information

MDPI

Article

# A Computational Study on Fairness of the Tendermint Blockchain Protocol

**Nicolas Lagaillardie †, Mohamed Aimen Djari and Önder Gürcan ***

CEA, LIST, Point Courrier 174, F-91191 Gif-sur-Yvette, France; nicolas.lagaillardie@etu.ese.fr (N.L.); Mohamed-Aimen.DJARI@ca.fr (M.A.D.)

* Correspondence: onder.gurcan@cea.fr; Tel.: +33-1-69-08-00-07

† Current address: Imperial College London, London SW7 2AZ, UK.

Received: 1 November 2019; Accepted: 25 November 2019; Published: 30 November 2019


**Abstract:** Fairness is a crucial property for blockchain systems since it affects the participation: the ones that find the system fair tend to stay or enter, the ones that find the system unfair tend to leave. While current literature mainly focuses on fairness for Bitcoin-like blockchains, little has been done to analyze Tendermint. Tendermint is a blockchain technology that uses a committee-based consensus algorithm, which finds an agreement among a set of block creators (called validators), even if some are malicious. Validators are regularly selected to the committee based on their investments. When a validator does not have enough asset to invest, it can increase it with the help of participants that delegate their assets to the validators (called delegators). In this paper, we implement the default Tendermint model and a Tendermint model for fairness in a multi-agent blockchain simulator where participants are modeled as rational agents who enter or leave the system based on their utility values. We conducted experiments for both models where agents have different investment strategies and with various numbers of delegators. In the light of our experimental evaluation, we observed that while, for both models, the fairness decreases and the system shrinks in the absence of delegators, the fairness increases, and the system expands for the second model in the presence of delegators.

**Keywords:** blockchain; tendermint; fairness; simulation

## 1. Introduction

Bitcoin, introduced by Satoshi Nakamoto [1], is the core of blockchain systems—decentralized transactional systems that are based on blockchain protocols. Participants following blockchain protocols can create together a distributed, economical, social and technical system where anyone can join and leave and perform transactions in between without either needing to trust each other or having a trusted third party. It is a very attractive technology since it maintains a *public*, *immutable* and *ordered* log of transactions that guarantees an *auditable* ledger accessible by anyone. The security and sustainability of blockchains, however, are not trivial and require increased participation, since each participant validates the diffused data, and keeps a replica of the entire blockchain. Participants consider it worthwhile to join and stay in the system over time if they find it fair.

Generally speaking, a blockchain system is an open and distributed system composed of participants called users and block creators. Users create transactions and broadcast across the blockchain network for being confirmed. Note that, due to the characteristics of the network, every participant will receive in different orders and also some transactions might be lost during communication. In this sense, block creators are responsible for ordering transactions totally as blocks for appending to the blockchain in an immutable manner. Block creators (based on the consensus algorithm used and the blockchain technology, block creators are named as miners [1], validators Ref [2], bakers [3], committee members [4,5], and so on, respectively.) agree on how and who to create a block

(consensus) either by themselves (Proof-of-Work [6], Proof-of-Stake, Delegated Proof-of-Stake, etc.) or by a committee (Practical Byzantine Fault-Tolerance [7], etc.)—see [8] for a review. Blockchain systems are usually classified with respect to their consensus algorithms as *Bitcoin-like blockchains* (Since Bitcoin came up by Proof-of-Work, blockchains using Proof-of-*something* algorithms are classified as Bitcoin-like.) (e.g., Bitcoin [1], Ethereum [2]), *committee-based blockchains* (e.g., Tendermint [4], RedBelly Ref [9]) and *hybrid blockchains* (e.g., PeerCensus [10], Byzcoin [11]). Considering Bitcoin-like blockchains, several studies have been conducted so far analyzing and improving fairness both for block creators Ref [12–20] and for users [21–23]. However, fairness in committee-based blockchains has not been studied much yet.

Based on this observation, in this paper, we study the fairness of the committee-based blockchain Tendermint [4]. We have chosen Tendermint because, in [24], it has been shown by mathematical analyzes that Tendermint, as a closed system, is *unfair* for some of its participants. However, we consider Tendermint as an open system and thus aim to study its fairness by computational analyzes. To this end, we modeled each participant, like in [21], as a *rational agent* who finds the system *fair* if the total satisfaction of its expectations is above a certain degree. We, then, realized two implementations of Tendermint: (1) as proposed by Buchman et al. [4], which we call the *Tendermint Default* protocol and (2) as proposed by us, which we call call the *Tendermint Fairness* protocol. We conducted various simulation experiments with different initial conditions and different agent strategies.

### *1.1. Contributions*

The contributions of this study are as follows:

- A *fairness* improvement for the Tendermint protocol;
- A *computational* analysis of the fairness in the Tendermint protocol.

### *1.2. Organization*

The organization of this paper is as follows. Section 2 provides background about the Tendermint protocol. Section 3 provides a formalization of the existing Tendermint protocol. Section 4 improves this formalization by improving fairness of the behaviors for participants. In Section 5, make simulation analyzes of both models. Results and discussion are presented in Section 6. Section 7 presents the related work and, finally, Section 8 concludes the paper.

## 2. Background

In this section, we give background information about the Tendermint blockchain protocol. Tendermint is a *committee-based* blockchain protocol that uses an adaptation of the Practical Byzantine Fault Tolerance (PBFT) algorithm [7] as its consensus algorithm. In the following, we first describe the PBFT algorithm (Section 2.1) and then we describe the Tendermint protocol (Section 2.2).

### *2.1. Practical Byzantine Fault Tolerance*

The PBFT algorithm [7] is a variant of traditional Byzanytine Fault Tolerant (BFT) consensus algorithms that are resilient to both byzantine and crash failures and generally work under partial synchrony assumptions and bounded communication latency. In PBFT, an agreement among $n$ nodes is reached through the transmission of $O(n^2)$ messages; it does so relying on a three phase round division where in each round a value (e.g., a block) is validated passing through a *pre-prepared*, *prepared* and *commit* steps. Each peer's proposal accesses to the next phase only with the 2/3 network approval. Therefore, the algorithm requires at least $2f + 1$ honest replicas to tolerate $f$ faulty nodes.

Figure 1 illustrates the normal operation of this protocol. Here, the line 'C' represents the operations of a peer requesting an agreement on something (in particular, sending a transaction in the context of the blockchain), and waiting for the reply. The lines '0–3' represent the operation of peers running the PBFT protocol, '3' is faulty and '0' is called the *proposer*.

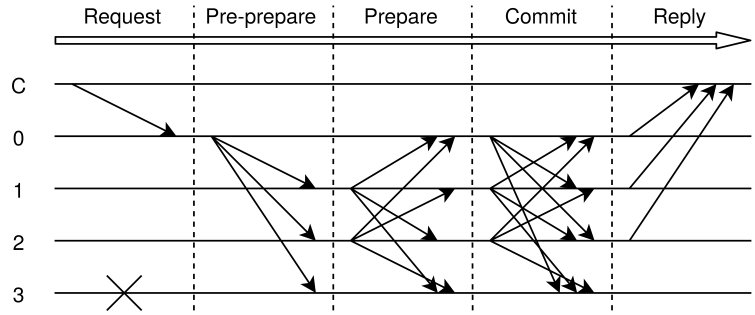


**Figure 1.** Normal execution of the Practical Byzantine Fault Tolerance (PBFT) protocol [7].

### *2.2. Tendermint Protocol*

Tendermint is a *committee-based* blockchain protocol that relies on the *proof-of-stake* (PoS) algorithm for the selection of the committee. The Tendermint protocol defines three types of participants [25]: the *users*, the *validators*, and the *delegators*; and two types of assets: the *balance* and the *stake*. Users create transactions, validators create blocks, and delegators invest in validators. For each new block, a committee is *selected* among the validator, a validator can only create a block if it is in the committee. Each participant keeps an account balance on the blockchain data structure. All participants can use their balances to send or receive assets to or from other participants. Validators and delegators can *invest* their balances in stakes, and stakes can be *withdrawn* into balances. The more stakes a validator has, the higher its chance to become a *block creator* (i.e., PoS). The delegators, on the other hand, do not directly enter into the committee and create blocks, but invest in validators they choose to increase their chance to be in the committee. In general, validators who are aware that they do not have enough stakes to be selected as block creators become delegators. All participants can switch roles at any moment: the amount of personal stakes and where they are determine the role.

For each new block that needs to be created, a committee is *selected* among the validators. The selection of validators for the committee is a deterministic process: each participant knows the committee size and can compute the stakes of the validators thanks to its own local copy of blockchain for selecting the wealthiest. During this process, if a participant acts incorrectly and this is detected by the others, its stakes and the stakes of the delegators which invested in it are *slashed*, i.e., the stakes are burnt and remain unusable.

When a committee is selected, its members go through a PBFT process as explained in Section 2.1. This process is illustrated in Figure 2 (even though in the original Tendermint paper [7] the consensus protocol steps are named *propose*, *prevote*, *precommit*, and *commit*, hereafter, in this paper, we prefer to use *prepropose*, *propose*, *vote*, and *commit*, respectively, as proposed in [7].) At round $r$, the $r$th validator, modulo the size of the committee, is considered as the *proposer*. The proposer sends a *prepropose message* containing a block *proposal* to all validators in the committee, including itself. It can propose a block that has already been proposed in a different round for the same height. Validators check the proposal, and, if they found it valid, they send a *propose message* containing the proposal to the other validators.

When a validator receives at least *2/3* of the proposed messages containing the same valid block, it broadcasts to the whole blockchain system a *vote message* containing the proposal. When at least *2/3* of the vote messages containing the same valid block are received by a participant, it *commits* the block: i.e., it appends it to its own blockchain. Otherwise, it considers the block invalid and moves to a new round with a new proposer. A block can be proposed more than once in different rounds for the same height. Upon a successful creation of a new block, the committee members are rewarded. However, unlike Bitcoin-like blockchains where block creators are rewarded directly by themselves, in Tendermint, block creators (committee members) are rewarded by the following committee.

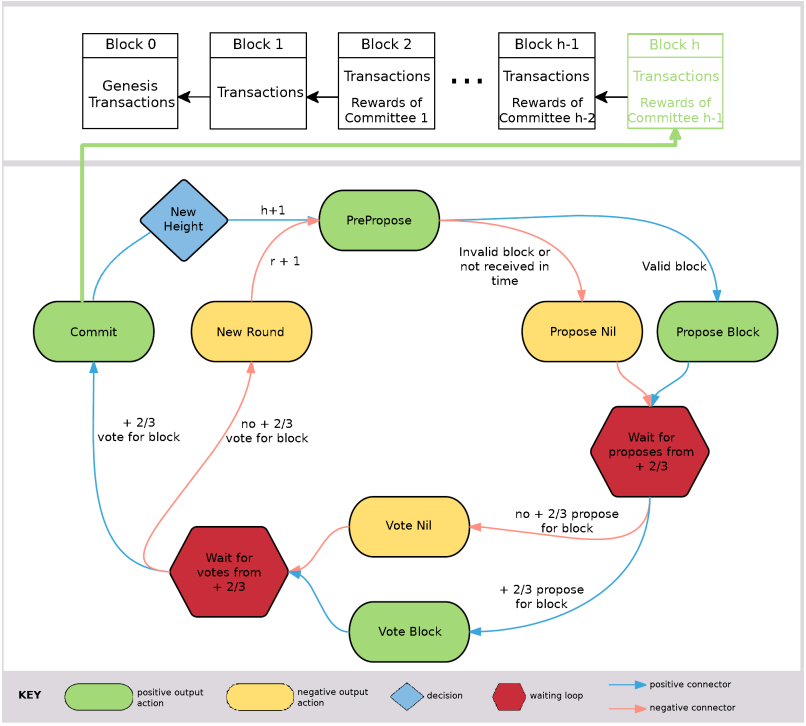


**Figure 2.** Illustration of the Tendermint consensus protocol for the creation of block $h$.

## 3. Tendermint Default Model

In this section, we provide a Tendermint protocol model based on [4,25] and various core features from the official protocol given in the official Github repositor (Tendermint Github repository, https://github.com/tendermint/tendermint, last access on 9 September 2019.). In the following, we first introduce the overall system model (Section 3.1), after we give the blockchain data structure model (Section 3.3), and then we each present validators and delegators by providing high-level detailed pseudocodes of their key actions (Sections 3.4 and 3.5, respectively).

### *3.1. Overall System Model*

We model the Tendermint blockchain network as a dynamic directed graph $G = (N, E)$, where $N$ denotes the dynamic node (vertex) set, $E$ denotes a dynamic directed link (edge) set. A node $n$ can enter and leave $G$ by using its $join(G)$ and $leave(G)$ actions, respectively. Upon joining $G$, $n$ discovers neighbor nodes to connect to. A link $\langle n, m \rangle \in E$ represents a directed link $n \rightarrow m$, where $n, m \in N$, $n$ is the owner of the link, and $n$ is the neighbor of $m$.

A node $n$ can communicate with a set of recipient nodes $R_n$ (where $\forall m \in R_n | \{n, m\} \in E$) by exchanging messages of the form $\langle n, msg, d \rangle$, where $n$ is the sender, $msg$ is the type, and $d$ is the data contained.

Each node $n \in N$ has a list of its neighbors $N_n$, where $N_n \subseteq N$ and $\forall m \in N_n | \langle n, m \rangle \in E$. A node $n$ adds and removes another node $m$ as its neighbor using its $addNeighbour(m)$ and $removeNeighbour(m)$ actions, respectively. Each neighbor $m \in N_n$ is represented as a 2-tuple $\langle m, t_m \rangle$ where $t_m$ is the last communication time. $t_m$ is updated at each message receipt and, if $m$ has not communicated for more than some time, $m$ is removed from $N_n$.

Each node $n$ has a memory pool $\Theta_n$ in which it keeps unconfirmed transactions that have input transactions, an orphan pool $\bar{\Theta}_n$ in which they keep unconfirmed transactions that have one or more missing input transactions (orphan transactions) and a blockchain ledger $B_n$ in which they keep confirmed transactions where $\Theta_n \cap \bar{\Theta}_n = \emptyset$, $\Theta_n \cap B_n = \emptyset$ and $\bar{\Theta}_n \cap B_n = \emptyset$ always hold. The total wealth of $n$, i.e., its balance ($\phi_n$), is calculated from $B_n$.

### 3.2. Fairness Model

We model fairness based on the fairness model proposed in [21]. In this sense, we model all nodes in the blockchain network $G$ as rational agents. A rational agent behaves according to its local perceptions and local knowledge, models uncertainty via expected values of variables or actions, and always chooses to perform the actions with the optimal expected outcome (among all feasible actions) for maximizing its utility [26]. Each rational agent $n \in N$ has a set of actions $A_n$ and a utility function $\mathcal{U}_n$. Using $A_n$ and $\mathcal{U}_n$, $n$ uses a decision process where it identifies the possible sequences of actions to execute. We call these sequences rational behaviors of $n$ and denote as $\beta$. The objective of $n$ is to choose the behaviors that selfishly keep $\mathcal{U}_n$ as high as possible.

We model the utility function of a rational agent $n \in N$ as $\mathcal{U}_n = u_0 + \sum_{i=1}^{k} \mathcal{U}(\beta_i)$, where $u_0$ is the initial utility value, $k \geq 0$ is the number of behaviors executed so far, and $\mathcal{U}(\beta_i)$ is the utility value of the behavior $\beta_i$. A utility value $\mathcal{U}(\beta_i)$ can also be interpreted as the *degree of satisfaction* experienced by the realization of $\beta_i$. The utility value $\mathcal{U}(\beta_i)$ is calculated as $\mathcal{R}(\beta_i) - \mathcal{C}(\beta_i)$, where $\mathcal{R}(\beta_i)$ is the overall reward gained and $\mathcal{C}(\beta_i)$ is the overall cost spent for the execution of $\beta_i$.

> **Fairness:** A rational agent $n \in N$ finds a system (i.e., the blockchain network) $G$ *fair*, if the total satisfaction of its expectations $\mathcal{U}_n$ is above a certain degree $\tau_n$, where $\tau_n < u_0$.

If, at any time, an agent $n$ finds $G$ *unfair* ($\mathcal{U}_n \leq \tau_n$), it may decide to leave $G$ if, from its points of view, it will not be possible to increase its overall utility above $\tau_n$ by calculating the expected values of its possible future behaviors. In other words, $n$ may decide to leave $G$ if $\mathcal{U}_n + \sum_{j=k}^{m} \mathcal{E}(\beta_j) \leq \tau_n$ where $\beta_k$, ..., $\beta_m$ are sufficiently enough desired future behaviors of $n$.

### 3.3. Blockchain Model

We model the blockchain ledger of an agent $n$ as a dynamic append-only linked list $B_n = \{b_0 \xleftarrow{r_0} b_1 \xleftarrow{r_1} \ldots \xleftarrow{r_{h-1}} b_h\}$, where each block $b_i$ ($0 < i \leq h$) contains a cryptographic reference $r_{i-1}$ to its previous block $b_{i-1}$, $h = |B_n|$ is the depth of $B_n$, $b_0$ is the root block which is also called the *genesis block*, and $b_h$ is the furthest block from the genesis block, which is referred to as the *blockchain head*.

We denote a block as $b_i = \langle \mathtt{h}_i, x\Psi_i \rangle$ where $\mathtt{h}_i$ is the block header and $\Psi_i$ is the block data. The block data $\Psi_i$ contains all the transactions organized as a Merkle tree [27]. Basically, the copies of each transaction are hashed, and the hashes are then paired, hashed, paired again, and hashed again until a single hash remains, the Merkle root of a Merkle tree. We denote a Merkle tree and its root as $\Psi_i$ and $\psi_i$, respectively. A Merkle tree $\Psi_i$ is created using the action of the form $createMerkleTree(X_{i-1}, \theta_v)$, where $X_{i-1}$ is the set of coinbase transactions that reward the validator agents $V$ and the delegator agents $d \in D$ of the previous block $b_{i-1}$ with a total block reward $\gamma$ for their work, $\theta_v \subseteq \Theta_v$ is the set of candidate transactions chosen for this block. The set of candidate transactions $\theta_v$ is selected using the action of the form $selectTransactions(\Theta_v) : \theta_v$, where $\Theta_v$ is the memory pool of the validator.

The block header is denoted as $\mathtt{h}_i = \{v_n, \mathcal{H}(\mathtt{h}_{i-1}), \mathcal{H}(\mathtt{h}_i), t_{b_i}, \psi_i\}$, where $v_n$ is the version number of the protocol used by $n$, $\mathcal{H}(\cdot)$ is the cryptographic hash function, $\mathcal{H}(\mathtt{h}_{i-1})$ is the cryptographic hash code of the header of the previous block $b_{i-1}$ ($i > 0$), $\mathcal{H}(\mathtt{h}_i)$ is the cryptographic hash code of $\mathtt{h}_i$ generated by $m$, $t_{b_i}$ is the current timestamp, and $\psi_i$ is the root of the Merkle tree.

We model a transaction as $tx = \langle I, O \rangle$ where $I$ is a list of inputs ($I \neq \emptyset$) and $O$ is a list of outputs ($O \neq \emptyset$). We model the outputs as $o_i = \langle m, ¢_{o_i} \rangle$, where $m \in N$ is the receiver of the coins $¢_{o_i}$ ($¢_{o_i} \geq 0$). All inputs of a transaction have to be spent in that transaction and the total input coins $¢_I$ has to be greater than or equal to the total output coins $¢_O$. The fee $f_{tx}$ of a transaction $tx$ is then modeled as $f_{tx} = ¢_I - ¢_O$. Depending on the fee to be paid, if there are still some coins left to be spent, the sender can add an output that pays this remainder to itself.

A coinbase transaction $tx \in X_i$ is a special transaction that collects and spends any transaction fees paid by transactions included in a block and exceptionally it does not have any input set ($I = \emptyset$). It is the first transaction in a block and can only be created by a validator when it is proposing a block.

### *3.4. Validator Model*

An agent $v \in V$, where $V \subseteq N$, is said to be a *validator* if it is capable of participating in the creation of blocks (consensus) for confirming the transactions in its memory pool $\Theta_v$. To do so, $v$ can use its actions of the form *prepropose*(), *propose*(), *vote*() and *commit*() (see Algorithm 1). When $v$ receives a message $m$, it handles it using the action *handleMessage*($m$) to store $m$ into one of the corresponding received messages list, i.e., $M_{pre}$, $M_{pro}$ and $M_{vote}$.

**Algorithm 1** The actions of a validator agent $v$ for creation of the $i$th block

1: **action prepropose()**
2: $b_i \leftarrow$ createBlock()
3: send($\langle$ $v$,"prepropose", $b_i$ $\rangle$, $C_i$)
4:
5: **action propose()**
6: **if** (valid($b_i$)) **then**
7: send($\langle$ $v$,"propose", $b_i$ $\rangle$, $C_i$)
8: **else**
9: send($\langle$ $v$,"propose", *nil* $\rangle$, $C_i$)
10: **endif**
11:
12: **action vote()**
13: **if** ($|M_{pro}^{b_i}| > 2/3\ |C_i|$) **then**
14: send($\langle$ $v$,"vote", $b_i$ $\rangle$, $C_i$)
15: **else**
16: send($\langle$ $v$,"vote", *nil*, $n$ $\rangle$, $C_i$)
17: **endif**
18:
19: **action commit()**
20: **if** ($|M_{vote}^{b_i}| > 2/3\ |C_i|$) **then**
21: $B_v \leftarrow B_v \cup \{b_i\}$
22: **else**
23: newRound()
24: **endif**
25:
26: **action createCoinbaseTxs()**
27: **for each** ($z \in C_{i-1}$) **do**
28: **for each** ($d \in D(z)$) **do**
29: $\gamma_d = \frac{\gamma_{i-1}}{|C_{i-1}|} \times \frac{s_z^d}{s_z}$
30: $X_{i-1} \leftarrow \{d,\gamma_d\}$
31: **endfor**
32: $\gamma_z = \frac{\gamma_{i-1}}{|C_{i-1}|} \times \frac{s_z^z}{s_z}$
33: $X_{i-1} \leftarrow \{z,\gamma_z\}$
34: **endfor**
35: **return** $X_{i-1}$
36:
37: **action createBlock()**
38: $\theta_v \leftarrow$ selectTransactions($\Theta_v$)
39: $X_{i-1} \leftarrow$ createCoinbaseTxs()
40: $\Psi_i \leftarrow$ createMerkleTree($X_{i-1}, \theta_v$)
41: $b_i \leftarrow \langle h_i, \Psi_i \rangle$
42: **return** $b_i$
43:
44: **action invest(**$a$**)**
45: **if** ($\phi_{v_i} \geq a$) **then**
46: $\phi_{v_i} \leftarrow \phi_{v_i} - a$
47: $s_{v_i} \leftarrow s_{v_i} + a$
48: **endif**
49:
50: **action withdraw(**$a$**)**
51: **if** ($s_{v_i} \geq a$) **then**
52: $s_{v_i} \leftarrow s_{v_i} - a$
53: $\phi_{v_i} \leftarrow \phi_{v_i} + a$
54: **endif**
55:
56: **action handleMessage(**$m$**)**
57: **switch** ($m$)
58: **case** : "*prepropose*"
59: $M_{pre} \leftarrow M_{pre} \cup \{m\}$
60: **case** : "*propose*"
61: $M_{pro} \leftarrow M_{pro} \cup \{m\}$
62: **case** : "*vote*"
63: $M_{vote} \leftarrow M_{vote} \cup \{m\}$
64: **endswitch**

During the creation of block $b_{h+1}$, $v$ is said to be a committee member ($v \in C_{h+1}$ where $C_{h+1} \subseteq V$) if it has enough stakes $s_v$. If $v$ does not have enough stakes, it can invest a certain amount $a$ into stakes in order to be selected in the committee using the action *invest*($a$) where $a \leq$ balance $\phi_v$. At any time, $v$ can withdraw a certain amount $a$ from its investment $s_v$ into its balance $\phi_v$ using the action *withdraw*($a$).

Blocks are created using the action *createBlock*(), and they contain the reward transactions for the last committee members and their delegators. These transactions are created using the action *createCoinbaseTxs*() that takes into account the personal stakes of the validator $v_i$, called $s_{v_i}^{v_i}$, the stakes

invested by a delegator $d_i$ into a validator $v_i$, called $s_{v_i}^{d_i}$, and the global stakes of the validator $v_i$, which is the sum of its personal stakes and the stakes the delegators gave it, called $s_{v_i} = s_{v_i}^{v_i} + \sum s_{v_i}^{d_i}$.

### *3.5. Delegator Model*

An agent $n \in D$, where $D \subseteq N$ and $D \cap V = \emptyset$, is said to be a *delegator* if it is capable of delegating a certain amount $a$ to a validator $v \in V$ for helping $v$ to enter into the committee using the action of the form $invest(a, v)$ (see Algorithm 2). Hereafter, each time $v$ is rewarded for participating in a block creation, $n$ is also rewarded. At any time, $n \in D$ can withdraw a certain amount $a$ of its investments into its balance using the action $withdraw(a, v)$.

**Algorithm 2** The actions of a delegator agent $d$

1: **action invest(**$a, v$**)**
2: **if** $(\phi_{v_i} \geq a)$ **then**
3: $\phi_{v_i} \leftarrow \phi_{v_i} - a$
4: $s_{T_v} \leftarrow s_{T_v} + a$
5: $s_i(d, v) \leftarrow s_i(d, v) + a$
6: **endif**
7:
8: **action withdraw(**$a, v$**)**
9: **if** $(s_i(d, v) \geq a)$ **then**
10: $s_i(d, v) \leftarrow s_i(d, v) - a$
11: $s_{T_v} \leftarrow s_{T_v} - a$
12: $\phi_d \leftarrow \phi_d + a$
13: **endif**
14:

## 4. Tendermint Fairness Model

In this section, we improve the model given in Section 3 by improving the behaviors of validators in order to increase the fairness of the system. The improvements are as follows.

It has been shown in [24] that when there is less than $f$ Byzantines, and there is bad communication between some committee members, the block creation process can be frozen. To tackle this problem, rather than storing only the last valid proposed block, the committee members store all valid proposed blocks for creation block $h$. This way, if there is bad communication between the members and there are less than $f$ Byzantines, they can still remember the previously valid proposed blocks and vote for them when they are proposed once again.

As mentioned in Section 3, the committee rewards the previous one for successful creation of a new block. However, if there are Byzantines in the previous committee, the current committee will most probably not have enough time to detect this. This may cause unfair rewarding of the participants. To tackle this problem, we propose *delayed rewarding* of the committees where a previous committee is rewarded after a certain number of blocks. We model this improvement by modifying the actions $commit()$ and $createCoinbaseTxs()$ as shown in Algorithm 3. When a validator of a committee commits a block, it sends a message to the whole network that it has committed that block. The proposer of a following committee will add the related coinbase transactions to its proposal.

**Algorithm 3** The actions of a validator agent $v$ for creation of the $i$th block in the Fairness model

**action commit($b_i$)**
  **if** ($|M^{b_i}_{vote}| > 2/3\ |C_i|$) **then**
    $B_v \leftarrow B_v \cup \{b_i\}$
    **if** ($v \in C_i$) **then**
      send($\langle\ v$,"hascommit", $b_i\ \rangle$, $C_i$)
    **endif**
  **else**
    newRound()
  **endif**

**action createCoinbaseTxs()**
  **for each** ($z \in hascommit(b_i)$) **do**
    **for each** ($d \in D(z)$) **do**
      $\gamma_d = \frac{\gamma_{i-1}}{|C_{i-1}|} \times \frac{s^d_z}{s_z}$
      $X_{i-1} \leftarrow \{d,\gamma_d\}$
    **endfor**
    $\gamma_z = \frac{\gamma_{i-1}}{|C_{i-1}|} \times \frac{s^z_z}{s_z}$
    $X_{i-1} \leftarrow \{z,\gamma_z\}$
  **endfor**
  **return** $X_{i-1}$

## 5. Simulations

To quantitatively analyze the aforementioned models in terms of *fairness*, we designed and conducted several simulation experiments. In the following, we first give the various simulation experiments we designed (Section 5.1) and then we present our experimental setup (Section 5.2).

### *5.1. Simulation Experiments*

We designed experiments in order to analyze *fairness* of the Tendermint models given in Sections 3 and 4.

The assumptions we made for all the experiments are as follows:

- There are no Byzantines, i.e., there are no crashes or no erroneous messages.
- Each participant always plays the same role.
- The size of the committee is always 5.
- There are no fees for the transactions.
- All agents process the messages as soon as they receive them.
- The total block reward $\gamma_b$ is 50.
- There is no message transmission delay in the network.
- The reliability of the network is 100%, i.e., every message is received by its receivers.
- Starting from the creation of the first block, each 30 ticks, the simulator calculates the overall fairness of the system and adds new validators depending on the overall fairness according to

$$w = \begin{cases} log_2(\mathcal{K}), & \text{if } \mathcal{K} > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

  where $\mathcal{K} = \tau_n - \frac{1}{|N|} * \sum_{i=1}^{|N|} \mathcal{U}_i$.
- The initial balance for validators ($\alpha_v$) is calculated using a normal distribution between 50 and 100.
- The initial balance for delegators ($\alpha_d$) is calculated using a normal distribution between 25 and 75.
- The utility function of all agents is $U_n : \mathcal{N} \rightarrow \mathcal{N}$ and for every agent $n$ the *threshold* $\tau_n = -10$.
- $\forall$ *agent* $n : u_0 = 0$.

For each agent $n$, its utility follows the law:

$$\mathcal{U}_n = \begin{cases} \mathcal{U}_n + 1, & \text{if } b_h \textit{ rewards } n, \\ \mathcal{U}_n - 1, & \text{otherwise.} \end{cases} \tag{2}$$

We define two different investment strategies for validators:

- $\beta_1$: investing all balance in stakes,
- $\beta_2$: investing random amounts from balance in stakes,

We define one investment strategy for delegators:

- $\beta_3$: investing random amount from their balance in one random validator at each new block.

Using the different validator strategies and different initial number of delegators, we designed eight different experiments (see Table 1).

**Table 1.** Settings of the experiments.

| Experiment # | Model | Validator Strategy | Initial # Validators | Initial # Delegators |
|---|---|---|---|---|
| Ex 1 | Default | $\beta_1$ | 10 | 0 |
| Ex 2 | Default | $\beta_1$ | 10 | 5 |
| Ex 3 | Default | $\beta_2$ | 10 | 0 |
| Ex 4 | Default | $\beta_2$ | 10 | 5 |
| Ex 5 | Fairness | $\beta_1$ | 10 | 0 |
| Ex 6 | Fairness | $\beta_1$ | 10 | 5 |
| Ex 7 | Fairness | $\beta_2$ | 10 | 0 |
| Ex 8 | Fairness | $\beta_2$ | 10 | 5 |

### *5.2. Experimental Setup*

To study blockchain systems in general, we developed an agent-based simulator (the simulator is called Multi Agent eXperimenter (MAX) and has not been made public yet) based on the MaDKit (MaDKit, http://www.madkit.net/madkit/, last access on 9 September 2019.) framework. MaDKit is a lightweight Java library for designing and simulating multi-agent systems [28]. We run our experiments on an Ubuntu OS computer powered by an *Intel® Core™ i7-8850H CPU @ 2.60 GHz x 12 threads* and 32 GB of RAM (Gif-sur-Yvette, France).

We run each experiment five times until the 150th block is committed to the blockchain. We report and discuss the results of these executions in the following section.

## 6. Results and Discussion

In section, we report and discuss the results of the conducted simulations.

### *6.1. Results*

In order to study our experiments, we first get the utilities of each agent of the experiments at each height. For each experiment, we compute the median of the utilities. Then, we compute the average of the medians. We compare the experiments by computing the difference of those results.

Figures 3 and 4 give an overview of the differences between the different models and settings. Figure 4 displays the number of agents over time. Figure 3 displays the median utilities of the experiments overtime.

### *6.2. Discussion*

Figure 4 gives an overview of the differences between the different models and settings. This chart displays the number of agents overtime: the higher the numbers are, the better is the tested model. We can see that, in general, the Fairness model is usually more fair than the Default model after a certain amount of time. In the beginning, due to the presence or absence of delegators, there might be more or less agents in the system. However, as they are still agents, they might leave if they think the

system is not fair enough. In addition, some experiments have the same settings, but different models, which means that we should first compare them before transiting to others.

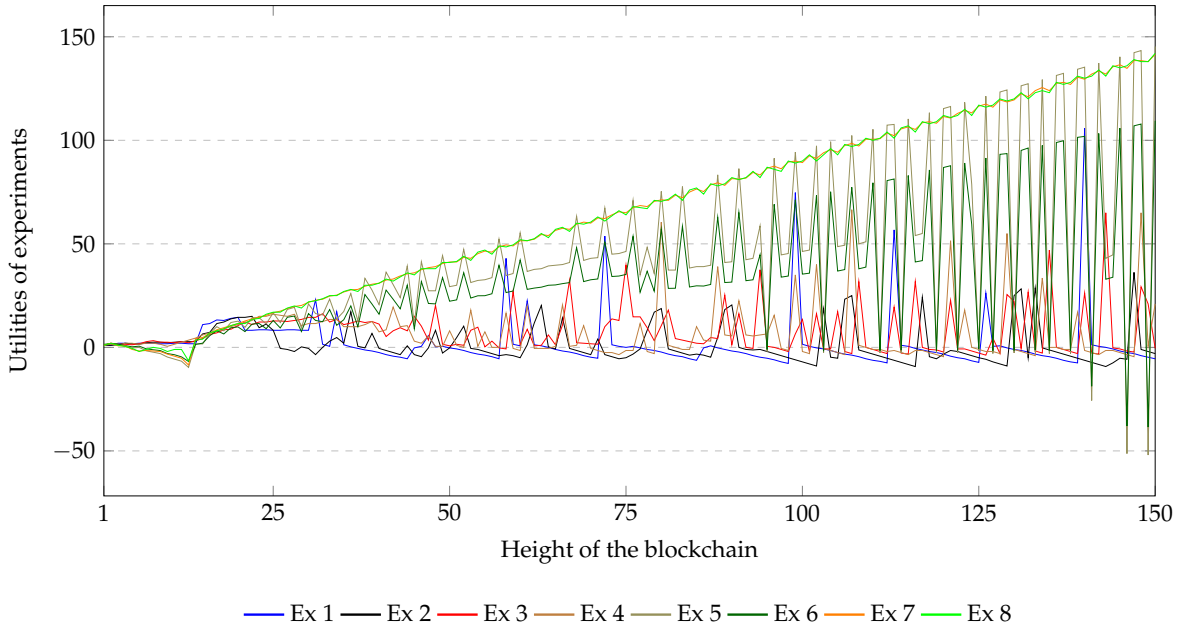


**Figure 3.** Evolution of the utilities of the experiments by height.

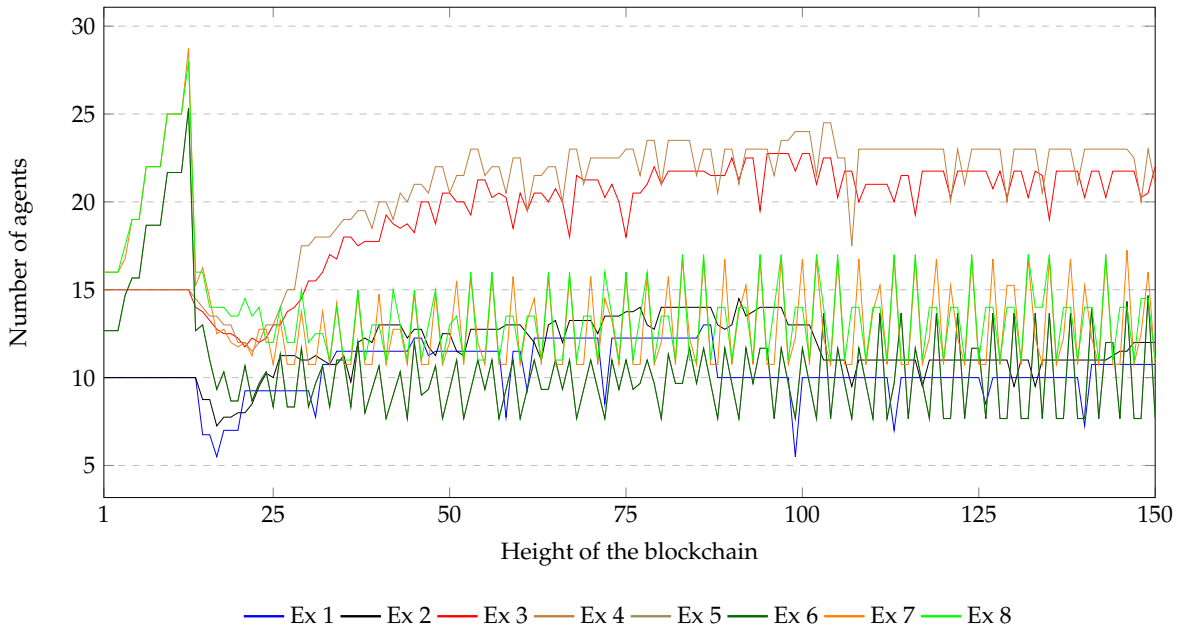


**Figure 4.** Evolution of the number of agents by height.

We can also observe that most models seem to stabilize at a certain number of agents which is often close to the initial number of agents. This can be due to multiple factors. First, the utility threshold is, in our experiments, the same for every agent, which, in reality, is really different. Moreover, the utility is only based on *when* an agent is rewarded, not how much. In real life, an agent can be rewarded at every block, but the reward could be very low. Furthermore, because no agents withdraw their stakes, the stakes of each agent will always increase, mathematically speaking. Hence, a validator selected in a committee will have a huge chance to be selected at the following ones because the non selected validators do not have enough stakes at first. The two ways to change this are:

1. a non selected validator has a higher sum of balance and stakes than a selected one, but lower stakes at one height. Then, at a further height, this non-selected validator chooses to invest enough balance in stakes to have more assets than the selected one, hence being selected afterwards.
2. delegators invest enough stakes in non selected validators rather than selected validators to reverse the process.

The first option happens only with half of the experiments: those where the behavior of the validators is $\beta_2$. In real life, this can happen with a decreasing probability over time because the system enters in a *virtuous* circle for multiple validators and a *vicious* circle for the others. Indeed, the gap of the sum of the two assets, balance and stakes, between the validators will increase over time.

The second option happens only with half of the experiments too, but not the same ones. This is what is expected to happen in order to make the system more fair: thanks to this process, more agents would be rewarded. Of course, the delegators could invest only in the wealthiest validators, but if we refer to [26], the delegators could be less rewarded than investing in a little bit poorer validators, which could be then selected in a committee.

## 7. Related Work

One solution for solving *partially* the fairness issue in Bitcoin-like blockchains was found by its participants: the use of *mining pools* [16]. Those structures allow more agents to be involved in the system and be rewarded, even with small power. There are fees for being part of a pool, but they remain low enough to be profitable even with a low-powered machine. However, such mining pools are controlled by owners that reduce the distributed aspect of the Blockchain. Some Blockchain protocols proposed solutions, such as the *FruitChains protocol* [29], to distribute the block rewards in a more fair way. However, they still rely on energy consumption to secure the blockchain.

Other issues triggered by Bitcoin-like blockchain systems are summed up in [21]:

- There is no mechanism for a user to cancel an already issued transaction, i.e., once it decides to engage in the game, it can never abandon it. This implies expected values going to minus infinity. In decision theory terms, this would mean assuming a user having an infinite interest on a transaction, which is in fact hard to assume in real settings.
- Given different fees with their associated probabilities, more flexibility to guarantee fairness would have been reached by allowing the user (experiencing a long waiting time) to resend the transaction with a lower fee. This flexibility, on the other hand, is not achieved in Bitcoin-like blockchain systems mainly due to two reasons: (1) there is no deterministic guarantee that the system chooses a given copy of the transaction, it is only guaranteed with high probability that only one copy will be inserted into the blockchain, and (2) the miner behavior favors the transaction with the highest fee to get included in a block.
- The block size is a fixed parameter today. Obviously, if the volume of transactions increases over time, the space left for more transactions can become a scarce resource, and miners will be more apt to deliberately delaying a transaction with a low fee. This will dramatically drop the probabilities for low fees and their corresponding expected values by time.
- Fees are decided by the users, leading to a possible race among user fees. Such a situation would make the probabilities difficult to predict, and expected values of users might drop even faster.
- The fairness is a locally perceived concept, and it must indeed be tracked. This needs further detection mechanisms not included in the current systems.

In general, most of the work is focused on fairness to users, such as [21], but very few on block creators. There is even less about the Tendermint protocol. In [30], the fairness policy in Tendermint is discussed in order to correctly punish Byzantines. This aspect is crucial as the correct agents would find a system unfair if incorrect ones are rewarded. This paper also addresses Factom [31], Enigma [32], and Hawk [33], protocols which provide various ways of managing such policies. Jalalzai et al. [34] implemented and evaluated the performance of different BFT based protocols for blockchains under

normal conditions as well as when byzantine failures are encountered in the network, and also calculated the reliability of each protocol under the desired throughput.

## 8. Conclusions and Future Work

Tendermint introduces a class of Blockchain protocols which relies on committees, stakes, and delegators. It aims at solving different issues from the Bitcoin-like blockchains, including the fairness. In this paper, we show that we implemented a simulation model of Tendermint to analyze its fairness using MaDKit. We then highlight the strengths and weaknesses of Tendermint based on the results generated by the execution of this implementation. Thanks to its PBFT consensus, more agents are involved in the block creation process (and rewarded for their involvement) and thus the protocol is more fair compared to Bitcoin-like blockchains. We also show that the delegators play a crucial role by giving more importance to agents who want to be involved without having enough *stakes*. However, the Tendermint protocol suffers from flaws that exclude agents for various reasons, especially when they do not have enough information on time or do not adapt rapidly. We proposed to delay the rewarding and included the improvements proposed in [24]. Those modifications upgrade the original model as seen in the results.

In this paper, we did not study the security limits of the Tendermint protocol with incorrect agents. We also did not study more *intelligent* agents who could take more profitable actions for them and the system. For instance, the delegators could have *smarter* processes for selecting their validators and how much they invest. We also did not study the influence of fees for the transaction selections and rewarding. Indeed, the agents have to be rewarded as soon as their messages are received. In a more realistic system, validated transactions can be chosen by the proposers and they may refuse some of them. All those issues can be studied in further research.

**Author Contributions:** Conceptualization, Ö.G.; methodology, Ö.G.; software, M.A.D. and N.L.; validation, N.L., M.A.D. and Ö.G.; investigation, N.L., M.A.D. and Ö.G.; writing–original draft preparation, Ö.G.; writing–review and editing, N.L., M.A.D. and Ö.G.; visualization, M.A.D.; supervision, Ö.G.; project administration, Ö.G.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyzes, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| PBFT | Practical Byzantine Fault Tolerance |
| PoS | Proof-of-Stake |
| MAX | Multi-Agent eXperimenter |
| MaDKit | Multi-Agent Development Kit |

## References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: http://www.bitcoin.org/bitcoin.pdf (accessed on 29 November 2019).
2. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Available online: http://gavwood.com/Paper.pdf (accessed on 27 November 2019).
3. Goodman, L.M. Tezos – s Self-Amending crypto-ledger. White Paper. 2014. Available online: https://tezos.com/static/white_paper-2dc8c02267a8fb86bd67a108199441bf.pdf (accessed on 29 November 2019)
4. Buchman, E.; Kwon, J.; Milosevic, Z. The latest gossip on BFT consensus. *arXiv* **2018**, arXiv:1807.04938
5. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; Caro, A.D.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *arXiv* **2018**, arXiv:1801.10228.

6. Back, A. Hashcash-A Denial of Service Counter-Measure; Technical Report. 2002. Available online: http://www.hashcash.org/papers/hashcash.pdf (accessed on 29 November 2019).
7. Castro, M.; Liskov, B. Practical Byzantine Fault Tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, LA, USA, 22–25 February 1999; USENIX Association: Berkeley, CA, USA, 1999; pp. 173–186.
8. Wang, W.; Hoang, D.T.; Hu, P.; Xiong, Z.; Niyato, D.; Wang, P.; Wen, Y.; Kim, D.I. A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks. *IEEE Access* **2019**, *7*, 22328–22370. doi:10.1109/ACCESS.2019.2896108. [CrossRef]
9. Crain, T.; Gramoli, V.; Larrea, M.; Raynal, M. (Leader/Randomization/Signature)-Free Byzantine Consensus for Consortium Blockchains. Available online: http://csrg.redbellyblockchain.io/doc/ConsensusRedBellyBlockchain.pdf (accessed on 27 November 2019).
10. Decker, C.; Seidel, J.; Wattenhofer, R. Bitcoin Meets Strong Consistency. In Proceedings of the 17th International Conference on Distributed Computing and Networking Conference (ICDCN), Singapore, 4–7 January 2016.
11. Kokoris-Kogias, E.; Jovanovic, P.; Gailly, N.; Khoffi, I.; Gasser, L.; Ford, B. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. In Proceedings of the 25th USENIX Conference on Security Symposium, Austin, TX, USA, 10–12 August 2016; USENIX Association: Berkeley, CA, USA, 2016; pp. 279–296.
12. Eyal, I.; Sirer, E.G. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin, Germany, 2014; pp. 436–454.
13. Lewenberg, Y.; Sompolinsky, Y.; Zohar, A. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin, Germany, 2015; pp. 528–547.
14. Eyal, I. The miner's dilemma. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015; pp. 89–103.
15. Sapirshtein, A.; Sompolinsky, Y.; Zohar, A. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin, Germany, 2016; pp. 515–532.
16. Eyal, I.; Gencer, A.E.; Sirer, E.G.; Van Renesse, R. Bitcoin-NG: A Scalable Blockchain Protocol. In Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation, Santa Clara, CA, USA, 16–18 March 2016; pp. 45–59.
17. Göbel, J.; Keeler, H.P.; Krzesinski, A.E.; Taylor, P.G. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Perform. Eval.* **2016**, *104*, 23–41. [CrossRef]
18. Pass, R.; Shi, E. FruitChains: A Fair Blockchain. Cryptology ePrint Archive, Report 2016/916. 2016. Available online: http://eprint.iacr.org/2016/916.pdf (accessed on 27 November 2019).
19. Carlsten, M.; Kalodner, H.; Weinberg, S.M.; Narayanan, A. On the instability of bitcoin without the block reward. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 154–167.
20. Liu, J.; Li, W.; Karame, G.O.; Asokan, N. Towards Fairness of Cryptocurrency Payments. *arXiv* **2016**, arXiv:1609.07256.
21. Gürcan, Ö.; Del Pozzo, A.; Tucci-Piergiovanni, S. On the Bitcoin Limitations to Deliver Fairness to Users. In *On the Move to Meaningful Internet Systems. OTM 2017 Conferences*; Panetto, H., Debruyne, C., Gaaloul, W., Papazoglou, M., Paschke, A., Ardagna, C.A., Meersman, R., Eds.; Springer International Publishing: Cham, Switzerland, 2017; Volume 10573, pp. 589–606. doi:10.1007/978-3-319-69462-7_37. [CrossRef]
22. Gürcan, Ö.; Ranchal Pedrosa, A.; Tucci-Piergiovanni, S. On Cancellation of Transactions in Bitcoin-Like Blockchains. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*; Panetto, H., Debruyne, C., Proper, H.A., Ardagna, C.A., Roman, D., Meersman, R., Eds.; Springer International Publishing: Cham, Switzerland, 2018; Volume 11229, pp. 516–533. doi:10.1007/978-3-030-02610-3_29. [CrossRef]
23. Gürcan, Ö. Multi-Agent Modelling of Fairness for Users and Miners in Blockchains. In Proceedings of the 2nd Workshop on Block Chain Technologies 4 Multi-Agent Systems (BCT4MAS), Co-Located with PAAMS 2019, Avila, Spain, 26–28 June 2019.
24. Amoussou-Guenou, Y.; Del Pozzo, A.; Potop-Butucaru, M.; Tucci-Piergiovanni, S. *Correctness and Fairness of Tendermint-Core Blockchains*; Research Report; LIP6 UMR 7606; UPMC Sorbonne Universités: Paris, France, 2018.

25. Kwon, J. Tendermint: Consensus without Mining; White Paper. 2014. Available online: https://pdfs.semanticscholar.org/df62/a45f50aac8890453b6991ea115e996c1646e.pdf (accessed on 29 November 2019).
26. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed.; Prentice Hall Press: Upper Saddle River, NJ, USA, 2009.
27. Merkle, R.C. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology—CRYPTO '87: Proceedings*; Pomerance, C., Ed.; Springer: Berlin/Heidelberg, Germany, 1988; pp. 369–378. doi:10.1007/3-540-48184-2_32. [CrossRef]
28. Gutknecht, O.; Ferber, J. The MadKit Agent Platform Architecture. In *Workshop on Infrastructure for Scalable Multi-Agent Systems at the International Conference on Autonomous Agents*; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1887. doi:10.1007/3-540-47772-1_5. [CrossRef]
29. Pass, R.; Shi, E. FruitChains: A Fair Blockchain. In Proceedings of the ACM Symposium on Principles of Distributed Computing, Washington, DC, USA, 25–27 July 2017; ACM Press: New York, NY, USA, 2017; pp. 315–324. doi:10.1145/3087801.3087809. [CrossRef]
30. Herlihy, M.; Moir, M. Enhancing accountability and trust in distributed ledgers. *arXiv* **2016**, arXiv:1606.07490.
31. Snow, P.; Deery, B.; Kirby, P.; Johnston, D. Factom Ledger by Consensus. 2015. Available online: https://factomize.com/uploads/FactomLedgerbyConsensus.pdf (accessed on 29 November 2019).
32. Kosba, A.; Miller, A.; Shi, E.; Wen, Z.; Papamanthou, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In Proceedings of the 2016 IEEE symposium on security and privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 839–858.
33. Zyskind, G.; Nathan, O.; Pentland, A. Enigma: Decentralized computation platform with guaranteed privacy. *arXiv* **2015**, arXiv:1506.03471.
34. Jalalzai, M.M.; Richard, G.; Busch, C. An Experimental Evaluation of BFT Protocols for Blockchains. In *Blockchain—ICBC 2019*; Joshi, J., Nepal, S., Zhang, Q., Zhang, L.J., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 34–48.