount of x in transaction fees and miners get all the fees from the blocks they create under Bitcoin. Finally, note that the expected total gain of the network is prx for both Bitcoin and Fruitchain for this execution.

We also note that if we are interested in finding the elapsed number of rounds between blocks/fruits of M, we can reason to Geometric distribution. So, if R is the elapsed number rounds between blocks of M, we have $R \sim \text{Geom}(ph_M)$. This means, in expectation, the number of rounds between every two block of M is given by $\mathrm{E}[R] = 1/ph_M$.

3.3 Reward Properties of Fruitchain

Before analyzing the reward schemes in terms of *fairness* and *profitability*, we need to derive formulas for reward properties of Fruitchain, i.e. how much reward a block/fruit brings to its creator. We make use of the following theorem in our analysis.

Theorem 1. *Wald's Identity. Let N be a random variable assuming positive integer values. Let X_i be a sequence of i.i.d random variables with $\mathrm{E}[X_i] = \mathrm{E}[X]$. Then,*

$$\mathrm{E}[\sum_{i=1}^{N} X_i] = \mathrm{E}[N] \cdot \mathrm{E}[X].$$

First, we show how many fruits a block contains in expectation to find the

expected normal reward per fruit (n_0 from Figure 2.6).

Lemma 1. *Let N be the number of fruits in a block. Then we have,* $\mathrm{E}[N] = c_0$.

Proof. Let F_i be the number of fruits created in round i. Let R be the elapsed number between the head of BC at round i and the next block. We note F_i's are i.i.d. random variables with $F \sim \mathrm{Bern}\left(p_f\right)$ and $R \sim \mathrm{Geom}\left(p\right)$. If we enumerate rounds between subsequent blocks from 1 to R, we can write $N = \sum_{i=1}^{R} F_i$. Using Theorem 1 gives us $\mathrm{E}[N] = \mathrm{E}[R] \cdot \mathrm{E}[F]$. Finally, plugging $\mathrm{E}[R] = \frac{1}{p}$ and $\mathrm{E}[F] = p_f$ yields the result. ■

Corollary 1. *Let N_0 denote the normal reward per fruit which is defined as,*

$$N_0 = \frac{\textit{Reward distributed over a window}}{\textit{Number of fruits in a window}}.$$

Then we have,

$$\mathrm{E}[N_0] = \frac{(1 - c_1) \cdot x}{k \cdot (\mathrm{E}[N] + 1)}.$$

Proof. Reward distributed over every window is simply $(1 - c_1)x$. Since Fruitchain scheme assumes an implicit fruit in each block, we have $k \cdot (\mathrm{E}[N] + 1)$ fruits in every window[3]. Applying the definition of N_0 yields the result. ■

We can now derive formulas for reward properties.

Lemma 2. *Let R_f denote the reward of mining a fruit. Then we have,*

$$\mathrm{E}[R_f] = \frac{(1 - c_1)x \cdot (1 - c_2 + c_3)}{\mathrm{E}[N] + 1}.$$

Proof. Since fruits always point to the head of BC, we have $d(\ell(f)) = c_3$ for each fruit f. A fruit gets window reward for k times and in each time, it is rewarded by $\mathrm{E}[N_0] \cdot (1 - c_2 + c_3)$. Its total reward is then given by $\mathrm{E}[R_f] = k \cdot \mathrm{E}[N_0] \cdot (1 - c_2 + c_3)$. Plugging the value of $\mathrm{E}[N_0]$ yields the result. ■

Lemma 3. *Let R_b denote the reward of mining a block. Then we have,*

$$\mathrm{E}[R_b] = c_1 x + \frac{(1 - c_1)x}{\mathrm{E}[N] + 1} \cdot \Big(1 + \mathrm{E}[N](c_2 - c_3)\Big).$$

[3]Of course, this is in expectation.

Proof. A block gets the direct reward once and the window reward k times. Direct reward is simply $c_1 x$. Block gains $\mathrm{E}[N_0]$ for his implicit fruit and gains $\mathrm{E}[N_0] \cdot (c_2 - c_3)$ for each other fruit that it contains. There are $\mathrm{E}[N]$ such fruits and by using Theorem 1, we can see the block gains,

$$\mathrm{E}[N_0] + \mathrm{E}[N] \cdot \mathrm{E}[N_0] \cdot (c_2 - c_3),$$

for each window reward. Thus, in total we have,

$$E[R_b] = c_1 x + k\Big(E[N_0] + E[N] \cdot E[N_0] \cdot (c_2 - c_3)\Big).$$

Plugging the value of $E[N_0]$ and simplifying the terms yields the result. ■

3.4 Fairness and Profitability

In this section, we introduce our definitions of *fairness* and *profitability*. We then compare Fruitchain and Bitcoin with respect to them.

3.4.1 Fairness

Fairness is the distribution of rewards in proportion to hashrate of miners. It is an important property that a reward scheme should satisfy in order to promote participation in the mining process. Formally, we define fairness as follows.

Definition 1. *Fairness of a reward scheme. Let M be a miner in an execution with hashrate fraction h_M. Let R_M denote the reward M gains and let R_N denote the reward network gains at the end of this execution. We call a reward scheme fair if for every miner M, we have $\frac{E[R_M]}{E[R_N]} = h_M$.*

We now show Bitcoin and Fruitchain reward schemes are *fair*. We adopt the notation from Definition 1 in our proofs.

Lemma 4. *Bitcoin reward scheme is fair.*

Proof. Let M be an arbitrary miner in an execution that takes r rounds. Let B denote the number of blocks he creates. Since M creates a block at a round with probability ph_M, we have $\mathrm{E}[B] = ph_M r$. Then, the expected total reward of M is given by $\mathrm{E}[R_M] = E[B] \cdot x = ph_M rx$. Finally, the expected total reward of the network is simply $\mathrm{E}[R_N] = prx$. Thus, $\frac{\mathrm{E}[R_M]}{\mathrm{E}[R_N]} = h_M$. ■

Lemma 5. *Fruitchain reward scheme is fair.*

Proof. Let M be an arbitrary miner in an execution that takes r rounds. Let B denote the number of blocks and let F denote the number of fruits M creates. Then, $R_M = \sum_{i=1}^{B} R_{b_i} + \sum_{i=1}^{F} R_{f_i}$ where R_{b_i} is the reward of i'th block and R_{f_i} is the reward of i'th fruit that M creates. Applying Theorem 1 gives us $E[R_M] = E[B] \cdot E[R_b] + E[F] \cdot E[R_f]$. Since M creates a block with probability ph_M and creates a fruit with probability $p_f h_M$ in each round, we have $E[B] = ph_M r$ and $E[F] = p_f h_M r$. Plugging the values of $E[R_b]$ and $E[R_f]$ from Lemmas 2 and 3 and simplifying the terms yields

$E[R_M] = ph_M rx$ [4]. Finally, the expected total reward of the network is simply $E[R_N] = prx$. Thus, $\frac{E[R_M]}{E[R_N]} = h_M$. ■

3.4.2 Profitability

Miners should make regular payments to keep their businesses running (e.g. electric bills). This means they must receive their rewards regularly. Due to this, we believe profitability is an important property that a reward scheme should satify. Formally, we define profitability of a miner as follows.

Definition 2. *Profitability of a miner. Let M be a miner in an execution. Let R_m denote his reward at the end of this execution and let T_m denote his average reward gap*[5] *during this execution. Then, we define profitability of M for this execution as* $\frac{\mathrm{E}[R_m]}{\mathrm{E}[T_m]}$.

To compare Bitcoin and Fruitchain with respect to profitability, we first deduce the expected average reward gap of a miner under them.

Lemma 6. *Let M be a miner in an execution. Then under Bitcoin reward scheme, we have* $\mathrm{E}[T_m] = \frac{1}{ph_M}$.

Proof. Let p_{avg} denote the average probability of getting a reward at a round for miner M. Then we have $T_m \sim \text{Geom}\left(p_{avg}\right)$. For every round, probability of getting a reward is equal to probability of mining a block under Bitcoin. Thus, $p_{avg} = ph_M$. Using the expected value of geometric distribution gives us $\mathrm{E}[T_m] = 1/ph_M$. ■

Lemma 7. *Let M be a miner in an execution. Then under Fruitchain reward scheme, we have*

$$\mathrm{E}[T_m] = \frac{1}{p(1-(1-h_M)^{k+1+c_0 k})}.$$

Proof. Let p_{avg} denote the average probability of getting a reward at a round for miner M. Then we have $T_m \sim \text{Geom}\left(p_{avg}\right)$.

[4] We give this simplification in the Appendix.

[5] Number of rounds between two instants in which a miner gains his rewards. For example, if a miner gains a reward at rounds 2 ands 4, his reward gap for this interval is given by 4-2+1 = 3 (4 is inclusive). If a miner does not gain any reward during an execution, his reward gap is equal to the length of this execution.

Under Fruitchain, M does not earn a reward at a round in two cases: either nobody creates a block or another miner than M creates a block and M does not have any fruit or block in the window. The probability of the first case is $1-p$ for all rounds. The probability of the second case is $p(1-h_M)^{k+1+c_0k}$ on average due to the multiplication of,

i. Probability[6] of a miner other than M creates a block, $p(1-h_M)$,

ii. Average probability of M not having any block in the window, $(1-h_M)^k$,

iii. Average probability of M not having any fruit in the window, $(1-h_M)^{c_0k}$.

Thus we see that,

$$\begin{aligned} p_{avg} &= 1-\big(1-p+p(1-h_M)^{k+1+c_0k}\big) \\ &= p(1-\big(1-h_M\big)^{k+1+c_0k}\big). \end{aligned}$$

Finally, the expected value of geometric distribution gives us,

$$\mathrm{E}[T_m] = \frac{1}{p(1-(1-h_M)^{k+1+c_0k})}.$$

■

We can now show Fruitchain is more profitable than Bitcoin for any miner M with hashrate fraction $0 < h_M < 1$. For $h_M = 1$, it is trivial to see both schemes are equivalent as M creates all the blocks/fruits.

Lemma 8. *Let M be an arbitrary miner in an execution. Then, Fruitchain is more profitable than Bitcoin for M. That is,*

$$\frac{\mathrm{E}[R_{BTC_M}]}{\mathrm{E}[T_{BTC_M}]} < \frac{\mathrm{E}[R_{FTC_M}]}{\mathrm{E}[T_{FTC_M}]},$$

where R_{BTC_M} is the total reward and T_{BTC_M} is the average reward gap of M under Bitcoin (R_{FTC_M} and T_{FTC_M} denote the same for Fruitchain, respectively).

Proof. Due to Lemmas 4 and 5, we have $\mathrm{E}[R_{FTC_M}] = \mathrm{E}[R_{BTC_M}]$. Thus proving the statement reduces down to showing,

$$ph_M < p(1-(1-h_m)^{k+1+c_0k}.$$

[6]This is same for all rounds, thus equal to its average

By canceling p and rewriting the inequality we obtain,

$$(1 - h_M)^{1+k+c_0 k} < 1 - h_M,$$

which holds for any $k \geq 1, c_0 \geq 1$ as $0 < h_M < 1$. ■

3.5 Extending the Results to a Dynamic Model

We like to note that all of our results obtained in Sections 3.3, 3.4.1 and 3.4.2 hold under a dynamic version of our model that allows miners to change their hashrates and that allows miners to join/leave between rounds. To simulate join/leave, we consider a miner's hashrate as 0 for rounds he is not present.

Results from Section 3.3 are already valid under the dynamic model as they only depend on execution parameters p and p_f which do not vary with individual miners.

To see why our results from Sections 3.4.1 and 3.4.2 hold, consider an execution in which a miner M changes his hashrate between rounds such that his average hashrate fraction is h_M. We can see this is equivalent to an execution where M's hashrate fraction is h_M for all rounds in terms of the expected number of blocks/fruits that M creates. This is simply due to the way mining is modeled, i.e. each round is independent.

This immediately shows our results on fairness hold as they only depend on our results from Section 3.3 and the expected number of fruits/blocks. For profitability, note that we can compute the expected average reward gap of M under Bitcoin as the ratio between the length of execution and the expected number of blocks created by M. We can apply the same reasoning to Fruitchain to derive probabilities we use in our proof of Lemma 7.

Chapter 4

Experimental Analysis of the Reward Schemes

In this chapter, we aim to verify the correctness of our results from Chapter 3 by doing measurements with a simulator. We also try to illustrate their implications on miners through simulations. In addition to that, we briefly compare Fruitchain with Bitcoin under the *selfish-mining attack*.

4.1 Simulator

We implemented a simulator of our execution model from Section 3.1 using Python 3.5.2 [22]. The parameters of the simulator are as follows.

- p: Probability of network mining a block at a round.
- p_f: Probability of network mining a fruit at a round.
- r: Number of rounds.
- n: Number of miners.
- h: List of hashrate fractions such that h_i is the hashrate fraction of miner i.

During the simulations, we simultaneously record how much reward each miner earns under Bitcoin and Fruitchain as we explain in Section 3.1. For Fruitchain, we use the default parameters, i.e. $c_1 = \frac{1}{100}, c_2 = \frac{1}{10}, c_3 =$

$\frac{1}{100}, k = 16$. Further, we assume every block has accumulated an amount of 12.5 BTC in transactions fees and ignore the fixed reward as Fruitchain only takes fees into account.

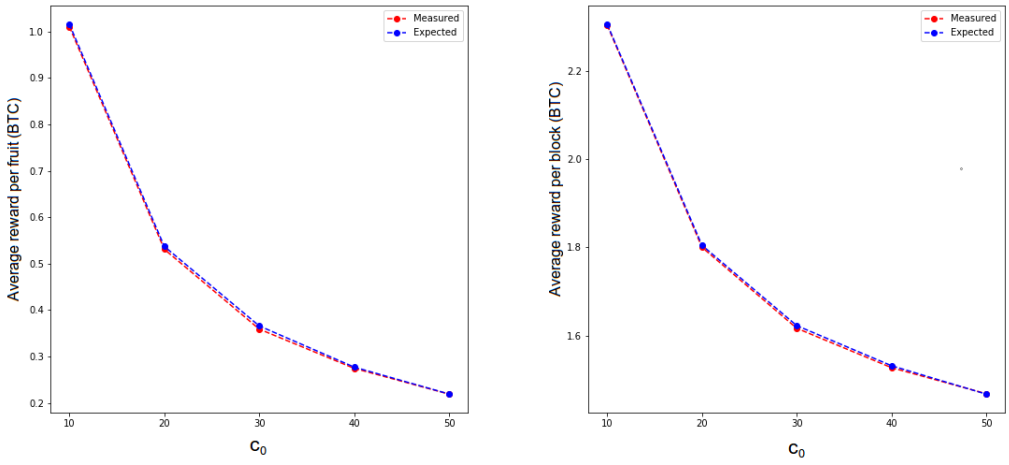
4.2 Correctness of Theoretical Results

We briefly show our measurements match with their expected values from Chapter 3.

4.2.1 Reward Properties of Fruitchain

We recall our formulas from Section 3.3 depend only on $c_0 = \frac{p_f}{p}$. So, we plot them against different c_0 values.

The plots in this section are obtained under the following settings: we fix $r = 10^5$, $n = 1$ and $h_1 = 1$ and vary c_0 from 10 to 50 by incrementing it 10 between simulations. To obtain different c_0 values, we arbitrarily fix $p_f = 0.5$ and adjust p accordingly.



(a) Average reward per fruit (Lemma 2).

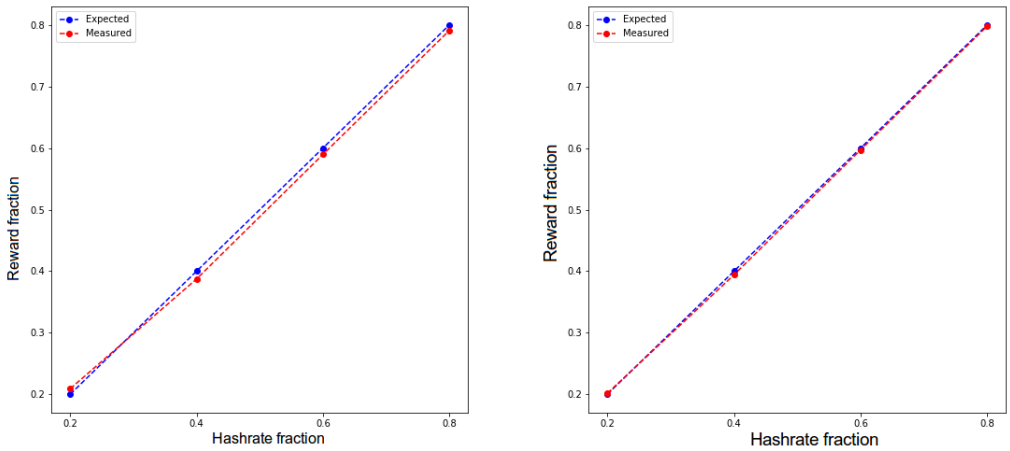
(b) Average reward per block (Lemma 3).

Figure 4.1: Reward properties of Fruitchain.

4.2.2 Fairness

In section 3.4.1, we proved a miner's reward fraction is equal to his hashrate fraction in expectation for both Bitcoin and Fruitchain. So, we plot reward fraction for different hashrate fractions.

The plots in this section are obtained under the following settings: we fix $r = 10^5, p = 0.01, p_f = 1, n = 2$ and vary h_2 from 0.2 to 0.8 by incrementing it 0.2 between simulations. We do our measurement for miner 2.



(a) Fairness of Bitcoin (Lemma 4).

(b) Fairness of Fruitchain (Lemma 5).

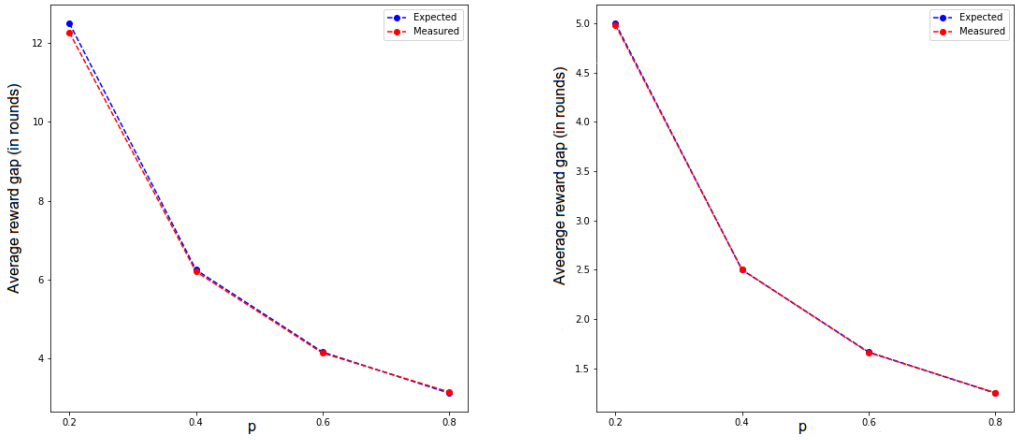
Figure 4.2: Fairness of Bitcoin and Fruitchain.

4.2.3 Profitability

Recall from Section 3.4.2, it is sufficient to analyze average reward gap to analyze profitability. Average reward gap of a miner under both Bitcoin and Fruitchain depends on p and the respective hashrate fraction of the miner. We arbitrarily decided to fix the hashrate of the miner and plot average reward gap for different p values.

The plots in this section are obtained under the following settings: we fix $r = 10^5, p_f = 1, n = 2, h_2 = 0.4$ and vary p from 0.2 to 0.8 by incrementing

it 0.2 between simulations. We do our measurement for miner 2.



(a) Reward gap under Bitcoin (Lemma 6).

(b) Reward gap under Fruitchain (Lemma 7).

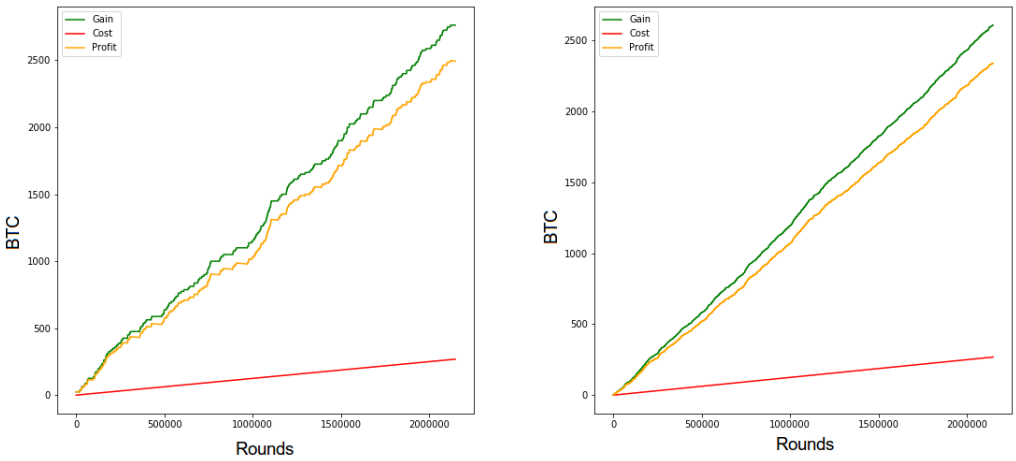
Figure 4.3: Reward gap under Bitcoin and Fruitchain.

4.3 Illustrating the Effect of Profitability

We now attempt to highlight the difference between Bitcoin and Fruitchain by explicitly showing a miner's profit under them. Basically, we wonder for what hashrate fractions profitability of Fruitchain becomes visible. To this end, we plot profit of a miner against his hashrate fraction for both Bitcoin and Fruitchain.

The plots in this section are obtained under the following settings: we fix $p = 0.001, p_f = 1, n = 2$ and adjust the running time of each simulation such that it corresponds to roughly 15 days, i.e. $r = \frac{1}{p} \cdot 144 \cdot 15 = 2.16 \cdot 10^6$ [1]. The cost of mining per round is arbitrarily set as 10% of the expected gain per round. We do 3 simulations for $h_2 = 0.1, 0.01$ and 0.001 and we do our measurements for miner 2.

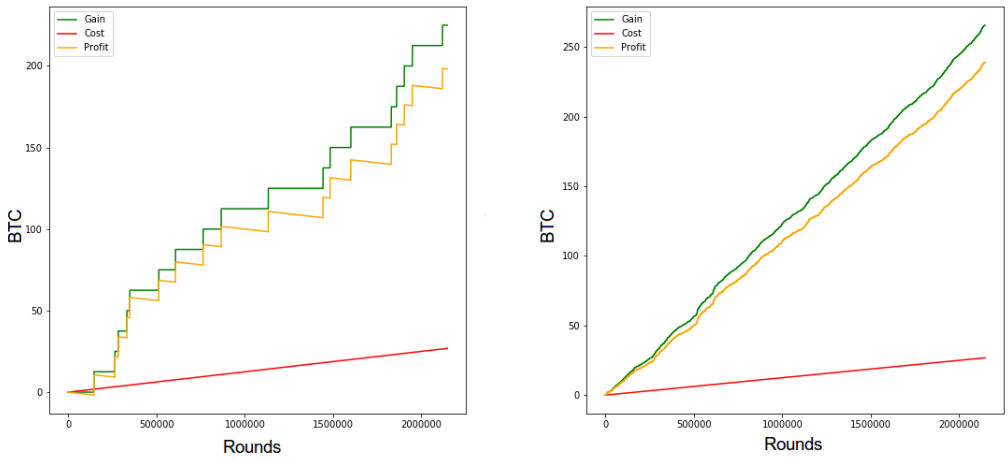
[1] Roughly, 144 blocks are created in each day in the Bitcoin network [23].



(a) Profit under Bitcoin.

(b) Profit under Fruitchain.

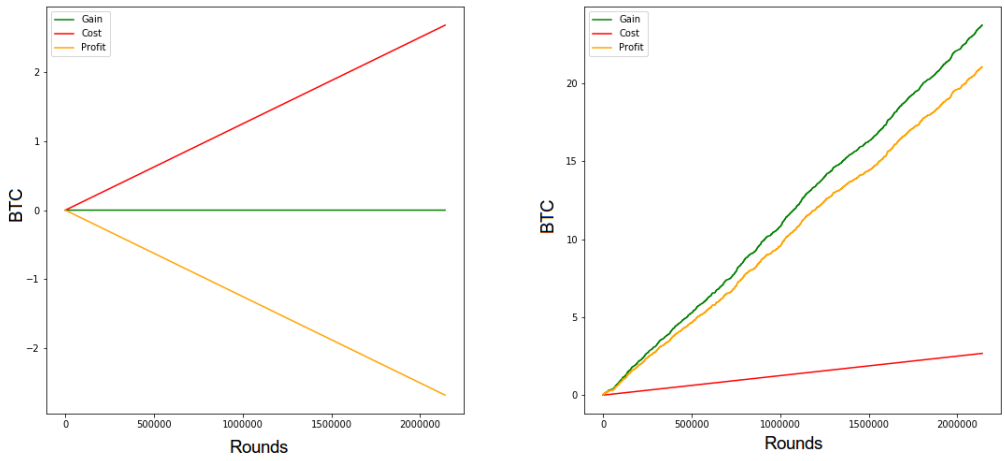
Figure 4.4: Profit under Bitcoin and Fruitchain with $h_2 = 0.1$.



(a) Profit under Bitcoin.

(b) Profit under Fruitchain.

Figure 4.5: Profit under Bitcoin and Fruitchain with $h_2 = 0.01$.



(a) Profit under Bitcoin.

(b) Profit under Fruitchain.

Figure 4.6: Profit under Bitcoin and Fruitchain with $h_2 = 0.001$.

We see that Bitcoin and Fruitchain are nearly equivalent in Figure 4.4. At the end of the simulation, miner's profit is roughly 2500 BTC for both under Bitcoin and Fruitchain.

In Figure 4.5, miner is better under Fruitchain. His profit is about 225 BTC under Fruitchain and slightly below 200 under Bitcoin. Note that graphs are scaled differently.

Finally, the advantage of Fruitchain is apparent in Figure 4.6. Under Bitcoin, miner's profit is negative as he was not able to create any block. However, under Fruitchain, his profit is positive due to rewards he gained from his fruits.

Our experiments show that a miner's profit is roughly the same for Bitcoin and Fruitchain if he has a relatively large portion of the hashrate (≈ 0.1). However, if he happens to have relatively a small portion of the hashrate (≈ 0.01), he clearly profits more under Fruitchain. The main reason is, due to low reward gap, rewards under Fruitchain converge faster to their expected values (recall that the expected reward is the same for Bitcoin and Fruitchain due to our results on fairness).

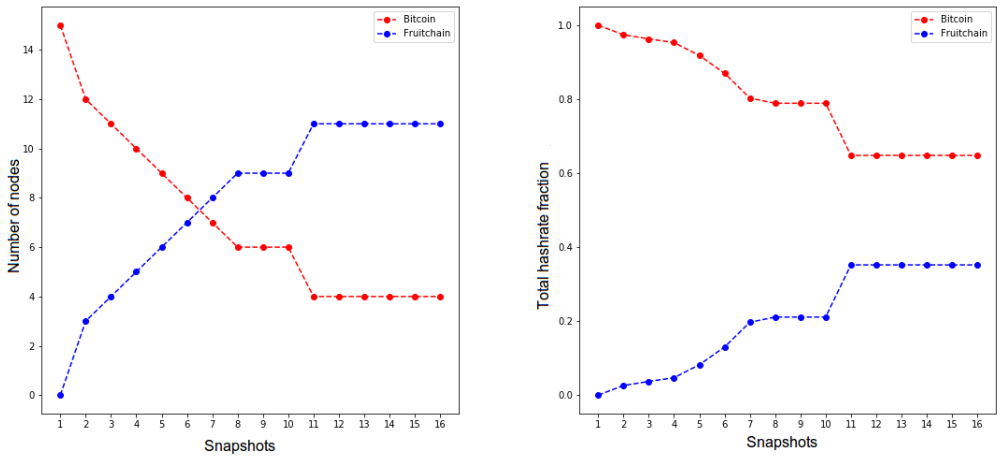
4.4 Reward Scheme Switching Simulation

We now try to illustrate the effect we captured in Section 4.3 on a network scale. To this end, we conduct a simple simulation.

Briefly, our simulation is as follows: we divide r rounds into t length intervals. For each interval, miners have a preferred reward scheme. During an interval, miners gain their rewards according to their preferred scheme. At the end of an interval, each miner compares his earnings with its expected value. If his earnings are less than x% of its expected for this interval, he switches his preferred reward scheme. Each interval is independent of the other, i.e. we reset earnings of miners between intervals.

We conduct our simulation with the following parameters. We have $n = 15$ miners such that their hashrates are distributed according to the Bitcoin network as of 21/02/2018 [24]. Preferred reward scheme of every miner is initialized to Bitcoin's. Other parameters are $t = 125 \cdot 10^3$, $r = 2 \cdot 10^6$, $x = 85$, $p = 0.001$, $p_f = 1$. We picked these values after experimenting with different settings and obtaining a stable state under those, i.e. nodes stop switching their reward schemes after some point.

During the simulation, we take a "snapshot" after each interval. That is, we record the number of nodes and their total hashrate fractions for both reward schemes. Resulting plots from our simulation are given in Figure 4.7.



(a) Change in the number of nodes.

(b) Change in the hashrate fractions.

Figure 4.7: Reward scheme switching experiment.

We see that miners keep switching rewarding schemes until 11th snapshot then, the network stabilizes. In the end, we have 11 miners that use Fruitchain reward scheme and 4 that use Bitcoin reward scheme. However, the total hashrate fraction of nodes that use Bitcoin's is about 0.6. This means miners with low hashrate are concentrated under Fruitchain's reward scheme. This is in-line with our simulations from Section 4.3. That is, miners with high hashrates do well under Bitcoin and however, Fruitchain is a much better option for miners with low hashrates.

4.5 Selfish-Mining

In addition to our analysis of profitability, we also provide a brief analysis of the *selfish-mining attack* [9] with randomized chain selection rule [2]. A theoretical analysis of this attack for Fruitchain is given in [6] and we merely confirm authors' results through our simulations.

We first recall the selfish-mining as summarized in [6]. For simplicity,

[2]Upon a fork, miners select a branch randomly.

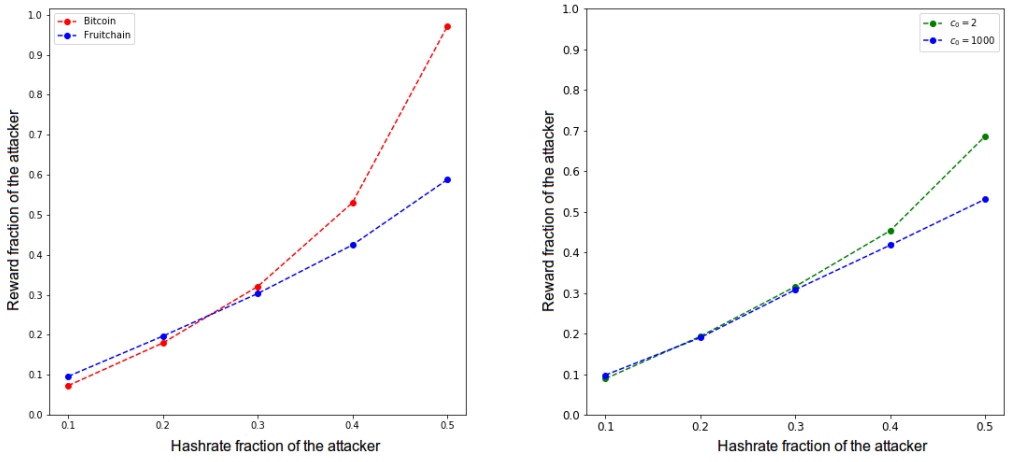
assume we have 2 miners: an attacker and an honest miner. The attacker attempts to build a secret chain that has d more blocks than the public (honest) chain and,

- if $d = 1$ and the honest node finds a block, the attacker broadcasts his secret block,
- if $d > 1$, the attacker tries to extend his secret chain until $d = 1$. He broadcasts his secret blocks once $d = 1$.

The analysis in [9] shows that the attacker creates 38.4% of blocks with 33% of the hashrate with selfish-mining.

In the context of Fruitchain, the attacker always broadcast his fruits and follows the selfish-mining strategy for broadcasting his blocks. In our simulations, we assume fruits of the attacker bring the maximal freshness bonus and fruits of the honest miner does not bring any freshness bonus to maximize the effect of selfish-mining in favor of the attacker.

We run a simulation with the following setting: we fix $r = 10^5, p = 0.001, p_f = 1, n = 2$ and vary h_2 from 0.1 to 0.5 by incrementing it 0.1 between simulations. We record earnings of miner 2 who mines according to selfish-mining.



(a) Bitcoin vs. Fruitchain under selfish-mining. (b) Effect of c_0 to selfish-mining under Fruitchain.

Figure 4.8: Selfish Mining.

As can be seen in Figure 4.8 (a), Fruitchain is much more resistant to selfish-mining. Intuitively, the reason is this: in Bitcoin, attacker's advantage in blocks directly transfer to an advantage in rewards. However, in Fruitchain, this is not the case. Rewards are distributed over both blocks and fruits. Fruits of the honest miner are invalid under the secret chain, so the attacker cannot include these fruits in his blocks and as a consequence, his blocks worth less than the honest miner's blocks as the worth of a block is roughly proportional to the number of fruits contained in it.

We also plot earnings of the attacker under Fruitchain for different c_0 values in Figure 4.8 (b). Rewards concentrate on blocks as c_0 gets smaller which causes the advantage in the number of blocks to transfer to an advantage in rewards. This increases the rewards of the attacker as can be seen.

Chapter 5

Conclusions

The main theme of our work was capturing and highlighting the differences between rewarding mechanisms of Bitcoin and Fruitchain. To this end, we first introduced a simplified execution model that solely focuses on the rewarding mechanisms of the protocols. Then, we explicitly defined two important properties for miners: *fairness* and *profitability*. Following that, we have analyzed Bitcoin's and Fruitchain's performance in terms of satisfying these properties. Through our analysis, we have proved Fruitchain is more profitable than Bitcoin and they are equivalent in terms of fairness, that is, according to our definitions.

After establishing our results, we showed their implications for miners through simulations. We showed the profitability of Fruitchain is especially important to miners with small hashrates by comparing their profits under Bitcoin and Fruitchain. Concretely, we showed miners were able to earn profits in a steady manner under Fruitchain. This confirms that computing power is less likely to be centralized under Fruitchain. Lastly, we also did a brief experimental analysis of the protocols under the *selfish mining attack* and illustrated that attackers gain considerably less in Fruitchain than in Bitcoin.

Appendix A

Simplifying terms in the proof of Lemma 5

We adopt the following notation:
$a = (1 - c_1)x$,
$b = c_2 - c_3$,
$t = E[N + 1] = \frac{p_f + p}{p}$.

We like to show the following,

$$\begin{aligned} ph_M rx &\stackrel{?}{=} E[F] \cdot E[R_f] + E[B] \cdot E[R_b] \\ &= p_f h_M r \cdot E[R_f] + ph_M r \cdot E[R_b] \\ px &\stackrel{?}{=} p_f \cdot E[R_f] + p \cdot E[R_b]. \end{aligned}$$

We first deal with the term $p_f \cdot E[R_f]$,

$$\begin{aligned} p_f \cdot E[R_f] &= p_f (\frac{a(1-b)}{t}) \\ &= \frac{p_f a - p_f ab}{t}. \end{aligned}$$

Now with the term $p \cdot E[R_b]$,

$$\begin{aligned} p \cdot E[R_b] &= p\big(x - a + \frac{a(1+(t-1)b}{t}\big) \\ &= p\big(\frac{xt - at + a + atb - ab}{t}\big) \\ &= \frac{(p_f + p)x - (p_f + p)a + pa + (p_f + p)ab - pab}{t} \\ &= \frac{(p_f + p)x - p_f a + p_f ab}{t}. \end{aligned}$$

Finally, we complete the simplification as follows,

$$\begin{aligned} px &\stackrel{?}{=} p_f \cdot E[R_f] + p \cdot E[R_b] \\ &= \frac{p_f a - p_f ab + (p_f + p)x - p_f a + p_f ab}{t} \\ &= \frac{(p_f + p)x}{t} \\ &= px. \end{aligned}$$

Bibliography

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. https://bitcoin.org/bitcoin.pdf.

[2] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. Cryptology ePrint Archive, Report 2016/916, 2016. http://eprint.iacr.org/2016/916.pdf.

[3] Mining pools. https://en.bitcoin.it/wiki/Pooled_mining. Accessed: 2018-02-27.

[4] Majority attack. https://en.bitcoin.it/wiki/Majority_attack. Accessed: 2018-02-27.

[5] Bitcoin hashrate distribution. https://blockchain.info/pools. Accessed: 2018-02-27.

[6] Iddo Bentov, Yuncong Hu, Rafael Pass, Elaine Shi, and Siqiu Yao. Decentralized pooled mining: An implementation of fruitchain.

[7] P2pool. http://p2pool.org/. Accessed: 2018-03-03.

[8] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S. Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 919–927, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[9] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[10] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[11] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167. ACM, 2016.

[12] Double-spending attack. `https://bitcoin.stackexchange.com/questions/4974/what-is-a-double-spend`. Accessed: 2018-02-27.

[13] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. *Communications and Multimedia Security*, pages 258–272, 1999.

[14] The protocol rules of bitcoin. `https://en.bitcoin.it/wiki/Protocol_rules`. Accessed: 2018-03-14.

[15] The utxo pool. `https://bitcoin.stackexchange.com/questions/37397/where-is-the-utxo-data-stored`. Accessed: 2018-03-14.

[16] Bitcoins the hard way: Using the raw bitcoin protocol. `http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html`. Accessed: 2018-03-02.

[17] Bitcoin developer guide. `https://bitcoin.org/en/developer-guide#block-chain`. Accessed: 2018-03-03.

[18] Ralph C. Merkle. *A Digital Signature Based on a Conventional Encryption Function*, pages 369–378. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988.

[19] Blockchain figure. `https://en.wikipedia.org/wiki/Blockchain`. Accessed: 2018-02-27.

[20] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. *The Bitcoin Backbone Protocol: Analysis and Applications*, pages 281–310. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[21] Shares in mining pools. `https://bitcoin.stackexchange.com/questions/1505/what-is-a-share-can-i-find-it-while-mining-solo-or-only-when-pool-mining`. Accessed: 2018-03-13.

[22] Python 3.5.2. https://www.python.org/download/releases/3.5.2/. Accessed: 2018-02-25.

[23] Confirmation times. https://en.bitcoin.it/wiki/Confirmation. Accessed: 2018-03-08.

[24] Hashrate distribution. https://blockchain.info/pools. Accessed: 2018-02-21.