Received: 29 August 2022 | Revised: 6 January 2023 | Accepted: 10 January 2023

DOI: 10.1049/blc2.12023

ORIGINAL RESEARCH

Waterfall: Gozalandia. Distributed protocol with fast finality and proven safety and liveness

Sergii Grybniak[1] | **Yevhen Leonchyk**[2] | **Igor Mazurok**[2] | **Oleksandr Nashyvan**[1] | **Ruslan Shanin**[2]

[1]Odessa National Polytechnic University Institute of Computer Systems, Institute of Computer Systems, Odesa, Ukraine

[2]Odessa I.I. Mechnikov National University, Faculty of Mathematics, Physics and Information Technologies, Odesa, Ukraine

Correspondence
Sergii Grybniak, Odessa National Polytechnic University Institute of Computer Systems, Institute of Computer Systems, Odesa, Ukraine.
Email: sergii.grybniak@ieee.org

Funding information
Fuente Labs LLC funded the research work and AWS servers to carry out the load tests.

Abstract

A consensus protocol is a crucial mechanism of distributed networks by which nodes can coordinate their actions and the current state of data. This article describes a BlockDAG consensus algorithm based on the Proof of Stake approach. The protocol provides network participants with cross-voting for the order of blocks, which, in the case of a fair vote, guarantees a quick consensus. Under conditions of dishonest behavior, cross-voting ensures that violations will be quickly detected. In addition, the protocol assumes the existence of a Coordinating network containing information about the approved ordering, which qualitatively increases security and also serves to improve network synchronization.

1 | INTRODUCTION

Blockchain technology is now firmly entrenched in our lives. The core of a decentralized platform is a consensus layer that allows all honest participants (ie properly operating and following the network protocol) to arrive at the same decisions and have the same state of the distributed ledger [1]. PoW (Proof of Work) based consensus mechanisms were the first to be implemented [2]. Unfortunately, PoW-based consensus mechanisms have two significant drawbacks: poor scalability and high power consumption [3]. This shortcoming engendered the search for new consensus approaches. One solution to the problem of high energy costs is a consensus mechanism based on Proof of Stake (PoS) (see, for example, Algorand [4], Cardano project [5], Ouroboros [6–8], Snow white [9]).

Another solution is to use BFT consensuses (e.g. see [10, 11]), but these consensuses only work well with a small number of nodes.

Another problem of classical blockchains is that concurrent transactions/blocks cannot be accepted at the same time, resulting in slow addition and confirmation of transactions. This state of affairs motivates the emergence of protocols based on directed acyclic graph (DAG) structures (see, e.g. [12–16]). DAG structures allow us to create multiple blocks at the same time. In particular, we get the following benefits:

- increase the number of transactions accepted per round of network operation;
- split the transaction pool to reduce network load.

An overview of protocols based on DAG structures can be found in [17]. We also note, for the sake of completeness, that blockchains are also BlockDAGs. They are typically represented as a PolyTree (or a directed tree) with a longest chain rule [2] or its modification as a fork choice rule.

There are typically two types of DAG based protocols: transactional DAGs (txDAGs) and block DAGs (BlockDAG) [17]. In txDAGs, vertices of the DAG are transactions [18], while in BlockDAGs they are—blocks [13, 19, 20]. Referential structure specificities pose their own challenges, in particular, the achievement of network decentralization.

The goal of the present paper is to build a DAG-based PoS consensus protocol upon the assumption that there will be tens, or even hundreds of thousands of validator-nodes, while remaining scalable and processing thousands of transactions per second. Some examples of other DAG-based PoS consensus

TABLE 1 Network load test results

Number of				Loading					
				Idling		First day		Third day	
Servers	Nodes	Coordinators	Total	CPU	RAM	CPU	RAM	CPU	RAM
t3, medium	32	32	1024	1.9	0.6	2.5	1.3	2.2	1.8
	32	16	512	2.3	0.7	2.3	1.3	0.56	2.2
	8	32	256	1.9	0.6	2.5	1.3		
	8	16	128	2.3	0.7	2.3	1.3	0.53	2.66
	8	8	64	2.3	0.5	2.3	1.3		
t3, small	8	8	64	1.9	0.5	2.2	1.3		

protocols can be found in [13, 21–23]. Also in [22] it can be found some typical problems of PoS and DAG.

1.1 | Our results

We present "Waterfall: Gozalandia," a new BlockDAG model based on PoS. The following new results have been obtained:

- a new simple PoS oriented blockDAG ordering algorithm was proposed;
- a model of the divided work of validators (voting for the ordering of the blockDAG) and the creators of blocks with transactions was proposed;
- a new validator voting model has been proposed: Validators do not vote for each block, but for "spine blocks" that determine the ordering. This approach reduces the amount of information that needs to be distributed to make a sequencing decision.
- under optimistic assumptions about the network operation, an estimate of the number of slots for ordering stability is obtained and the speed of decision making in the liveness property is obtained.
- an algorithm for time adjustment of duration of slots is proposed.

Our protocol is safe under a number of plausible assumptions:

1) the network is synchronous in the sense that there is an upper bound within which any honest participant can communicate with any other participant;
2) honest validators are available as needed to participate in each epoch;
3) honest validators do not remain offline for long periods of time.

We use the scheme for selecting committees, leaders, and block producers proposed in [24]. This method guarantees sufficient security and protection against manipulation by attacking nodes.

In our review of the protocol, we focus on the properties of liveness and safety [25], which are the two main formal properties of a reliable block registry. Viability means that every properly created block available to nodes in the network will, after some time, be added to the ordering, and all its consistent transactions will be accepted. Security means that if some honest node has accepted some block ordering, then all other honest nodes will also adopt the same block ordering.

This consensus protocol allows for further implementation of smart contract technology by Waterfall. The concept of smart contracts was introduced for the first time by Nick Szabo in 1994 [26]. It later became widespread primarily due to Ethereum. Smart contracts are becoming increasingly popular for improving business transactions around the world [27–29], especially in Decentralized Finance (DeFi) [30, 31]. Nowadays, the international community continues to actively study and develop new smart contract technologies, in particular, Ethereum v2.0 [32].

Finality is a key issue for the implementation of smart contracts. The property of finality [33, 34] ensures that a block that has been committed and recorded will not be dropped in the future for all honest nodes. Thus, smart contract results cannot be reversed.

However, most modern finality consensuses have one significant limitation: knowledge of the number of nodes taking part in the consensus is required. One of our main goals is to provide a consensus protocol with the property of finality that allows a dynamic number of network nodes (validators). In doing so, the platform will be able to remain fully decentralized and highly-scalable, relative to both the number of nodes and transaction throughput. All of these issues require the development of a new method for creating a DAG referential structure, and its further ordering and finalization.

1.2 | Implementation

Currently, the main elements of "Waterfall: Gozalandia" have been implemented. We have carried out the load tests, using AWS t3.small and t3.medium servers with 2-core CPU and 2 or 4 GB of RAM, respectively. The algorithm has been tested with 8, 32 and 64 nodes and with 16 and 32 workers per node.

We tested the network by generating approximately 96,000 transactions with the slot length is 4 seconds and the number of blocks in the slot is 4. At this state, the fully operational network handles a throughput of 3200–3600 transactions per second, where the average waiting time for an confirmation is about 20 seconds.

Moreover, we also studied relationship between the load on the CPU or RAM and the number of Workers running on servers. From the Table 1, one can see that there is no significant correlation between these indicators. We also noticed that the load structure changes throughout time. On the third day, the impact of program adaptation mechanisms became apparent. The workload on the processor was reduced by 75–77% as a result of putting some of the data needed for work in a pool in RAM. However, this only happened in situations where there was enough memory available to store all the data required for all the Workers operating on the node. As an experiment, we evaluated hashing of data with insufficient RAM for 32 Workers. Despite the 38% increase in memory usage, the reduction in CPU resources were only 12%. Thus, we came to the conclusion that doubling the minimum amount of RAM allows for a more than four-fold reduction in CPU load on the third day of work. At Amazon Web Services prices, these factors result in an average 64% savings in server maintenance costs for the Waterfall network node.

2 | AUXILIARY DEFINITIONS AND RESULTS

The "Waterfall: Gozalandia" system consists of two parts: the Coordinating and BlockDAG networks. Each node of the Coordinating network is connected to a node of the BlockDAG network and vice versa.

The time of the work of the Coordinating and the BlockDAG networks are divided into slots and epochs. Slots define the rounds of work of the networks and epochs combine slots and serve to summarize the intermediate results of work of the network.

2.1 | Directed acyclic graphs

For our further explanation, we will need some concepts from the Graph Theory.

A (simple) directed graph G is an ordered pair (V, E) of disjoint sets, where $V = V(G)$ is a nonempty set of vertices and $E = E(G)$ is a set of edges which are ordered pairs (a, b) of some elements from the set $V(G)$. The first vertex of the ordered pair is the tail of the edge, and the second is the head; together, they are the endpoints. We say that an edge is an edge from its tail to its head (see, e.g. [35, p. 53]).

If there is an edge from a to b, then we will say that a refers to b. We will write (a, b), $a \to b$ or $b \leftarrow a$ for the case if there is an edge from a to b.

Let G be a directed graph. A graph Y is a subgraph of G if the inclusions $V(Y) \subseteq V(G)$ and $E(Y) \subseteq E(G)$ hold. Let the vertices of subgraph P can be renumbered such that

$$V(P) = \{a_0, a_1, \dots, a_k\},\ k \geqslant 1,$$

$$E(P) = \left\{(a_k, a_{k-1}), \dots, (a_1, a_0)\right\},$$

where all a_i are different for $i = 1, \dots, k-1$. If $a_0 = a_k$ and $k \geqslant 2$, then the subgraph P is called a cycle. If all a_i are different, then the subgraph P is called a path joining vertices a_0 and a_k. In this case, we will also say that P is the path from a_k to a_0 and that a_0 is the beginning of the path P and a_k is the end of the path P. For the path P we will also use the following notation

$$P = P_{a_k,a_0} = a_0 \leftarrow a_1 \leftarrow \dots \leftarrow a_{k-1} \leftarrow a_k.$$

The number of edges of a path P is called *length* of the path P and denote $l(P)$.

If the graph G does not have any cycles, then G is acyclic. In what follows, for directed acyclic graph G, we will say that G is a DAG to be short.

Let G be a directed acyclic graph and $\mathcal{P}$ be a family of paths P. A graph P^* is intersection of paths of the family $\mathcal{P}$,

$$P^* = \bigcap_{P \in \mathcal{P}} P,$$

if the set of vertices of P^* is intersection of the sets of vertices and the set of edges is intersection of the sets of edges of paths of family $\mathcal{P}$, respectively,

$$V(P^*) = \bigcap_{P \in \mathcal{P}} V(P), \quad E(P^*) = \bigcap_{P \in \mathcal{P}} E(P).$$

As vertices of the DAG we will consider the blocks created in the BlockDAG network. The edges will be defined by block references to previous blocks. In this paper we will consider DAGs for which there is a block b_G such that for every block b of the graph G there is a path $P_{b,b_G} \subset E(G)$ connecting blocks b and b_G. Such the block b_G will be called *genesis*, its creation starts the work of the network.

For the set X denote by $|X|$ the number of elements in the set X. For $b \in V(G)$ we denote by $\deg^+(b)$ the number of edges with tail b, $\deg^+(b) = \left|\{(b, a) : (b, a) \in E(G)\}\right|$, and denote by $\deg^-(b)$ the number of edges with head b, $\deg^-(b) = \left|\{(a, b) : (a, b) \in E(G)\}\right|$. If $\deg^-(b) = 0$, then we say that $b \in \mathrm{tips}(G)$.

Since the operating time of BlockDAG network is divided into periods called slots, to the usual structure of directed acyclic graph we add information about time of a block creation, number of the slot in which the block was created. For a block b created in the n-th slot we will write $b \in \mathrm{Slot}(n)$. By definition, we assume $b_G \in \mathrm{Slot}(0)$. This is the only block of the zero slot. Obviously, in a well-synchronized network, block b is created in the n-th slot if and only if $\max_{P_{b,b_G} \subset E(G)} l(P_{b,b_G}) = n$ holds.

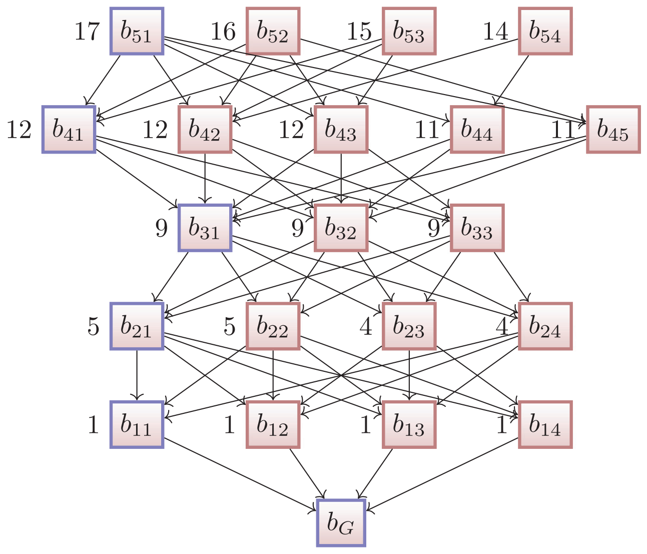

FIGURE 1 The ordering of the blocks in the slots with labels of $|\operatorname{past}(\cdot,\cdot)|$ in the DAG of the BlockDAG network. The spine blocks are shown in blue

We will say that a block a belongs to the past of a block b, $a \in \operatorname{past}(b, G)$, if there is a path $P_{b,a} \subseteq E(G)$ connecting blocks b and a.

Let us construct the order in n-slot of a directed acyclic graph with slots. Let $b_1,\ b_2 \in \operatorname{Slot}(n)$. We will say that the block b_1 precedes the block b_2, $b_1 \prec_{sl} b_2$, if

1) $|\operatorname{past}(b_1, G)| > |\operatorname{past}(b_2, G)|$.
2) If $|\operatorname{past}(b_1, G)| = |\operatorname{past}(b_2, G)|$, then $\deg^+(b_1) > \deg^+(b_2)$.
3) If $|\operatorname{past}(b_1, G)| = |\operatorname{past}(b_2, G)|$ and $\deg^+(b_1) = \deg^+(b_2)$, then $\operatorname{hash}(b_1) < \operatorname{hash}(b_2)$ (see Figure 1).

A block $b \in \operatorname{Slot}(n)$ will be called a *spine block* if it precedes all other blocks in the slot.

3 | PROTOCOL DESCRIPTION

Let us start with a description of how the Coordinating network works.

3.1 | Description of the Coordinating network

One of the main components of "Waterfall: Gozalandia" is a Coordinating network. This is a blockchain network that performs the following functions:

1) Maintaining a register of validators — nodes in the network that have blocked a certain amount of coins to participate in the network. In "Waterfall: Gozalandia," the blocked amount is *fixed* and the same for all validators. The set of all validators will be denoted by $\mathcal{V}$;
2) allocation of validators to committees and slots;
3) assign roles to validators: committee member, committee leader, block creator;
4) getting information about spine blocks from the BlockDAG network;
5) voting for spine blocks and creating a block in your network with the results of the vote.

We assume that all members of the Coordinating network can authenticate messages from each other. We also assume that all messages are eventually delivered to all honest members and that they can synchronize in a time not exceeding T_{sync}. In doing so, network nodes that have not received a message in time are considered faulty.

The work of the Coordinating network begins with the creation of a genesis block, which contains the initial list of validators.

The Coordinating network works according to the following scheme:

1) At the beginning of each epoch, a list of N_C committees for the next epoch is randomly generated for each slot.
2) At the beginning of each slot, the leaders of those committees are chosen at random from among the members of the committees.
3) At the beginning of each slot, nodes are randomly chosen from the committee members to create blocks.
4) At the beginning of each slot, each node of the Coordinating network sends to the connected nodes of the BlockDAG network information about the epoch, the slot, the list of block creators and information about the finalized blocks of the BlockDAG network (see below for definition of finalized blocks).
5) At the beginning of each slot, each node of the Coordinating network receives information about spine blocks from the BlockDAG network node connected to it.

The finalized spine block of the n-th slot will be denoted by the symbol f_n. By definition, we set $f_0 = b_G$, where b_G is the genesis block.

Let us describe the procedure for creating a block in the Coordinating network, assuming that the finalized spine block f_n is already selected. Denote by Σ_{n+s-1} the list of blocks up to $n+s-1$-th slot of the Coordinating network and denote by $\mathcal{B}_i$ the list of blocks $n+i$-th slot of the BlockDAG network.

1) Each committee member sends messages to the other committee members with a list of visible spine blocks starting at $n+1$ slots, a block of the previous state, and a signature, $m_{v,1} = \big((b_1, b_2, \dots, b_s), \sigma, v\big)$, where $b_i \in \mathcal{B}_i$, $v \in \mathcal{V}$ is the signature and $\sigma \in \Sigma_{n+s-1}$ is chosen according to the following rule
   a) if there are no forks, then it is the last correctly created block of the Coordinating network;
   b) if there are forks and the longest chain, it is the last block of the longest chain;

   c) if there are forks and chains generated by them of the same length, then the earliest of the last blocks of forks chains.
2) Let a sequence $\overline{b} = (b_1, b_2, \ldots, b_{k_s})$ be the beginning of some sequence $(b_1, b_2, \ldots, b_s)$ published some validator v. Then we say that the sequence $\overline{b}$ receive a vote of the validator v. As a result of the previous point, two options are possible:
   a) There is a sequence $\overline{b} = (b_1, b_2, \ldots, b_{k_s})$ which received more than $2/3$ votes, such that no longer sequence received more than $2/3$ votes.
   b) No sequence received more than $2/3$ of votes. In this case, we set $\overline{b} = \varnothing$.
3) Each committee member who supports $\overline{b}$ sends the message $m_{v,2} = \left(\overline{b}, \sigma, v\right)$ to the other committee members.
4) The leader of the committee forms a message

$$m_l = \left(\overline{b}, \sigma, (v_1, \ldots, v_i)\right),$$

and sends it to the other leaders of the committees.
5) If more than $2/3$ of the committees receive a nonempty sequence, each leader sends the message $(\overline{m}, v_l)$ to the other leaders, where $\overline{m} = (m_{l_1}, \ldots, m_{l_{N_C}})$ and the first leader forms and sends out the block $(\overline{m}, \overline{v}, v_{l_1})$, where $\overline{v} = (v_{l_1}, \ldots, v_{l_{N_C}})$. Otherwise the block is not published.

Let us move on to the rules for accepting a block in the Coordinating network

1) If a correctly created block of the Coordinating network is obtained no later than the next slot, then it is sent further and added to the chain. In this case we will say that the block is obtained in time.
2) If a correctly created block of the Coordinating network is obtained through one slot, then it is sent further and added to the waiting list. In this case we will say, that the block is obtained with a delay.
3) If a block, which belongs to the waiting list, is referenced by a block created in time or with a delay, then it is added to the chain.
4) If a block of the Coordinating network is obtained later than 2 slots after it was created, it is deleted and not sent further.

Definition 3.1. A sequence $(b_1, \ldots, b_k)$ is called accepted in block b if in more than $2/3$ of the committee solutions of that block it is the initial sequence and no longer sequence has this property.

Definition 3.2. Let $P_1 = b_0 \leftarrow b_{1,1} \leftarrow b_{1,2} \leftarrow \ldots \leftarrow b_{1,s_1}$, $P_2 = b_0 \leftarrow b_{2,1} \leftarrow b_{2,2} \leftarrow \ldots \leftarrow b_{2,s_2}$ be paths and let $P_i(j)$ be a subpath of the path P_i containing only blocks created before $j +$ 1-th slot. We will say that the path P_1 less than the path P_2 and write $P_1 \precsim P_2$, if the following statements

$$\exists k \in \mathbb{N}\ \forall j < k \left| l(P_1(j)) - l(P_2(j)) \right| \leqslant 2,$$

and $l(P_2(k)) = l(P_1(k)) + 3$ are valid. If for paths P_1 and P_2 we have $P_1 \not\precsim P_2$ and $P_2 \not\precsim P_1$, then we will write $P_1 \parallel P_2$ and say that P_1 and P_2 are not comparable.

Definition 3.3. Let G be a DAG of the Coordinating network. The path $P = b_1 \leftarrow \ldots \leftarrow b_k$ is called to be continuous in G if, for every $i \in \{\mathrm{slot}(b_1), \ldots, \mathrm{slot}(b_k)\}$, we have that $a_i \in G$ implies $a_i \in P$, where a_i is a block created in i-th slot. In other words, the path P is continuous if all blocks, created between b_1 and b_k, belong to the path P.

Definition 3.4. Let G be a DAG of the Coordinating network. Denote by $\mathcal{P}$ the set of all paths P such that there is no path P' for which $P \precsim P'$ and let P^* be intersection of paths P of the family $\mathcal{P}$. Then the path P^*_{r-2} is called *accepted*, where P^*_{r-2} is the path P^* without the last two blocks. If block b belongs to an accepted path P, then b is called to be accepted.

Definition 3.5. Let $\mathcal{B}_i$ be the list of blocks $n + i$-th slot of the BlockDAG network, Σ_{n+i} be the list of blocks up to $n + i$-th slot of the Coordinating network. A sequence of blocks $(b_1, \ldots, b_k)$, where $b_i \in \mathcal{B}_i$, $\mathcal{B}_i$ is the list of blocks $n + i$-th slot, is called finalized if this sequence is accepted in N_A accepted blocks, where N_A is a network parameter.

3.2 | BlockDAG network description

Let's move on to describing how the BlockDAG network works.

At the beginning of the BlockDAG network, a genesis block is created, containing service information about the network.

The blocks in the BlockDAG network "Waterfall: Gozalandia" contain hashes of blocks that do not belong to the same slot in which the block is created and that are not referenced in the visibility area of the node during block creation (no blocks containing hash of those blocks) and an ordered list of transactions.

Let us describe the procedure of adding a new block by a BlockDAG node of the network. When a new block is received, the BlockDAG node of the network checks if the block was created correctly:

1) the correctness of the creation slot and node (could the node create a block in this slot);
2) the content of the block is correct;
3) if the references to previous blocks are correct
   a) no references to multiple blocks that are created by the same node in the same slot;
   b) no references to incorrectly created blocks.

Let us describe the block ordering procedure in the BlockDAG network.

Let the sequence f_0, ..., f_n of finalized spine blocks be given in the graph G. Define the expanding sequence of subgraphs G_n of graph G associated with this sequence. Assume by definition that G_0 is a graph with $V(G_0) = \{b_G\}$ and $E(G_0) = \varnothing$. Let us define the graph G_n as follows

$$V(G_n) = V(G_{n-1}) \cup \text{past}(f_n) \cup \{f_n\},$$

and, for every $a_1, a_2 \in V(G_n)$, $(a_1, a_2) \in E(G_n)$ if and only if $(a_1, a_2) \in E(G)$.

Let us construct a linear order on G_n (see Figure 2):

1) For every $a \in V(G_{n-1})$ and $b \in V(G_n) \setminus V(G_{n-1})$ we set $a \prec b$.
2) For every $a \in V(G_n) \setminus \{f_n\}$ we set $a \prec f_n$.
3) For every $a, b \in (V(G_n) \setminus V(G_{n-1}))$, if $a, b \in \text{Slot}(i)$, then we set $a \prec b$ if $a \prec_{sl} b$. If $a \in \text{Slot}(i)$, $b \in \text{Slot}(j)$, $i \neq j$, then we set $a \prec b$ if $i < j$.

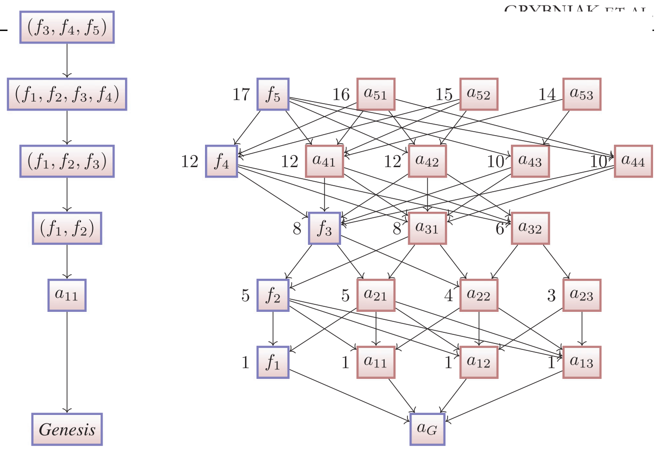

FIGURE 2 The structure of the Coordinating and BlockDAG networks. The finalized spine blocks are shown in blue. The final ordering is as follows: $a_G, f_1, a_{11}, a_{12}, a_{13}, f_2, a_{21}, a_{22}, f_3, a_{23}, a_{31}, a_{32}, f_4, a_{41}, a_{42}, a_{43}, a_{44}, f_5$

4 | PROPERTIES OF THE PROTOCOL

4.1 | On the probability of creating blocks

We will analyze the protocol under the assumption that the time of spreading information over the network T_{sync} among honest nodes does not exceed the slot time T_{sl}, $T_{sync} < T_{sl}$.

Evaluate the probability of a committee with an unfair majority. Let there be n nodes in the network and among them n_c dishonest nodes and let the size of the committee be k nodes. Denote by k_m the smallest natural number such that $k_m > (2/3)k$. Then the probability $P(k_{cor} \geqslant k_m)$ that more than $2/3$ of the committee members are dishonest is

$$P(k_{cor} \geqslant k_m) = \sum_{i=k_m}^{k} \frac{\binom{n_c}{i}\binom{n-n_c}{k-i}}{\binom{n}{k}}.$$

Similarly, the probability $P(k_{hon} \geqslant k_m)$ that more than $2/3$ of the committee members will be honest is

$$P(k_{hon} \geqslant k_m) = \sum_{i=k_m}^{k} \frac{\binom{n_c}{k-i}\binom{n-n_c}{i}}{\binom{n}{k}}.$$

To see the order of these values, consider a particular case.

Example 4.1. Let us set $n = 10{,}000$, $n_c = 3333$, $k = 100$, $N_C = 4$. Then $k_m = 67$,

$$P(k_{cor} \geqslant 67) = \sum_{i=67}^{100} \frac{\binom{3333}{i}\binom{6667}{100-i}}{\binom{10000}{100}} \approx 5 \cdot 10^{-12},$$

$$P(k_{hon} \geqslant 67) = \sum_{i=67}^{100} \frac{\binom{3333}{100-i}\binom{6667}{i}}{\binom{10000}{100}} \approx 0.52.$$

Assuming that the probabilities of choosing honest committees are $p \approx 0.52$ (they are not, but they are approximately equal), the probability that at least 3 out of 4 committees will be honest is

$$P(Nc_{hon} \geqslant 3) = \binom{4}{3} p^3 (1-p) + \binom{4}{4} p^4 \approx 0.34.$$

These results show that in about a third of cases, a correct block will be published in the Coordination network, with a very low probability (about 10^{-33}) several blocks will be published (multicreating), and in about $2/3$ of cases a block may not be published. Note also that as the number of honest nodes grows, the probability of publishing a correct block quickly approaches 1.

Due to the fact that some of Coordinators may turn out to be faulty, we need to find out how this will affect the decision in the committees and the final decision on finalization. In this case, it is necessary to estimate what is the maximum proportion of faulty Coordinators that is acceptable without stopping the decision-making process. In other words, for a given number of Coordinators, find such a number of committees and determine their number, at which the average number of faulty slots per epoch will be in a certain sense "minimal". Obviously, the number of faulty slots will depend on the proportion of faulty Coordinators and their distribution within the committees. At the same time, the number of sent messages should also be taken into account, possibly minimizing it.

Using simulation modeling, it was found that the largest percentage of faulty Coordinators, at which there is no significant delay in block finalization, is 20%. The best form for the number of committee members and committees is $3 \cdot k + 1$ (k is integer) to reduce the number of missed slots in comparison with nearby options. In addition, during the simulation, it is found that the best result with a fixed number of Coordinators (lower average number of faulty slots per epoch) is achieved with a decrease in the number of committees and with a corresponding increase in the number of their members. Figure 3 depicts the number of faulty slots per epoch depending on the

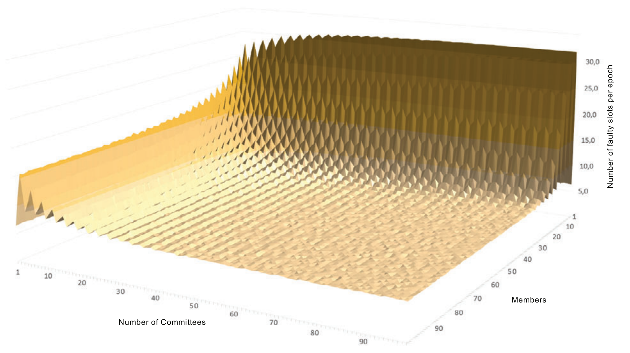

FIGURE 3 Dependency of the number of missed slots on the number of committees and committee members

number of committees and committee members. However, due to technical restrictions on the exchange of messages within the committee, the number of committee members should not exceed approximately 100.

Remark 4.2. By Example 4.1, we will assume that for every path consisting of two blocks there is a block from this path which is created by honest committees and is obtained by some honest nodes in time. In particular, if the second block is created by honest committees, then each of the blocks are created by honest committees.

Consensus protocols are used by nodes in a distributed network to make a consensus decision from possible non-consensus decisions. Each consensus protocol must have the following two properties:

- Liveness: if a block has become available to nodes in the network, after some time it will be added to the ordering and all consistent transactions will be accepted;
- Safety: the accepted decision is the same for all honest nodes.

Property 4.3. If a block is created by honest committees, then some honest nodes obtain the block in time and all honest nodes obtain the block either in time or with a delay.

Remark 4.4. If Property 4.3 is valid and, for a block b created in i-th slot, there is a block referred to b, then, by Remark 4.2, all honest node see the block b in $i+3$-th slot.

For the following we assume that Property 4.3 is valid.

Theorem 4.5. *Let G be a directed acyclic graph of the Coordinating network in some node Nd_1 and let P, P^* be paths with the same beginning such that $P \precsim P^*$. Then the path P will be dropped.*

Proof. Let $P = b_0 \leftarrow b_{1,1} \leftarrow \dots \leftarrow b_{1,s}$ and $P^* = b_0 \leftarrow b_{2,1} \leftarrow \dots \leftarrow b_{2,r}$ be paths and let $P \precsim P^*$ hold. It implies that the following statements

$$\exists k \,\forall j < k \left| l(P(j)) - l(P^*(j)) \right| \leqslant 2, \tag{4.1}$$

and

$$l(P^*(k)) = l(P(k)) + 3, \tag{4.2}$$

are valid. Denote

$$P_{s_1} = P(k) = b_0 \leftarrow b_{1,1} \leftarrow \dots \leftarrow b_{1,s_1},$$

$$P^*_{r_1} = P^*(k) = b_0 \leftarrow b_{2,1} \leftarrow \dots \leftarrow b_{2,r_1}.$$

It is easy to see that statements (4.1) and (4.2) are valid only in the following two cases (see Figure 4)

1) The path $b_{2,r_1-2} \leftarrow b_{2,r_1-1} \leftarrow b_{2,r_1}$ is continuous.
2) There is a block created between b_{2,r_1-2} and b_{2,r_1-1} and the path $b_{2,r_1-1} \leftarrow b_{2,r_1}$ is continuous.

Let b_{j_1} and b_{j_2} be blocks created after block b_{2,r_1}. Then in the time of creating the block b_{j_2} all honest nodes see the block b_{2,r_1-1}. Consequently, the block b_{2,r_1-2} are added in the chain in all honest nodes. Thus, since in the first case we have $l(P^*_{r_1-2}) = l(P_{s_1}) + 1$ and the block b_{2,r_1-2} was created before the block b_{j_1}, all blocks, created after $k+1$-th slot, do not refer to the path P. For the second case, we have

$$l(P^*_{r_1-2}) = l(P_{s_1}) + 2.$$

Consequently, all blocks created after $k+1$-th slot do not refer to the path P. Thus, we have that in both cases the path P is dropped. □

From the proof of Theorem 4.5 it follows the following corollaries.

Corollary 4.6. *Let G be a directed acyclic graph of the Coordinating network in some node Nd_1 and let P, P^* be paths with the same beginning such that $P \precsim P^*$. Then there is at most one block of P created after the last block of the path P^*.*

Corollary 4.7. *Let G be a directed acyclic graph of the Coordinating network in some node Nd_1 and let b_0 be an accepted block. If there is the path $P^* = b_0 \leftarrow b_1 \leftarrow \dots \leftarrow b_r$ such that, for every path P of G with the same beginning, we have either $P = P^*$ or $P \precsim P^*$, then all paths P with the same beginning different of P^* will be dropped and the path P^*_{r-2} will be continued.*

Proof. It follows from the proof of Theorem 4.5. □

The following theorem shows that the protocol, we are considering, has the property of Safety.

Theorem 4.8 (Safety). *If block f was finalized by some honest node, it will be finalized by every active honest node.*

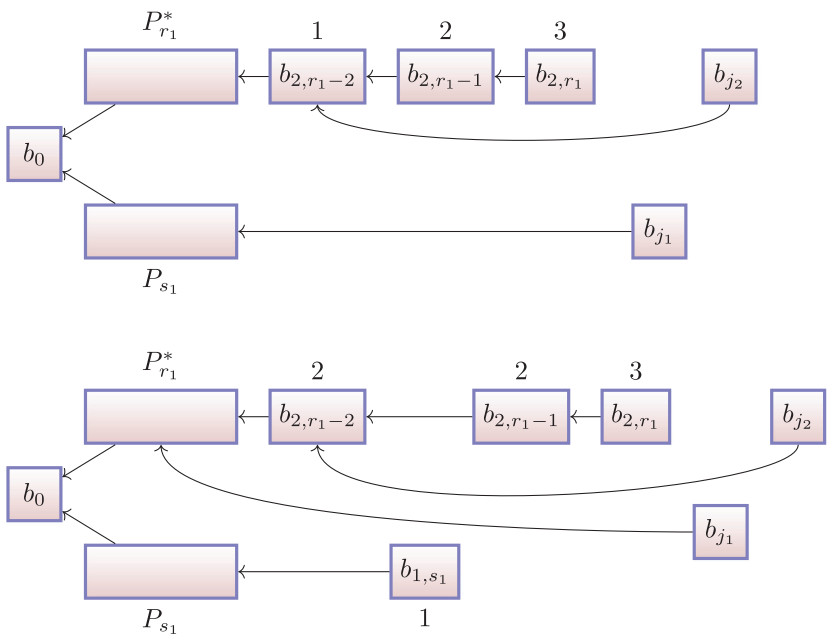

FIGURE 4 The path P will be dropped and the path $P^*_{r_1-2}$ will be continued

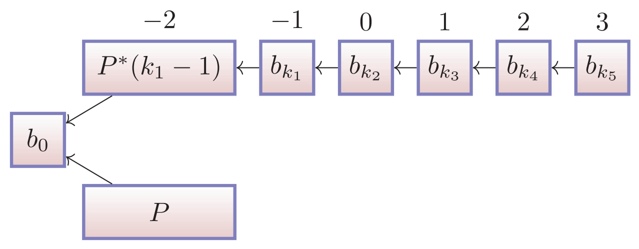

FIGURE 5 The path P less than P^*

Proof. It follows from the Corollary 4.7 that the sets of accepted blocks in any two honest nodes coincide. Thus, the sets of finalized blocks also coincide. □

In the following theorem we give a simple sufficient condition that the path is maximal.

Theorem 4.9. *Let a path P_0 be continuous, $l(P_0) \geqslant 4$ and let a path P^* with beginning in b_0 such that $P_0 \subseteq P^*$ hold. Then, for every path P with beginning in b_0 satisfying $P_0 \not\subseteq P$, we have $P \precsim P^*$.*

Proof. Let $P_0 = b_{k_1} \leftarrow b_{k_2} \leftarrow \ldots \leftarrow b_{k_5}$ hold, where $\text{slot}(b_{k_i}) = k_i$. Consider the paths $P_1^* = P^*(k_5)$ and $P_1 = P(k_5)$. Since the path P_0 is continuous, the last block of the path P_1 have a slot less than k_1. Hence, it follows that $l(P^*(k_1)) - l(P) \geqslant -1$ holds (see Figure 5).

Indeed, if we have $l(P^*(k_1)) - l(P) \leqslant -2$, then the inequality $P^*(k_1 - 1) \precsim P$ holds, and, by Corollary 4.6, $P^*(k_1 - 1)$ has at most one block after the last block of P, that contradicts to suggestion of the theorem.

Consequently, the blocks, which have the slot more than $k_5 + 1$, refer to blocks which have the slots more than k_2 and have in the past the block b_{k_2} because $l(P^*(k_3)) - l(P) = 1$ and there is at most one block referred to the path P, which is created after the slot k_3. □

Let us now turn to the consideration of the property of Liveness. It can be shown that Property 4.3 is not enough, in general, for validity of Liveness.

Property 4.10. If a block is created by honest committees, then all honest nodes obtain the block in time.

Remark 4.11. If Property 4.10 is valid and, for a block b created in i-th slot, there is a block referred to b, then, by Remark 4.2, all honest node see the block b in $i + 2$-th slot.

For the proof of the following theorem we assume that Property 4.10 is valid.

Theorem 4.12 (Liveness). *If a block has become available to nodes on the network, it will be added to the ordering after some time.*

Proof. For the proof of the theorem we must prove that every block of the Coordinating network will be either accepted or dropped after some time.

Let b_0 be the last accepted block and $\text{slot}(b_0) < k_0$. Let us show that if from $k_0 + 1$-th slot it starts to be valid Property 4.10, then there is a path with beginning in b_0 which will be accepted.

First of all, we show that for every $k \geqslant k_0$ there is the unique block b_k^* with $\text{slot}(b_k^*) > k$ referred to a block created before $k + 1$-th slot such that every path P not containing b_k^* has less than four blocks created after k-th slot.

Suppose contrary that there are two paths

$$P_1 = P_1(k) \leftarrow a_{1,1} \leftarrow a_{1,2} \leftarrow a_{1,3},$$

$$P_2 = P_2(k) \leftarrow a_{2,1} \leftarrow a_{2,2} \leftarrow a_{2,3},$$

where $a_{1,1} \neq a_{2,1}$, which have not less than three blocks. By Remark 4.2, both of the paths have not less than two blocks created by honest committees. Without loss of generality, we may assume that $\mathrm{slot}(a_{1,1}) < \mathrm{slot}(a_{2,1})$. Since the blocks $a_{1,1}$, $a_{1,2}$, $a_{2,1}$, $a_{2,2}$ are created by honest committees and, by Property 4.10, are obtained in time, the block $a_{2,2}$ does not refer to the block $a_{2,1}$, that contradicts supposition.

Let us show now that the blocks b_k^* will be accepted. For every $k \geqslant k_0$, there is a path P_k containing the block b_k^* such that P_k has more than 5 blocks created after k-th slot. From proved above the part of theorem it follows that every path P, not containing the block b_k^*, has no more than 3 blocks created after k-th slot and, consequently, $P \precsim P_k$. Thus, if $\mathcal{P}$ is the family containing all paths P such that there is no path P' for which $P \precsim P'$ and P^* is intersection paths of the family $\mathcal{P}$, then P^* contains the block b_k^*. □

5 | TIME ADJUSTMENT

The value of block finalization time is significantly affected by the duration of slots T_{sl}^1 and T_{sl}^2 in the Coordinating and BlockDAG networks, respectively. However, the parameters and characteristics of the communication channels in the public network may vary over time, which will lead to changes in the propagation time of both the blocks themselves and the service messages. Obviously, if slot times are specified with a margin (overestimate), this will lead to an overestimate of the finalization time. On the other hand, if the slot time is insufficient to propagate blocks and reach consensus, then all nodes that do not manage to get information in the allotted time will have to be counted as foul. Thus, the number of these nodes along with the actual foul nodes may exceed the expected limit of one-third of the nodes. To eliminate this threat, the current protocol uses an adaptive time control mechanism.

Changing of slot duration is done at the beginning of the next epoch for BlockChain and BlockDAG networks separately on the basis of results demonstrated in the previous epoch and fixed in the first block of each epoch. The results of the last two blocks of the previous epoch are not taken into account. Note that if it is necessary to react more quickly to changes in network connection parameters, time adjustment can be done twice—both at the beginning and in the middle of an epoch. The general concept is that slot duration smoothly decreases during observed successes in protocol operation and increases during a series of failures.

5.1 | Coordinating network case

The measure of success in a given epoch is the number of multisignature signed lists of visible spine blocks $\overline{b}$. Let us define m as the fraction of such lists from the maximal possible number in the period under consideration, and set two reference points $0 < s < S < 1$. Then

- if $m < s$, then the value T_{sl}^1 increases by 10%;
- if $m > S$, then the value T_{sl}^1 decreases by 10%. However, T_{sl}^1 is bounded below by T_{sync};
- then $s \leqslant m \leqslant S$ we assume that the system is working normally and that there is no need for time adjustment.

5.2 | BlockDAG case

In this case the success of node communication can be judged by the depth of links of new blocks. If quite a few blocks of a given slot refer to blocks that were created in slots deeper than the block before, it means that the communication of the nodes is delayed. In other words, nodes do not have time to exchange new blocks within the same slot.

Each honest block producer in the BlockDAG network accompanies his block with links to all known tips-blocks. Since each block contains information about the slot number in which it was created, we have the ability to calculate how long it traveled from a given honest node to another honest node. However, dishonest nodes can produce blocks with any reference, not just tips-blocks. So, to rely on the full structure of DAG as a reliable reflection of the speed of information dissemination is impossible. It is necessary to isolate only that part of the structure which is potentially formed by honest links. In this case, the ability of foul nodes to underestimate the speed is much greater than the possibility of overestimation. To understate, they only need to refer to long-known blocks while ignoring new ones. To overestimate, they need to produce new blocks by colluding. However, such an overestimation actually reflects the real rate of information propagation in a group of colluding foul nodes, possibly located on the same or close technical devices.

Each node observes its own topology. This means that applying the time settings of the BlockDAG network slots within the same network is not possible. This means that the time settings of the BlockDAG network slots must be fixed within the same network. Consider the characteristics of the topology (O_t) of the network based on the current situation observed by a particular node Nd based on the links of the block b he created.

5.2.1 | Observable topology

Let $b \in V(G)$ be the BlockDAG network. Denote by

$$\mathrm{child}(b) = \{a \in V(G) : (b, a) \in E(G)\}$$

is the set of blocks to which block b refers and $\mathrm{slot}(b)$ is the number of the slot in which block b is created.

Each node calculates the characteristic of the observed topology when forming its block by all observed blocks to which it refers:

$$O_t(b) = \frac{\sum_{a \in \text{child}(b)} \left(\sum_{u \in \text{child}(a)} (\text{slot}(a) - \text{slot}(u)) \right)}{\sum_{a \in \text{child}(b)} \text{slot}(a)}.$$

Thus, each block created by an honest node reliably shows from which nodes it had time to receive information and the degree of its relevance. A valid block created by a foul node shows only part of its information links and the possibility of any large underestimation of the real speed indicators. We will use this information to determine reliable information flows and perform fine-tuning of slot durations.

5.2.2 | BlockDAG network synchronization speed

Each node can count the current observed synchronization rate of the network. For this purpose, a set of honest observations is constructed in which the highest O_t^{sup} and the lowest O_t^{inf} observed rates are discarded:

$$O_t^{hon} = O_t \setminus \left(O_t^{\text{inf}} \cup O_t^{\text{sup}} \right),$$

where $O_t = \{O_t(b) : b \in G\}$,

$$O_t^{\text{inf}} = \bigcup_{b \in G : |A_*(b)| < |O_t|/3} A_*(b),$$

$$O_t^{\text{sup}} = \bigcup_{b \in G : |A^*(b)| > 2|O_t|/3} A^*(b),$$

and

$$A_*(b) = \left\{ a \in G : O_t(a) < O_t(b) \right\},$$

$$A^*(b) = \left\{ a \in G : O_t(a) > O_t(b) \right\}.$$

Recall that there are b_* and $b^* \in G$ such that $O_t^{\text{inf}} = A_*(b_*)$ and $O_t^{\text{sup}} = A^*(b^*)$. Indeed, for all pairwise different blocks b_1, b_2, without loss of generality, we can assume that $O_t(b_1) \leqslant O_t(b_2)$ holds. Then

$$A_*(b_1) = \left\{ a \in G : O_t(a) < O_t(b_1) \right\}$$
$$\subseteq \left\{ a \in G : O_t(a) < O_t(b_2) \right\} = A_*(b_2)$$

and

$$A^*(b_2) = \left\{ a \in G : O_t(a) > O_t(b_2) \right\}$$
$$\subseteq \left\{ a \in G : O_t(a) > O_t(b_1) \right\} = A^*(b_1).$$

The values of the set of valid observations are averaged to obtain an integrated characteristic of the BlockDAG network synchronization rate:

$$T_S^{DAG} = \frac{3T_{sl}}{|O_t|} \sum_{t \in O_t^{hon}} t.$$

Thus the computational complexity of such operation is $\underline{O}(|O_t| \ln |O_t|)$.

This characteristic can be used to determine the properties of node accessibility (the rate at which it receives information and information from it) and to estimate its foulness.

To determine the characteristics of the block rate to and from a given node, two characteristics must be taken into account:

- Visibility(Nd) — how many slots after block creation the blocks referring to it appear;
- Sight(Nd) — how many slots in the past from the block are the blocks to which it refers.

Sight node characteristic can be calculated at any time after receiving a block b from this node Nd by the following formula:

$$\text{Sight(Nd)} = \frac{1}{\deg^+(b)} \sum_{a \in \text{child}(b)} \left(\text{slot}(b) - \text{slot}(a) \right).$$

The characteristic Visibility is calculated similarly, but cannot be calculated immediately because you have to wait for the blocks that will use this one as a tips-block for reference.

We denote by create(Nd) the set of all blocks created by node Nd. With long observation, we can construct a dynamic edge-weighted graph $G_t(N, S, C)$ of the information flow topology of the BlockDAG network, where N is the set of BlockDAG network nodes,

$$S = \{(\text{Nd}_1, \text{Nd}_2) \in N \times N : \exists (a, b) \in E(G)$$
$$a \in \text{create}(\text{Nd}_1), b \in \text{create}(\text{Nd}_2)\}$$

is a set of arcs, which indicate which nodes refer to blocks of which other nodes, C is the set of weights, which specify visibility areas, Sight(Nd) is estimation of information propagation speed between pairs of nodes.

The proposed model allows us to build speed estimates based on algorithms for solving maximal flow problems. This will not only optimize the slot time in the network to achieve the speed, but also highlight the bottlenecks in the propagation of blocks. In the further development of the system, the economic stimulation of nodes to create additional communication channels in specific places of the information network linking DAG-nodes is assumed.

6 | CONCLUSION

This article defines main the aspects of "Waterfall: Gozalandia" in the Coordinating and BlockDAG networks. We have designed the distributed fast finality algorithm and have proven its liveness and safety. The use of DAG structures makes it possible for the protocol to achieve high scalability and network performance when conducting transactions and executing smart contracts. The protocol can be applied in both public and private networks. However, its advantages are more manifested in the public case, where nodes are large in number and can be online/offline without any confirmation, at the discretion of users or technical circumstances, while private networks are characterized by a more predictable behavior of nodes.

Further, the consensus protocol can be empowered by economic leverages to provide the system with sustainable development (e.g. see [36]). Positive actions of validators should be incentivized by tokenomics, and their malicious behavior should not be beneficial, but should be punished with penalties. The incentivizing system can facilitate appropriate protection from various kinds of attacks like Nothing-at-stake, Sybil etc., as well as some possible threats related to a DAG referential structure. Thus, a favorable ecosystem will be created for the provision of a wide spectrum of services in a public decentralized environment.

In the future works we will analyze possible attacks and responses to them. In addition, future research will be devoted to increasing the number of BlockDAG shards managed by the Coordinating network. Each shard will be optimized according to the operating conditions within a given metric.

AUTHOR CONTRIBUTIONS

Sergii Grybniak: Conceptualization, project administration, writing - review and editing. Yevhen Leonchyk: Formal analysis, writing - review and editing. Igor Mazurok: Formal analysis, writing - review and editing. Oleksandr Nashyvan: Conceptualization, software, writing - review and editing. Ruslan Shanin: Conceptualization, formal analysis, writing - original draft, writing - review and editing.

CONFLICT OF INTEREST

All authors declare that they have no conflicts of interest.

DATA AVAILABILITY STATEMENT

Data available on request from the authors

ORCID

Sergii Grybniak https://orcid.org/0000-0001-6817-8057

REFERENCES

1. Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., Danezis, G.: SoK: Consensus in the age of blockchains. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies, pp. 183–198. ACM, New York (2019)
2. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
3. Bach, L.M., Mihaljevic, B., Zagar, M.: Comparative analysis of blockchain consensus algorithms. In: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1545–1550. IEEE, Piscataway (2018)
4. Jing, C., Silvio, M.: Algorand. arXiv:1607.01341v9, pp. 1–75 (2017). https://arxiv.org/pdf/1607.01341v9.pdf
5. Cardano Team: Documentation for the Cardano ecosystem (2015). https://docs.cardano.org/
6. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, pp. 913–930. Association for Computing Machinery, New York (2018)
7. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: EUROCRYPT 2018. Lecture Notes in Computer Science, Part II, vol. 10821, pp. 66–98. Springer, Heidelberg (2018)
8. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: CRYPTO 2017. Lecture Notes in Computer Science, vol. 10401, pp. 357–388. Springer, Heidelberg (2017)
9. Daian, P., Pass, R., Shi, E.: Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proof of Stake. In: Goldberg, I., Moore, T. (eds.) Financial Cryptography and Data Security, pp. 23–41. Springer International Publishing, Cham (2019)
10. Castro, M., Liskov, B.: Practical Byzantine Fault Tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation, pp. 173–186. USENIX Association, Berkeley, CA (1999)
11. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Trans. Program. Lang. Syst. 4(3), 382–401 (1982)
12. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements forcryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51–68. ACM, New York (2017)
13. Nguyen, Q., Cronje, A., Kong, M., Lysenko, E., Guzev, A.: Lachesis: Scalable Asynchronous BFT on DAG Streams. arXiv:2108.01900v1, pp. 1–45 (2021). https://arxiv.org/pdf/2108.01900v1.pdf
14. Sompolinsky, Y., Lewenberg, Y., Zohar, A.: SPECTRE: Serialization of proof-of-work events—Confirming transactions via recursive elections (2017)
15. Sompolinsky, Y., Wyborski, S., Zohar, A.: PHANTOM GHOSTDAG: A scalable generalization of Nakamoto consensus. In: Proceedings of the 3rd ACM Conference on Advances in Financial Technologies, pp. 57–70. Association for Computing Machinery, New York (2021)
16. Sompolinsky, Y., Zohar, A.: Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. IACR Cryptology ePrint Archive 2013, 881 (2013)
17. Wang, Q., Yu, J., Chen, S., Xiang, Y.: SoK: Diving into DAG-based blockchain systems. arXiv:2012.06128, pp. 1–36 (2020). https://arxiv.org/pdf/2012.06128.pdf
18. Popov, S.: The tangle (2015). https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf
19. Baird, L., Harmon, M., Madsen, P.: Hedera: A public hashgraph network & governing council (2020). https://hedera.com/hh_whitepaper_v2.1-20200815.pdf
20. Choi, S.M., Park, J., Nguyen, Q., Cronje, A.: Fantom: A scalable framework for asynchronous distributed systems. arXiv:1810.10360v1, pp. 1–36 (2018). https://arxiv.org/pdf/1810.10360v1.pdf.
21. Nguyen, Q., Cronje, A., Kong, M., Kampa, A., Samman, G.: StakeDag: Stake-based consensus for scalable trustless systems. arXiv:1907.03655v1 (2019). https://arxiv.org/pdf/1907.03655v1.pdf
22. Tian, H., Lin, H., Zhang, F.: Design a proof of stake based directed acyclic graph chain. In: Frontiers in Cyber Security, FCS 2020. Communications in Computer and Information Science, vol. 1286, pp. 150–165. Springer, Singapore (2020)
23. Zhou, T., Li, X., Zhao, H.: DLattice: A permission-less blockchain based on DPoS-BA-DAG consensus for data tokenization. IEEE Access 7, 39273–39287 (2019)
24. Hoang, V.T., Morris, B., Rogaway, P.: An Enciphering Scheme Based on a Card Shuffle. In: Advances in Cryptology – CRYPTO 2012, pp. 1–13. Springer, Berlin Heidelberg (2012)
25. Lamport, L.: Proving the correctness of multiprocess programs. IEEE Trans. Software Eng. SE-3(2), 125–143 (1977)

26. Szabo, N.: Smart contracts (1994). https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html
27. Ante, L.: The Non-Fungible Token (NFT) Market and its relationship with Bitcoin and Ethereum. BRL Work. Pap. Ser. 21, 1–15 (2021). https://doi.org/10.2139/ssrn.3861106
28. Hoffmann, C. H.: Blockchain use cases revisited: micro-lending solutions for retail banking and financial inclusion. J. Syst. Sci. Inf. 9(1), 1–15 (2021)
29. Malamud, S., Rostek, M.: Decentralized exchange. Am. Econ. Rev. 107(11), 18–25 (2017). https://doi.org/10.1257/aer.20140759
30. Schär, F.: Decentralized finance: On blockchain- and smart contract-based financial markets. Federal Reserve Bank of St. Louis Review, Second Quarter 103(2), 153–174 (2021). https://doi.org/10.20955/r.103.153-74
31. Zetzsche, D.A., Arner, D.W., Buckley, R.P.: Decentralized finance. J. Financ. Regul. 6(2), 172–203 (2020). https://doi.org/10.1093/jfr/fjaa010
32. Lau, K.: Ethereum 2.0: An introduction (2020). https://assets.ctfassets.net/hfgyig42jimx/7j3AVp5aCnx2Ct2P6T6kY/e8991d5da1972d5d760233868f237609/Crypto.com_Macro_Report_-_Ethereum_2.0.pdf
33. Anceaume, E., Pozzo, A., Rieutord, T., Tucci-Piergiovanni, S.: On finality in blockchains. arXiv:2012.10172v1 (2020). https://arxiv.org/pdf/2012.10172.pdf
34. Anceaume, E., Pozzo, A., Rieutord, T., Tucci-Piergiovanni, S.: On finality in blockchains (2021). https://hal-cea.archives-ouvertes.fr/cea-03080029v2/file/Eventual_Finality.pdf
35. West, D.B.: Introduction to Graph Theory, 2nd ed. Pearson Education, Inc., Boston (2001)
36. Mazurok, I., Pienko, V., Leonchyk, Y.: Empowering fault-tolerant consensus algorithm by economic leverages. In: ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer. Part II: 7th International Workshop on Information Technologies in Economic Research, pp. 465–472. Springer, Heidelberg (2019)

How to cite this article: Grybniak, S., Leonchyk, Y., Mazurok, I., Nashyvan, O., Shanin, R.: Waterfall: Gozalandia. Distributed protocol with fast finality and proven safety and liveness. IET Blockchain 1–12 (2023). https://doi.org/10.1049/blc2.12023