Game Theoretical Analysis of DAG-Ledgers Backbone

Simone Galimberti, Maria Potop-Butucaru

Game Theoretical Analysis of DAG-Ledgers Backbone

Simone Galimberti[1] and Maria Potop-Butucaru[1]

LIP6, UMR 7606 Sorbonne University - CNRS, 4 place Jussieu 75252 Paris Cedex 05, France

Abstract. We study the rational behaviors of participants in *DAG*-Based Distributed Ledgers. We analyze generic algorithms that encapsulate the main actions of participants in a *DAG*-based distributed ledger: voting for a block, and checking its validity. Knowing that those actions have costs, and validating a block gives rewards to users who participated in the validation procedure, we study using game theory how strategic participants behave while trying to maximize their gains. We consider scenarios with different type of participants and investigate if there exist equilibria where the properties of the protocols are guaranteed. The analysis is focused on the study of equilibria with trembling participants (i.e. rational participants that can do unintended actions with a low probability). We found that in presence of trembling participants, there exist equilibria where protocols properties may be violated.

Keywords: Blockchains, · Game Theory, · Directed Acyclic Graph.

1 Introduction

Blockchain and distributed ledger technologies, [2], have emerged as one of the most revolutionary developments in recent years, with the goal of eliminating centralised intermediaries and installing distributed trusted services. They facilitate trustworthy trades and exchanges over the Internet, power cryptocurrencies, ensure transparency for documents, and much more. Traditionally, blockchain systems maintain a continuously-growing list of ordered blocks that include one or more transactions that have been verified by the members of the system, called miners. Blocks are linked using cryptography and the order of blocks in the blockchain is the result of a form of agreement among the system participants. After the releasing of the most popular blockchains (e.g., Bitcoin or Ethereum) with a specific focus on economical transactions their huge potential for various other applications became evident. However, quickly after their release blockchains reached their limits in terms of throughput, blocksize and unforeseen behaviors. Therefore, non academic research further concentrate in proposing alternatives by improving some performance aspects but with non zero costs either on security or throughput. One of these solutions extends the blockchain to a DAG overlay and provide an ordering over all blocks and transactions, and outputs a consistent set of accepted transactions. In the research

along this line transactions are still organized on blocks. All these DAG-based systems structure blocks in a Directed Acyclic Graph. Each block can include several references to predecessors. The objective of this paper is to analyse from the game theoretical point of view a generic backbone of that captures the characteristics of existing DAG-based blockchains (e.g. Spectre [5], GhostDAG [6], Phantom [7], IOTA [4] etc) into a unified framework.

It should be noted that the analysis of strategic behaviors in blockchains (see [1] for an extended state of the art). To the best of our knowledge, no work has been dedicated to analyze or discuss the rational behaviors among participants in *DAG*-based blockchains. The only work proposed in the context of the IOTA DAG ([3]) studies the attachment preferences.

In this paper we extract the backbone of a DAG-based blockchain we analyze the behavior of rational participants. We show the different equilibria that exist given different scenarios where other type of participants are present. We analyze if the equilibria do satisfy the DAG-backbone invariants. Additionally, we used the notion of "trembling hand". The trembling hand can be viewed as a failure of rational participants. With low probability, the player can tremble and do an unintended action.

2 Model and Problem Definition

We consider a system composed of a finite and ordered set Π of n players, of synchronous sequential players denoted by their index, namely $\Pi = \{1, \ldots, n\}$.

Each participant has an internal state and proceeds in rounds, it is important to note that users are not synchronized. Every user has an internal clock that is different from the others so in every phase the transmission of blocks or informations may have delay compared to the sending time of other participants in the network. At the end of each round, the participant goes to the next round and so on. A round is composed of three phases:

- A *send phase*: It is during the send phase that a participant can send a block to the other participants. The block to send should be prepared during the previous rounds. After the send phase, the participant goes to the delivery phase.
- A *delivery phase*: during this phase, a participant collects from the network blocks previously sent. We consider that participants have an unbounded memory, *i.e.*, they can store all blocks they collect. When a participant effectively collects a whole block and can process it, we say that it has delivered the block, otherwise, even if part of the block is collected, it has not delivered the block. The delivery phase has a finite and positive duration. The duration represents the time the participant plans to wait for collecting messages sent. After its delivery phase, the participant goes to the compute phase.
- A *compute phase*: In compute phase, a participant uses all blocks it delivered to update its local *DAG* and to prepare the block for the next round if it will

issue one. After the compute phase, the participant goes to the send phase of the next round.

Participants communicate by sending and receiving messages through a *synchronous network*. We assume that each participant proceeds in rounds. A *round* consists of three sequential phases, in order: the send, the delivery, and the compute phases. The delivery phase has a fixed duration that allows collecting all the messages sent by the participants. At the end of a round, a participant exits from the current round and starts the next one. We assume the existence of a *reliable broadcast* primitive (see Appendix for the formal definition).

When a participant i delivers a message, it knows the participant j that created the message.
When a participant sends a message, other participants do not deliver the block instantaneously, there is a *message (transmission) delay* or a *transmission latency*. Let $\Delta \geq 0$ be the maximum message delay between any two participants, *i.e.*, $\forall i, j \in \Pi$, if i sends a message to j at date (time) t, then at date $t + \Delta$, j has delivered the message.

- **Synchronous Communication:** In synchronous communication systems, Δ is finite and known by the participants. If a participant broadcasts a block at a time t, then at time $t + \Delta$, all participants have delivered the block.

Since the maximum delay Δ is known, participants can use that information in their execution; typically, they can match the duration of their delivery phase with Δ to ensure the reception of all blocks sent at the beginning of the round.

In a blockchain system, all participants do not necessarily have the same behaviour. The first two types of participants are the classical types we can find in the distributed systems' literature: the *correct* and the *Byzantine* participants. Both correct and Byzantine participants do not care about the costs of their actions, nor the global execution of the system. For the sake of clarity and consistency, we use the terminology *obedient* instead of correct.

A participant is *obedient* with respect to a protocol $\mathcal{A}$ if always follows protocol $\mathcal{A}$.

A participant is Byzantine with respect to a protocol $\mathcal{A}$ if she can arbitrarily deviate from protocol $\mathcal{A}$.

A participant is *rational* if she is self-interested. It does an action instead of another action if and only if doing so increases her expected gain.

A rational participant has preferences and takes actions to have her most preferred execution of the system, taking into account what the other participants do. She does not necessarily want to hurt the system; she wants to benefit from it. We now introduce two refinements of the rational participants: the *strategic*, and the *malicious* participants. Having a property on the system's executions, one can know whether a given execution satisfies that property. In particular, a property defines a partition on the executions of the system; in one hand, the executions that satisfy the property and on the other hand, the executions that do not satisfy the property.

A participant is *strategic* with respect to a property $\mathcal{P}$ ifshe is rational, and she prefers executions that satisfy $\mathcal{P}$, *i.e.*, she assigns a positive value to executions where $\mathcal{P}$ is satisfied, and a negative value to executions where $\mathcal{P}$ is not satisfied.

A strategic participant prefers the executions where the system achieves its properties, but think about her own interest in the first place.

A participant is *malicious* with respect to a property $\mathcal{P}$if she is rational, and it prefers executions that do not satisfy $\mathcal{P}$, *i.e.*, she assigns a positive value to executions where $\mathcal{P}$ is not satisfied.

A malicious participant's objective is to hurt the system. A malicious participant takes all actions she can to increase the chance of having a global execution that does not satisfy the properties of the system.

Problem definition. We study the behaviour of strategic participants in a DAG-based ledger. Each node in the network maintains locally a data-structure *Directed Acyclic Graph* which is an extension of the traditional linear sequence of *blockchains*. Each node of the DAG data-structure is a block containing one or more transactions (in the case of IOTA for example each node in the DAG is a single transaction). A *block* $\mathcal{B} \in$ contains the *(i) identifier* of the user $w$hich built $\mathcal{B}$, *(ii)* a transaction tr, *(iii)* a couple of hashes of *parents* blocks in the DAG structure.

A DAG-based ledger verifies the following properties:

- *DAG-Growth.* For every user i given two different instants of time $t_2 > t_1$ it follows that $DAG_i^{t_1} \subseteq DAG_i^{t_2}$, which means $DAG_i^{t_1}$ is a prefix of $DAG_i^{t_2}$;
- *DAG-Uniformity.* For every two users i, j, in the absence of new blocks appending, eventually $DAG_i = DAG_j$;
- *DAG-Liveness.* If a correct user, i, appends a correct block $\mathcal{B}$ to DAG_i, all correct users, j, eventually append $\mathcal{B}$ to their own DAG_j;
- *DAG-Validity.* For any correct user, j, each block in DAG_j is valid; it satisfies a predefined validity predicate;

3 DAG-Based Ledger Backbone Protocol with Obedient Users

We propose the backbone of a DAG-Based Ledger with only obedient participants and one correct issuer.

Algorithm 1: Issuer i Block $\mathcal{B}$

1 $Tips := select\_tips.(DAG_i)$
2 $\mathcal{B} := generate\ (Tips, tr)$
3 $DAG_i := update(DAG_i, \mathcal{B})$
4 **Broadcast**$(< \mathcal{B} >)$
5 **Wait** Δ

A *selection tip algorithm* is run by the function *select_tips*(DAG_i) (*line 1*) to select the tips to approve in the local DAG_i, *i.e.*, the protocol IOTA perform a *random walker algorithm* to select the best two tips, this procedure is computational expensive compared to a randomized choice but the former has many advantages: *(i)* Discourages lazy behaviour and encourages approving fresh tips, *(ii)* continuously merges small branches into a single large branch, thus increasing confirmation rate.*(iii)* In case of conflicts, kills off all but one of the conflicting branches. Issuer i embedded (*line 2*) the block $\mathcal{B}$ with her own id, the transaction tr and the hashes referring to the parents.

Finally the user updates her own DAG_i (*line 3*) and broadcast (*line 4*) the block to other users and wait a time Δ to be sure that anyone receives the block.

Algorithm 2: Receiver i Block $\mathcal{B}$

1 **Upon receiving** $\mathcal{B}$ **:**
2 $DAG_i := update(DAG_i, \mathcal{B})$

We note that in this simply version of the protocol the user receive a block, without checking the validity since it is guaranteed, then she updates her own local DAG_i. The following lemma proves that Algorithms 1 and 2 verify the properties of a DAG-based ledger.

Lemma 1. *Algorithms 1 and 2 verify DAG-Liveness, DAG-Uniformity, DAG-Validity and DAG-Growth.*

Proof.
- *DAG-Liveness*: *DAG-Liveness* property is straightforward. Indeed, every the user who receives a block, she will be append it since it is valid.
- *DAG-Validity*: Since there is no trembling issuer neither malicious participant each block is valid by construction.
- *DAG-Uniformity*: Assume by contradiction that $DAG_i \neq DAG_j$ this means that one the two user has a block in her own DAG that is not included in the other tree. This is not possible, since we have a unique issuer who issues the same blocks and broadcast to all users, so every users receive a valid block and adds it to the DAG.
- *DAG-Growth*: Given two different instants of time $t_2 > t_1$ we have two cases: no block is issued between t_2 and t_1, or some blocks are issued. In the first case $DAG_i^{t_2} = DAG_i^{t_1}$, while if blocks are issued these are added to the DAG since they are valid blocks and so $DAG_i^{t_1} \subseteq DAG_i^{t_2}$.

Remark 1. Note that in this simplified version we can not talk about Nash Equilibrium since there are no strategic players, there is no game but only users who follows prescribed instructions.

Remark 2. The case where the issuer produces only valid blocks is not interesting from a game theoretical point of view because in that scenario there would not

be the risk of updating the structure with invalid blocks. The most challenging scenario is when an invalid block may be produced by every user even if obedient or rational.

Due to space limitations we propose the backbone of a DAG-Based Ledger with obedient participants and a trembling hand issuer in the Appendix.

4 DAG-Based Distributed Ledger Backbone with Strategic Players

In this section we study the backbone of DAG-Based Distributed Ledgers considering strategic players. In this framework users will try to maximize their own utility, so they may decide not to follow the protocol, *i.e.*, we talk about behaviours of users that may decide to make an action or not. For this reason it is necessary now to introduce a system of reward and penalty to incentivate players to behave correctly.

In the following we assume that when an invalid block is inserted, all participants are punished by incurring in a cost κ. While a reward R, is given to the participants when a block is produced.
We already presented before, the cost c_{check} of checking validity, moreover another action will be possible for users which will be able to communicate by paying c_{send} of sending a message in order to coordinate.
Additionally, we assume that the reward obtained when a block is produced is smaller than the cost κ of producing an invalid block and the cost of checking is greater than sending. That is,

$$\kappa > R > c_{check} > c_{send} > 0.$$

Note that when a block is produced, in protocols which allow communication only the participants which sent a message are rewarded (and receive R).

4.1 Correct Issuer and No Communication between Users

In this section we consider the situation where the issuer is correct and strategic users do not communicate with each other.

Algorithm 3: Issuer i Block $\mathcal{B}$

1 $Tips := select\_tips.(DAG_i)$
2 $\mathcal{B} := generate\ (Tips, tr)$
3 **Broadcast**$(< \mathcal{B} >)$
4 $DAG_i := update(DAG_i, \mathcal{B})$
5 **Wait** Δ

Algorithm 4: Strategic Receiver i Block $\mathcal{B}$

```
1  Initialisation:
2    action^check :=∅
3    validValue:= false                          // validValue ∈{true,false}
4  Upon receiving B :
5    action^check← σ_i^check()    // σ_i^check() ∈ {true,false} sets the action of
                                  // checking or not the validity of the block
6    if action^check == false then
7      DAG_i:= update(DAG_i, B)
8    if action^check == true then
9    validValue ← isValid(B)   // The execution of isValid(B) has a cost
                                                                  // c_check
10     if validValue == true
11       DAG_i:= update(DAG_i, B)
```

Strategic participants choice is represented in Algorithm 4 by dedicated variable, namely, $action^{check}$. The action, initialized at a default value $\varnothing$ (*line 2*), can take value from the set {`false`,`true`}. For participant i, the value of $action^{check}$ is set by calling respectively the functions $\sigma_i^{check}()$, returning its strategy (*line 5*). The strategy $\sigma_i^{check}()$ determines if i, the receiving participant chooses to check the validity of the proposal or not, which is a costly action.
When the issuer is correct no user will choose to check the validity (*line 5*) paying a cost c_{check}.
It is not in her interest paying when they know for sure that the block is valid.

4.2 Trembling Issuer and No Communication between Users

In this section we consider the situation where the issuer is trembling and strategic users do not communicate with each other.

Algorithm 5: Trembling Issuer i Block $\mathcal{B}$

```
1  Tips := select_tips.(DAG_i)
2  B := generateTrembling (Tips, tr)
3  Broadcast(< B >)
4  DAG_i:= update(DAG_i, B)
5  Wait Δ
```

Algorithm 9 (see Appendix) is the same of Algorithm 4, but in this case the issuer is trembling, so participants will actually choose to check the validity (*line 8*) or not. The choice will depend strictly on the relation between the reward and the penalty, users decide if it is worth paying the cost of checking or not,

while when the issuer was correct no one pay for double check the correctness of the proposal.

Proposition 1. *Participants will not check the validity if and only if* $\kappa < c_{check}/p + R$.

Proof. First we prove that if the inequality holds players will not check the validity:
If a strategic player does not check the validity of a block her average gain is $R - p\kappa$, since she will always include the block in her own DAG and receives the penalty of an invalid block with probability p. On the other hand if the user checks, her average gain is $(1-p)R - c_{check}$, she will always pay the cost of checking but getting the reward only when the block is valid and so added to the DAG. Not checking is the best strategy if $R - p\kappa > (1-p)R - c_{check}$, solving this inequality we get $\kappa < c_{check}/p + R$.
Now we prove that if users are not checking the inequality holds:
If a strategic player decides not to check this means that the utility is grater compared to the action of checking: $R - p\kappa > (1-p)R - c_{check}$ which solved gives $\kappa < c_{check}/p + R$.

Lemma 2. *Algorithms 5 and 9 verify DAG-Uniformity, DAG-Growth and DAG-Liveness. DAG-Validity holds if* $\kappa \geq c_{check}/p + R$.

Proof.
- *DAG-Validity*: When a trembling user is present each block may be checked or not, if $\kappa \geq c_{check}/p + R$ holds by *Proposition 1* only valid block will be appended while invalid block discarded. Otherwise also invalid blocks will be added.
- *DAG-Liveness*: When a block is issued there are two possible scenarios. The users do not check the validity of the block, if $\kappa < c_{check}/p + R$ holds, otherwise the block is checked. In both cases valid blocks will be appended.
- *DAG-Uniformity*: Note that by hypothesis at time t_0 the local DAG_p of each user p is initialized with the same genesis block.
  - *Base case:* Let $\mathcal{B}$ be the first block issued by q after t_0. When the inequality of *Proposition 1* does not hold if the block is checked and is valid, then $\mathcal{B}$ is appended to the genesis block. Otherwise if the block is checked and it is invalid no user will append it to her own DAG. If the inequality of *Proposition 1* holds the block will be appended in any case. Let DAG_p the tree of p at time $t_0 + \Delta$. By Δ time any users delivered $\mathcal{B}$ by the broadcast primitive, hence the DAG of any process is identical to DAG_p. It follows that δ time after the first proposal, any two correct processes i and j verify the property: $DAG_i = DAG_j$.
  - *Inductive case:* Suppose the lemma is true after a sequence of h blocks and prove it holds for the $h+1$ block. Let t be the time when the h block is proposed. By inductive hypothesis, we have that by time $t + \Delta$, all users have the same DAG. Let $t'(t' \geq t + \Delta)$ be the time when the $h+1$ block is invoked by q. Let DAG_q be the local tree of q. By inductive hypothesis, by time t', all the other users have the same local DAG. Two cases have to be considered:

  - The block proposed at time t' is valid : at time t' the user checks (or does not check) the validity of the block. Since by assumption of the inductive case they have the same tree as q, then by time $t' + \Delta$, $\mathcal{B}$ is appended to the DAG. It follows that any two honest users i and j verify the property: $DAG_i = DAG_j$.
  - The block proposed at time t' is invalid: in this case if users check the validity, the block $\mathcal{B}$ is not appended, otherwise they update the local DAG.

– *DAG-Growth*: Given two different instants of time $t_2 > t_1$ we have two cases: no block is issued between t_2 and t_1, or some blocks are issued. In the first case $DAG_i^{t_2} = DAG_i^{t_1}$, while when blocks are issued if they are checked (or not checked and added) and are valid these are added to the DAG and so $DAG_i^{t_1} \subseteq DAG_i^{t_2}$.

4.3 Trembling Issuer and Communication between Users

Now we add the broadcast of the opinion of users about the validity of the block $\mathcal{B}$. After the checking phase each user will receive the decisions of the others and according to a simple majority rule she will add or not the block to the DAG_i. To ensure that the reward covers the costs of checking and sending in this setting we assume that $(1-p)(R - c_{send}) - c_{check} > 0$, that is, the reward covers the costs. We also note that it is better for the participants to send (resp. not to send) without checking than checking and sending (resp. not sending) irrespective to the block validity; that would mean incurring a cost $-c_{check}$ for nothing. It is also not in their best interest to check the validity of the proposal and vote if the proposal is invalid, that would mean increasing the chances of producing an invalid block and incurring a cost $-\kappa$.

Strategic participants may decide to check the validity of the block $\mathcal{B}$ paying a cost c_{check} (*line 8*), in the same way they can also decide if sending or not the validity of the block paying an additional cost c_{send} (*line 11*). In this version of the protocol users can communicate and for this purpose two more variables are needed. *timerVote* (*line 2*) is a timer which allows a user to receive every vote expressed by the others, this is strictly required since every participants has his own time clock and maybe some delay is present between votes' arrival. From *line 14* to *line 17* the receiver must wait Δ and stores the votes in the vector *votes* which were initialised as empty.

In the *Compute* phase each user receives the opinions of the others and insert the block $\mathcal{B}$ according to the majority.

Due to space limitation the correctness of Algorithm 6 is proposed in the Appendix.

Algorithm 6: Receiver i Block $\mathcal{B}$ (Strategic Participant)

Initialisation:
  $timerVote := 0$ // Timer for receiving votes
  $votes[] := \{\varnothing,...,\varnothing\}$ // Vector storing the votes about the validity of a block
  $action^{check} := \varnothing$
  $action^{send} := \varnothing$
  $validValue :=$ `false` // $validValue \in$ {true,false}
Upon receiving $\mathcal{B}$ **:**
  $action^{check} \leftarrow \sigma_i^{check}()$ // $\sigma_i^{check}() \in$ {true,false} sets the action of checking or not the validity of the block
  if $action^{check} ==$ `true` **then**
    $validValue \leftarrow isValid(\mathcal{B})$ // The execution of $isValid(\mathcal{B})$ has a cost c_{check}
    $action^{send} \leftarrow \sigma_i^{send}(validValue)$ // $\sigma_i^{send}() \in$ {true,false} sets the action of sending or not the validity of the block
    if $action^{send} ==$ `true` **then**
      broadcast $(< validValue >)$ // The execution of the broadcast has a cost c_{send}
set $timerVote$ to Δ
while $TimerVote$ not expires **do**
  upon receiving $< validValue_j >$ **:**
    $votes_j \leftarrow validValue_j$ // The participants collect all the opinions on the validity of the block
Compute phase:
  if $|< votes >==$ `true` $|>|< votes >|/2$
    $DAG_i := update(DAG_i, \mathcal{B})$

5 Game Theoretical Analysis for Trembling Issuer and Rational and Byzantine Users

In this section we describe the game, *i.e.*, the possible actions of players,the game tree and finally the utility.

Action space. After receiving the proposal block, each participant decides whether to check the block's validity or not (at cost c_{check}), and second decides whether to send a message (at cost c_{send}) or not.

Information sets. At the beginning of each round $t > 1$, the information set of the participant, η_i^t, includes the observation of the round number t, the participant's own type θ_i, as well as the observation of what happened in previous rounds, namely (i) whether the participant decided to check validity, and in that case, she knows the validity of the block, (ii) whether the participant decided to send the validity. Then, in each round $t > 1$, the participant decides whether to

check the validity of the current block. At this point, denoting by $\mathcal{B}_t$ the block proposed at round t, when the participant does not decide to check validity, the *isValid($\mathcal{B}_t$)* function is set to `false`, while if the participant decides to check, *isValid($\mathcal{B}_t$)* is equal to `true` if the block is valid and `false` otherwise. Therefore, at this stage, the participant information set becomes $H_i^t = \eta_i^t \cup isValid(\mathcal{B}_t)$, which is η_i^t augmented with the validity information participant i has about $\mathcal{B}_t$, the proposed block.

Strategies. At each round $t \geq 1$, the strategy of participant i is a mapping from her information set into her actions. At the point at which the participant can decide to check block validity, her strategy is given by $\sigma_i^{check}(\eta_i^t)$. Finally, after making that decision, the participant must decide whether to send a message or not, and that decision is given by $\sigma_i^{send}(H_i^t)$. The decision tree of a participant is depicted in Figure 1. We note that when the participant does not check the validity of the proposal, she does not know if the block is valid or not. We denote by $\sigma = (\sigma_1, ..., \sigma_n)$, the *strategy profile* where $\forall i \in \{1, ..., n\}$, participant i uses strategy σ_i , where $\sigma_i(H_i^t)$ is the pair $(\sigma_i^{check}(\eta_i^t)$,$\sigma_i^{send}(H_i^t))$.

Objective of Strategic Participants. The expected gain of strategic participant i is:

$$U_i = E\Big[R \times \mathbb{1}_{(\sigma_i^{send}(H_i^t)=\texttt{true})} \times \mathbb{1}_{(valid\ block\ produced)} - \kappa \times \mathbb{1}_{(invalid\ block\ produced)} - c_{check} \times \mathbb{1}_{(\sigma_i^{check}(\eta_i^t)=\texttt{true})} - c_{send} \times \mathbb{1}_{(\sigma_i^{send}(H_i^t)=\texttt{true})} \mid \eta_i^t\Big]$$

where $\mathbb{1}(.)$ denotes the indicator function, taking the value 1 if its argument is `true`, and 0 if it is `false`.

5.1 Equilibria with only Strategic Participants

In this section we analyse the different Nash Equilibria which may be possible in the framework of a trembling issuer with strategic players who communicate described in Algorithm 6.

Proposition 2. *With only strategic participants and if there is a probability p that the proposer proposes an invalid block, there may be two type of Nash equilibria. In equilibrium either (i) if $\kappa < min\{R/p - c_{send}/p, R - c_{send} + c_{check}/p\}$, all participants do not check the validity of the proposal and vote it as valid, (ii) if $\kappa > R + c_{check}/p - c_{send}$, $\lfloor n/2 \rfloor + 1$ participants check the validity and send the vote, since all the players are simmetric we have $\binom{n}{\lfloor n/2 \rfloor + 1}$ equilibria.*

Proof. We prove that the strategy profiles described in the proposition are Nash equilibria.

First, we prove that the strategy profile where all participants do not check

the proposal validity and vote is a Nash equilibrium.

(i) The expected gain at equilibrium of any participant is $R - c_{send} - p\kappa$. If one participant deviates and does not send the vote, her gain at deviation is 0; if she deviates checking, the gain is $(1-p)(R - c_{send}) - c_{check}$. These utilities are lower than the gain at equilibrium if and only if $\kappa < min\{R/p - c_{send}/p, R - c_{send} + c_{check}/p\}$, which is our assumption, therefore, the gain at equilibrium is better than the gain if the participant deviates. The strategy profile where all participants don't check the proposal validity is a Nash equilibrium.

We now prove that the strategy profile where $\lfloor n/2 \rfloor + 1$ participants check the proposal validity but all users vote is a Nash equilibrium.

(ii) The expected gain at equilibrium of participants who check is $(1-p)(R - c_{send}) - c_{check}$. This gain is positive by assumption and consistency that reward covers the costs, so no users will deviate not sending. If a participant deviates not checking her gain at deviation is $R - c_{send} - p\kappa$, which is lower than gain at equilibrium if and only if $\kappa > R + c_{check}/p - c_{send}$, which is our assumption, therefore, the gain at equilibrium is better than the gain if the participant deviates. The strategy profile where $\lfloor n/2 \rfloor + 1$ participants check the proposal validity and vote is a Nash equilibrium.

Remark 3. There are two Nash Equilibria in Proposition 2. In the equilibrium where all participants do not check, if the proposal is invalid, there is no *DAG-Validity*, however, *DAG-Liveness*, *DAG-Uniformity* and *DAG-Growth* are always ensured. While in the second equilibrium where $\lfloor n/2 \rfloor + 1$ participants check we are in the case of Lemma 4 so all invariants hold.

Proposition 3. *With $\lfloor n/2 \rfloor$ obedient participant, if there is a probability p that the proposer proposes an invalid block, if $\kappa > R + c_{check}/p - c_{send}$ there are $\lfloor n/2 \rfloor$ Nash Equilibria where one strategic player checks the validity. (This case can be extended when there are less than $\lfloor n/2 \rfloor$ obedient participants,* i.e.*, if there are $\lfloor n/2 \rfloor$ -1 obedient the Nash Equilibria are $\binom{\lfloor n/2 \rfloor + 1}{2}$ when two strategic participants check the validity).*

Proof. We prove that the strategy profiles described in the Proposition 3 are Nash Equilibria.
The gain at equilibrium of the strategic participant who checks is $(1-p)(R - c_{send}) - c_{check}$. If she deviates and does not check, her gain at deviation is $R - c_{send} - p\kappa$, which is lower than gain at equilibrium if $\kappa > R + c_{check}/p - c_{send}$, which is our assumption, therefore, the gain at equilibrium is better than the gain if the participant deviates. The strategy profile where one participant checks the proposal validity and vote is a Nash equilibrium.

Remark 4. In these equilibria we are in the case of Lemma 11 so all invariants hold.

Equilibria with Faulty Participants We analyse the situation where faulty participants are present. A Faulty participant is a user that crashes and stops following the protocol (the proof is similar to *Proposition 2, Proof (i)).*

Proposition 4. *With faulty participants and if there is a probability p that the proposer proposes an invalid block, there may be a Nash equilibrium. If* $\kappa < min\{R/p - c_{send}/p, R - c_{send} + c_{check}/p\}$ *, in the equilibrium all participants do not check the validity of the proposal and vote it as valid.*

Remark 5. In the equilibrium if the proposal is invalid, there is no *DAG-Validity*, however, *DAG-Liveness*, *DAG-Uniformity* and *DAG-Growth* are always ensured.

5.2 Equilibria with Rational Malicious Participants

We consider now a new type of users **b** *rational malicious participants* whose objective is harming the network but they incur, differently from malicious ones, in the costs for checking and sending, their utility is defined as following:

Payoff of Rational Malicious Participants. Let ω be an outcome of the game. If ω does not satisfy *DAG-Validity*, then the malicious participants have a payoff of $\tilde{\kappa}_{Validity} > 0$.

$$\tilde{\kappa}(\omega) = \begin{cases} \tilde{\kappa}_{Validity}, & \text{if } \omega \text{ does not satisfy } DAG\text{-}Validity \\ 0, & \text{if } \omega \text{ satisfies } DAG\text{-}Validity \end{cases} \tag{1}$$

Objective of Rational Malicious Participants. The expected gain of malicious participant i is:

$$\tilde{U}_i = E\Big[\tilde{\kappa}_{Validity} \times \mathbb{1}_{(invalid\ block\ is\ produced)} - c_{check} \times \mathbb{1}_{(\sigma_i^{check}(\eta_i^t)=\texttt{true})} - c_{send} \times \mathbb{1}_{(\sigma_i^{send}(H_i^t)=\texttt{true})} \mid \eta_i^t\Big].$$

Proposition 5. *When the proposer is trembling rational, there may be a Nash Equilibrium. We indicate with* **b** *the number of rational malicious participants and with* **r** *the number of strategic ones. If:*

$$\kappa > R - c_{send}/p + \frac{c_{send} - R + c_{check} + \mathbb{P}(\mathbf{r} > \mathbf{b})[R(1-p) - c_{send}]}{p(1 - \mathbb{P}(\mathbf{r} > \mathbf{b}))}$$

All strategic participants check the validity and send the vote, while rational malicious participants do nothing if $\mathbf{r} > \mathbf{b}$*, they all send the vote without checking when* $\mathbf{r} < \mathbf{b}$*.*

Proof. We prove that the strategy profile described in the proposition is a Nash Equilibrium.

The expected gain at equilibrium of strategic participant is $(1-p)(R-c_{send})-c_{check}$. This gain is positive by assumption and consistency that reward covers the costs, so no users will deviate not sending. Since a strategic participant does not know the exact number of the malicious ones, she can only compute the expected reward when deviating not checking, *i.e.*, if $\mathbf{r} > \mathbf{b}$, the gain is $(1-p)R-c_{send}$, while it is $Rp-c_{send}-p\kappa$ when $\mathbf{r} < \mathbf{b}$. These utilities are lower than the gain at equilibrium if and only if:

$$\kappa > R - c_{send}/p + \frac{c_{send} - R + c_{check} + \mathbb{P}(\mathbf{r} > \mathbf{b})[R(1-p) - c_{send}]}{p(1 - \mathbb{P}(\mathbf{r} > \mathbf{b}))},$$

which is our assumption, therefore, the gain at equilibrium is better than the gain if the participant deviates. Regarding malicious rational participants, we can suppose that in the worst case scenario they know how many they are, and so, they will not check neither send the vote if $\mathbf{r} > \mathbf{b}$ in order to avoid any useless cost. While if $\mathbf{r} < \mathbf{b}$, all malicious strategic participants will send the vote without checking with a gain of $p\tilde{\kappa}_{Validity} - pc_{send}$. This is an equilibrium for the malicious rational participants, if one deviates its reward would be 0.

The strategy profile where all strategic participants check the proposal validity and vote is a Nash equilibrium.

Remark 6. We are in the case of Lemma 4 so all invariants hold. However $DAG-Validity$ property falls if $\mathbf{r} < \mathbf{b}$.

5.3 Equilibria with Malicious Participants

Finally we consider $\mathbf{m}$ *malicious participants*, these users do not incur any costs, no matter their actions, and their objective is damaging the network by validating invalid blocks. For this reason we can assume that they will always check the block and vote only if this in invalid. Strategic participants are aware of the presence of these other participants but they do not know their precise number.

Payoff of Rational Malicious Participants. Let ω be an outcome of the game. If ω does not satisfy *DAG-Validity*, then the malicious participants have a payoff of $\tilde{\kappa}_{Validity} > 0$, we also assume that malicious participants do not incur any costs, no matter their actions.

$$\tilde{\kappa}(\omega) = \begin{cases} \tilde{\kappa}_{Validity}, & \text{if } \omega \text{ does not satisfy } DAG\text{-}Validity \\ 0, & \text{if } \omega \text{ satisfies } DAG\text{-}Validity \end{cases} \tag{2}$$

Objective of Malicious Participants. The expected gain of malicious participant i is:

$$\tilde{U}_i = E\Big[\tilde{\kappa}_{Validity} \times \mathbb{1}_{(invalid\ block\ is\ produced)} \mid \eta_i^t\Big].$$

Proposition 6. *When the proposer is trembling rational, there may be one Nash equilibrium. We indicate with* $\mathbf{m}$ *the number of malicious participants and with* $\mathbf{r}$ *the number of rational ones. If:*

$$\kappa > R - c_{send}/p + \frac{c_{send} - R + c_{check} + \mathbb{P}(\mathbf{r} > \mathbf{m})[R(1-p) - c_{send}]}{p(1 - \mathbb{P}(\mathbf{r} > \mathbf{m}))}$$

All strategic participants check the validity and send the vote.

Proof. For the proof see *Proposition 5, Proof.* In this scenario malicious participants will always check and send the vote since they have no costs.

Remark 7. We are in the case of Lemma 4 so all invariants hold. However *DAG-Validity* property falls if $\mathbf{r} < \mathbf{b}$.

Considering $c_{send} > c_{check}$ it may happen that sending without checking is no more the best choice for strategic participants, since they would pay more and also incur in the risk of add an invalid block that can be avoided at a less price by checking. As in the previous case to ensure that the reward covers the costs of checking and sending in this setting we assume that $(1-p)(R-c_{send})-c_{check} > 0$, that is, the reward covers the costs.

By observing the equilibria presented above it is possible to see that they do not depend on the relation between c_{send} and c_{check}. For this reason even if we consider a cost of sending greater the cost of checking the equilibria will remain the same. However $c_{send} > c_{check}$ may lead in a modification of the values of the thresholds for these equilibria, *i.e.*, in the ones where all users check the validity the corresponding threshold that is in the assumption, will be much easily satisfied. On the other hand the condition for the equilibria where no one checks will be more stricter than the case where $c_{send} < c_{check}$.

6 Conclusion

We analyze the behavior of rational players in *DAG* based protocols. This paper studies the case where there is one proposer at the time. The case of multi proposers is left open. We found that in presence of trembling participants, there exist equilibria where the backbone properties may be violated. The results of our work can be used in improving the design of DAG-based ledgers. Our findings are summarized in Table 5.3.

Participants	DAG-Growth	DAG-Uniformity	DAG-Liveness	Validity
Only Strategic				
- No one check the validity	Yes	Yes	Yes	No
- $\lfloor n/2 \rfloor + 1$ users check	Yes	Yes	Yes	Yes
Strategic & Faulty				
- No one check the validity Proposition 4	Yes	Yes	Yes	No
Strategic & Rational Malicious				
- All users check Proposition 5	Yes	Yes	Yes	Yes
Strategic & Malicious				
- All users check Proposition 6	Yes	Yes	Yes	Yes

Table 1. Summary of the Equilibria.

References

1. Amoussou-Guenou, Y.: Governing the commons in blockchains. (Gouvernance des biens communs dans les blockchains). Ph.D. thesis, Sorbonne University, France (2020), https://tel.archives-ouvertes.fr/tel-03173517
2. Anta, A.F., Georgiou, C., Herlihy, M., Potop-Butucaru, M.: Principles of Blockchain Systems. Synthesis Lectures on Computer Science, Morgan & Claypool Publishers (2021). https://doi.org/10.2200/S01102ED1V01Y202105CSL014, https://doi.org/10.2200/S01102ED1V01Y202105CSL014
3. Popov, S., Saa, O., Finardi, P.: Equilibria in the tangle. Comput. Ind. Eng. **136**, 160–172 (2019). https://doi.org/10.1016/j.cie.2019.07.025, https://doi.org/10.1016/j.cie.2019.07.025
4. Popov, S.Y.: The tangle (2015)
5. Sompolinsky, Y., Lewenberg, Y., Zohar, A.: Spectre: A fast and scalable cryptocurrency protocol. IACR Cryptol. ePrint Arch. **2016**, 1159 (2016)
6. Sompolinsky, Y., Wyborski, S., Zohar, A.: PHANTOM GHOSTDAG: a scalable generalization of nakamoto consensus: September 2, 2021. In: Baldimtsi, F., Roughgarden, T. (eds.) AFT '21: 3rd ACM Conference on Advances in Financial Technologies, Arlington, Virginia, USA, September 26 - 28, 2021. pp. 57–70. ACM (2021). https://doi.org/10.1145/3479722.3480990, https://doi.org/10.1145/3479722.3480990
7. Sompolinsky, Y., Zohar, A.: PHANTOM: A scalable blockdag protocol. IACR Cryptol. ePrint Arch. p. 104 (2018), http://eprint.iacr.org/2018/104

Appendix

Specification of the Reliable broadcast

– **Byzantine Reliable Broadcast.** It ensures that a message sent by a correct process is received by all correct processes, and that all correct processes eventually receive the same set of messages. The service provides two operations, BRB-broadcast and BRB-delivery; the first broadcasts a message to all processes, and the second delivers a message that was previously broadcast. The service is used by the servers, and from their point of view, the BRB service guarantees the following properties:
  - *Validity*: if a correct process p_i BRB-delivers a message m from a correct process p_j, then p_j BRB-broadcast m.
  - *Integrity*: a message is BRB-delivered at most once by a correct server.
  - *Termination 1 (local)*: if a correct process BRB-broadcasts a message, it BRB-delivers it.
  - *Termination 2 (global)*: if a correct process BRB-delivers a message, all correct processes BRB-deliver it.

Validity relates outputs to inputs. Validity and integrity concern safety. Termination is on the fact that messages must be BRB-delivered; it concerns liveness. It follows that all correct processes BRB-deliver the same set of messages, which includes all the messages they BRB-broadcast.

Notions of Game Theory

One can defined *Game theory* as a mathematical formalisation of conflicts and cooperation between rational decision-makers in a given environment. Game theory has various domains of application as of computer science, economics, evolutionary biology, mathematics, political science, *etc.* It is mainly used to model and study rational behaviours between different participants. Using game theory appears to be useful according to our goal of studying rational behaviours in blockchain systems.
Formally, a game is a tuple $\mathcal{G} = (N, (A_i)_{i\in\mathbb{N}}, (\Theta_i)_{i\in\mathbb{N}}, (g_i)_{i\in\mathbb{N}})$, where:

– N is a non-empty set of participants;
– A_i is the set of actions of participant i;
– Θ_i is the type of participant i;
– g_i is the gain function of participant i;

Static Games. At a beginning of a static game, all participants choose simultaneously the actions they will play, without knowing which actions the others will choose. The gain of participant i is of the form $g_i : A_1 \times \cdots \times A_n \to \mathbb{R}$. Static games are also called simultaneous game. The most popular *simultaneous*

game is perhaps *rock-paper-scissors*.

Dynamic or Repeated Games. A repeated game is a sequence of static games where participants may have some information thanks to the previous static games played. We say that the first static game is the game at stage 1, the second is the game at stage 2, and so on. To accommodate the information each participant has, we denote by h_i^t the information set of participant i at time t. At each time of the game, each participant has an information set that contains information of what happened previously. Let $\mathcal{H}_i^t$ be the set of information sets participant i can have at time t. A *pure strategy* or simply a *strategy* of a participant i at time t is a function $\sigma_i^t : \mathcal{H}^t \to A_i$. A strategy of the participant is a family $(\sigma_i^t)_{t\geq}$. However, in the rest of the manuscript, we will write σ_i to represent the strategy of participant i, and we will denote by $\mathcal{S}_i$ the set of strategies of participant i. A vector of strategies for each participant $\sigma = (\sigma_1, ..., \sigma_n)$ is called a *strategy profile*. In repeated games, the gain of participant i is of the form $g_i : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \mathbb{R}$.

Complete Information Games. In complete information games, all participants know the other participants' information, their types, their gain function, their actions set, *etc.* At each time in the game, all participants know what actions the other participants did previously.

Incomplete Information Games. When a game is not with complete information, we say that it is an *incomplete information* game. In incomplete information games, at least one participant is not sure about the type of another participant. We can say that participants have private information that other participants do not know about. For example, it can be known that there is a malicious participant in the system, but except the malicious, no one else knows which participant it is. An initial distribution of the type of participants is set at the beginning of the game, but participants do not know it exactly. Participants only have at each time a probability distribution over the other participants' type; the probability distribution is updated following *Bayes' rule* and is kept in the information set of each participant. That is why incomplete information games are also called *Bayesian games*.
Let A and B be two events such that $\mathbb{P}[B] > 0$. *Bayes' theorem* also called *Bayes' rule* is the following equality:

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[B|A] \cdot \mathbb{P}[A]}{\mathbb{P}[B]}$$

Solution Concept. Now that the game is defined, we can try to predict the behaviour of rational participants. To do so, different concepts exist, the main one being the concept of *Nash equilibrium*. Intuitively, a strategy profile is a Nash equilibrium when each participant has a strategy that maximises its gain with respect to the other participants' strategies in the strategy profile. A Nash equilibrium can also be seen as a strategy profile where no participant can increase its gain by deviating alone from the strategy profile. Let $\sigma = (\sigma_1, \ldots, \sigma_n)$

be a strategy profile, and let $\sigma_i{}' \in \mathcal{S}_i$ be a strategy of participant i. We denote the strategy profile $(\sigma_1, ..., \sigma_{i-1}, \sigma_i{}', \sigma_{i+1}, ..., \sigma_n)$ by $(\sigma_i, \sigma_i{}').(\sigma_i, \sigma_i{}')$ is the strategy profile where participant i deviates by doing $\sigma_i{}'$ instead of σ_i, and all other participants continue with the same strategies. We can now formally define a Nash equilibrium.

Definition 1. *(Nash Equilibrium). A strategy profile σ is a Nash equilibrium if and only if:*

$$\forall i \in N, g_i(\sigma) \geq g_i((\sigma_{-i}, \sigma_i')).$$

The definition of a Nash equilibrium can be seen as too restrictive in some games, and the concept of *approximate Nash equilibrium* is used. An *ϵ-approximate Nash equilibrium* is a strategy profile where if one participant deviates, she can gain at most ϵ more than her gain at equilibrium. Formally, a strategy profile σ is an *ϵ-approximate Nash equilibrium* if and only if $\forall i \in N, g_i(\sigma) \geq g_i((\sigma_{-i}, \sigma_i')) - \epsilon$. Approximate Nash equilibria are not stable situations, since a participant can prefer to deviate, even if her gain is just ϵ, so we do not consider them in this manuscript. On another hand, and especially in repeated games, the concept of Nash equilibrium allows too many behaviours, and some are not interesting or coherent. The concept of *subgame perfect equilibrium* is defined to specifically capture equilibria that are coherent throughout the whole game. At any history, the "remaining game" can be regarded as a game on its own. We call such a remaining game a *subgame* of the game. Note that the whole game is also its own subgame.

Definition 2. *(Subgame Perfect Nash Equilibrium). A strategy profile σ is a subgame perfect Nash equilibrium if in all subgames, σ restricted to the subgame is a Nash equilibrium.*

Bayesian Equilibria. In Bayesian games, the concept of Nash equilibrium is also not well suited. In Bayesian game, since at least one participant is unsure of the type of other participants, a strategy profile may give more than one execution, according to the type distributions. An analogous concept to Nash equilibrium, the *Bayesian equilibrium* is defined for Bayesian games. Contrarily to Nash equilibrium, in a Bayesian equilibrium, each participant's goal is to maximise her expected gain, given her knowledge about the types' distribution; in particular, each participant's beliefs are consistent with Bayes' law when computing probabilities conditional on events that have positive probability on the equilibrium path. In Bayesian games, the gain function g is a probability distribution over the gain of all different executions that correspond to the given strategy profile.

Definition 3. *(Bayesian Equilibrium). A strategy profile σ is a Bayesian equilibrium at time t if and only if:*

$$\forall i \in N, \mathbb{E}\big[g_i(\sigma)|h_i^t\big] \geq \mathbb{E}\big[g_i(\sigma_{-i}, \sigma_i')|h_i^t\big].$$

As for Nash equilibria, the concept of subgame perfection exists in Bayesian games, and such equilibrium is called a *perfect Bayesian equilibrium.*

Definition 4. *(Perfect Bayesian Equilibrium). A strategy profile σ is called a subgame perfect Bayesian equilibrium if in all subgames, σ restricted to the subgame is a Bayesian equilibrium.*

A Comparison of Possible Executions from Participants' Type Let $\mathcal{A}$ be a protocol. We can see from Definition **??** that a rational participant can be viewed as a special subclass of Byzantine participants, rational participants can deviate from the protocol, but only if that deviation is beneficial to them. Let $\mathcal{C}$ be a class of rational participants such that their objective is to follow the protocol $\mathcal{A}$ no matter what is the global execution of the system and what the other participants do. The class $\mathcal{C}$ consists of exactly all the obedient participants with respect to $\mathcal{A}$, and only them. We can consider that some rational participants can behave as obedient, if their gain function is defined in that case. That exhibits that the behaviour of rational participants depends really on their gains. Therefore, the obedient participants are a subclass of rational participants where their objective is to follow the protocol. Usually, a protocol is designed to solve a problem, or equivalently to satisfy a property. It can then happen for strategic participants to behave as obedient. However, if running the protocol is (too) costly, strategic participants may deviate to improve their gain. The class of obedient participants is not a subclass of strategic participants, but both can intersect. As mentioned earlier, malicious and strategic participants are rational participants, but they have opposing objectives, so they do not intersect. A summary of the comparison of possible execution can be viewed in Fig. 2.

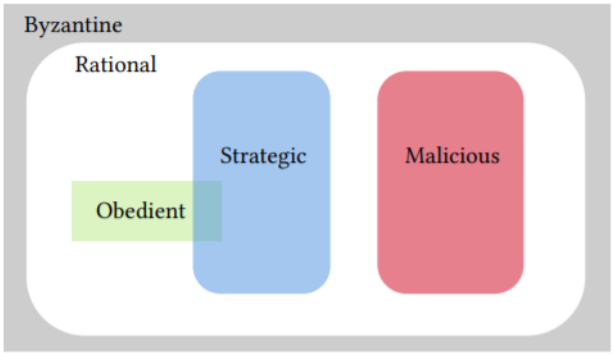

Fig. 1. A Comparison of Participant's Types

7 DAG-Based Ledger Backbone Protocol with Obedient Users: Trembling Issuer Case

Now, we assume that there is some negligible probability p for the issuer to generate an invalid proposal, and all participants are aware of the trembling effect. When proposing a value, there is a probability that the hand of the issuer trembles and proposes an invalid block instead of a valid block; *i.e.*, in some sense, we take into account the possibility of making a mistake for the proposal. Algorithm 7 corresponds to Algorithm 1, there is only one difference, the function that generates the block, the new *generateTrembling* encapsulates the probability of producing invalid transactions which were not present before.

Algorithm 7: Trembling Issuer i Block $\mathcal{B}$

1 $Tips := select\_tips.(DAG_i)$
2 $\mathcal{B} := generateTrembling(Tips, tr)$
3 **Broadcast**$(< \mathcal{B} >)$
4 **Wait** Δ

Since users may create unintentionally invalid blocks and everyone is aware about, a new action is available to the users, they may pay a cost c_{check} in order to check the validity of the block.
Note that checking the validity of a block may be important, there is a risk of producing an invalid block, breaking the *DAG-Validity* property.
The function *isValid* may be called for this purpose:

- *isValid*: $\mathcal{B} \rightarrow$ {`false`,`true`} is an application-dependent predicate that is satisfied if the given block is valid. If there is a block $\mathcal{B}$ such that $isValid(\mathcal{B})$ = `true`, we say that $\mathcal{B}$ is valid. We note that for any block, we set *isValid* to `false`, the validity of the block depends on the blockchain and the application, and *isValid* is known by all participants.

Obedient participants choice is represented in Algorithm 8. Upon receiving a block (*line 3*) they check the validity of the proposal, which is a costly action, and then update the knowledge about the validity of the proposal. Otherwise, the participants keep not knowing if the proposal is valid or not ($validValue$ remains `false`). Note that this value remains `false` even if the participant is the proposer. This is because we assumed, without loss of generality, that checking validity has a cost and that the only way of checking validity is by executing the $isValid(\mathcal{B})$ function.

Lemma 3. *Algorithms 7 and 8 verify DAG-Uniformity, DAG-Validity, DAG-Growth and DAG-Liveness.*

Proof. – *DAG-Liveness*: When a trembling user is present if the block produced is invalid no users will append the block since being obedient they

Algorithm 8: Receiver i Block $\mathcal{B}$ (Obedient Participant)

```
1 Initialisation:
2    validValue:= false                          // validValue ∈{true,false}
3 Upon receiving B :
4    validValue ← isValid(B)  // The execution of isValid(B) has a cost
                                                              // c_check
5 Compute phase:
6    if valiValue == true
7       DAG_i:= update(DAG_i, B)
```

will check the validity of the proposal and decide not to add the block. While if a user receive a valid block after checking it will be appended, and every obedient users will do the same.
- *DAG-Validity*: When a trembling user is present each block is checked, so only valid block will be appended while invalid block discarded.
- *DAG-Uniformity*: Assume by contradiction that $DAG_i \neq DAG_j$ this means that one the two user has a block in her own DAG that is not included in the other tree. This is not possible, since we have a unique issuer who issues the same blocks and broadcast to all users, so every users receive a block and if it is valid she adds it to the DAG.
- *DAG-Growth*: Given two different instants of time $t_2 > t_1$ we have two cases: no block is issued between t_2 and t_1, or some valid blocks are issued. In the first case $DAG_i^{t_2} = DAG_i^{t_1}$, while if blocks are issued these are added to the DAG since they are valid blocks and so $DAG_i^{t_1} \subseteq DAG_i^{t_2}$.

Remark 8. Note that in this simplified version we still can not talk about Nash Equilibrium since there are no strategic players, there is no game but only users who follows prescribed instructions.

7.1 Correctness of Algorithm 6

In the following we are interested about the correctness of Algorithm 6.

Lemma 4. *In the trembling issuer case and communication between users Algorithm 6 all invariants.*

Proof.
- *DAG-Validity*: When a trembling user is present each block must be voted before added to the DAG, for this reason only valid blocks will be appended while invalid blocks discarded.
- *DAG-Liveness*: When the issuer is trembling invalid blocks may be produced. When a user receives a block, after the voting system the block will be appended or considered as invalid. When a valid block is issued it will be voted by all the network and each correct user will update her own DAG.
- *DAG-Uniformity*: Note that by hypothesis at time t_0 the local DAG_p of each user p is initialized with the same genesis block.

Algorithm 9: Strategic Receiver i Block $\mathcal{B}$

Initialisation:
  $action^{check} := \varnothing$
  $validValue$:= `false` // $validValue \in$ {`true`,`false`}
Upon receiving $\mathcal{B}$ **:**
  $action^{check} \leftarrow \sigma_i^{check}()$ // $\sigma_i^{check}() \in$ {`true`,`false`} sets the action of checking or not the validity of the block
  if $action^{check}$ == `false` **then**
    $DAG_i := update(DAG_i, \mathcal{B})$
  if $action^{check}$ == `true` **then**
  $validValue \leftarrow isValid(\mathcal{B})$ // The execution of $isValid(\mathcal{B})$ has a cost c_{check}
    if $validValue$ == `true`
      $DAG_i := update(DAG_i, \mathcal{B})$

- *Base case:* Let $\mathcal{B}$ be the first block issued by q after t_0. If the block is valid, then $\mathcal{B}$ is appended to the genesis block. Otherwise if the block is voted as invalid no user will append it to her own DAG. Let DAG_p the tree of p at time $t_0 + \Delta$. By Δ time any users delivered $\mathcal{B}$ by the broadcast primitive, hence the DAG of any process is identical to DAG_p. It follows that δ time after the first proposal, any two processes i and j verify the property: $DAG_i = DAG_j$.
- *Inductive case:* Suppose the lemma is true after a sequence of h blocks and prove it holds for the $h + 1$ block. Let t be the time when the h block is proposed. By inductive hypothesis, we have that by time $t + \Delta$, all users have the same DAG. Let $t'(t' \geq t + \Delta)$ be the time when the $h+1$ block is invoked by q. Let DAG_q be the local tree of q. By inductive hypothesis, by time t', all the other users have the same local DAG. Two cases have to be considered:
  - The block proposed at time t' is valid : at time t' the networks votes the block as eligible to be appended, and then by time $t' + \Delta$ all users delivered $\mathcal{B}$. Since by assumption of the inductive case they have the same tree as q, then by time $t' + \Delta$ $\mathcal{B}$ is appended to the DAG. It follows that any two honest users i and j verify the property: $DAG_i = DAG_j$.
  - The block proposed at time t' is invalid : in this case block $\mathcal{B}$ is voted as invalid and it is not appended to the local DAG.

– *DAG-Growth*: Given two different instants of time $t_2 > t_1$ we have two cases: no block is issued between t_2 and t_1, or some blocks are issued. In the first case $DAG_i^{t_2} = DAG_i^{t_1}$, while when blocks are issued if they are voted as valid these are added to the DAG and so $DAG_i^{t_1} \subseteq DAG_i^{t_2}$, otherwise they are discarded and $DAG_i^{t_2} = DAG_i^{t_1}$.