National prior rights within the meaning of Article 139(2) EPC are not a bar to the grant of a European patent in proceedings before the EPO. Therefore, the EPO is not required to search for and assess such rights (GL, H-III, 4.4). Applicants may, however, consider the procedural option under Rule 138 EPC in view of the effects of such rights in national proceedings and/or before the Unified Patent Court (Article 3 Regulation (EU) No 1257/2012). As a support service free of charge for the applicant in this context, the applicant is hereby offered non-binding information on a search for and prima facie relevance assessment of national prior rights by the examining division. It is the applicant's responsibility to assess such national prior rights and any use of the procedural option under Rule 138 EPC (GL, H-III, 4.4). The applicant is informed that no prima facie relevant national prior rights were found.

Europäisches Patentamt
European Patent Office
Office européen des brevets

European Patent Office
80298 MUNICH
GERMANY
Tel: +49 89 2399 0
Fax: +49 89 2399 4465

Annex to EPO Form 2004, Communication pursuant to Rule 71(3) EPC

Bibliographical data of European patent application No. 20 807 112.6

For the intended grant of the European patent, the bibliographical data are set out below, for information:

Title of invention:

- COMPUTERIMPLEMENTIERTES VERFAHREN ZUM ERREICHEN EINES VERTEILTEN KONSENSUS IN EINEM BLOCKCHAIN-NETZWERK UND KNOTEN ZUR DURCHFÜHRUNG DES VERFAHRENS
- COMPUTER-IMPLEMENTED METHOD FOR REACHING A DISTRIBUTED CONSENSUS IN A BLOCKCHAIN NETWORK AND NODE IMPLEMENTING THE METHOD
- PROCÉDÉ MIS EN OEUVRE PAR ORDINATEUR POUR ATTEINDRE UN CONSENSUS DISTRIBUÉ DANS UN RÉSEAU À CHAÎNE DE BLOCS ET NOEUD METTANT EN OEUVRE LE PROCÉDÉ

Classification: INV. H04L9/00 H04L9/32

Date of filing: 15.10.2020

Priority claimed: IT / 15.10.2019 / ITA201900018935
IT / 31.10.2019 / ITA201900020218

Contracting States* for which fees have been paid: AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

Extension States* for which fees have been paid:

Validation States* for which fees have been paid:

Applicant(s):**

Ailia SA
Weinberghohe 27
6300 Zug
CH

Inventor(s):

ANTINO, Giovanni
c/o Ailia SA Weinberghohe 27
6300 Zug
CH

DIMNI, Iris
c/o Ailia SA Weinberghohe 27
6300 Zug
CH

Europäisches Patentamt
European Patent Office
Office européen des brevets

European Patent Office
80298 MUNICH
GERMANY
Tel: +49 89 2399 0
Fax: +49 89 2399 4465

PASETTO, Francesco
c/o Ailia SA Weinberghohe 27
6300 Zug
CH

*)	If the time limit for the payment of designation fees according to Rule 39(1) EPC has not yet expired and the applicant has not withdrawn any designation, **all Contracting States/Extension States/Validation States** are currently still deemed to be designated. See also Rule 71a(3) EPC and, if applicable, the above Note to users of the automatic debiting procedure.
**)	If two or more applicants have designated different Contracting States, this is indicated here.

Europäisches Patentamt
European Patent Office
Office européen des brevets

European Patent Office
80298 MUNICH
GERMANY
Tel: +49 89 2399 0
Fax: +49 89 2399 4465

Mola, Edoardo
Praxi Intellectual Property S.p.A.
Corso Vittorio Emanuele II, 3
10125 Torino
ITALIE

Formalities Officer
Name: Gottar, Chantal
Tel: +49 89 2399 - 3203
or call
+31 (0)70 340 45 00

Application No.	Ref.	Date
20 807 112.6 - 1218	P3183EPPC	01.03.2023
Applicant Ailia SA		

Communication under Rule 71(3) EPC

1. Intention to grant

You are informed that the examining division intends to grant a European patent on the basis of the above application, with the text and drawings and the related bibliographic data as indicated below.

A copy of the relevant documents is enclosed.

1.1 In the text for the Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

Description, Pages

1-38 as published

Claims, Numbers

1-15 filed in electronic form on 05-12-2022

Drawings, Sheets

1/12-12/12 as published

With the following amendments to the above-mentioned documents proposed by the division

Description, Pages 3

Claims, Numbers 1, 14

Comments

DESCRIPTION
Page 3: Mention of relevant prior art in the description (Rule 42(1) EPC)
Page 3: Translation from Italian into English
CLAIMS
Pages 1, 8, CLAIMS 1, 14: Scope of claim unclear - clarified, missing antecedent (Art. 84 EPC)
Page 1, Claim 1: Absent or incorrect two-part form of claim (Rule 43(1) EPC)

See also the comments in enclosed EPO Form 2906.

1.2 Bibliographic data

The title of the invention in the three official languages of the European Patent Office, the international patent classification, the designated contracting states, the registered name(s) of the applicant(s) and the other bibliographic data are shown on **EPO Form 2056** (enclosed).

2. Invitation

You are invited, **within a non-extendable period of four months** of notification of this communication,

2.1 to EITHER approve the text communicated above and verify the bibliographic data (Rule 71(5) EPC)

(1) by filing a translation of the claim(s) in the other two official languages of the EPO

	Fee code	EUR
(2a) by paying the fee for grant including the fee for publication:	**007**	990.00
minus any amount already paid (Rule 71a(5) EPC):		0.00
	Total amount:	990.00
(3) by paying additional claims fees under Rule 71(4) EPC; number of claims fees payable: 0	**016**	0.00
minus any amount already paid (Rule 71a(5) EPC):		0.00
	Total amount:	0.00

Important: If the translations of the claims and fees have already been filed and paid respectively in reply to a previous communication under Rule 71(3) EPC, e.g. in the case of resumption of examination after approval (see Guidelines C-V, 6), **agreement as to the text to be granted** (Rule 71a(1) EPC) must be expressed within the same time limit (e.g. by approving the text and verifying the bibliographic data, by confirming that grant proceedings can go ahead with the documents on file and/or by stating which translations of the claims already on file are to be used).

Note 1: See "Notes concerning fee payments" below.
Note 2: Any overpaid "minus" amounts will be refunded when the decision to grant (EPO Form 2006A) has been issued.
Note 3: For the calculation of the grant fee under Article 2(2), No. 7, RFees (old fee structure), the number of pages is determined on the basis of a clean copy of the application documents, in which text deleted as a result of any amendments by the examining division is not shown. Such clean copy is made available via on-line file inspection only.

2.2 OR, in the case of disapproval, to request reasoned amendments or corrections to the text communicated above or keep to the latest text submitted by you (Rule 71(6) EPC).

In this case the translations of the claims and fee payments mentioned under point 2.1 above are NOT due.

The terms "amendment(s)" and "correction(s)" refer only to amendments or corrections of the application documents and not of other documents (e.g. bibliographic data, the designation of the inventor, etc.).

If filing amendments, you must identify them and indicate the basis for them in the application as filed. Failure to meet either requirement may lead to a communication from the examining division requesting that you correct this deficiency (Rule 137(4) EPC).

2.3 Bibliographic data

Where you request a change or correction of bibliographic data in response to the Rule 71(3) communication, this will **not** cause the sending of a further communication under Rule 71(3) EPC. You will still have to pay the fees and file translations in reply to the Rule 71(3) communication in the case of 2.1 above, unless you also file a reasoned request for amendments or corrections in response to the Rule 71(3) communication (see case 2.2 above).

3. Loss of rights

If neither of the two possible actions above (see points 2.1 or 2.2) is performed in due time, the European patent application will be deemed to be withdrawn (Rule 71(7) EPC).

4. Further procedure

4.1 In the case of point 2.1 above

4.1.1 The decision to grant the European patent will be issued, and the **mention of the grant** of the patent will be published in the European Patent Bulletin, if the requirements concerning the translation of the claims and the payment of all fees are fulfilled and there is agreement as to the text to be granted (Rule 71a(1) EPC).

Note on payment of the renewal fee:
If a renewal fee becomes due before the next possible date for publication of the mention of the grant of the European patent, publication will be effected only after the renewal fee and any additional fee have been paid (Rule 71a(4) EPC).

Under Article 86(2) EPC, the obligation to pay renewal fees to the European Patent Office terminates with the payment of the renewal fee due in respect of the year in which the mention of the grant of the European patent is published.

Note on payment of the designation fee(s):
If the designation fee(s) become(s) due after the communication under Rule 71(3) EPC, the mention of the grant of the European patent will not be published until these fees have been paid (Rule 71a(3) EPC).

4.1.2 After publication, the **European patent specification** can be downloaded free of charge from the EPO publication server https://data.epo.org/publication-server.

4.1.3 Filing of translations in the contracting states

As regards translation requirements prescribed by the contracting states under Article 65(1) EPC, please consult the website of the European Patent Office
www.epo.org ➞Law & practice ➞Legal texts, National law relating to the EPC
www.epo.org ➞Law & practice ➞All Legal texts ➞London Agreement

In the case of a valid extension or validation

As regards translation requirements prescribed by the extension or validation states, please consult the website of the European Patent Office
www.epo.org ➞Law & practice ➞Legal texts, National law relating to the EPC

Failure to supply a prescribed translation in a contracting state, or in an extension or validation state may result in the patent being deemed to be void *ab initio* in the state concerned (Art. 65(3) EPC).

4.2 In the case of 2.2 above

If the present communication under Rule 71(3) EPC is based on an auxiliary request and, within the time limit, you maintain the main request or a higher ranking request which is not allowable, the application will be refused (Art. 97(2) EPC).

If the examining division gives its consent to the requested amendments or corrections, it will issue a new communication under Rule 71(3) EPC; otherwise, it shall resume the examination proceedings (Rule 71(6) EPC).

5. Filing of a divisional application

Any divisional application relating to this European patent application must be filed directly with the European Patent Office in Munich, The Hague or Berlin and will be in the language of the proceedings for the present application, or if the latter was not in an official language of the EPO, the divisional application may be filed in the language of the present application as filed (see Article 76(1) and Rule 36(2) EPC). Any such divisional application must be filed while the present application is still pending (Rule 36(1) EPC; Guidelines A-IV, 1.1.1).

6. Notes concerning fee payments

6.1 Making payments

For payments made via deposit account, please note that as from 1 December 2017 debit orders will only be carried out if filed in an electronically processable format (xml), using an accepted means of filing as laid down in the Arrangements for deposit accounts (ADA), published in the Supplementary publication in the Official Journal.

All relevant information related to the modes of payment of fees to the EPO can be retrieved from the EPO website at "**Making Payments**".

6.2 Information concerning fee amounts

Procedural fees are usually adjusted every two years, on even years, with effect from 1 April. Therefore, before making a payment, parties should verify the amounts actually due on the date of payment using the applicable version of the Schedule of fees and expenses, published as a Supplement to the Official Journal of the EPO, available on the EPO website (**www.epo.org**) at **www.epo.org/schedule-of-fees**. The "Schedule of fees" table allows the viewing, downloading and searching of individual fee amounts, both current and previous.

6.3 Note to users of the automatic debiting procedure

The fee for grant, including the fee for publication, and any additional claims fees due under Rule 71(4) EPC will be debited automatically on the date of filing of the translations of the claims, or on the last day of the period of this communication. However, if the designation fee(s) become(s) due as set out in Rule 71a(3) EPC and/or a renewal fee becomes due as set out in Rule 71a(4) EPC, these should be paid separately by another permitted way of payment in order not to delay the publication of the mention of the grant. The same applies in these circumstances to the payment of extension and validation fees.

Examining Division:

Chairman:	Apostolescu, Radu
2nd Examiner:	Jascau, Adrian
1st Examiner:	Di Felice, M

Gottar, Chantal
For the Examining Division
Tel. No.: +49 89 2399 - 3203

Enclosures: Text intended for grant

EPO Form 2056
EPO Form 2906

Europäisches Patentamt
European Patent Office
Office européen des brevets

Application No.: EP20807112

Intention to grant a European patent

Electronically authenticated

01.03.2023

Date

Radu Apostolescu	Massimiliano Di Felice	Adrian Jascau
Chairperson	1st examiner	2nd examiner

"COMPUTER-IMPLEMENTED METHOD FOR REACHING A DISTRIBUTED CONSENSUS IN A BLOCKCHAIN NETWORK AND NODE IMPLEMENTING THE METHOD"

DESCRIPTION

FIELD OF INVETION

The present invention concerns a computer-implemented method for reaching a proof of stake based distributed consensus for a distributed ledgers, such as the blockchain technology.

STATE OF THE ART

The invention is particularly suited, but not limited to, achieving higher level of distribution of a proof of stake based network in order to not have too much stake concentrated in too few nodes, thus increasing the robustness of the network and lowering its vulnerability to any one node being compromised.

In this document we use the term blockchain to include all forms of electronic, computer-based, distributed ledgers. These include, but are not limited to blockchain and transaction-chain technologies, permissioned and un-permissioned ledgers, consensus-based ledgers, shared ledgers and variations thereof. The most widely known application of blockchain technology is the Bitcoin ledger, although other blockchain implementations have been proposed and developed. A blockchain is a consensus-based, electronic ledger which is implemented as a computer-based decentralised, distributed system made up of blocks which in turn are made up of transactions and other information. Bitcoin is an example of a proof-of-work blockchain in which miners perform expensive computations in order to facilitate transactions on the blockchain.

Proof-of-work based blockchains have been criticized due to the large computing resources required, which requires a large amount of power consumption to operate and also due to the fact that block-generation may be irregular and slow. Moreover, several blocks must be built on top of a given block before the given block is considered to be confirmed (i.e., sufficiently unlikely to be reverted).

Proof-of-stake based blockchains have been proposed as an alternative to proof-of-work blockchains. In a proof-of-stake blockchain network, the blockchain is secured by proof-of-stake rather than proof-of-work. Under proof-of-stake, miners hold a stake (deposit some tokens) in a special account. This stake may be referred to as a security deposit and the probability of being selected as the node to mine a block is proportional to the quantity of the digital assets provided as a security deposit. Proof-of-stake blockchain networks can be used to avoid the computational expense and energy required to mine on proof-of-work blockchains. Further, proof-of-stake blockchains can allow for higher frequency and more regular block creation than proof-of-work blockchains. At least some proof-of-stake blockchains also have a low probability of forking and a block may be effectively confirmed as soon as it is added to the blockchain.

As mentioned before, although the economy of scale leads to the concentration of mining resources to increase network efficiency, in developing a distributed network, concentration is a problem concerning network control, network robustness because it is impossible to carry out planned maintenance operations on the nodes. The main issue in small, concentrated networks, has to do with low redundancy in case of failures, whether it's network, hardware or software problems. The opposite situation, in which there is an excessive fragmentation, can also be counterproductive. This is because information transmission and replication always present an overhead within a real system. Furthermore assigning stakes directly to mining nodes may lead to problem of

overloading if distribution constraints are present.

Thus, there is a need to tackle the abovementioned concentration challenges and improve the proof of stake consensus protocol to obtain a protocol encouraging distribution and at the same time efficiency of the network.

C. Badertscher et al: "Ouroboros Genesis Composable Proof-of-Stake Blockchains with Dynamic Availability"; A. Kiayas et al: "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol"; and T. Deepak et al: "CloudPoS: A Proof-of-Stake Consensus Design for Blockchain Integrated Cloud" disclose various examples of incentive management in proof-of-stake based consensus mechanisms for blockchain networks.

OBJECTIVES OF THE INVENTION

One objective of the present invention, according to a first of its aspects, is obtaining a computer-implemented method able to encourage the distribution of the network and efficiency of the miners, and at the same time optimizing blockchain scalability.

A second objective of the present invention is obtaining a computer-implemented method able to improve robustness of the network.

A further objective is to optimize the computer-implemented method in order to increase the global security of the blockchain network.

A further objective of the present invention is to provide a the computer-implemented method that encourages the use of honest performing nodes to mine the subsequent block and discourage the use of nodes that do not perform well or that have intentionally bad or malicious behaviour.

An objective of the present invention is also to keep low the energy consumption of the network and possibly increasing node computational power only when needed.

A further objective of the present invention is to provide a system implementing the computer-implemented method easily to be designed, with an optimized energy and computational power consumption and extremely efficient.

BRIEF DESCRIPTION OF THE INVENTION

Hereinafter are summarized some technical aspects of the present inventions which enable some of the most important purposes to be achieved.

According to a first aspect this invention relates to a computer-implemented method for

reaching a proof of stake based distributed consensus for a blockchain network, thus allowing to achieve higher level of distribution of the network and to not have too much stake concentrated in too few nodes, said blockchain network comprising one or more MAIN addresses, for each MAIN addresses being associated in turn one or more OVERFLOW addresses/nodes, the last being able to act as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node, comprising the following processes:

-a redistribution process being configured to calculate the minimum stake necessary for an OVERFLOW node to mine, being minimum_stake_value=total_amount_at stake/predetermined_target_number_of_nodes, to calculate the maximum stake allowing efficient mining, being maximum_stake_value=minimum_stake_value*2, for each MAIN address to distribute the stake among the OVERFLOW nodes, to scroll through the OVERFLOW nodes of a given MAIN address and to assign to every OVERFLOW node the minimum_stake_value, until the remaining stake is insufficient to activate a miner or every overflow node has been assigned its stake;

-a penalty process being configured to freeze an amount of reward corresponding to the computational work for having mined a number of slots over a predetermined limit Slot_MAX in an $epoch_{i+1}$, and to recover it in the following epochs starting from $epoch_{i+1}$, a little at a time, according to a number of parts proportional to the number of slots mined over said predetermined limit Slot_MAX, being one part assigned to each following slot the miner is delegable to mine, thus in order to naturally incentive the distribution of the blockchain network.

Such a computer implemented method naturally enables high distribution of the blockchain network.

According to a second aspect this invention relates to a computer-implemented method

wherein the redistribution process may further be configured in such a way that if there is any stake left it is distributed between the OVERFLOW nodes until as many nodes as possible have a stake equal to maximum stake value and any further stake is assigned to the first OVERFLOW node according a predetermined order

It enables to optimize the redistribution of the total stake assigned to a pool of OVERFLOW nodes linked to a MAIN address and makes that only first OVERFLOW nodes according an univocal progressive predetermined numeration may be under penalty regime.

According to a third aspect this invention relates to a computer-implemented method further comprising an initializing process being configured to list all MAIN addresses of the blockchain network, for each MAIN address list its assigned OVERFLOW addresses/nodes, and then for each OVERFLOW addresses/node transfer the stake if it has any, to its assigned MAIN address.

It enables to optimize the redistribution process in case there may be any stake on a OVERFLOW node before the redistribution process be implemented.

According to a fourth aspect this invention relates to a computer-implemented method further comprising a maintenance process, comprising the following steps, being given a MAIN address and at least two or more OVERFLOW addresses and a node desired to be maintained:

110: sending a transaction including a first input providing a substitute OVERFLOW address using its private key, thus meaning said substitute OVERFLOW address being a node substituting the node desired to be maintained

120: assigning said substitute OVERFLOW address to the corresponding MAIN address using private key of the MAIN address

130 deregistering the desired to be maintained using its private key. Alternatively the

MAIN address can UNASSIGN an overflow node previously assigned to it.

It enables to keep the blockchain network highly distributed even in case of failure of one or more mining nodes.

According to a further aspects this invention relates to:

- a node configured to implement a method for reaching a proof of stake based distributed consensus for a blockchain network, thus allowing to achieve higher level of distribution of the network and to not have too much stake concentrated in too few nodes, said node comprising an interface device, a processor coupled to said interface device, a memory coupled to the processor, the memory having stored thereon computer executable instructions which, when executed, configure the processor to perform the corresponding steps of the computer implemented method disclosed.

-a computer readable storage medium comprising computer-executable instructions which, when executed, configure a processor to perform the disclosed computer implemented method.

-a computer network for performing a blockchain-based transaction from a first user to a second user in an electronic transfer network of connected nodes, wherein the nodes includes at least a first node, a second node, disclosed, wherein each node share a same blockchain, and wherein each node stores a computer readable storage medium disclosed.

According to an aspect of the present invention, a completely decentralized scheme is proposed for determining the order of access to a shared resource. The basic idea is to order the nodes, e.g. OVERFLOW, enabled to access the shared resource according to their own identifier. Each enabled node is assigned a numerical interval whose size is proportional to the number of time slots in which the same node has the right to have access to the shared resource.

A node is enabled for access when it has the right to access the shared resource for at

least one time slot.

Consequently, the numerical intervals are consecutive and also ordered according to the same order of the unique identifiers of the nodes.

This alignment of numerical intervals is shared and known by all nodes. Over time, this alignment may change in relation to the presence and breadth of individual numerical intervals.

For each epoch, the nodes enabled to access the shared resource calculate or acquire a seed, that is a random or pseudorandom number. By epoch it is meant, in summary according to the TDMA - "Time division multiple access" approach, a macro interval of subdivision of time, in which each epoch is divided into so-called time slots. The number of time slots assigned to each node sharing a shared resource is predetermined according to a predetermined distribution logic.

Each node independently performs the following operations in succession for a number of cycles equal to the number of time slots at the time:

- i-th calculation of the i-th Hash of the seed,
- Calculation of the remainder of the division of said i-th Hash by the sum of the amplitudes of said numerical intervals,
- Determination of the interval in which said remainder falls,
- Allocation of the i-th slot to the node corresponding to said interval.

The network can be formed by a greater number than the enabled nodes. The redistribution of time slots between nodes can depend on authoritative or random policies.

The seed is a random or pseudo-causal number that can be extracted from a specific node that, for example, has the sole task of generating the seed.

When the present invention is applied to the field of blockchains, the shared resource

consists of the task of undermining a block and the seed can be determined on the basis of a hash built on the basis of one or more blocks of a previous era.

For example, it is possible to divide the number of mined blocks of a previous epoch into three or four groups and to consider the hash of the first block of each group or the hash of the second block of each group, or the first, second and third hash of the second group or the hash of the last, penultimate and third last block of the first set of blocks.

The hashes extracted in this way are concatenated. Their concatenation or the hash of their concatenation (hash of hash) defines the seed for calculating the order of access to the shared resource of the next epoch.

Advantageously, since all nodes keep a copy of the blockchain, each node is able to calculate the order in which each enabled node has the right to access the shared resource. The essential elements that each node must know are the distribution of the time slots between the enabled nodes, the unique identifier of all the enabled nodes and the seed.

According to the application of this construction variant to blockchains, these are of the Proof of Stake type, in which there is no competition to complete the construction or mining of the blocks. The slots are assigned to the nodes according to the method described above and each node has the task of completing a block within this time slot, inserting queued and pending transactions at the time the time slot begins.

Advantageously, the present embodiment allows a blockchain to continue in its tasks even in the event of random and momentary disconnection of the nodes.

The performance of a node can be appreciated in relation to the number of transactions that it can include in a block generated in the assigned time slot.

At the end of the block generation, the node assigning the time slot forwards it to the network of nodes, which accept or reject this block, if it is correct. When the block is accepted by 50% + 1 of the nodes, the block is considered as definitively accepted and

therefore, the blockchain can continue to grow by appending additional blocks to that block.

According to a preferred variant, once the time slot has elapsed, if the assignee node does not send the block generated in the same time slot in time, therefore called "late block", the assignee node of the next time slot starts generating its own block, called “new block”, ignoring the missed or late reception of the late block. This implies that some of the transactions present in the late block can be included in the new block. A bifurcation and competition between the late block and the new block is therefore determined. The competition is won by the block that first receives 50% + 1 approval from the remaining nodes of the network. Therefore, the acceptance of the delayed block or the new one may depend on the extent of the delay.

A further block is therefore appended to the block that wins the contest. It must be taken into account that, in blockchains, the block that is queued includes the hash of the block to which it is queued.

BRIEF DESCRIPTION OF THE FIGURES

The structural and functional features of the present invention and its advantages with respect to the known prior art, will become even clearer from the underlying claims, and in particular from an examination of the following description, made with reference to the attached figures which show a preferred but not limited schematic embodiment of the invented computer-implemented method, system, device, wherein:

Figure 1 illustrates a flow chart of the computer-implemented method according to the present invention;

Figure 2 illustrates an operative example of the registration process of the computer-implemented method according to the present invention;

Figure 3 illustrates an example scheme of the network including one MAIN address and

three OVERFLOW nodes according to the present invention;

Figure 4 illustrates a flow chart of the initialization process of the computer-implemented method according to the present invention;

Figure 5 illustrates an operative flow chart of the initialization process of the computer-implemented method according to the present invention;

Figure 6 illustrates an operative example of the initialization process of the computer-implemented method according to the present invention;

Figure 7 illustrates a flow chart of the redistribution process of the computer-implemented method according to the present invention;

Figure 8 illustrates an operative flow chart of the redistribution process of the computer-implemented method according to the present invention;

Figure 8 illustrates an operative flow chart of the redistribution process of the computer-implemented method according to the present invention;

Figure 9 illustrates an operative example of the redistribution process of the computer-implemented method according to the present invention;

Figure 10 illustrates a flow chart of the penalty process of the computer-implemented method according to the present invention.

Figure 11 is an association table of entities necessary for the autonomous calculation of access to a shared resource;

Figure 12 shows a numerical range;

Figure 13 shows a scheme for accessing a shared resource according to a preferred embodiment of the present invention;

Figure 14 shows a flow chart of a method according to a preferred embodiment of the present invention.

In the context of this description, the term "second" component does not imply the

presence of a "first" component. These terms are in fact used as labels to improve clarity and should not be understood in a limiting way.

DETAILED DESCRIPTION OF THE INVENTION

In general, this disclosure describes a computer-implemented method, system for reaching a proof of stake based distributed consensus for a distributed ledgers, such as the blockchain technology. In particular this invention relates to a proof of stake based blockchain technology where stakeholders cannot and do not have to communicate with mining nodes.

The blockchain technology according to the present invention provides for two main actors: stakeholders i.e. who owns tokens allowing control of the chain, and miner i.e. who control mining nodes actively operating in the network and entrusted with a sufficient amount of stake in order to obtain assignment of a certain number of work slots where they mine a correspondent certain number of block thus lengthening the chain.

The blockchain is considered a meta actor and it is the union of the nodes operating on the network. Sending a transaction to the blockchain leads that this transaction reaches each operating node which updates its internal status consistently. These update procedures are constructed in such a way as to ensure that the same initial conditions produce identical status updates in every node in a strictly deterministic way. This meta actor represents the massive update procedure on each node of the network.

From now on it is intended:

- a slot a time interval of 30 seconds. These slots are uniquely assigned to the mining nodes and within a given slot one and only one node has the right to produce a block. This assignment is unique and strictly deterministic and is performed before the start of each epoch;

- an epoch is intended a grouping of contiguous slots. Each epoch consists of an equal

number of slots. Assignment and redistribution of slots and stakes are calculated once for each epoch and remain unchanged until the end of the same.

- a stake is a numerical representation of the right to vote of each stakeholder of the network. The stake may be delegated with specific rules and operations to the nodes involved in the block creation.

- a MAIN address: this address is not associated with a physical object (a MAIN address in not associated to an active mining node) and acts as a placeholder indicating a physical resource group which the blockchain is desired to access, thus exposing its private key only during assignment transaction of OVERFLOW address to MAIN address and not during other operative transaction, increasing the global security of the chain; this MAIN address (for instance alice_main_address) is declared by its owner and its managed using its private key (for instance alice_key). Only the MAIN address may declare which OVERFLOW nodes are assigned to it;

- an OVERFLOW address: this address is associated with a physical object and acts as an active mining node; this OVERFLOW address (for instance node_1_overflow_address) is managed using its private key (for instance node_1_key); it is possible that the MAIN address manager and the OVERFLOW node manager are the same entity but they use different keys when impersonating Alice or Node_1; the owner of each MAIN address (for instance Alice) using a specific transaction associates each registered overflow node to its MAIN address. Only the MAIN address in the network may link an OVERFLOW address to itself, meaning that only the MAIN address may send an ASSIGN type transaction linking an OVERFLOW to a MAIN address. This operation is not commutative, i.e. an OVERFLOW address may not assign itself to a MAIN address. OVERFLOW address/node may not be shared between MAIN addresses, an OVERFLOW address may be assigned to only one MAIN address. Only an address

previously declared as OVERFLOW may be associated with a MAIN address. An OVERFLOW node may also deregister and by doing so, removes itself from both the pool of available nodes and the MAIN address it was assigned to beforehand. This is to give both MAIN and OVERFLOW nodes the possibility to protect themselves from unwanted situations in which they get linked together, whether it's the result of an honest mistake or malicious behaviour.

If an OVERFLOW address is not linked to any MAIN address it will be operative only as a replication node.

The computer implemented method according to the present invention (fig. 1) comprises, a redistribution process, a penalty process and may comprise a registration process, an initializing process and a maintenance process.

The computer implemented method according to the present invention is set up according to the following parameters and values, said values are to be intended as examples only and may be modified according specific technical needs:

Time Slot duration=30 seconds

Epoch Slots (E_S)= 24000 => number of slots for each epoch

Penalty Factor=2 => determines the amount of time penalty for the current protocol violation

Recovery Factor=1 => determines the age of return from the penalty regime

Target number of nodes (MAX_N)=400 => optimal number of nodes making the network

Minimum number of nodes (min_N)=200

Minimum slot per node Slot_min (E_S/MAX_N)=60

Maximum slot per node Slot_MAX (E_S/min_N)=120

Epoch Evaluation Point (EEP) (E_S/3)=8000

The registration process is configured to register one or more MAIN address, to register one or more OVERFLOW address/node and to associate said one or more OVERFLOW address/node to a MAIN address and to register said association.

The registration process (fig. 2, 3) implemented on each overflow node of the blockchain network comprises the following steps:

5: via the blockchain network, receiving a transaction including as input MAIN address and registering it in an address register using a private key of the manager of the MAIN address

10: via the blockchain network, receiving a transaction including as input one or more OVERFLOW addresses and registering said one or more OVERFLOW address in the address register using a private key of the manager of each OVERFLOW address

15: via the blockchain network, receiving a transaction including as input assigning the one or more OVERFLOW addresses to the MAIN address registered in the step 5.

The computer implemented method according to the present invention provides for setting a minimum amount of stake (S_min_VAL) assigned to an OVERFLOW node to be activated and considered as mining node and in order to avoid having all the stake concentrated on only a few nodes is settled a soft upper limit to the amount of stake that can be assigned to a node, beyond which mining node starts incurring in penalties. This should make concentration of stake beyond efficient ranges less appealing to nodes looking to reap the maximum possible rewards. The maximum amount of stake desirable (S_MAX_VAL) may be double the value of S_min_VAL.

The algorithm is designed to maximise the number of active nodes by spreading the excess stake among as many overflow nodes as possible in order to expand the network thus increasing its resilience.

The computer implemented method comprises an initializing process in order to recover

the incorrectly pointed stake on an OVERFLOW node and to collect them to the corresponding MAIN address before performing a redistribution process according to computer implemented method of the invention. The initializing process is useful when stakes are incorrectly bet on one or more OVERFLOW nodes, thus making the distribution between OVERFLOW nodes reliable. Furthermore the initializing process is useful to ensure the OVERFLOW addresses have no stakes assigned before performing the redistribution process.

The initializing process (fig. 4) is configured to list all MAIN addresses of the blockchain network, for each MAIN address list its assigned OVERFLOW addresses/nodes, and then for each OVERFLOW addresses/node transfer the stake if it has any, to its assigned MAIN address.

The initializing process (fig. 5, 6) implemented on each overflow node of the blockchain network in particular comprises the following steps:

20: having a blockchain network comprising one or more MAIN addresses, for each MAIN addresses are associated in turn one or more OVERFLOW addresses/nodes acting as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node;

25: via the blockchain network, receiving a transaction including as input addresses from address register comprising MAIN addresses and their corresponding OVERFLOW addresses;

30: associating to each MAIN address a univocal progressive number;

35: for each MAIN address verifying if it contains an amount of stake and if this value is >0 assigning this value to parameter M_A_S, and if this value is <=0 implementing step 35 to the next MAIN address member of the blockchain network and generating a transaction including output providing the M_A_S value associated to the corresponding

MAIN address,

40: associating to each OVERFLOW node a univocal progressive number

45: for each OVERFLOW node assigned to each corresponding MAIN address verifying if it contains an amount of stake defined OVERFLOW_ADDRESS.assigned_stake and if this value is >0 assigning this value to parameter O_A_S, calculating value MAIN_ADDRESS.assigned_stake as M_A_S+O_A_S and assigning O_A_S=0, and generating a transaction including output providing the updated value of the M_A_S value associated to the corresponding MAIN address, then implementing step 45 to the next OVERFLOW address assigned to the corresponding MAIN address considered under step 35,

and if OVERFLOW_ADDRESS.assigned_stake is <=0 implementing step 45 to the next OVERFLOW address assigned to the corresponding MAIN address considered under step 35

50: generating a transaction including output of the updated value of TOTAL_AT_STAKE value for each MAIN address of the blockchain network corresponding to the total amount of stake on each MAIN address after performing preceding steps.

The redistributing process (fig. 7) is configured to calculate the minimum stake necessary for an OVERFLOW node to mine, being minimum_stake_value=total_amount_at stake/predetermined_target_number_of_nodes, to calculate the maximum stake allowing efficient mining, being maximum stake value=minimum stake value*2, for each MAIN address to distribute the stake among the OVERFLOW nodes, to scroll through the OVERFLOW nodes of a given MAIN address and to assign to every OVERFLOW node the minimum_stake_value, until the remaining stake is insufficient to activate a miner or every overflow node has been assigned its stake. The redistribution process may further

be configured in such a way that if there is any stake left it is distributed between the OVERFLOW nodes until as many nodes as possible have a stake equal to maximum stake value and any further stake is assigned to the first OVERFLOW node according a predetermined order.

The redistributing process (fig. 8, 9) comprises in turn a first iteration redistribution process where the algorithm assigns to every available OVERFLOW node assigned to each MAIN address the minimum required amount of stake to became an active mining node, and a second iteration redistributing process the algorithm distributes the stake trying to maximise the number of overflow nodes operating at full stake in view of S_MAX_VAL settled for any OVERFLOW node.

The redistribution process comprises in particular the following steps:

52: having a blockchain network comprising one or more MAIN addresses, for each MAIN addresses are associated in turn one or more OVERFLOW addresses/nodes acting as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node;

55: getting as input TOTAL_AT_STAKE value for each MAIN address of the blockchain network corresponding to the total amount of stake on each MAIN address

60: calculating S_min_VAL= TOTAL AT STAKE/TARGET NUMBER OF_NODES being TARGET NUMBER OF_NODES = total number of OVERFLOW mining nodes in the chain network,

Calculating S_MAX_VAL=S_min_VAL * 2

Assuming a counter value ITERATION=0

65: if ITERATION value is <2 then updating the counter value ITERATION=ITERATION+1

70: for each MAIN address getting as input the MAIN_ADDRESS.assigned_stake value

and if this value is >0, thus meaning that the MAIN address under consideration contains an amount of stake, assigning this value to parameter M_A_S,

and if M_A_S value is >= S_MAX_VAL repeating step 70 to the next MAIN address,

75: if M_A_S value is < S_MAX_VAL getting as input OVERFLOW_ADDRESS.assigned_stake value for each OVERFLOW address associated to the MAIN under consideration, and

if OVERFLOW_ADDRESS.assigned_stake value is =0 repeating step 75 to the next OVERFLOW address associated to the MAIN under consideration, and

if OVERFLOW_ADDRESS.assigned_stake value is ≠0, assigning OVERFLOW_ADDRESS.assigned_stake value to O_A_S parameter, calculating MAIN_ADDRESS.assigned_stake = M_A_S-S_min_VAL, and calculating OVERFLOW_ADDRESS.assigner_stake =O_A_S+S_min_VAL, and repeating step 75 to the next OVERFLOW address associated to the MAIN under consideration.

The penalty process makes mining operations that do not respect settled rules disadvantageous, thus incentivising nodes to naturally expand the network thus increasing its resilience as already discussed.

From now on it is intended:

$epoch_i$ the epoch when miners registered their MAIN and OVERFLOW addresses,

$epoch_{i+1}$ the epoch when miners mine their assigned slots,

$epoch_{i+2}$ the epoch when rewards and penalties are evaluated,

The subscript "j" concerns $block_j$ written in the $slot_j$

The subscript "i" concerns $epoch_i$ and what is relevant to $epoch_i$.

The computer implemented method according to this invention includes a reward for miner for remunerating their calculation effort during their mining operation. If a miner is not able to write a block in the corresponding slot or he mines in a number of slots

over a predetermined limit, he will not receive in the following epoch the corresponding reward for the block not written or for the block mined over the predetermined limit (in each slot only one block may be written), but this reward (for the blocks mined over the limit) is frozen and will be recovered very slowly a little at a time (for instance according to a number of parts equal to twice the number of slots mined under penalty regime) for each following slots will have been mined by the miner, in particular according to a number of parts proportional to the number of slots mined over said predetermined limit Slot_MAX, being one part assigned to each following slot the miner is delegable to mine, thus in order to naturally incentive the distribution of the blockchain network.

As mentioned before a $reward_{i+2}$ corresponds to the reward for mining a $block_{j+1}$ in the $epoch_{i+1}$.

This $reward_{i+2}$ includes one coinbase (i.e. a new created coin) for the $block_{j, i+1}$ mined, and a $collected\_fee_{j, i+1}$. This collected $fee_{j, i+1}$ in turn includes the collected fee paid by users wanting a transaction to be performed and written in the relevant $block_{j, i+1}$ of blockchain network.

From now on it is intended:

$Penalty\_slots_{i+1}$= number of slots mined over the limit in the $epoch_{i+1}$ i.e. number of slots over Slot_MAX*2

$Frozen\ fee_{i+1}$= is the amount of reward not dispensed to the miners in the epoch $_{i+1}$ for having mined a number of slots over the limit Slot_MAX, thus having been under penalty regime.

The penalty process (fig. 10) implemented on each overflow node of the blockchain network comprises in particular the following steps:

80: calculating total $frozen\_fee_{i+1}$ as the amount of $reward_{i+2}$ not dispensed to the miners in a specific $epoch_{i+1}$ for having mined a number of slots over the limit Slot_MAX

85: calculating penalty_slots$_{i+1}$ as preferably twice (or other proportional law) the number of slots mined over the limit i.e. number of slots over Slot_MAX

90: calculating recovered_fee$_{j, i+1}$ value as frozen_fee$_{i+1}$/penalty_slots$_{i+1}$ i.e. the fee to be recovered to each slot

95: adding the recovered_fee$_{j, i+1}$ value to the reward of the MAIN_address for mining the block$_i$ in the epoch$_{i+1}$ as reward$_{j, i+1}$=one coinbase+collected_fee$_{j, i+1}$+recovered_fee$_{j, i+1}$

100: calculating the new value of frozen_fee$_{i+1}$ parameter as frozen_fee$_{i+1}$-recovered_fee$_{j, i+1}$ and the new value of penalty_slots$_{i+1}$ as penalty_slots$_{i+1}$-1 and repeat step 95 for the following block$_{j+1}$ in the epoch$_{i+1}$ while frozen_fee$_{i+1}$>0 or while penalty_slots$_{i+2}$>0.

At the end of each epoch, corresponding with the last block mined, penalty_slots and frozen_fee are transferred from the first OVERFLOW node to its MAIN address. Likewise, at the beginning of every epoch, corresponding with the first block mined, the penalty_slots and frozen_fee are transferred from the MAIN address to its first OVERFLOW node.

This procedure is necessary to avoid malicious behaviour such as undue transfers of tokens from one pool to another.

Also, since it's always possible to register a new overflow node and calculate its address so that it precedes in the order of overflows the one with the assigned penalties. By doing so, a pool would recover frozen fees without re-entering within the optimal parameters or extending the network. Considering that under normal circumstances fees outweigh the generated coinbase this would reduce significantly the efficacy of the penalties.

With regard to events of scheduled maintenance in order to keep the blockchain network highly distributed even in case of failure of one or more mining nodes, the computer-implemented invention according to the present invention may comprise a maintenance process comprising in turn the following steps, being given a MAIN address and at least

two or more OVERFLOW addresses and a node desired to be maintained:

110: sending a transaction including a first input providing a substitute OVERFLOW address using its private key, thus meaning said substitute OVERFLOW address being a node substituting the node desired to be maintained

120: assigning said substitute OVERFLOW address to the corresponding MAIN address using private key of the MAIN address

130 deregistering the node desired to be maintained using its private key. Alternatively the MAIN address can UNASSIGN an overflow node previously assigned to it.

With regard to blockchain security aspects according to the present invention, as per the way the assignment system is structured, MAIN address private keys come into play only for the short period of time necessary to generate registration and assignment transaction but is no longer involved in any subsequent processing phase. This exempts it from being available online on an object that may be subject to malicious attack. On the other hand presence constantly online of a physical servers and the corresponding keys necessary to operate on the network allows a malicious third party to steal private keys from them and to take over all the stake delegated to the MAIN address farm.

The present invention further concerns a node (preferably an OVERFLOW node) configured to implement the computer implemented method disclosed, said node comprising an interface device, a processor coupled to said interface device, a memory coupled to the processor, the memory having stored thereon computer executable instructions which, when executed, configure the processor to perform the corresponding steps of the computer implemented method disclosed.

The present invention further concerns a computer readable storage medium comprising computer-executable instructions which, when executed, configure a processor to perform the computer implemented method disclosed.

The present invention further concerns a computer network for performing a blockchain-based transaction from a first user to a second user in an electronic transfer network of connected nodes, wherein the nodes includes at least a first node, a second node, as disclosed before, wherein each node share a same blockchain, and wherein each node stores a computer readable storage medium disclosed before.

Hereinafter are described some operative examples of the computer implemented method according this invention.

First example: Maliciuos Operator

Suppose two MAIN addresses:

- Alice, alice_main_addr, node_a1, node_a2, … , node_a10 Stake 30.000
- Mallory, mallory_main_addr, node_m1, node_m2, … , node_m5 Stake 30.000

Total Stake: 1.200.000
Minimum Stake: 3.000 Maximum Stake: 6.000
Slot min: 60 Slot max: 120

To each block generated is assigned a reward of 1 token

In epoch 14 Alice and Mallory completed the registration of their main addresses and assigned them their nodes.

These will be active starting from epoch 15

In epoch 16 the reward will be calculated.

At the beginning of epoch 15 Alice will have 10 active nodes (node_a1, ..., node_a10) each with 60 slots assigned.

At the beginning of the epoch 15, Mallory will have active 5 nodes (node_m1, ..., node_m5) each with 120 slots assigned.

The slots of Alice and Mallory are the same because they have the same active stake. However, the distribution changes. Mallory has only 5 nodes while Alice has 10. None of them is penalized. Alice's nodes

are active with the minimum, Mallory's nodes with the maximum. Assuming a perfect execution of the protocol in epoch 15, without transactions to simplify the example.

The ten Alice nodes have assigned 60 slots, thus producing 60 blocks, in each block there will be a reward of 1 token. Being ten nodes (60 * 10 * 1) the reward will be 600 tokens. At the beginning of epoch 16, Alice's balance will increase from 30,000 to 30,600 tokens.

Let's now pass to Mallory, the five nodes of Mallory have assigned 120 slots, thus producing 120 blocks, in each block there will be a reward of 1 token. Being five nodes (120 * 5 * 1) the expected reward will be 600 tokens. At the beginning of epoch 16, Mallory's balance should increase from 30,000 to 30,600 tokens. Only three of Mallory's nodes, node_m3, node_m4, node_m5, have been modified to operate incorrectly. The causes of this behaviour can be various and now we will list some of them.

- An intentional change was made that attempts to perform invalid operations (duplicate transactions, accept false transactions with incorrect key, ...) = Voluntary malicious interaction.
- The server on which the node is installed is malfunctioning and fails the mathematical operations by marking as invalid valid transactions eg. Carol, with a budget of 100 tokens, sends 25 to Bob. The operation is legal but is marked as illegal in the product block. = Involuntary malevolent interaction.
- The server's network connection is unreliable and the block is not transmitted in time. = Involuntary malevolent interaction.

• The block is connected not to its immediate valid predecessor but to another predecessor violating the rule of extending the heavier chain. In this way, the node aims to make unwanted transactions disappear, for example a payment, from the history of the chain = Involuntary malevolent interaction.

These aspects cause the blocks generated by node_m5 to be rejected by the other nodes and not become part of the blockchain.

In this way the reward collected by Mallory will be:

Node	Collected Reward
node_m1	120
node_m2	120
node_m3	0
node_m4	0
node_m5	0
Total	240

Let's now pass to an evaluation of the remaining part of the network. Alice and Mallory have in total (30,000 Alice and 30,000 Mallory) 60,000 stake tokens.

On the rest of the network (1,200,000 - 60,000) 1,140,000 token in stake remain.

Assuming that all the rest of the network works in ideal conditions and without transactions to the other nodes making up the network, they will be assigned:

Total Slots	Alice's Slots	Mallory's	Slots

		Slots	assigned to the rest of the network
24000	-600	-600	=22800

22,800 blocks will be generated and therefore a reward of 22,800 tokens.

Total Stake	Alice's Stake	Mallory's Stake	Stake assigned to the rest of the network
1200000	-30000	-30000	=1140000

In the epoch 15 the total stake of the network is 1,200,000, the stake assigned to non-Alice or Mallory nodes is 1,140,000. In epoch 16 the reward for the aforementioned nodes will be 22,800 tokens, bringing their stake to 1,162,800 tokens.

Considering Mallory's main collected a lower stake, we recalculate the values for the next epoch.

The overall stake is 1,200,000 + 600 + 240 + 22,800 = 1,223,640

The following table compares the network control percentage in epoch 15 with the revaluation in epoch 16 (expected) and 16 (real)

	Epoch 15	**Epoch 16 (expected)**	**Epoch 16 (real)**
Alice	Stake	Stake	Stake
	30.000	30.600	30.600

	Slot assigned	Slot assigned	Slot assigned
	600	600	600
	Network control	Network control	Network control
	2,5%	2,5%	2,5%
Mallory	Stake	Stake	Stake
	30.000	30.600	30.240
	Slot assigned	Slot assigned	Slot assigned
	600	600	593 (-7) (-1,17%)
	Network control	Network control	Network control
	2,5%	2,5%	2,47% (-0.03%)
Other participants	Stake	Stake	Stake
	1.140.000	1.162.800	1.162.800
	Slot assigned	Slot assigned	Slot assigned
	22.800	22.800	22.807 (+7)
	Network control	Network control	Network control
	95%	95%	95,03% (+0,03%)

As you can see, Mallory's computing power was not eroded (the algorithm is not punitive in the classical sense of removing something) but it has grown more slowly than that of the other components of the network. This caused, at the point of

recalculation of the epoch 16, a change in the slot allocation with Mallory that we saw take off 0.03% of the computing power compared to the total of the network slots. Instead, comparing it to its previous slot number, the erosion corresponds to the loss of 1.17% of the slots.

Second example: Unbalanced stake with fee

Suppose two MAIN addresses:

- Alice, alice_main_addr, node_a1, node_a2, … , node_a10
  Stake 30.000
- Mallory, mallory_main_addr, node_m1
  Stake 30.000

Total Stake: 1.200.000
Minimum Stake: 3.000 Maximum Stake: 6.000
Slot min: 60 Slot max: 120

To each block generated is assigned a reward of 1 token

In epoch 14 Alice and Mallory completed the registration of their main addresses and assigned them their nodes.

These will be active starting from epoch 15

In epoch 16 the reward will be calculated.

At the beginning of epoch 15 Alice will have ten active nodes (node_a1, ..., node_a10) each with 60 slots assigned.

At the beginning of epoch 15, Mallory will have one node_m1 node active with 600 slots assigned.

The slots of Alice and Mallory are the same because they have the same active stake. However, the distribution changes. Mallory has only one node while Alice has ten of them. Mallory is penalized. Alice's nodes are active with the minimum, the Mallory node is 480 slots beyond the limit and for this she will receive a penalty.

Assuming a perfect execution of the protocol in epoch 15, we assume that an average of 5 tokens / block is the amount of collected fee for greater simplicity.

The ten Alice nodes have assigned 60 slots, thus producing 60 blocks, in each block there will be a reward of 1 token. Being ten nodes (60 * 10 * 1) the reward will be 600 tokens. At the beginning of epoch 16, Alice's balance will increase from 30,000 to 30,600 tokens. To these we will add the rewards of operations 600 * 5 = 3.000 tokens. Alice will total 33,600 tokens as a result of epoch 15 operations.

Let's now pass to Mallory, the Mallory node has assigned 600 slots, it will therefore produce 600 blocks, for the first 120 blocks a reward of 1 token will be assigned. For the following 480 the reward will be 0. As for the rewards, using the homogeneous distribution indicated in the preconditions, Mallory will be awarded only in the first 120 slots. For these you will receive 5 * 120 = 600 tokens. The remaining 2,400 are frozen to be returned in a period of 480 * 2 = 960 slots within the limits. Assuming that the configuration of the nodes does not change from epoch 15 to epoch 16, we evaluate what happens to the EEP of epoch 16 for epoch 17.

At the beginning of epoch 16, Mallory's balance should increase from 30,000 to 30,600 tokens as a result of the coinbase and should be assigned a further 3,000 tokens due to the fees. Only that Mallory has activated a number of nodes that are insufficient to meet the network's requirements.

Node	Collected Reward

node_m1	120 (coinbase) + 600 (fee) *2400 are frozen for 960 slots and they may not enter in the reward calculation*
Total	720

Let's now pass to an evaluation of the remaining part of the network. Alice and Mallory have in total (30,000 Alice and 30,000 Mallory) 60,000 stake tokens.

On the rest of the network (1,200,000 - 60,000) = 1,140,000 token in stake remain.

Assuming that all the rest of the network works in ideal conditions and without transactions to the other nodes making up the network, they will be assigned:

Total Slots	Alice's Slots	Mallory's Slots	Slots assigned to the rest of the network
24000	-600	-600	=22800

22,800 blocks will be generated and then a reward of 22,800 tokens for the coinbase portion. For the portion of fees 114,000 tokens

Stake Previous other nodes	1.140.000	+	
Coinbase other nodes	22.800	+	
Fee other nodes	114.000	=	
Total other nodes			***1.276.800*** **+**
Stake Previous Epoch Alice	30.000	+	
Coinbase Alice	600	+	

Fee Alice	3.000 =	
	Total Alice	***33.600 +***
Stake Previous Epoch Mallory	30.000 +	
Coinbase Mallory	120 +	
Fee Mallory	600 =	
	Total Mallory	***30.720 =***
Total stake for new epoch		**1.341.120**

The following table compares the network control percentage in epoch 15 with the revaluation in epoch 16 (expected) and 16 (real)

	Epoch 15	**Epoch 16 (expected)**	**Epoch 16 (real)**
Alice	Stake	Stake	Stake
	30.000	33.600	33.600
	Slot assigned	Slot assigned	Slot assigned
	600	600	601 (+1)
	Network control	Network control	Network control
	2,5%	2,5%	2,51% (+0,01)
Mallory	Stake	Stake	Stake
	30.000	33.600	30.720
	Slot assigned	Slot assigned	Slot assigned
	600	600	550 (-50) (-

			8,33%)
	Network control	Network control	Network control
	2,5%	2,5%	2,29% (-0.21%)
Other participants	Stake	Stake	Stake
	1.140.000	1.276.800	1.276.800
	Slot assigned	Slot assigned	Slot assigned
	22.800	22.800	22.849 (+49)
	Network control	Network control	Network control
	95%	95%	95,20% (+0,20%)

As it can be seen, Mallory's computational power was not eroded (the algorithm is not punitive in the classical sense of removing something) but it has grown more slowly than that of the other components of the network. This has caused, at the point of recalculation of the epoch 16, a change in the slot allocation with Mallory which has seen itself take away 0.21% of the computing power on the network. Comparing it to its previous slot number the erosion corresponds to the loss of 8.33% of the slots. The algorithm, in the presence of the fees (therefore of a charged network), becomes heavily punitive. This is because the greater the network load, the more the violations become critical.

Third example: Recovery of frozen stake

In this section the return mechanism for the stakes that have been

frozen starting from the simulated condition in Unbalanced stake with fee will be presented. In this way the re-entry into the optimal operating parameters of the network is rewarded.

Mallory reconfigures the pool to fit the optimal mining parameters

Starting from the condition mentioned above (second example), before the EEP of epoch 16 it adds 4 nodes. With the addition of 4 nodes its 550 slots are divided into 550/5 = 110 <120. As always, the change will take effect only from the next epoch.

With this change, it completely leaves the penalty area.

As starting conditions we take those highlighted in grey of the last column.

Let's now simulate the execution in case Mallory corrects his behaviour with the addition of 4 nodes.

In this case you will get a reward for its 550 slots:

coinbase = 550 * 1 = 550

collected_fee = 550 * 5 = 2750

From the previous epoch, Mallory inherited a frozen fee of 2400 tokens and a number of penalty slots equal to 960 (twice the slots for which it remained outside the parameters).

The formula to calculate the recovered fee for a slot is as follows:

recovered_feeE17S1 = frozen_feeE17S1 / penalty_slotsE17S1

At this point the values of the frozen_fee and the penalty_slots are updated:

frozen_feeE17S2 = frozen_feeE17S1 - recovered_feeE17S1

penalty_slotsE17S2 = penalty_slotsE17S1 - 1

For notational convenience, the absolute value epoch is indicated in the subscript, in this case 17, but in slot 2 is indicated the slot assigned to Mallory after one. The two slots may not be consecutive. In this case, after the "s", the sequence number of the slot assigned to Mallory is shown.

The following table shows the evolution of values as slots pass.

Assigned Slot	frozen_fee	penalty_slots	recovered_fee
E17S1	2400	960	2,5
E17S2	2397,5	959	2,5
E17S3	2395	958	2,5
...			
E17S109	2130	852	2,5
E17S110	2127,5	851	2,5

At the end of the epoch, Mallory will have recovered 275 tokens for the 2400 frozen. This allows him to partially bridge the gap with the other nodes of the network and serves as an incentive to favour the virtuous behaviour of having added nodes.

In frozen fees there are no coinbases which have not been generated. It is always and in any case more convenient for a node to operate in compliance with the rules than to use this recovery procedure.

Furthermore, according to a preferred embodiment, an access to a shared resource is calculated autonomously by all nodes e.g. OVERFLOW nodes of the network. In this regard, in the proof of stake type blockchains, it is not negligible to guarantee an appropriate redistribution of the construction tasks (called “mining”) of the blocks. While it is simple to centrally determine the order of assignment of the blocks to be mined between the nodes of the blockchain, a fully distributed determination of assignments is

somewhat complex.

The systems for determining the order of access to a time division transmission medium are particularly known as TDMA from the Anglo-Saxon acronym "Time division multiple access". These systems have been conceived in the field of telecommunications to avoid collisions between messages generated by multiple nodes, which lead to a rapid decrease in the network capacity, called "throughput".

The TDMA approach involves dividing time into macro intervals called epochs, in which each epoch is divided into so-called time slots. The number of time slots assigned to each node sharing a shared resource is predetermined according to a predetermined distribution logic.

With reference to Figure 11, each node N1, N2, N4, etc., which accesses the shared resource has its own unique identifier ID1, ID2, ID4, etc .. generally numeric.

The identifiers can be sorted according to a list, as shown in figure 11.

Each identifier is associated with a number of time slots to which the node has the right of access at a predetermined time. An epoch is an ordered grouping over time, of time slots.

Each node is also associated with a numerical interval of width proportional to the number of time slots associated with the same node.

Therefore, if N1 has the right to access two time slots, the relative numerical interval is for example 20, while if N2 has the right to access only one time slot, the relative numerical interval is 10.

The various number ranges are contiguous and do not overlap.

For example, for N1 the range goes from 0 to 19. For N2, the range goes from 19 to 29. For N4 the range goes from 29 to 59.

The other nodes N3 and N5, obviously, do not have the right to access the shared

resource at the time under consideration.

The overall numerical range is given by the sum SUM.

Since each node knows

- the ID of the other nodes,
- the number of time slots to which each node has the right to access,
- the seed,

then it can independently calculate the order of access to the shared resource according to the steps performed cyclically for a number of cycles equal to the number of time slots at the time:

- i-th calculation of the seed hash,
- Calculation of the remainder of the division of said Hash by the sum SUM of the amplitudes of said numerical intervals,
- Determination of the interval in which said remainder falls,
- Allocation of the i-th slot to the node corresponding to said interval.

For example, if the Hash of the seed leads to a numerical value of 600, then, since SUM is equal to 60, the division 610/60 brings as remainder 10. The value 10 falls in the first numerical interval R1 which belongs to N1 . Therefore the first time slot is assigned to N1.

The Hash of the previously calculated Hash is then calculated. This time, the remainder of the division leads to a value equal to 32 which falls in the interval R4 which belongs to N4. Therefore the second time slot belongs to N4.

Since R4 has a greater interval than the others, the probability of being assigned a time slot is proportional to the width of its interval and therefore, the correct redistribution of time slots is guaranteed, but with an assignment order that is maximally dependent on the seed.

The procedure continues until all time slots are assigned.

As regards the division, one must take into account that a Hash can have a numerical representation whatsoever, for example mZOhnZ2ZSUva Cq16HEy0 + + = 1I87DHFhVUKI28QcswrfQ and can be converted into hexadecimal notation becoming 9993a19d9d99494bdaf82ab5e87132d3ed48f3b0c7161554288dbc41cb30adf4 or in decimal notation becoming 391875681579805356411831496321815245512789534030969319853065217075543948136635096802576932411078811361721 3.

Therefore, the aforementioned division can be achieved by carrying out any conversions of divisor or dividend notation so that they are compatible.

This also applies to any unique identifier.

Figure 14 shows an explanatory flow chart of the method object of the present invention, which includes the following preliminary steps in succession:

- (i) sharing among all nodes of all the unique identifiers of at least said predetermined number of nodes (N1, N2, N4) having the right to access the shared resource,
- (ii) sharing of said respective predetermined number of time slots assigned to each node,
- (iii) sharing a seed;

the method including the following steps in succession

- (iv) ordering of said identifiers,
- (v) association to each node of a numerical interval (R1, R2, R4) of size proportional to said respective predetermined number of associated time slots,
- (vi) alignment of said numerical intervals so as to be contiguous and without overlapping according to an order defined by said order of identifiers,
- (vii) calculation of a sum (SUM) of said numerical intervals,
- (viii) cyclic execution of the following steps for a number of cycles equal to the number

of time slots of the time:

a. i-th calculation of the i-mo Hash of the seed,

b. Calculation of the remainder of a division of said i-th Hash by said sum (SUM),

c. Determination of the numerical interval (R1, R2, R4) in which said remainder falls,

d. Attribution of the i-th slot to the node corresponding to said numerical range in which said remainder falls.

Figure 13 shows a random allocation scheme of the time slots according to the present invention, now described assuming that the shared resource is a transmission medium.

It is assumed that there are a predetermined number of transmitting stations N1, N2, N4, indicated as "Nodes" and another number A, B, C of flow generators, indicated as "Holders". It must be understood that it is just a coincidence that the transmitting stations and the flow generators are equal in number.

Holders, for example, can be access servers to a portion of a telematic network, for example the internet.

Flow generator A is entitled to use three time slots, flow generator B is entitled to five time slots, while flow generator C is entitled to seven time slots.

Evidently, each flow generator can forward a request for access to the transmission medium to a transmitting station.

A wishes to request node N1 to transmit its data during the assigned time slots.

B distributes two assignments to N1, three in loads to N2 and two and N3.

The calculation of the time slots to be accessed can be performed by the flow generators or by the transmitting stations. Furthermore, the identifiers ID can belong to the flow generators or to the transmission nodes.

Therefore, the time slots of the time (Epoch) of Figure 13 are assigned by the present invention to N2, N1, N3, N2, N3, etc ..

Figure 13 can also be interpreted as a situation in which in a proof of stake blockchain, the flow generators, indicated in the figure as "Holders" are servers that receive / collect transaction requests, while the nodes N1 - N4 correspond to the miners.

In both cases, Holders can choose the node to which to entrust their transactions (or their data flow), in relation to the reliability of the node itself. For example, by keeping track of the efficiency in the execution of tasks assigned in previous times.

The present invention can be advantageously realized by means of a computer program which comprises coding means for carrying out one or more steps of the method, when this program is executed on a computer. Therefore it is intended that the scope of protection extends to said computer program and further to computer readable means comprising a recorded message, said computer readable means comprising program coding means for carrying out one or more steps of the method. , when said program is run on a computer.

P3183PC00

CLAIMS

1. A computer-implemented method for reaching a proof of stake based distributed consensus for a blockchain network, thus allowing to achieve higher level of distribution of the network and to not have too much stake concentrated in too few nodes for increasing network robustness and reducing vulnerability at a compromised node, said blockchain network comprising one or more MAIN addresses, for each MAIN addresses being associated in turn one or more OVERFLOW addresses/nodes, the last being able to act as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node, characterized by comprising the following processes:

-a redistribution process being configured to calculate the minimum stake necessary for an OVERFLOW node to mine, being minimum_stake_value=total_amount_at stake/predetermined_target_number_of_nodes, to calculate the maximum stake allowing efficient mining, being maximum stake value=minimum stake value*2, for each MAIN address to distribute the stake among the OVERFLOW nodes, to scroll through the OVERFLOW nodes of a given MAIN address and to assign to every OVERFLOW node the minimum_stake_value, until the remaining stake is insufficient to activate a miner or every overflow node has been assigned its stake;

-a penalty process being configured to freeze an amount of reward corresponding to the computational work for having mined a number of slots over a predetermined limit Slot_MAX in an $epoch_{i+1}$, and to recover it in the following epochs starting from $epoch_{i+2}$ progressively according to a number of parts proportional to the number of slots mined over said predetermined limit Slot_MAX, being one part assigned to each following slot the miner is delegable to mine, thus in order to naturally incentive the distribution of the blockchain network.

P3183PC00

2. The computer-implemented method according to claim 1 wherein the redistribution process may further be configured in such a way that if there is any stake left it is distributed between the OVERFLOW nodes until as many nodes as possible have a stake equal to maximum stake value and any further stake is assigned to the first OVERFLOW node according a predetermined order.

3. The computer-implemented method according to one of the preceding claims wherein the redistribution process comprises in particular the following steps:

52: having a blockchain network comprising one or more MAIN addresses, for each MAIN addresses are associated in turn one or more OVERFLOW addresses/nodes acting as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node;

55: getting as input TOTAL_AT_STAKE value for each MAIN address of the blockchain network corresponding to the total amount of stake on each MAIN address

60: calculating S_min_VAL= TOTAL AT STAKE/TARGET NUMBER OF_NODES being TARGET NUMBER OF_NODES = total number of OVERFLOW mining nodes in the chain network,

Calculating S_MAX_VAL=S_min_VAL * 2

Assuming a counter value ITERATION=0

65: if ITERATION value is <2 then updating the counter value ITERATION=ITERATION+1

70: for each MAIN address getting as input the MAIN_ADDRESS.assigned_stake value and if this value is >0, thus meaning that the MAIN address under consideration contains an amount of stake, assigning this value to parameter

P3183PC00

M_A_S,

and if M_A_S value is >= S_MAX_VAL repeating step 70 to the next MAIN address,

75: if M_A_S value is < S_MAX_VAL getting as input OVERFLOW_ADDRESS.assigned_stake value for each OVERFLOW address associated to the MAIN under consideration, and

if OVERFLOW_ADDRESS.assigned_stake value is =0 repeating step 75 to the next OVERFLOW address associated to the MAIN under consideration, and

if OVERFLOW_ADDRESS.assigned_stake value is ≠0, assigning OVERFLOW_ADDRESS.assigned_stake value to O_A_S parameter, calculating MAIN_ADDRESS.assigned_stake = M_A_S - S_min_VAL, and calculating OVERFLOW_ADDRESS.assigner_stake = O_A_S + S_min_VAL, and repeating step 75 to the next OVERFLOW address associated to the MAIN under consideration.

4. The computer-implemented method according to one of the preceding claims 1, 2 wherein the penalty process comprises in particular the following steps:

80: calculating total frozen_fee$_{i+1}$ as the amount of reward$_{i+2}$ not dispensed to the miners in a specific epoch$_{i+1}$ for having mined a number of slots over the limit Slot_MAX

85: calculating penalty_slots$_{i+1}$ as preferably twice (or other proportional law) the number of slots mined over the limit i.e. number of slots over Slot_MAX

90: calculating recovered_fee$_{j,\ i+1}$ value as frozen_fee$_{i+1}$/penalty_slots$_{i+1}$ i.e. the fee to be recovered to each slot

95: adding the recovered_fee$_{j,\ i+1}$ value to the reward of the MAIN_address for mining the block$_i$ in the epoch$_{i+1}$ as reward$_{j,\ i+1}$=one coinbase+collected_fee$_{j,\ i+1}$ +

P3183PC00

recovered_fee$_{j,\ i+1}$

100: calculating the new value of frozen_fee$_{i+1}$ parameter as frozen_fee$_{i+1}$ - recovered_fee$_{j,\ i+1}$ and the new value of penalty_slots$_{i+1}$ as penalty_slots$_{i+1}$ - 1 and repeat step 95 for the following block$_{j+1}$ in the epoch$_{i+1}$ while frozen_fee $_{i+1}$ >0 or while penalty_slots$_{i+2}$>0, being that when the last block$_j$ is mined at the end of each epoch$_i$, penalty_slots$_i$ and frozen_fee$_i$ are transferred from the first OVERFLOW node to its MAIN address and when the first block$_j$ is mined at the beginning of every epoch$_i$ the penalty_slots$_i$ and frozen_fee$_i$ are transferred from the MAIN address to its first OVERFLOW node.

5. The computer-implemented method according to one of the preceding claims further comprising an initializing process being configured to list all MAIN addresses of the blockchain network, for each MAIN address list its assigned OVERFLOW addresses/nodes, and then for each OVERFLOW addresses/node transfer the stake if it has any, to its assigned MAIN address.

6. The computer-implemented method according to claim 5 wherein the initializing process comprises in particular the following steps:

20: having a blockchain network comprising one or more MAIN addresses, for each MAIN addresses are associated in turn one or more OVERFLOW addresses/nodes acting as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node;

25: via the blockchain network, receiving a transaction including as input addresses from address register comprising MAIN addresses and their corresponding OVERFLOW addresses;

30: associating to each MAIN address a univocal progressive number;

35: for each MAIN address verifying if it contains an amount of stake and if this

P3183PC00

value is >0 assigning this value to parameter M_A_S, and if this value is <=0 implementing step 35 to the next MAIN address member of the blockchain network and generating a transaction including output providing the M_A_S value associated to the corresponding MAIN address,

40: associating to each OVERFLOW node a univocal progressive number

45: for each OVERFLOW node assigned to each corresponding MAIN address verifying if it contains an amount of stake defined OVERFLOW_ADDRESS.assigned_stake and if this value is >0 assigning this value to parameter O_A_S, calculating value MAIN_ADDRESS.assigned_stake as M_A_S+O_A_S and assigning O_A_S=0, and generating a transaction including output providing the updated value of the M_A_S value associated to the corresponding MAIN address, then implementing step 45 to the next OVERFLOW address assigned to the corresponding MAIN address considered under step 35,

and if OVERFLOW_ADDRESS.assigned_stake is <= 0 implementing step 45 to the next OVERFLOW address assigned to the corresponding MAIN address considered under step 35

50: generating a transaction including output of the updated value of TOTAL_AT_STAKE value for each MAIN address of the blockchain network corresponding to the total amount of stake on each MAIN address after performing preceding steps.

7. The computer-implemented method according to one of the preceding claims further comprising a registration process being configured to register one or more MAIN address, to register one or more OVERFLOW address/node and to associate said one or more OVERFLOW address/node to said one or more MAIN address and to register said

P3183PC00

association.

8. The computer-implemented method according to one of the preceding claims further comprising a maintenance process, in order to keep the blockchain network highly distributed even in case of failure of one or more mining nodes, said maintenance process comprising in turn the following steps, being given a MAIN address and at least two or more OVERFLOW addresses and a node desired to be maintained:

110: sending a transaction including a first input providing a substitute OVERFLOW address using its private key, thus meaning said substitute OVERFLOW address being a node substituting the node desired to be maintained

120: assigning said substitute OVERFLOW address to the corresponding MAIN address using private key of the MAIN address

130 deregistering the node desired to be maintained using its private key.

9. Method according to any of the preceding claims and distributed for determining an access order to a shared resource by a predetermined number of nodes OVERFLOW (N1, N2, N4) having a right to access to the shared resource in a predetermined time (epoch) for a respective predetermined number of time slots, in which an epoch is formed by a plurality of time slots, and wherein each node (N1, N2, N4, etc.), has its own unique identifier (ID1, ID2, ID4), the method comprising the following preliminary steps in succession:

- (i) sharing all the univocal identifiers of at least said predetermined number of nodes (N1, N2, N4, etc.) among all the nodes having the right of access to the shared resource,
- (ii) sharing of said respective predetermined number of time slots assigned to each node,
- (iii) sharing a seed,

the method comprising the following steps in succession

- (iv) ordering of said identifiers,

P3183PC00

- (v) association to each node of a numerical interval (R1, R2, R4) of amplitude proportional to said respective predetermined number of associated time slots,

- (vi) alignment of said numerical intervals so as to be contiguous and without overlapping according to an order defined by said identification order,

- (vii) calculation of a sum (SUM) of said numerical intervals,

- (viii) cyclic execution of the following steps for a number of cycles i equal to the time slot number of the epoch:

a. i-th calculation of the seed Hash,

b. Calculation of the rest of a division of said i-th Hash for said sum (SUM),

c. Determination of the numerical range (R1, R2, R4) in which the rest falls,

d. Assignment of the i-th slot to the node corresponding to said numeric interval in which the rest falls.

10. Method according to claim 9, wherein said shared resource is a transmission medium.

11. Method according to claim 10, wherein said shared resource consists of the task of generating a block to be shared by a blockchain.

12. Method according to claim 11, wherein said seed is given by the concatenation of two or more Hash of blocks previously constructed and suitably selected.

13. Method according to claim 12, wherein said blocks are selected according to the following steps:

- distribution of the blocks generated in a previous epoch into three or more groups

- extraction of the first block of each group or the first, second and third hash of the second group or the last, second last and third last block of the first group of blocks;

- concatenation of the Hashs related to the blocks extracted in the previous step.

14. A node configured to implement a method for reaching a proof of stake based distributed consensus for a blockchain network, thus allowing to achieve higher level of

distribution of the network and to not have too much stake concentrated in too few nodes for increasing network robustness and reducing vulnerability at a compromised node, said node comprising an interface device, a processor coupled to said interface device, a memory coupled to the processor, the memory having stored thereon computer executable instructions which, when executed, configure the processor to perform the corresponding steps of the computer implemented method of any of claims 1 to 13.

15. Computer program comprising program coding means for realizing all steps or processes of any of claims 1 to 13, when said program is run on a computer.

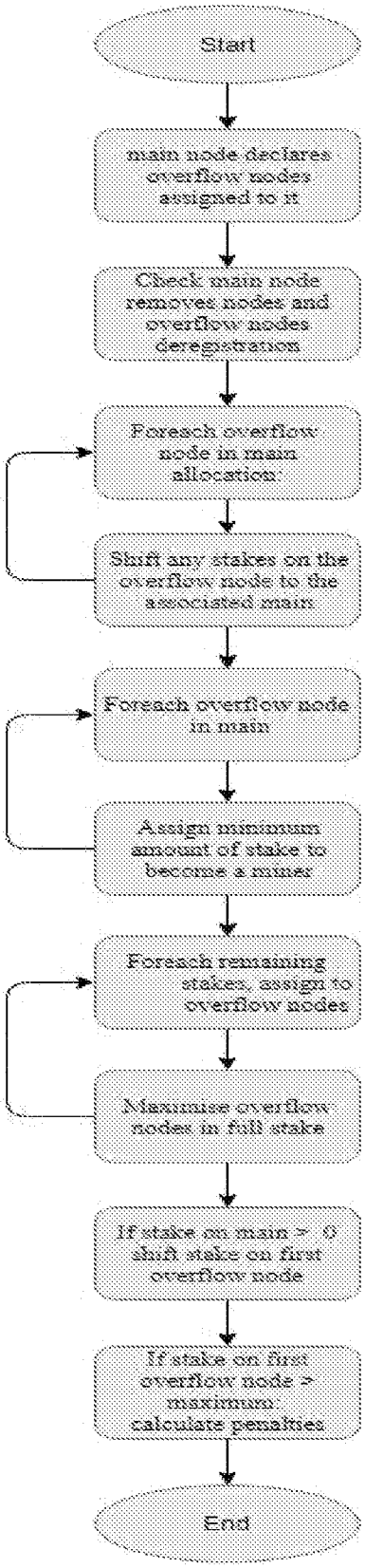
Start
main node declares overflow nodes assigned to it
Check main node removes nodes and overflow nodes deregistration
Foreach overflow node in main allocation:
Shift any stakes on the overflow node to the associated main
Foreach overflow node in main
Assign minimum amount of stake to become a miner
Foreach remaining stakes, assign to overflow nodes
Maximise overflow nodes in full stake
If stake on main > 0 shift stake on first overflow node
If stake on first overflow node > maximum: calculate penalties
End

Fig. 1

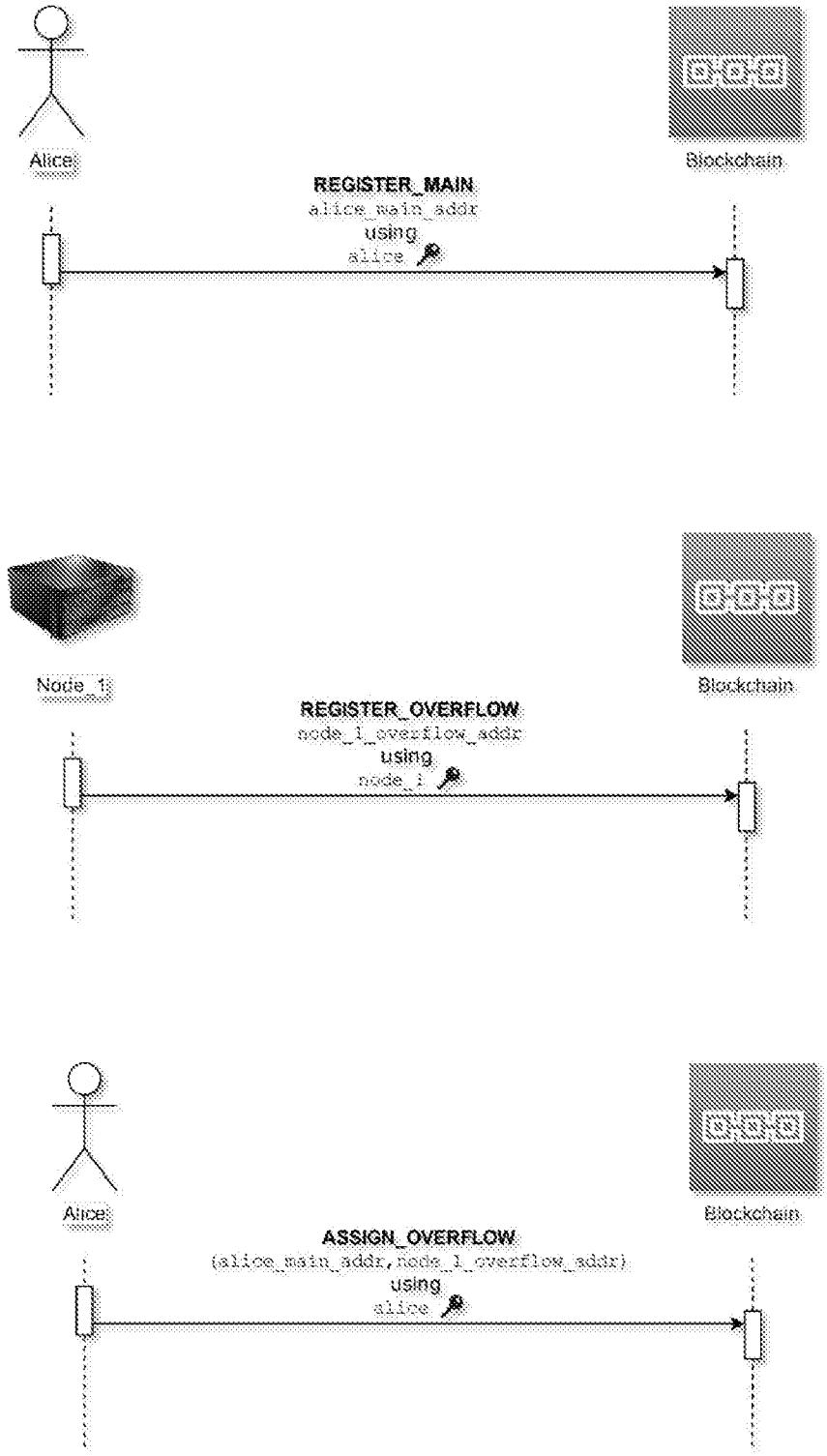
Alice
Blockchain
REGISTER_MAIN
alice_main_addr
using
alice
Node_1
Blockchain
REGISTER_OVERFLOW
node_1_overflow_addr
using
node_1
Alice
Blockchain
ASSIGN_OVERFLOW
(alice_main_addr,node_1_overflow_addr)
using
alice

Fig. 2

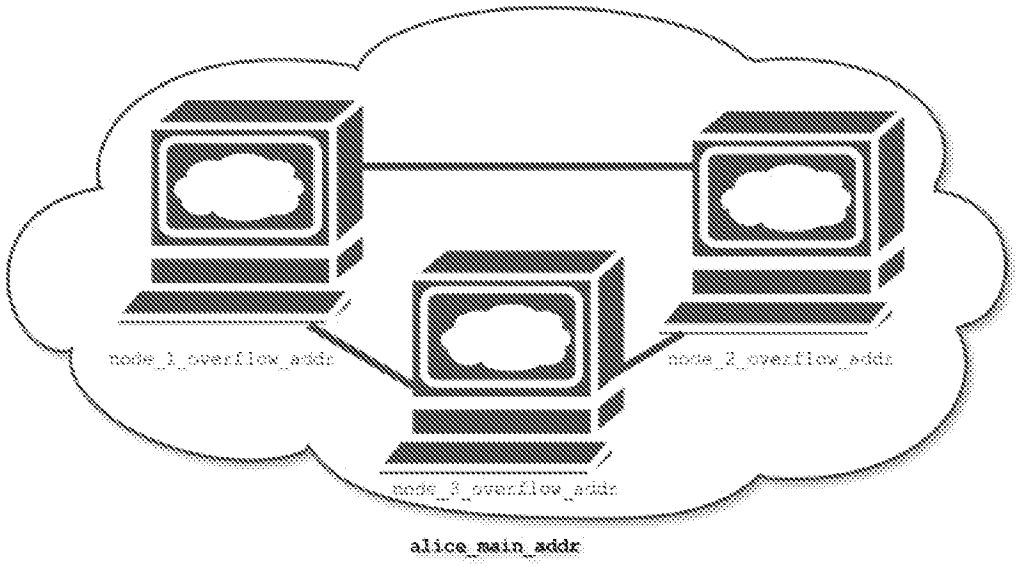
node_1_overflow_addr
node_2_overflow_addr
node_3_overflow_addr
alice_main_addr

Fig. 3

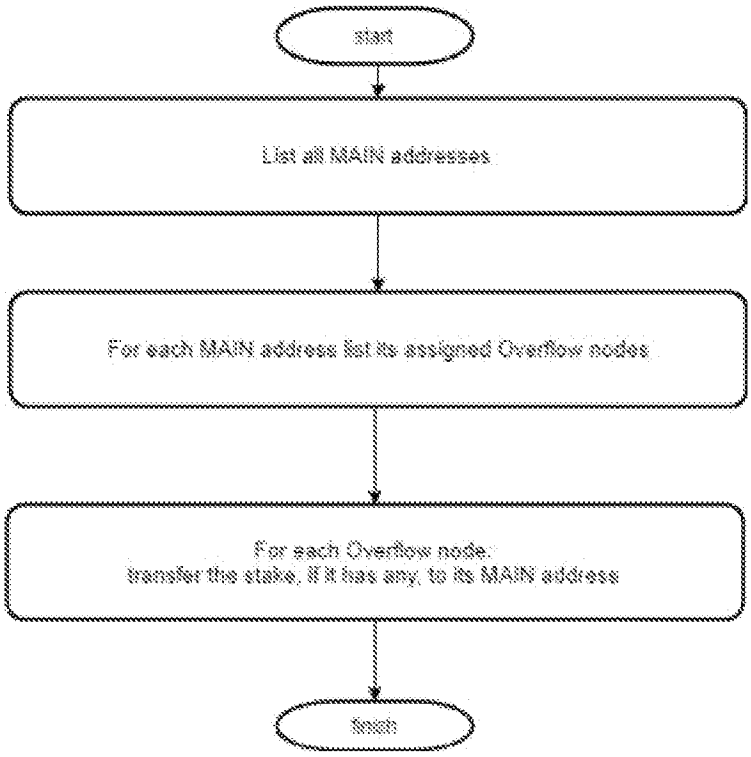
start
List all MAIN addresses
For each MAIN address list its assigned Overflow nodes
For each Overflow node:
transfer the stake, if it has any, to its MAIN address
finish

Fig. 4

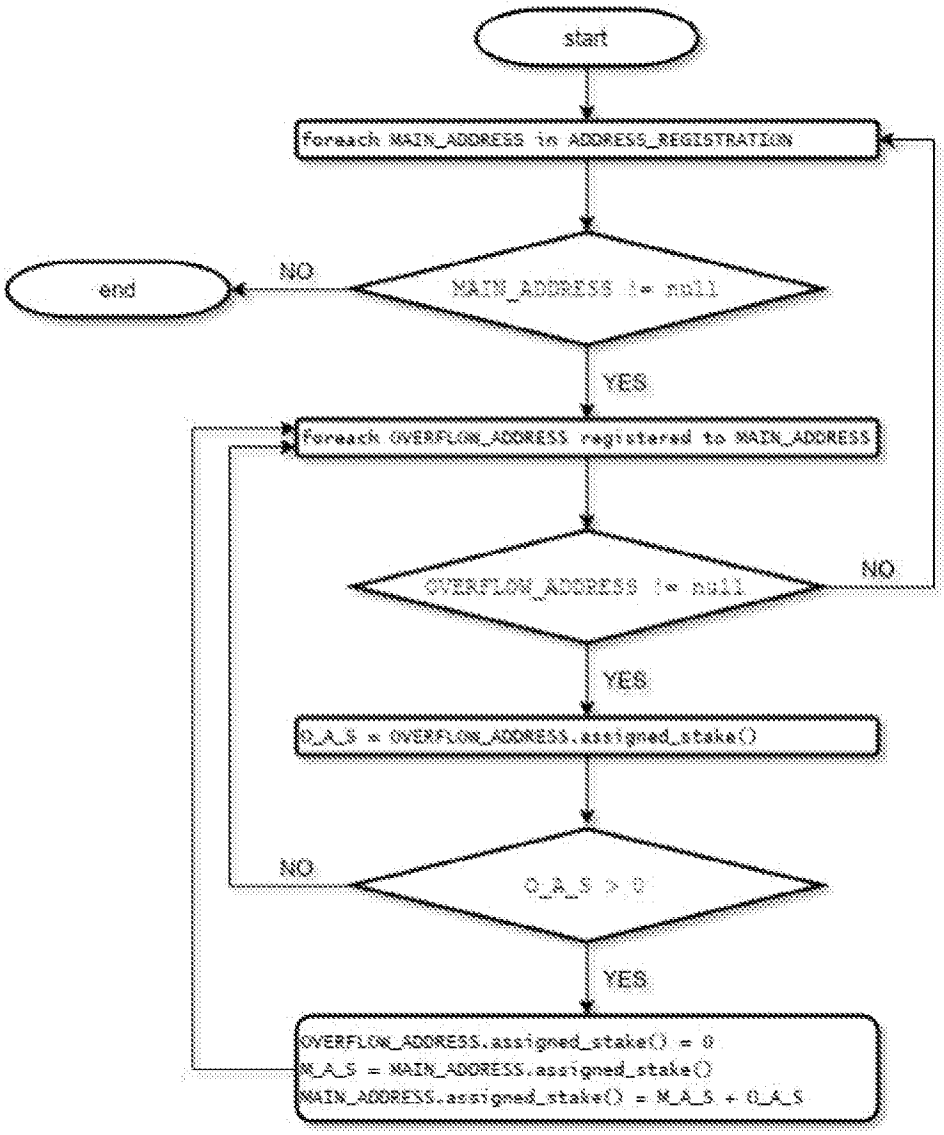

Fig. 5

	INITIAL STATE							
	not miner	full pay						
MAIN	S_min_VAL	S_min_VAL	S_min_VAL	S_min_VAL	S_min_VAL	S_min_VAL	S_min_VAL	RND
OVERFLOW	0							
OVERFLOW	0							
OVERFLOW	0							

Fig. 6

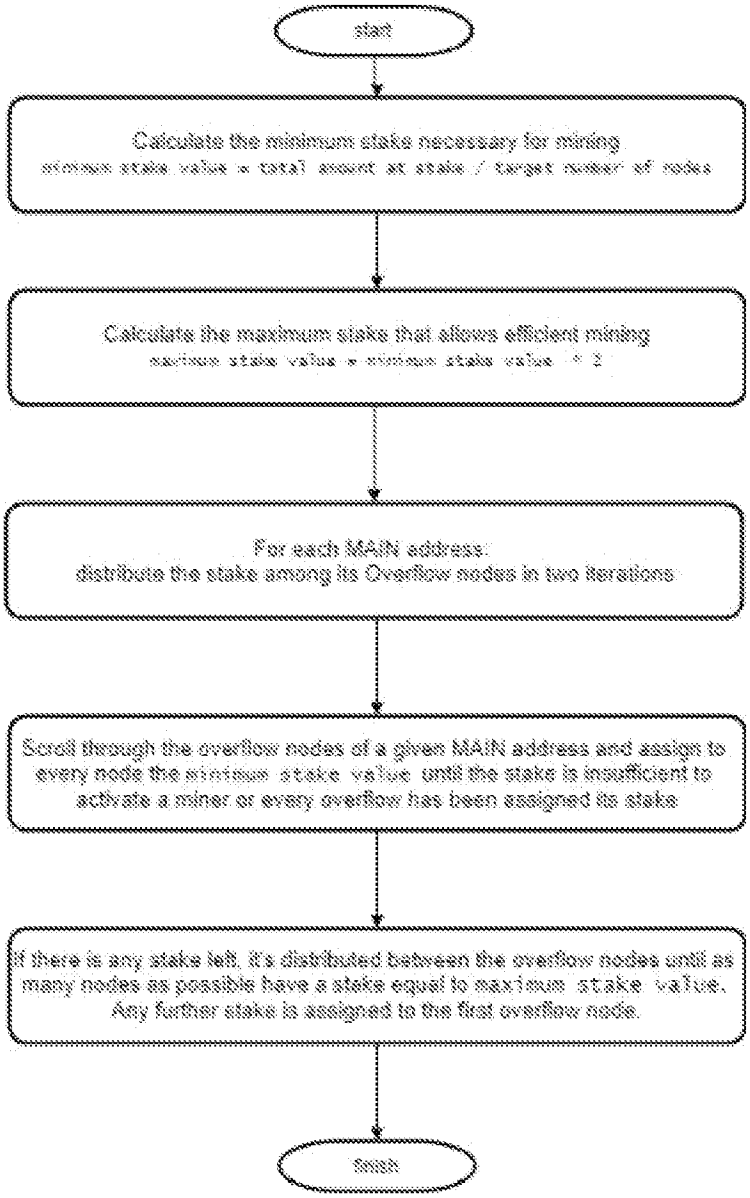
start
Calculate the minimum stake necessary for mining
minimum stake value = total amount at stake / target number of nodes
Calculate the maximum stake that allows efficient mining
maximum stake value = minimum stake value * 2
For each MAIN address:
distribute the stake among its Overflow nodes in two iterations
Scroll through the overflow nodes of a given MAIN address and assign to every node the minimum stake value until the stake is insufficient to activate a miner or every overflow has been assigned its stake
If there is any stake left, it's distributed between the overflow nodes until as many nodes as possible have a stake equal to maximum stake value. Any further stake is assigned to the first overflow node.
finish

Fig. 7

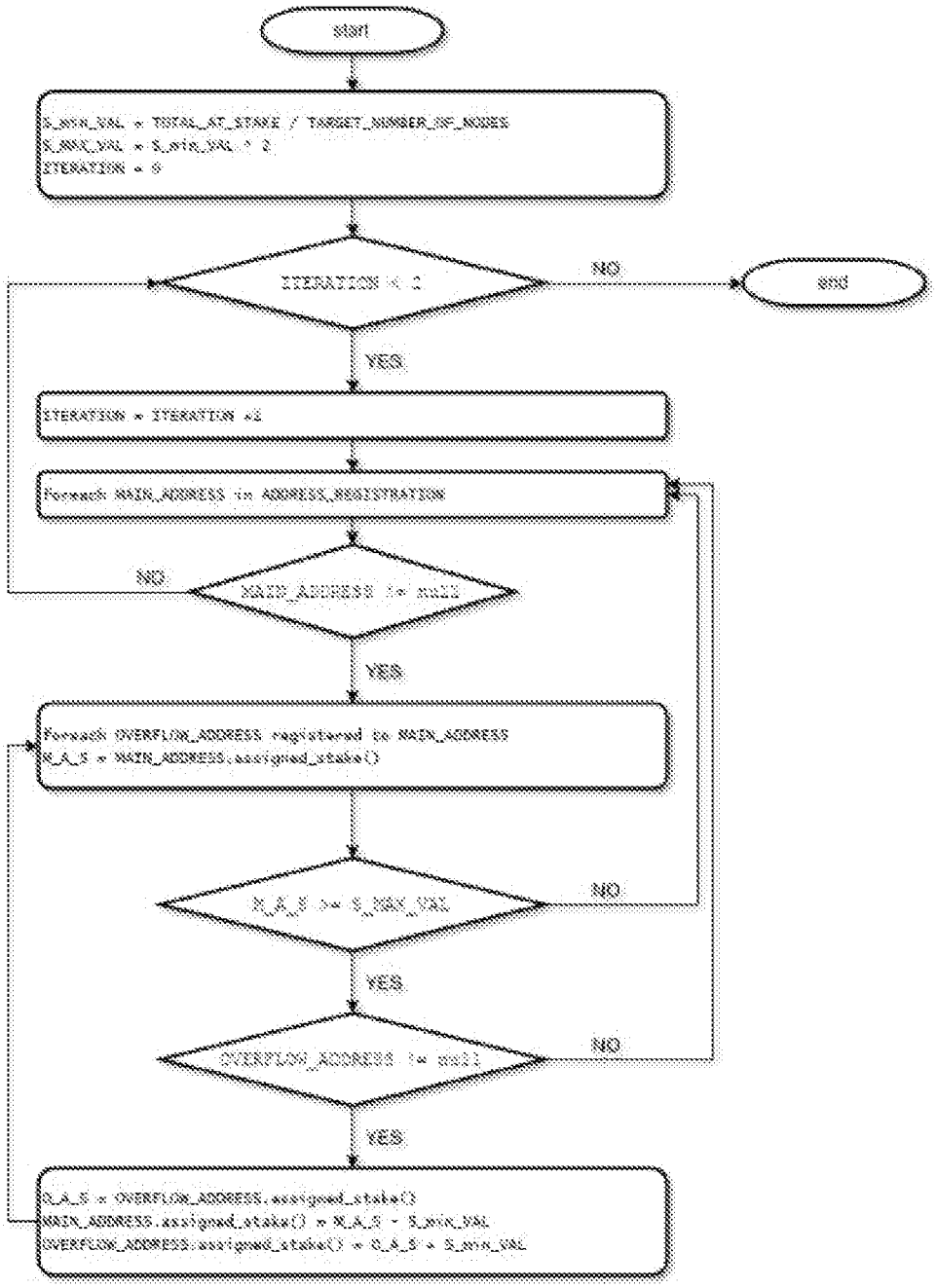
start
S_min_VAL = TOTAL_AT_STAKE / TARGET_NUMBER_OF_NODES
S_max_VAL = S_min_VAL * 2
ITERATION = 0
ITERATION < 2
NO
end
YES
ITERATION = ITERATION +1
Foreach MAIN_ADDRESS in ADDRESS_REGISTRATION
NO
MAIN_ADDRESS != null
YES
Foreach OVERFLOW_ADDRESS registered to MAIN_ADDRESS
O_A_S = MAIN_ADDRESS.assigned_stake()
M_A_S >= S_max_VAL
NO
YES
OVERFLOW_ADDRESS != null
NO
YES
O_A_S = OVERFLOW_ADDRESS.assigned_stake()
MAIN_ADDRESS.assigned_stake() = M_A_S - S_min_VAL
OVERFLOW_ADDRESS.assigned_stake() = O_A_S + S_min_VAL

Fig. 8

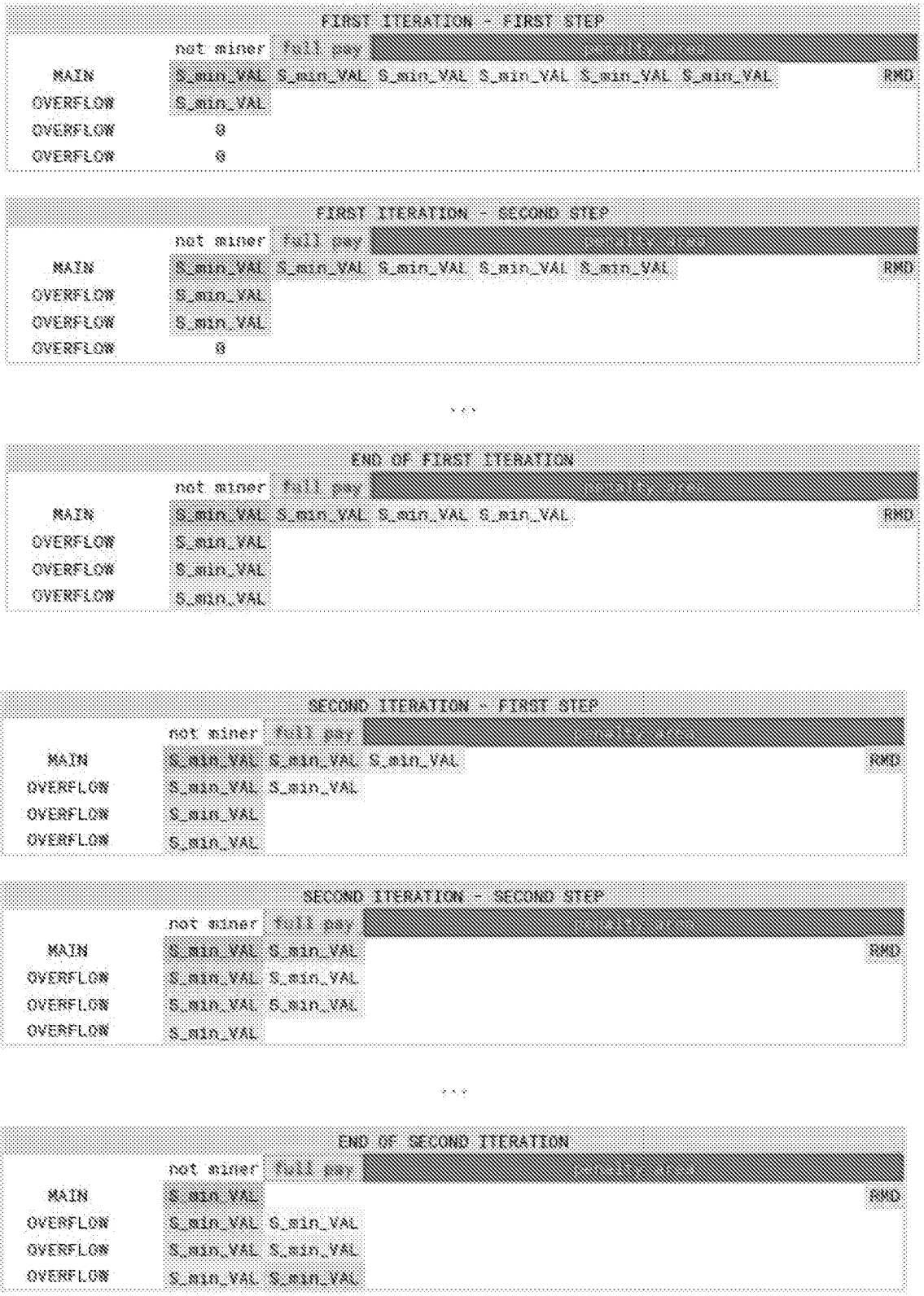
FIRST ITERATION - FIRST STEP
not miner full gas
MAIN S_min_VAL S_min_VAL S_min_VAL S_min_VAL S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL
OVERFLOW 0
OVERFLOW 0
FIRST ITERATION - SECOND STEP
not miner full gas
MAIN S_min_VAL S_min_VAL S_min_VAL S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL
OVERFLOW S_min_VAL
OVERFLOW 0
...
END OF FIRST ITERATION
not miner full gas
MAIN S_min_VAL S_min_VAL S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL
OVERFLOW S_min_VAL
OVERFLOW S_min_VAL
SECOND ITERATION - FIRST STEP
not miner full gas
MAIN S_min_VAL S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL
OVERFLOW S_min_VAL
SECOND ITERATION - SECOND STEP
not miner full gas
MAIN S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL
...
END OF SECOND ITERATION
not miner full gas
MAIN S_min_VAL RND
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL S_min_VAL

Fig. 9

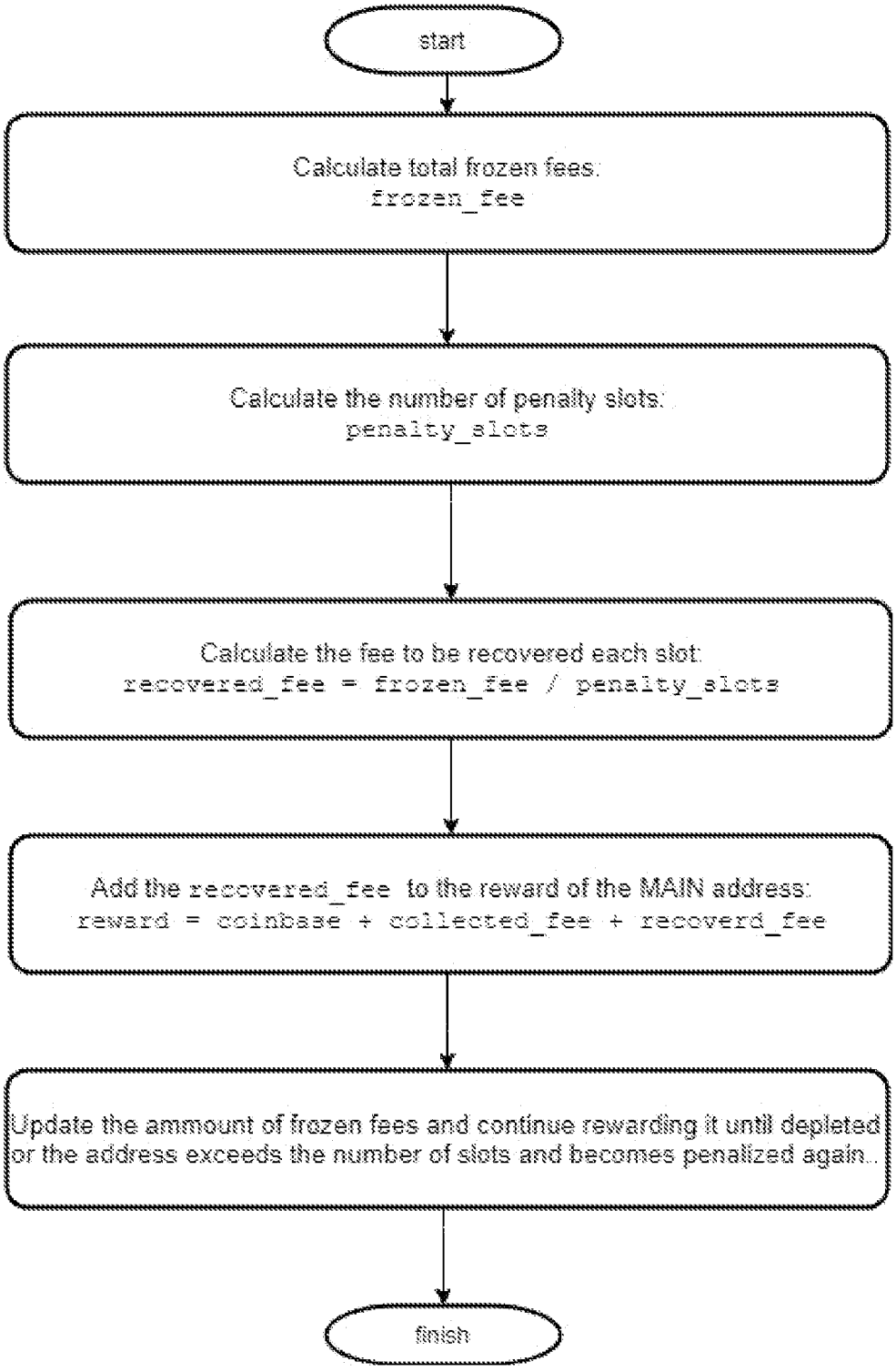
start
Calculate total frozen fees:
frozen_fee
Calculate the number of penalty slots:
penalty_slots
Calculate the fee to be recovered each slot:
recovered_fee = frozen_fee / penalty_slots
Add the recovered_fee to the reward of the MAIN address:
reward = coinbase + collected_fee + recoverd_fee
Update the ammount of frozen fees and continue rewarding it until depleted or the address exceeds the number of slots and becomes penalized again..
finish

Fig. 10

N1	ID1	2
N2	ID2	1
N3	ID3	0
N4	ID4	3
N5	ID5	0

Fig. 11

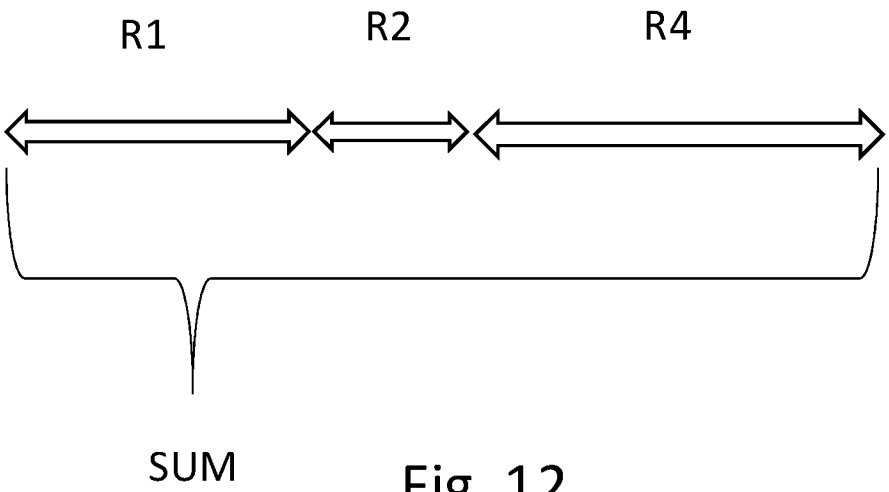

Fig. 12

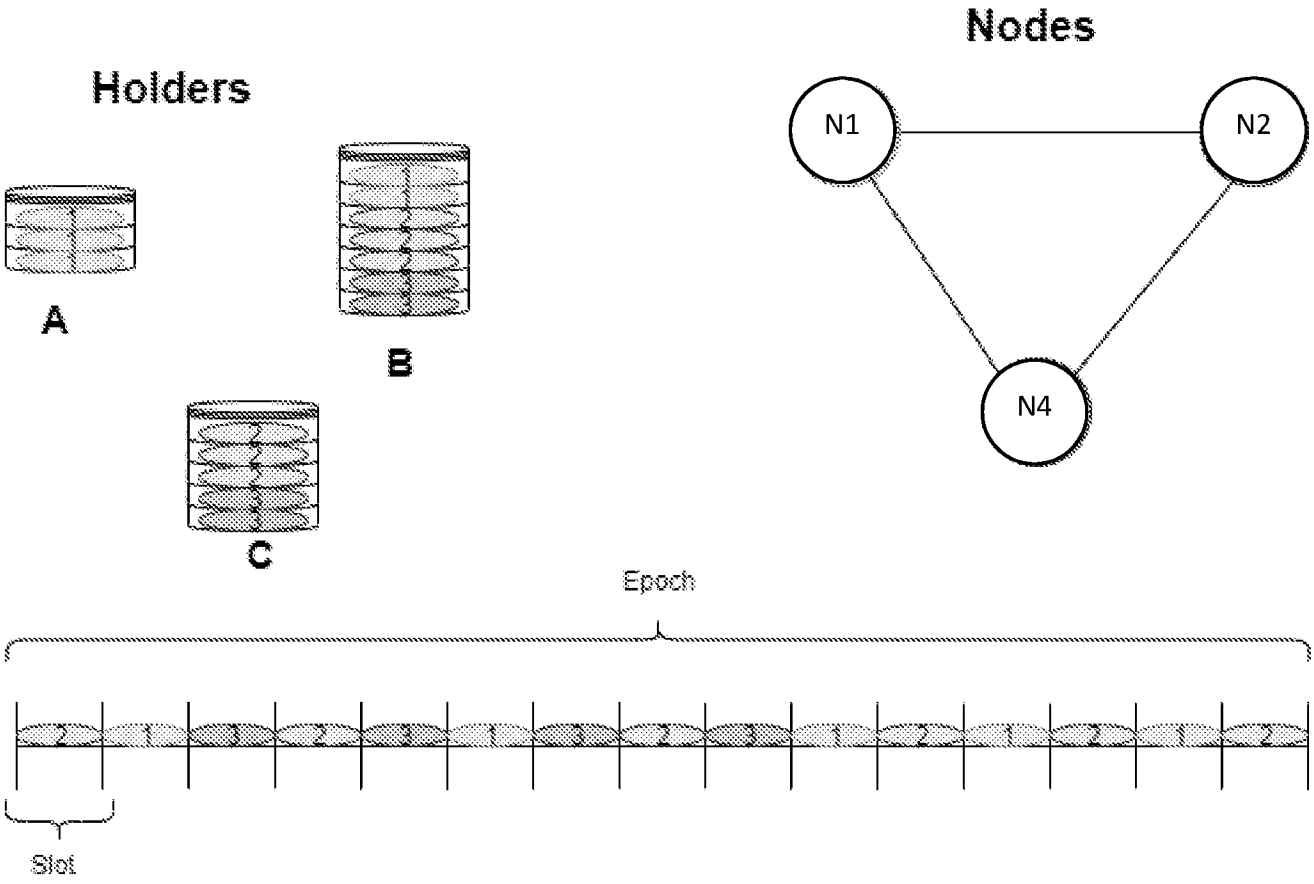

Fig. 13

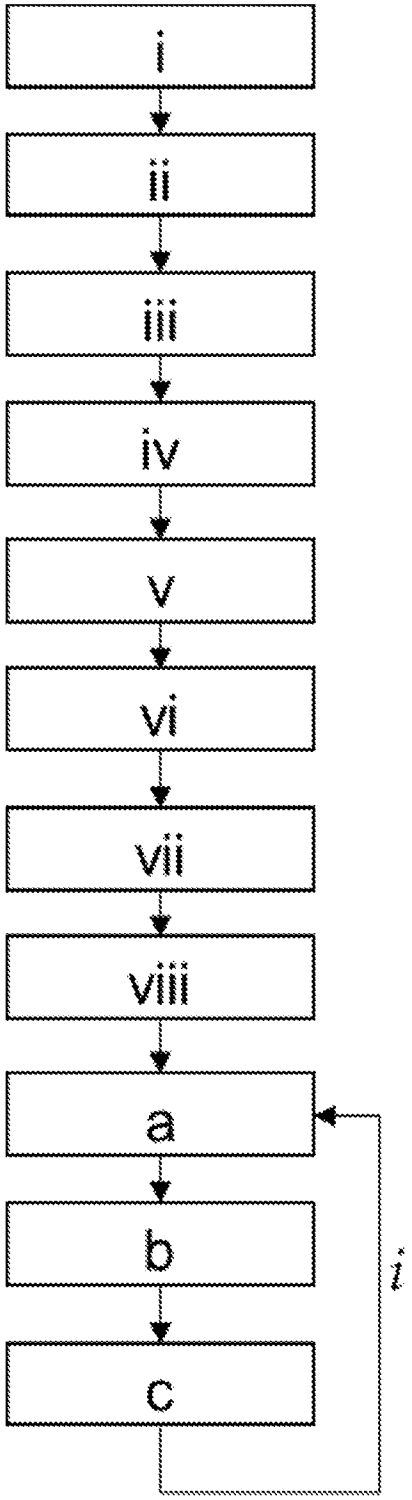

Fig. 14

"COMPUTER-IMPLEMENTED METHOD FOR REACHING A DISTRIBUTED CONSENSUS IN A BLOCKCHAIN NETWORK AND NODE IMPLEMENTING THE METHOD"

DESCRIPTION

FIELD OF INVETION

The present invention concerns a computer-implemented method for reaching a proof of stake based distributed consensus for a distributed ledgers, such as the blockchain technology.

STATE OF THE ART

The invention is particularly suited, but not limited to, achieving higher level of distribution of a proof of stake based network in order to not have too much stake concentrated in too few nodes, thus increasing the robustness of the network and lowering its vulnerability to any one node being compromised.

In this document we use the term blockchain to include all forms of electronic, computer-based, distributed ledgers. These include, but are not limited to blockchain and transaction-chain technologies, permissioned and un-permissioned ledgers, consensus-based ledgers, shared ledgers and variations thereof. The most widely known application of blockchain technology is the Bitcoin ledger, although other blockchain implementations have been proposed and developed. A blockchain is a consensus-based, electronic ledger which is implemented as a computer-based decentralised, distributed system made up of blocks which in turn are made up of transactions and other information. Bitcoin is an example of a proof-of-work blockchain in which miners perform expensive computations in order to facilitate transactions on the blockchain.

Proof-of-work based blockchains have been criticized due to the large computing resources required, which requires a large amount of power consumption to operate and also due to the fact that block-generation may be irregular and slow. Moreover, several blocks must be built on top of a given block before the given block is considered to be confirmed (i.e., sufficiently unlikely to be reverted).

Proof-of-stake based blockchains have been proposed as an alternative to proof-of-work blockchains. In a proof-of-stake blockchain network, the blockchain is secured by proof-of-stake rather than proof-of-work. Under proof-of-stake, miners hold a stake (deposit some tokens) in a special account. This stake may be referred to as a security deposit and the probability of being selected as the node to mine a block is proportional to the quantity of the digital assets provided as a security deposit. Proof-of-stake blockchain networks can be used to avoid the computational expense and energy required to mine on proof-of-work blockchains. Further, proof-of-stake blockchains can allow for higher frequency and more regular block creation than proof-of-work blockchains. At least some proof-of-stake blockchains also have a low probability of forking and a block may be effectively confirmed as soon as it is added to the blockchain.

As mentioned before, although the economy of scale leads to the concentration of mining resources to increase network efficiency, in developing a distributed network, concentration is a problem concerning network control, network robustness because it is impossible to carry out planned maintenance operations on the nodes. The main issue in small, concentrated networks, has to do with low redundancy in case of failures, whether it's network, hardware or software problems. The opposite situation, in which there is an excessive fragmentation, can also be counterproductive. This is because information transmission and replication always present an overhead within a real system. Furthermore assigning stakes directly to mining nodes may lead to problem of

overloading if distribution constraints are present.

Thus, there is a need to tackle the abovementioned concentration challenges and improve the proof of stake consensus protocol to obtain a protocol encouraging distribution and at the same time efficiency of the network. ¶

C. Badertscher et al: "Ouroboros Genesis Composable Proof-of-Stake Blockchains with Dynamic Availability"; A. Kiayas et al: "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol"; and T. Deepak et al: "CloudPoS: A Proof-of-Stake Consensus Design for Blockchain Integrated Cloud" disclose various examples of incentive management in proof-of-stake based consensus mechanisms for blockchain networks.

OBJECTIVES OF THE INVENTION

SCOPI DELL'INVENZIONE

One objective of the present invention, according to a first of its aspects, is obtaining a computer-implemented method able to encourage the distribution of the network and efficiency of the miners, and at the same time optimizing blockchain scalability.

A second objective of the present invention is obtaining a computer-implemented method able to improve robustness of the network.

A further objective is to optimize the computer-implemented method in order to increase the global security of the blockchain network.

A further objective of the present invention is to provide a the computer-implemented method that encourages the use of honest performing nodes to mine the subsequent block and discourage the use of nodes that do not perform well or that have intentionally bad or malicious behaviour.

An objective of the present invention is also to keep low the energy consumption of the network and possibly increasing node computational power only when needed.

A further objective of the present invention is to provide a system implementing the computer-implemented method easily to be designed, with an optimized energy and computational power consumption and extremely efficient.

BRIEF DESCRIPTION OF THE INVENTION

BREVE DESCRIZIONE DELL'INVENZIONE

Hereinafter are summarized some technical aspects of the present inventions which enable some of the most important purposes to be achieved.

According to a first aspect this invention relates to a computer-implemented method for

reaching a proof of stake based distributed consensus for a blockchain network, thus allowing to achieve higher level of distribution of the network and to not have too much stake concentrated in too few nodes, said blockchain network comprising one or more MAIN addresses, for each MAIN addresses being associated in turn one or more OVERFLOW addresses/nodes, the last being able to act as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node, comprising the following processes:

-a redistribution process being configured to calculate the minimum stake necessary for an OVERFLOW node to mine, being minimum_stake_value=total_amount_at stake/predetermined_target_number_of_nodes, to calculate the maximum stake allowing efficient mining, being maximum_stake_value=minimum_stake_value*2, for each MAIN address to distribute the stake among the OVERFLOW nodes, to scroll through the OVERFLOW nodes of a given MAIN address and to assign to every OVERFLOW node the minimum_stake_value, until the remaining stake is insufficient to activate a miner or every overflow node has been assigned its stake;

-a penalty process being configured to freeze an amount of reward corresponding to the computational work for having mined a number of slots over a predetermined limit Slot_MAX in an $epoch_{i+1}$, and to recover it in the following epochs starting from $epoch_{i+1}$, a little at a time, according to a number of parts proportional to the number of slots mined over said predetermined limit Slot_MAX, being one part assigned to each following slot the miner is delegable to mine, thus in order to naturally incentive the distribution of the blockchain network.

Such a computer implemented method naturally enables high distribution of the blockchain network.

According to a second aspect this invention relates to a computer-implemented method

wherein the redistribution process may further be configured in such a way that if there is any stake left it is distributed between the OVERFLOW nodes until as many nodes as possible have a stake equal to maximum stake value and any further stake is assigned to the first OVERFLOW node according a predetermined order

It enables to optimize the redistribution of the total stake assigned to a pool of OVERFLOW nodes linked to a MAIN address and makes that only first OVERFLOW nodes according an univocal progressive predetermined numeration may be under penalty regime.

According to a third aspect this invention relates to a computer-implemented method further comprising an initializing process being configured to list all MAIN addresses of the blockchain network, for each MAIN address list its assigned OVERFLOW addresses/nodes, and then for each OVERFLOW addresses/node transfer the stake if it has any, to its assigned MAIN address.

It enables to optimize the redistribution process in case there may be any stake on a OVERFLOW node before the redistribution process be implemented.

According to a fourth aspect this invention relates to a computer-implemented method further comprising a maintenance process, comprising the following steps, being given a MAIN address and at least two or more OVERFLOW addresses and a node desired to be maintained:

110: sending a transaction including a first input providing a substitute OVERFLOW address using its private key, thus meaning said substitute OVERFLOW address being a node substituting the node desired to be maintained

120: assigning said substitute OVERFLOW address to the corresponding MAIN address using private key of the MAIN address

130 deregistering the desired to be maintained using its private key. Alternatively the

MAIN address can UNASSIGN an overflow node previously assigned to it.

It enables to keep the blockchain network highly distributed even in case of failure of one or more mining nodes.

According to a further aspects this invention relates to:

- a node configured to implement a method for reaching a proof of stake based distributed consensus for a blockchain network, thus allowing to achieve higher level of distribution of the network and to not have too much stake concentrated in too few nodes, said node comprising an interface device, a processor coupled to said interface device, a memory coupled to the processor, the memory having stored thereon computer executable instructions which, when executed, configure the processor to perform the corresponding steps of the computer implemented method disclosed.

-a computer readable storage medium comprising computer-executable instructions which, when executed, configure a processor to perform the disclosed computer implemented method.

-a computer network for performing a blockchain-based transaction from a first user to a second user in an electronic transfer network of connected nodes, wherein the nodes includes at least a first node, a second node, disclosed, wherein each node share a same blockchain, and wherein each node stores a computer readable storage medium disclosed.

According to an aspect of the present invention, a completely decentralized scheme is proposed for determining the order of access to a shared resource. The basic idea is to order the nodes, e.g. OVERFLOW, enabled to access the shared resource according to their own identifier. Each enabled node is assigned a numerical interval whose size is proportional to the number of time slots in which the same node has the right to have access to the shared resource.

A node is enabled for access when it has the right to access the shared resource for at

least one time slot.

Consequently, the numerical intervals are consecutive and also ordered according to the same order of the unique identifiers of the nodes.

This alignment of numerical intervals is shared and known by all nodes. Over time, this alignment may change in relation to the presence and breadth of individual numerical intervals.

For each epoch, the nodes enabled to access the shared resource calculate or acquire a seed, that is a random or pseudorandom number. By epoch it is meant, in summary according to the TDMA - "Time division multiple access" approach, a macro interval of subdivision of time, in which each epoch is divided into so-called time slots. The number of time slots assigned to each node sharing a shared resource is predetermined according to a predetermined distribution logic.

Each node independently performs the following operations in succession for a number of cycles equal to the number of time slots at the time:

- i-th calculation of the i-th Hash of the seed,
- Calculation of the remainder of the division of said i-th Hash by the sum of the amplitudes of said numerical intervals,
- Determination of the interval in which said remainder falls,
- Allocation of the i-th slot to the node corresponding to said interval.

The network can be formed by a greater number than the enabled nodes. The redistribution of time slots between nodes can depend on authoritative or random policies.

The seed is a random or pseudo-causal number that can be extracted from a specific node that, for example, has the sole task of generating the seed.

When the present invention is applied to the field of blockchains, the shared resource

consists of the task of undermining a block and the seed can be determined on the basis of a hash built on the basis of one or more blocks of a previous era.

For example, it is possible to divide the number of mined blocks of a previous epoch into three or four groups and to consider the hash of the first block of each group or the hash of the second block of each group, or the first, second and third hash of the second group or the hash of the last, penultimate and third last block of the first set of blocks.

The hashes extracted in this way are concatenated. Their concatenation or the hash of their concatenation (hash of hash) defines the seed for calculating the order of access to the shared resource of the next epoch.

Advantageously, since all nodes keep a copy of the blockchain, each node is able to calculate the order in which each enabled node has the right to access the shared resource. The essential elements that each node must know are the distribution of the time slots between the enabled nodes, the unique identifier of all the enabled nodes and the seed.

According to the application of this construction variant to blockchains, these are of the Proof of Stake type, in which there is no competition to complete the construction or mining of the blocks. The slots are assigned to the nodes according to the method described above and each node has the task of completing a block within this time slot, inserting queued and pending transactions at the time the time slot begins.

Advantageously, the present embodiment allows a blockchain to continue in its tasks even in the event of random and momentary disconnection of the nodes.

The performance of a node can be appreciated in relation to the number of transactions that it can include in a block generated in the assigned time slot.

At the end of the block generation, the node assigning the time slot forwards it to the network of nodes, which accept or reject this block, if it is correct. When the block is accepted by 50% + 1 of the nodes, the block is considered as definitively accepted and

therefore, the blockchain can continue to grow by appending additional blocks to that block.

According to a preferred variant, once the time slot has elapsed, if the assignee node does not send the block generated in the same time slot in time, therefore called "late block", the assignee node of the next time slot starts generating its own block, called “new block”, ignoring the missed or late reception of the late block. This implies that some of the transactions present in the late block can be included in the new block. A bifurcation and competition between the late block and the new block is therefore determined. The competition is won by the block that first receives 50% + 1 approval from the remaining nodes of the network. Therefore, the acceptance of the delayed block or the new one may depend on the extent of the delay.

A further block is therefore appended to the block that wins the contest. It must be taken into account that, in blockchains, the block that is queued includes the hash of the block to which it is queued.

BRIEF DESCRIPTION OF THE FIGURES

The structural and functional features of the present invention and its advantages with respect to the known prior art, will become even clearer from the underlying claims, and in particular from an examination of the following description, made with reference to the attached figures which show a preferred but not limited schematic embodiment of the invented computer-implemented method, system, device, wherein:

Figure 1 illustrates a flow chart of the computer-implemented method according to the present invention;

Figure 2 illustrates an operative example of the registration process of the computer-implemented method according to the present invention;

Figure 3 illustrates an example scheme of the network including one MAIN address and

three OVERFLOW nodes according to the present invention;

Figure 4 illustrates a flow chart of the initialization process of the computer-implemented method according to the present invention;

Figure 5 illustrates an operative flow chart of the initialization process of the computer-implemented method according to the present invention;

Figure 6 illustrates an operative example of the initialization process of the computer-implemented method according to the present invention;

Figure 7 illustrates a flow chart of the redistribution process of the computer-implemented method according to the present invention;

Figure 8 illustrates an operative flow chart of the redistribution process of the computer-implemented method according to the present invention;

Figure 8 illustrates an operative flow chart of the redistribution process of the computer-implemented method according to the present invention;

Figure 9 illustrates an operative example of the redistribution process of the computer-implemented method according to the present invention;

Figure 10 illustrates a flow chart of the penalty process of the computer-implemented method according to the present invention.

Figure 11 is an association table of entities necessary for the autonomous calculation of access to a shared resource;

Figure 12 shows a numerical range;

Figure 13 shows a scheme for accessing a shared resource according to a preferred embodiment of the present invention;

Figure 14 shows a flow chart of a method according to a preferred embodiment of the present invention.

In the context of this description, the term "second" component does not imply the

presence of a "first" component. These terms are in fact used as labels to improve clarity and should not be understood in a limiting way.

DETAILED DESCRIPTION OF THE INVENTION

In general, this disclosure describes a computer-implemented method, system for reaching a proof of stake based distributed consensus for a distributed ledgers, such as the blockchain technology. In particular this invention relates to a proof of stake based blockchain technology where stakeholders cannot and do not have to communicate with mining nodes.

The blockchain technology according to the present invention provides for two main actors: stakeholders i.e. who owns tokens allowing control of the chain, and miner i.e. who control mining nodes actively operating in the network and entrusted with a sufficient amount of stake in order to obtain assignment of a certain number of work slots where they mine a correspondent certain number of block thus lengthening the chain.

The blockchain is considered a meta actor and it is the union of the nodes operating on the network. Sending a transaction to the blockchain leads that this transaction reaches each operating node which updates its internal status consistently. These update procedures are constructed in such a way as to ensure that the same initial conditions produce identical status updates in every node in a strictly deterministic way. This meta actor represents the massive update procedure on each node of the network.

From now on it is intended:

- a slot a time interval of 30 seconds. These slots are uniquely assigned to the mining nodes and within a given slot one and only one node has the right to produce a block. This assignment is unique and strictly deterministic and is performed before the start of each epoch;

- an epoch is intended a grouping of contiguous slots. Each epoch consists of an equal

number of slots. Assignment and redistribution of slots and stakes are calculated once for each epoch and remain unchanged until the end of the same.

- a stake is a numerical representation of the right to vote of each stakeholder of the network. The stake may be delegated with specific rules and operations to the nodes involved in the block creation.

- a MAIN address: this address is not associated with a physical object (a MAIN address in not associated to an active mining node) and acts as a placeholder indicating a physical resource group which the blockchain is desired to access, thus exposing its private key only during assignment transaction of OVERFLOW address to MAIN address and not during other operative transaction, increasing the global security of the chain; this MAIN address (for instance alice_main_address) is declared by its owner and its managed using its private key (for instance alice_key). Only the MAIN address may declare which OVERFLOW nodes are assigned to it;

- an OVERFLOW address: this address is associated with a physical object and acts as an active mining node; this OVERFLOW address (for instance node_1_overflow_address) is managed using its private key (for instance node_1_key); it is possible that the MAIN address manager and the OVERFLOW node manager are the same entity but they use different keys when impersonating Alice or Node_1; the owner of each MAIN address (for instance Alice) using a specific transaction associates each registered overflow node to its MAIN address. Only the MAIN address in the network may link an OVERFLOW address to itself, meaning that only the MAIN address may send an ASSIGN type transaction linking an OVERFLOW to a MAIN address. This operation is not commutative, i.e. an OVERFLOW address may not assign itself to a MAIN address. OVERFLOW address/node may not be shared between MAIN addresses, an OVERFLOW address may be assigned to only one MAIN address. Only an address

previously declared as OVERFLOW may be associated with a MAIN address. An OVERFLOW node may also deregister and by doing so, removes itself from both the pool of available nodes and the MAIN address it was assigned to beforehand. This is to give both MAIN and OVERFLOW nodes the possibility to protect themselves from unwanted situations in which they get linked together, whether it's the result of an honest mistake or malicious behaviour.

If an OVERFLOW address is not linked to any MAIN address it will be operative only as a replication node.

The computer implemented method according to the present invention (fig. 1) comprises, a redistribution process, a penalty process and may comprise a registration process, an initializing process and a maintenance process.

The computer implemented method according to the present invention is set up according to the following parameters and values, said values are to be intended as examples only and may be modified according specific technical needs:

Time Slot duration=30 seconds

Epoch Slots (E_S)= 24000 => number of slots for each epoch

Penalty Factor=2 => determines the amount of time penalty for the current protocol violation

Recovery Factor=1 => determines the age of return from the penalty regime

Target number of nodes (MAX_N)=400 => optimal number of nodes making the network

Minimum number of nodes (min_N)=200

Minimum slot per node Slot_min (E_S/MAX_N)=60

Maximum slot per node Slot_MAX (E_S/min_N)=120

Epoch Evaluation Point (EEP) (E_S/3)=8000

The registration process is configured to register one or more MAIN address, to register one or more OVERFLOW address/node and to associate said one or more OVERFLOW address/node to a MAIN address and to register said association.

The registration process (fig. 2, 3) implemented on each overflow node of the blockchain network comprises the following steps:

5: via the blockchain network, receiving a transaction including as input MAIN address and registering it in an address register using a private key of the manager of the MAIN address

10: via the blockchain network, receiving a transaction including as input one or more OVERFLOW addresses and registering said one or more OVERFLOW address in the address register using a private key of the manager of each OVERFLOW address

15: via the blockchain network, receiving a transaction including as input assigning the one or more OVERFLOW addresses to the MAIN address registered in the step 5.

The computer implemented method according to the present invention provides for setting a minimum amount of stake (S_min_VAL) assigned to an OVERFLOW node to be activated and considered as mining node and in order to avoid having all the stake concentrated on only a few nodes is settled a soft upper limit to the amount of stake that can be assigned to a node, beyond which mining node starts incurring in penalties. This should make concentration of stake beyond efficient ranges less appealing to nodes looking to reap the maximum possible rewards. The maximum amount of stake desirable (S_MAX_VAL) may be double the value of S_min_VAL.

The algorithm is designed to maximise the number of active nodes by spreading the excess stake among as many overflow nodes as possible in order to expand the network thus increasing its resilience.

The computer implemented method comprises an initializing process in order to recover

the incorrectly pointed stake on an OVERFLOW node and to collect them to the corresponding MAIN address before performing a redistribution process according to computer implemented method of the invention. The initializing process is useful when stakes are incorrectly bet on one or more OVERFLOW nodes, thus making the distribution between OVERFLOW nodes reliable. Furthermore the initializing process is useful to ensure the OVERFLOW addresses have no stakes assigned before performing the redistribution process.

The initializing process (fig. 4) is configured to list all MAIN addresses of the blockchain network, for each MAIN address list its assigned OVERFLOW addresses/nodes, and then for each OVERFLOW addresses/node transfer the stake if it has any, to its assigned MAIN address.

The initializing process (fig. 5, 6) implemented on each overflow node of the blockchain network in particular comprises the following steps:

20: having a blockchain network comprising one or more MAIN addresses, for each MAIN addresses are associated in turn one or more OVERFLOW addresses/nodes acting as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node;

25: via the blockchain network, receiving a transaction including as input addresses from address register comprising MAIN addresses and their corresponding OVERFLOW addresses;

30: associating to each MAIN address a univocal progressive number;

35: for each MAIN address verifying if it contains an amount of stake and if this value is >0 assigning this value to parameter M_A_S, and if this value is <=0 implementing step 35 to the next MAIN address member of the blockchain network and generating a transaction including output providing the M_A_S value associated to the corresponding

MAIN address,

40: associating to each OVERFLOW node a univocal progressive number

45: for each OVERFLOW node assigned to each corresponding MAIN address verifying if it contains an amount of stake defined OVERFLOW_ADDRESS.assigned_stake and if this value is >0 assigning this value to parameter O_A_S, calculating value MAIN_ADDRESS.assigned_stake as M_A_S+O_A_S and assigning O_A_S=0, and generating a transaction including output providing the updated value of the M_A_S value associated to the corresponding MAIN address, then implementing step 45 to the next OVERFLOW address assigned to the corresponding MAIN address considered under step 35,

and if OVERFLOW_ADDRESS.assigned_stake is <=0 implementing step 45 to the next OVERFLOW address assigned to the corresponding MAIN address considered under step 35

50: generating a transaction including output of the updated value of TOTAL_AT_STAKE value for each MAIN address of the blockchain network corresponding to the total amount of stake on each MAIN address after performing preceding steps.

The redistributing process (fig. 7) is configured to calculate the minimum stake necessary for an OVERFLOW node to mine, being minimum_stake_value=total_amount_at stake/predetermined_target_number_of_nodes, to calculate the maximum stake allowing efficient mining, being maximum stake value=minimum stake value*2, for each MAIN address to distribute the stake among the OVERFLOW nodes, to scroll through the OVERFLOW nodes of a given MAIN address and to assign to every OVERFLOW node the minimum_stake_value, until the remaining stake is insufficient to activate a miner or every overflow node has been assigned its stake. The redistribution process may further

be configured in such a way that if there is any stake left it is distributed between the OVERFLOW nodes until as many nodes as possible have a stake equal to maximum stake value and any further stake is assigned to the first OVERFLOW node according a predetermined order.

The redistributing process (fig. 8, 9) comprises in turn a first iteration redistribution process where the algorithm assigns to every available OVERFLOW node assigned to each MAIN address the minimum required amount of stake to became an active mining node, and a second iteration redistributing process the algorithm distributes the stake trying to maximise the number of overflow nodes operating at full stake in view of S_MAX_VAL settled for any OVERFLOW node.

The redistribution process comprises in particular the following steps:

52: having a blockchain network comprising one or more MAIN addresses, for each MAIN addresses are associated in turn one or more OVERFLOW addresses/nodes acting as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node;

55: getting as input TOTAL_AT_STAKE value for each MAIN address of the blockchain network corresponding to the total amount of stake on each MAIN address

60: calculating S_min_VAL= TOTAL AT STAKE/TARGET NUMBER OF_NODES being TARGET NUMBER OF_NODES = total number of OVERFLOW mining nodes in the chain network,

Calculating S_MAX_VAL=S_min_VAL * 2

Assuming a counter value ITERATION=0

65: if ITERATION value is <2 then updating the counter value ITERATION=ITERATION+1

70: for each MAIN address getting as input the MAIN_ADDRESS.assigned_stake value

and if this value is >0, thus meaning that the MAIN address under consideration contains an amount of stake, assigning this value to parameter M_A_S,

and if M_A_S value is >= S_MAX_VAL repeating step 70 to the next MAIN address,

75: if M_A_S value is < S_MAX_VAL getting as input OVERFLOW_ADDRESS.assigned_stake value for each OVERFLOW address associated to the MAIN under consideration, and

if OVERFLOW_ADDRESS.assigned_stake value is =0 repeating step 75 to the next OVERFLOW address associated to the MAIN under consideration, and

if OVERFLOW_ADDRESS.assigned_stake value is ≠0, assigning OVERFLOW_ADDRESS.assigned_stake value to O_A_S parameter, calculating MAIN_ADDRESS.assigned_stake = M_A_S-S_min_VAL, and calculating OVERFLOW_ADDRESS.assigner_stake =O_A_S+S_min_VAL, and repeating step 75 to the next OVERFLOW address associated to the MAIN under consideration.

The penalty process makes mining operations that do not respect settled rules disadvantageous, thus incentivising nodes to naturally expand the network thus increasing its resilience as already discussed.

From now on it is intended:

$epoch_i$ the epoch when miners registered their MAIN and OVERFLOW addresses,

$epoch_{i+1}$ the epoch when miners mine their assigned slots,

$epoch_{i+2}$ the epoch when rewards and penalties are evaluated,

The subscript "j" concerns $block_j$ written in the $slot_j$

The subscript "i" concerns $epoch_i$ and what is relevant to $epoch_i$.

The computer implemented method according to this invention includes a reward for miner for remunerating their calculation effort during their mining operation. If a miner is not able to write a block in the corresponding slot or he mines in a number of slots

over a predetermined limit, he will not receive in the following epoch the corresponding reward for the block not written or for the block mined over the predetermined limit (in each slot only one block may be written), but this reward (for the blocks mined over the limit) is frozen and will be recovered very slowly a little at a time (for instance according to a number of parts equal to twice the number of slots mined under penalty regime) for each following slots will have been mined by the miner, in particular according to a number of parts proportional to the number of slots mined over said predetermined limit Slot_MAX, being one part assigned to each following slot the miner is delegable to mine, thus in order to naturally incentive the distribution of the blockchain network.

As mentioned before a $reward_{i+2}$ corresponds to the reward for mining a $block_{j+1}$ in the $epoch_{i+1}$.

This $reward_{i+2}$ includes one coinbase (i.e. a new created coin) for the $block_{j, i+1}$ mined, and a $collected\_fee_{j, i+1}$. This collected $fee_{j, i+1}$ in turn includes the collected fee paid by users wanting a transaction to be performed and written in the relevant $block_{j, i+1}$ of blockchain network.

From now on it is intended:

$Penalty\_slots_{i+1}$= number of slots mined over the limit in the $epoch_{i+1}$ i.e. number of slots over Slot_MAX*2

$Frozen\ fee_{i+1}$= is the amount of reward not dispensed to the miners in the epoch $_{i+1}$ for having mined a number of slots over the limit Slot_MAX, thus having been under penalty regime.

The penalty process (fig. 10) implemented on each overflow node of the blockchain network comprises in particular the following steps:

80: calculating total $frozen\_fee_{i+1}$ as the amount of $reward_{i+2}$ not dispensed to the miners in a specific $epoch_{i+1}$ for having mined a number of slots over the limit Slot_MAX

85: calculating penalty_slots$_{i+1}$ as preferably twice (or other proportional law) the number of slots mined over the limit i.e. number of slots over Slot_MAX

90: calculating recovered_fee$_{j,\,i+1}$ value as frozen_fee$_{i+1}$/penalty_slots$_{i+1}$ i.e. the fee to be recovered to each slot

95: adding the recovered_fee$_{j,\,i+1}$ value to the reward of the MAIN_address for mining the block$_i$ in the epoch$_{i+1}$ as reward$_{j,\,i+1}$=one coinbase+collected_fee$_{j,\,i+1}$+recovered_fee$_{j,\,i+1}$

100: calculating the new value of frozen_fee$_{i+1}$ parameter as frozen_fee$_{i+1}$-recovered_fee$_{j,\,i+1}$ and the new value of penalty_slots$_{i+1}$ as penalty_slots$_{i+1}$-1 and repeat step 95 for the following block$_{j+1}$ in the epoch$_{i+1}$ while frozen_fee$_{i+1}$>0 or while penalty_slots$_{i+2}$>0.

At the end of each epoch, corresponding with the last block mined, penalty_slots and frozen_fee are transferred from the first OVERFLOW node to its MAIN address. Likewise, at the beginning of every epoch, corresponding with the first block mined, the penalty_slots and frozen_fee are transferred from the MAIN address to its first OVERFLOW node.

This procedure is necessary to avoid malicious behaviour such as undue transfers of tokens from one pool to another.

Also, since it's always possible to register a new overflow node and calculate its address so that it precedes in the order of overflows the one with the assigned penalties. By doing so, a pool would recover frozen fees without re-entering within the optimal parameters or extending the network. Considering that under normal circumstances fees outweigh the generated coinbase this would reduce significantly the efficacy of the penalties.

With regard to events of scheduled maintenance in order to keep the blockchain network highly distributed even in case of failure of one or more mining nodes, the computer-implemented invention according to the present invention may comprise a maintenance process comprising in turn the following steps, being given a MAIN address and at least

two or more OVERFLOW addresses and a node desired to be maintained:

110: sending a transaction including a first input providing a substitute OVERFLOW address using its private key, thus meaning said substitute OVERFLOW address being a node substituting the node desired to be maintained

120: assigning said substitute OVERFLOW address to the corresponding MAIN address using private key of the MAIN address

130 deregistering the node desired to be maintained using its private key. Alternatively the MAIN address can UNASSIGN an overflow node previously assigned to it.

With regard to blockchain security aspects according to the present invention, as per the way the assignment system is structured, MAIN address private keys come into play only for the short period of time necessary to generate registration and assignment transaction but is no longer involved in any subsequent processing phase. This exempts it from being available online on an object that may be subject to malicious attack. On the other hand presence constantly online of a physical servers and the corresponding keys necessary to operate on the network allows a malicious third party to steal private keys from them and to take over all the stake delegated to the MAIN address farm.

The present invention further concerns a node (preferably an OVERFLOW node) configured to implement the computer implemented method disclosed, said node comprising an interface device, a processor coupled to said interface device, a memory coupled to the processor, the memory having stored thereon computer executable instructions which, when executed, configure the processor to perform the corresponding steps of the computer implemented method disclosed.

The present invention further concerns a computer readable storage medium comprising computer-executable instructions which, when executed, configure a processor to perform the computer implemented method disclosed.

The present invention further concerns a computer network for performing a blockchain-based transaction from a first user to a second user in an electronic transfer network of connected nodes, wherein the nodes includes at least a first node, a second node, as disclosed before, wherein each node share a same blockchain, and wherein each node stores a computer readable storage medium disclosed before.

Hereinafter are described some operative examples of the computer implemented method according this invention.

First example: Maliciuos Operator

Suppose two MAIN addresses:

- Alice, alice_main_addr, node_a1, node_a2, … , node_a10 Stake 30.000
- Mallory, mallory_main_addr, node_m1, node_m2, … , node_m5 Stake 30.000

Total Stake: 1.200.000
Minimum Stake: 3.000 Maximum Stake: 6.000
Slot min: 60 Slot max: 120

To each block generated is assigned a reward of 1 token

In epoch 14 Alice and Mallory completed the registration of their main addresses and assigned them their nodes.

These will be active starting from epoch 15

In epoch 16 the reward will be calculated.

At the beginning of epoch 15 Alice will have 10 active nodes (node_a1, ..., node_a10) each with 60 slots assigned.

At the beginning of the epoch 15, Mallory will have active 5 nodes (node_m1, ..., node_m5) each with 120 slots assigned.

The slots of Alice and Mallory are the same because they have the same active stake. However, the distribution changes. Mallory has only 5 nodes while Alice has 10. None of them is penalized. Alice's nodes

are active with the minimum, Mallory's nodes with the maximum. Assuming a perfect execution of the protocol in epoch 15, without transactions to simplify the example.

The ten Alice nodes have assigned 60 slots, thus producing 60 blocks, in each block there will be a reward of 1 token. Being ten nodes (60 * 10 * 1) the reward will be 600 tokens. At the beginning of epoch 16, Alice's balance will increase from 30,000 to 30,600 tokens.

Let's now pass to Mallory, the five nodes of Mallory have assigned 120 slots, thus producing 120 blocks, in each block there will be a reward of 1 token. Being five nodes (120 * 5 * 1) the expected reward will be 600 tokens. At the beginning of epoch 16, Mallory's balance should increase from 30,000 to 30,600 tokens. Only three of Mallory's nodes, node_m3, node_m4, node_m5, have been modified to operate incorrectly. The causes of this behaviour can be various and now we will list some of them.

- An intentional change was made that attempts to perform invalid operations (duplicate transactions, accept false transactions with incorrect key, ...) = Voluntary malicious interaction.
- The server on which the node is installed is malfunctioning and fails the mathematical operations by marking as invalid valid transactions eg. Carol, with a budget of 100 tokens, sends 25 to Bob. The operation is legal but is marked as illegal in the product block. = Involuntary malevolent interaction.
- The server's network connection is unreliable and the block is not transmitted in time. = Involuntary malevolent interaction.

• The block is connected not to its immediate valid predecessor but to another predecessor violating the rule of extending the heavier chain. In this way, the node aims to make unwanted transactions disappear, for example a payment, from the history of the chain = Involuntary malevolent interaction.

These aspects cause the blocks generated by node_m5 to be rejected by the other nodes and not become part of the blockchain.

In this way the reward collected by Mallory will be:

Node	Collected Reward
node_m1	120
node_m2	120
node_m3	0
node_m4	0
node_m5	0
Total	240

Let's now pass to an evaluation of the remaining part of the network. Alice and Mallory have in total (30,000 Alice and 30,000 Mallory) 60,000 stake tokens.

On the rest of the network (1,200,000 - 60,000) 1,140,000 token in stake remain.

Assuming that all the rest of the network works in ideal conditions and without transactions to the other nodes making up the network, they will be assigned:

Total Slots	Alice's Slots	Mallory's	Slots

		Slots	assigned to the rest of the network
24000	-600	-600	=22800

22,800 blocks will be generated and therefore a reward of 22,800 tokens.

Total Stake	Alice's Stake	Mallory's Stake	Stake assigned to the rest of the network
1200000	-30000	-30000	=1140000

In the epoch 15 the total stake of the network is 1,200,000, the stake assigned to non-Alice or Mallory nodes is 1,140,000. In epoch 16 the reward for the aforementioned nodes will be 22,800 tokens, bringing their stake to 1,162,800 tokens.

Considering Mallory's main collected a lower stake, we recalculate the values for the next epoch.

The overall stake is 1,200,000 + 600 + 240 + 22,800 = 1,223,640

The following table compares the network control percentage in epoch 15 with the revaluation in epoch 16 (expected) and 16 (real)

	Epoch 15	**Epoch 16 (expected)**	**Epoch 16 (real)**
Alice	Stake	Stake	Stake
	30.000	30.600	30.600

	Slot assigned	Slot assigned	Slot assigned
	600	600	600
	Network control	Network control	Network control
	2,5%	2,5%	2,5%
Mallory	Stake	Stake	Stake
	30.000	30.600	30.240
	Slot assigned	Slot assigned	Slot assigned
	600	600	593 (-7) (-1,17%)
	Network control	Network control	Network control
	2,5%	2,5%	2,47% (-0.03%)
Other participants	Stake	Stake	Stake
	1.140.000	1.162.800	1.162.800
	Slot assigned	Slot assigned	Slot assigned
	22.800	22.800	22.807 (+7)
	Network control	Network control	Network control
	95%	95%	95,03% (+0,03%)

As you can see, Mallory's computing power was not eroded (the algorithm is not punitive in the classical sense of removing something) but it has grown more slowly than that of the other components of the network. This caused, at the point of

recalculation of the epoch 16, a change in the slot allocation with Mallory that we saw take off 0.03% of the computing power compared to the total of the network slots. Instead, comparing it to its previous slot number, the erosion corresponds to the loss of 1.17% of the slots.

Second example: Unbalanced stake with fee

Suppose two MAIN addresses:

- Alice, alice_main_addr, node_a1, node_a2, … , node_a10
  Stake 30.000
- Mallory, mallory_main_addr, node_m1
  Stake 30.000

Total Stake: 1.200.000
Minimum Stake: 3.000 Maximum Stake: 6.000
Slot min: 60 Slot max: 120

To each block generated is assigned a reward of 1 token

In epoch 14 Alice and Mallory completed the registration of their main addresses and assigned them their nodes.

These will be active starting from epoch 15

In epoch 16 the reward will be calculated.

At the beginning of epoch 15 Alice will have ten active nodes (node_a1, ..., node_a10) each with 60 slots assigned.

At the beginning of epoch 15, Mallory will have one node_m1 node active with 600 slots assigned.

The slots of Alice and Mallory are the same because they have the same active stake. However, the distribution changes. Mallory has only one node while Alice has ten of them. Mallory is penalized. Alice's nodes are active with the minimum, the Mallory node is 480 slots beyond the limit and for this she will receive a penalty.

Assuming a perfect execution of the protocol in epoch 15, we assume that an average of 5 tokens / block is the amount of collected fee for greater simplicity.

The ten Alice nodes have assigned 60 slots, thus producing 60 blocks, in each block there will be a reward of 1 token. Being ten nodes (60 * 10 * 1) the reward will be 600 tokens. At the beginning of epoch 16, Alice's balance will increase from 30,000 to 30,600 tokens. To these we will add the rewards of operations 600 * 5 = 3.000 tokens. Alice will total 33,600 tokens as a result of epoch 15 operations.

Let's now pass to Mallory, the Mallory node has assigned 600 slots, it will therefore produce 600 blocks, for the first 120 blocks a reward of 1 token will be assigned. For the following 480 the reward will be 0. As for the rewards, using the homogeneous distribution indicated in the preconditions, Mallory will be awarded only in the first 120 slots. For these you will receive 5 * 120 = 600 tokens. The remaining 2,400 are frozen to be returned in a period of 480 * 2 = 960 slots within the limits. Assuming that the configuration of the nodes does not change from epoch 15 to epoch 16, we evaluate what happens to the EEP of epoch 16 for epoch 17.

At the beginning of epoch 16, Mallory's balance should increase from 30,000 to 30,600 tokens as a result of the coinbase and should be assigned a further 3,000 tokens due to the fees. Only that Mallory has activated a number of nodes that are insufficient to meet the network's requirements.

Node	Collected Reward

node_m1	120 (coinbase) + 600 (fee) *2400 are frozen for 960 slots and they may not enter in the reward calculation*
Total	720

Let's now pass to an evaluation of the remaining part of the network. Alice and Mallory have in total (30,000 Alice and 30,000 Mallory) 60,000 stake tokens.

On the rest of the network (1,200,000 - 60,000) = 1,140,000 token in stake remain.

Assuming that all the rest of the network works in ideal conditions and without transactions to the other nodes making up the network, they will be assigned:

Total Slots	Alice's Slots	Mallory's Slots	Slots assigned to the rest of the network
24000	-600	-600	=22800

22,800 blocks will be generated and then a reward of 22,800 tokens for the coinbase portion. For the portion of fees 114,000 tokens

Stake Previous other nodes	1.140.000	+	
Coinbase other nodes	22.800	+	
Fee other nodes	114.000	=	
Total other nodes			***1.276.800*** **+**
Stake Previous Epoch Alice	30.000	+	
Coinbase Alice	600	+	

Fee Alice	3.000 =	
	Total Alice	***33.600 +***
Stake Previous Epoch Mallory	30.000 +	
Coinbase Mallory	120 +	
Fee Mallory	600 =	
	Total Mallory	***30.720 =***
Total stake for new epoch		**1.341.120**

The following table compares the network control percentage in epoch 15 with the revaluation in epoch 16 (expected) and 16 (real)

	Epoch 15	**Epoch 16 (expected)**	**Epoch 16 (real)**
Alice	Stake	Stake	Stake
	30.000	33.600	33.600
	Slot assigned	Slot assigned	Slot assigned
	600	600	601 (+1)
	Network control	Network control	Network control
	2,5%	2,5%	2,51% (+0,01)
Mallory	Stake	Stake	Stake
	30.000	33.600	30.720
	Slot assigned	Slot assigned	Slot assigned
	600	600	550 (-50) (-

			8,33%)
	Network control	Network control	Network control
	2,5%	2,5%	2,29% (-0.21%)
Other participants	Stake	Stake	Stake
	1.140.000	1.276.800	1.276.800
	Slot assigned	Slot assigned	Slot assigned
	22.800	22.800	22.849 (+49)
	Network control	Network control	Network control
	95%	95%	95,20% (+0,20%)

As it can be seen, Mallory's computational power was not eroded (the algorithm is not punitive in the classical sense of removing something) but it has grown more slowly than that of the other components of the network. This has caused, at the point of recalculation of the epoch 16, a change in the slot allocation with Mallory which has seen itself take away 0.21% of the computing power on the network. Comparing it to its previous slot number the erosion corresponds to the loss of 8.33% of the slots. The algorithm, in the presence of the fees (therefore of a charged network), becomes heavily punitive. This is because the greater the network load, the more the violations become critical.

Third example: Recovery of frozen stake

In this section the return mechanism for the stakes that have been

frozen starting from the simulated condition in Unbalanced stake with fee will be presented. In this way the re-entry into the optimal operating parameters of the network is rewarded.

Mallory reconfigures the pool to fit the optimal mining parameters

Starting from the condition mentioned above (second example), before the EEP of epoch 16 it adds 4 nodes. With the addition of 4 nodes its 550 slots are divided into 550/5 = 110 <120. As always, the change will take effect only from the next epoch.

With this change, it completely leaves the penalty area.

As starting conditions we take those highlighted in grey of the last column.

Let's now simulate the execution in case Mallory corrects his behaviour with the addition of 4 nodes.

In this case you will get a reward for its 550 slots:

coinbase = 550 * 1 = 550

collected_fee = 550 * 5 = 2750

From the previous epoch, Mallory inherited a frozen fee of 2400 tokens and a number of penalty slots equal to 960 (twice the slots for which it remained outside the parameters).

The formula to calculate the recovered fee for a slot is as follows:

recovered_feeE17S1 = frozen_feeE17S1 / penalty_slotsE17S1

At this point the values of the frozen_fee and the penalty_slots are updated:

frozen_feeE17S2 = frozen_feeE17S1 - recovered_feeE17S1

penalty_slotsE17S2 = penalty_slotsE17S1 - 1

For notational convenience, the absolute value epoch is indicated in the subscript, in this case 17, but in slot 2 is indicated the slot assigned to Mallory after one. The two slots may not be consecutive. In this case, after the "s", the sequence number of the slot assigned to Mallory is shown.

The following table shows the evolution of values as slots pass.

Assigned Slot	frozen_fee	penalty_slots	recovered_fee
E17S1	2400	960	2,5
E17S2	2397,5	959	2,5
E17S3	2395	958	2,5
...			
E17S109	2130	852	2,5
E17S110	2127,5	851	2,5

At the end of the epoch, Mallory will have recovered 275 tokens for the 2400 frozen. This allows him to partially bridge the gap with the other nodes of the network and serves as an incentive to favour the virtuous behaviour of having added nodes.

In frozen fees there are no coinbases which have not been generated. It is always and in any case more convenient for a node to operate in compliance with the rules than to use this recovery procedure.

Furthermore, according to a preferred embodiment, an access to a shared resource is calculated autonomously by all nodes e.g. OVERFLOW nodes of the network. In this regard, in the proof of stake type blockchains, it is not negligible to guarantee an appropriate redistribution of the construction tasks (called “mining”) of the blocks. While it is simple to centrally determine the order of assignment of the blocks to be mined between the nodes of the blockchain, a fully distributed determination of assignments is

somewhat complex.

The systems for determining the order of access to a time division transmission medium are particularly known as TDMA from the Anglo-Saxon acronym "Time division multiple access". These systems have been conceived in the field of telecommunications to avoid collisions between messages generated by multiple nodes, which lead to a rapid decrease in the network capacity, called "throughput".

The TDMA approach involves dividing time into macro intervals called epochs, in which each epoch is divided into so-called time slots. The number of time slots assigned to each node sharing a shared resource is predetermined according to a predetermined distribution logic.

With reference to Figure 11, each node N1, N2, N4, etc., which accesses the shared resource has its own unique identifier ID1, ID2, ID4, etc .. generally numeric.

The identifiers can be sorted according to a list, as shown in figure 11.

Each identifier is associated with a number of time slots to which the node has the right of access at a predetermined time. An epoch is an ordered grouping over time, of time slots.

Each node is also associated with a numerical interval of width proportional to the number of time slots associated with the same node.

Therefore, if N1 has the right to access two time slots, the relative numerical interval is for example 20, while if N2 has the right to access only one time slot, the relative numerical interval is 10.

The various number ranges are contiguous and do not overlap.

For example, for N1 the range goes from 0 to 19. For N2, the range goes from 19 to 29. For N4 the range goes from 29 to 59.

The other nodes N3 and N5, obviously, do not have the right to access the shared

resource at the time under consideration.

The overall numerical range is given by the sum SUM.

Since each node knows

- the ID of the other nodes,
- the number of time slots to which each node has the right to access,
- the seed,

then it can independently calculate the order of access to the shared resource according to the steps performed cyclically for a number of cycles equal to the number of time slots at the time:

- i-th calculation of the seed hash,
- Calculation of the remainder of the division of said Hash by the sum SUM of the amplitudes of said numerical intervals,
- Determination of the interval in which said remainder falls,
- Allocation of the i-th slot to the node corresponding to said interval.

For example, if the Hash of the seed leads to a numerical value of 600, then, since SUM is equal to 60, the division 610/60 brings as remainder 10. The value 10 falls in the first numerical interval R1 which belongs to N1 . Therefore the first time slot is assigned to N1.

The Hash of the previously calculated Hash is then calculated. This time, the remainder of the division leads to a value equal to 32 which falls in the interval R4 which belongs to N4. Therefore the second time slot belongs to N4.

Since R4 has a greater interval than the others, the probability of being assigned a time slot is proportional to the width of its interval and therefore, the correct redistribution of time slots is guaranteed, but with an assignment order that is maximally dependent on the seed.

The procedure continues until all time slots are assigned.

As regards the division, one must take into account that a Hash can have a numerical representation whatsoever, for example mZOhnZ2ZSUva Cq16HEy0 + + = 1I87DHFhVUKI28QcswrfQ and can be converted into hexadecimal notation becoming 9993a19d9d99494bdaf82ab5e87132d3ed48f3b0c7161554288dbc41cb30adf4 or in decimal notation becoming 391875681579805356411831496321815245512789534030969319853065217075543948136635096802576932411078811361721 3.

Therefore, the aforementioned division can be achieved by carrying out any conversions of divisor or dividend notation so that they are compatible.

This also applies to any unique identifier.

Figure 14 shows an explanatory flow chart of the method object of the present invention, which includes the following preliminary steps in succession:

- (i) sharing among all nodes of all the unique identifiers of at least said predetermined number of nodes (N1, N2, N4) having the right to access the shared resource,
- (ii) sharing of said respective predetermined number of time slots assigned to each node,
- (iii) sharing a seed;

the method including the following steps in succession

- (iv) ordering of said identifiers,
- (v) association to each node of a numerical interval (R1, R2, R4) of size proportional to said respective predetermined number of associated time slots,
- (vi) alignment of said numerical intervals so as to be contiguous and without overlapping according to an order defined by said order of identifiers,
- (vii) calculation of a sum (SUM) of said numerical intervals,
- (viii) cyclic execution of the following steps for a number of cycles equal to the number

of time slots of the time:

a. i-th calculation of the i-mo Hash of the seed,

b. Calculation of the remainder of a division of said i-th Hash by said sum (SUM),

c. Determination of the numerical interval (R1, R2, R4) in which said remainder falls,

d. Attribution of the i-th slot to the node corresponding to said numerical range in which said remainder falls.

Figure 13 shows a random allocation scheme of the time slots according to the present invention, now described assuming that the shared resource is a transmission medium.

It is assumed that there are a predetermined number of transmitting stations N1, N2, N4, indicated as "Nodes" and another number A, B, C of flow generators, indicated as "Holders". It must be understood that it is just a coincidence that the transmitting stations and the flow generators are equal in number.

Holders, for example, can be access servers to a portion of a telematic network, for example the internet.

Flow generator A is entitled to use three time slots, flow generator B is entitled to five time slots, while flow generator C is entitled to seven time slots.

Evidently, each flow generator can forward a request for access to the transmission medium to a transmitting station.

A wishes to request node N1 to transmit its data during the assigned time slots.

B distributes two assignments to N1, three in loads to N2 and two and N3.

The calculation of the time slots to be accessed can be performed by the flow generators or by the transmitting stations. Furthermore, the identifiers ID can belong to the flow generators or to the transmission nodes.

Therefore, the time slots of the time (Epoch) of Figure 13 are assigned by the present invention to N2, N1, N3, N2, N3, etc ..

Figure 13 can also be interpreted as a situation in which in a proof of stake blockchain, the flow generators, indicated in the figure as "Holders" are servers that receive / collect transaction requests, while the nodes N1 - N4 correspond to the miners.

In both cases, Holders can choose the node to which to entrust their transactions (or their data flow), in relation to the reliability of the node itself. For example, by keeping track of the efficiency in the execution of tasks assigned in previous times.

The present invention can be advantageously realized by means of a computer program which comprises coding means for carrying out one or more steps of the method, when this program is executed on a computer. Therefore it is intended that the scope of protection extends to said computer program and further to computer readable means comprising a recorded message, said computer readable means comprising program coding means for carrying out one or more steps of the method. , when said program is run on a computer.

P3183PC00

CLAIMS

1. A computer-implemented method for reaching a proof of stake based distributed consensus for a blockchain network, thus allowing to achieve higher level of distribution of the network and to not have too much stake concentrated in too few nodes for increasing network robustness and reducing vulnerability at a compromised node, said

blockchain network comprising one or more MAIN addresses, for each MAIN addresses being associated in turn one or more OVERFLOW addresses/nodes, the last being able to act as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node, characterized by comprising the following processes:

-a redistribution process being configured to calculate the minimum stake necessary for an OVERFLOW node to mine, being minimum_stake_value=total_amount_at stake/predetermined_target_number_of_nodes, to calculate the maximum stake allowing efficient mining, being maximum stake value=minimum stake value*2, for each MAIN address to distribute the stake among the OVERFLOW nodes, to scroll through the OVERFLOW nodes of a given MAIN address and to assign to every OVERFLOW node the minimum_stake_value, until the remaining stake is insufficient to activate a miner or every overflow node has been assigned its stake;

-a penalty process being configured to freeze an amount of reward corresponding to the computational work for having mined a number of slots over a predetermined limit Slot_MAX in an $epoch_{i+1}$, and to recover it in the following epochs starting from $epoch_{i+2}$ progressively according to a number of parts proportional to the number of slots mined over said predetermined limit Slot_MAX, being one part assigned to each following slot the miner is delegable to mine, thus in order to naturally incentive the distribution of the blockchain network.

P3183PC00

2. The computer-implemented method according to claim 1 wherein the redistribution process may further be configured in such a way that if there is any stake left it is distributed between the OVERFLOW nodes until as many nodes as possible have a stake equal to maximum stake value and any further stake is assigned to the first OVERFLOW node according a predetermined order.

3. The computer-implemented method according to one of the preceding claims wherein the redistribution process comprises in particular the following steps:

52: having a blockchain network comprising one or more MAIN addresses, for each MAIN addresses are associated in turn one or more OVERFLOW addresses/nodes acting as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node;

55: getting as input TOTAL_AT_STAKE value for each MAIN address of the blockchain network corresponding to the total amount of stake on each MAIN address

60: calculating S_min_VAL= TOTAL AT STAKE/TARGET NUMBER OF_NODES being TARGET NUMBER OF_NODES = total number of OVERFLOW mining nodes in the chain network,

Calculating S_MAX_VAL=S_min_VAL * 2

Assuming a counter value ITERATION=0

65: if ITERATION value is <2 then updating the counter value ITERATION=ITERATION+1

70: for each MAIN address getting as input the MAIN_ADDRESS.assigned_stake value and if this value is >0, thus meaning that the MAIN address under consideration contains an amount of stake, assigning this value to parameter

P3183PC00

M_A_S,

and if M_A_S value is >= S_MAX_VAL repeating step 70 to the next MAIN address,

75: if M_A_S value is < S_MAX_VAL getting as input OVERFLOW_ADDRESS.assigned_stake value for each OVERFLOW address associated to the MAIN under consideration, and

if OVERFLOW_ADDRESS.assigned_stake value is =0 repeating step 75 to the next OVERFLOW address associated to the MAIN under consideration, and

if OVERFLOW_ADDRESS.assigned_stake value is ≠0, assigning OVERFLOW_ADDRESS.assigned_stake value to O_A_S parameter, calculating MAIN_ADDRESS.assigned_stake = M_A_S - S_min_VAL, and calculating OVERFLOW_ADDRESS.assigner_stake = O_A_S + S_min_VAL, and repeating step 75 to the next OVERFLOW address associated to the MAIN under consideration.

4. The computer-implemented method according to one of the preceding claims 1, 2 wherein the penalty process comprises in particular the following steps:

80: calculating total frozen_fee$_{i+1}$ as the amount of reward$_{i+2}$ not dispensed to the miners in a specific epoch$_{i+1}$ for having mined a number of slots over the limit Slot_MAX

85: calculating penalty_slots$_{i+1}$ as preferably twice (or other proportional law) the number of slots mined over the limit i.e. number of slots over Slot_MAX

90: calculating recovered_fee$_{j,\ i+1}$ value as frozen_fee$_{i+1}$/penalty_slots$_{i+1}$ i.e. the fee to be recovered to each slot

95: adding the recovered_fee$_{j,\ i+1}$ value to the reward of the MAIN_address for mining the block$_i$ in the epoch$_{i+1}$ as reward$_{j,\ i+1}$=one coinbase+collected_fee$_{j,\ i+1}$ +

P3183PC00

recovered_fee$_{j,\ i+1}$

100: calculating the new value of frozen_fee$_{i+1}$ parameter as frozen_fee$_{i+1}$ - recovered_fee$_{j,\ i+1}$ and the new value of penalty_slots$_{i+1}$ as penalty_slots$_{i+1}$ - 1 and repeat step 95 for the following block$_{j+1}$ in the epoch$_{i+1}$ while frozen_fee $_{i+1}$ >0 or while penalty_slots$_{i+2}$>0, being that when the last block$_j$ is mined at the end of each epoch$_i$, penalty_slots$_i$ and frozen_fee$_i$ are transferred from the first OVERFLOW node to its MAIN address and when the first block$_j$ is mined at the beginning of every epoch$_i$ the penalty_slots$_i$ and frozen_fee$_i$ are transferred from the MAIN address to its first OVERFLOW node.

5. The computer-implemented method according to one of the preceding claims further comprising an initializing process being configured to list all MAIN addresses of the blockchain network, for each MAIN address list its assigned OVERFLOW addresses/nodes, and then for each OVERFLOW addresses/node transfer the stake if it has any, to its assigned MAIN address.

6. The computer-implemented method according to claim 5 wherein the initializing process comprises in particular the following steps:

20: having a blockchain network comprising one or more MAIN addresses, for each MAIN addresses are associated in turn one or more OVERFLOW addresses/nodes acting as potential mining nodes only if a predetermined minimum amount of stake is assigned to said one or more OVERFLOW address/node;

25: via the blockchain network, receiving a transaction including as input addresses from address register comprising MAIN addresses and their corresponding OVERFLOW addresses;

30: associating to each MAIN address a univocal progressive number;

35: for each MAIN address verifying if it contains an amount of stake and if this

P3183PC00

value is >0 assigning this value to parameter M_A_S, and if this value is <=0 implementing step 35 to the next MAIN address member of the blockchain network and generating a transaction including output providing the M_A_S value associated to the corresponding MAIN address,

40: associating to each OVERFLOW node a univocal progressive number

45: for each OVERFLOW node assigned to each corresponding MAIN address verifying if it contains an amount of stake defined OVERFLOW_ADDRESS.assigned_stake and if this value is >0 assigning this value to parameter O_A_S, calculating value MAIN_ADDRESS.assigned_stake as M_A_S+O_A_S and assigning O_A_S=0, and generating a transaction including output providing the updated value of the M_A_S value associated to the corresponding MAIN address, then implementing step 45 to the next OVERFLOW address assigned to the corresponding MAIN address considered under step 35,

and if OVERFLOW_ADDRESS.assigned_stake is <= 0 implementing step 45 to the next OVERFLOW address assigned to the corresponding MAIN address considered under step 35

50: generating a transaction including output of the updated value of TOTAL_AT_STAKE value for each MAIN address of the blockchain network corresponding to the total amount of stake on each MAIN address after performing preceding steps.

7. The computer-implemented method according to one of the preceding claims further comprising a registration process being configured to register one or more MAIN address, to register one or more OVERFLOW address/node and to associate said one or more OVERFLOW address/node to said one or more MAIN address and to register said

P3183PC00

association.

8. The computer-implemented method according to one of the preceding claims further comprising a maintenance process, in order to keep the blockchain network highly distributed even in case of failure of one or more mining nodes, said maintenance process comprising in turn the following steps, being given a MAIN address and at least two or more OVERFLOW addresses and a node desired to be maintained:

110: sending a transaction including a first input providing a substitute OVERFLOW address using its private key, thus meaning said substitute OVERFLOW address being a node substituting the node desired to be maintained

120: assigning said substitute OVERFLOW address to the corresponding MAIN address using private key of the MAIN address

130 deregistering the node desired to be maintained using its private key.

9. Method according to any of the preceding claims and distributed for determining an access order to a shared resource by a predetermined number of nodes OVERFLOW (N1, N2, N4) having a right to access to the shared resource in a predetermined time (epoch) for a respective predetermined number of time slots, in which an epoch is formed by a plurality of time slots, and wherein each node (N1, N2, N4, etc.), has its own unique identifier (ID1, ID2, ID4), the method comprising the following preliminary steps in succession:

- (i) sharing all the univocal identifiers of at least said predetermined number of nodes (N1, N2, N4, etc.) among all the nodes having the right of access to the shared resource,
- (ii) sharing of said respective predetermined number of time slots assigned to each node,
- (iii) sharing a seed,

the method comprising the following steps in succession

- (iv) ordering of said identifiers,

P3183PC00

- (v) association to each node of a numerical interval (R1, R2, R4) of amplitude proportional to said respective predetermined number of associated time slots,

- (vi) alignment of said numerical intervals so as to be contiguous and without overlapping according to an order defined by said identification order,

- (vii) calculation of a sum (SUM) of said numerical intervals,

- (viii) cyclic execution of the following steps for a number of cycles i equal to the time slot number of the epoch:

a. i-th calculation of the seed Hash,

b. Calculation of the rest of a division of said i-th Hash for said sum (SUM),

c. Determination of the numerical range (R1, R2, R4) in which the rest falls,

d. Assignment of the i-th slot to the node corresponding to said numeric interval in which the rest falls.

10. Method according to claim 9, wherein said shared resource is a transmission medium.

11. Method according to claim 10, wherein said shared resource consists of the task of generating a block to be shared by a blockchain.

12. Method according to claim 11, wherein said seed is given by the concatenation of two or more Hash of blocks previously constructed and suitably selected.

13. Method according to claim 12, wherein said blocks are selected according to the following steps:

- distribution of the blocks generated in a previous epoch into three or more groups

- extraction of the first block of each group or the first, second and third hash of the second group or the last, second last and third last block of the first group of blocks;

- concatenation of the Hashs related to the blocks extracted in the previous step.

14. A node configured to implement a method for reaching a proof of stake based distributed consensus for a blockchain network, thus allowing to achieve higher level of

distribution of the network and to not have too much stake concentrated in too few nodes for increasing network robustness and reducing vulnerability at ~~the~~ a compromised node,

said node comprising an interface device, a processor coupled to said interface device, a memory coupled to the processor, the memory having stored thereon computer executable instructions which, when executed, configure the processor to perform the corresponding steps of the computer implemented method of any of claims 1 to 13.

15. Computer program comprising program coding means for realizing all steps or processes of any of claims 1 to 13, when said program is run on a computer.

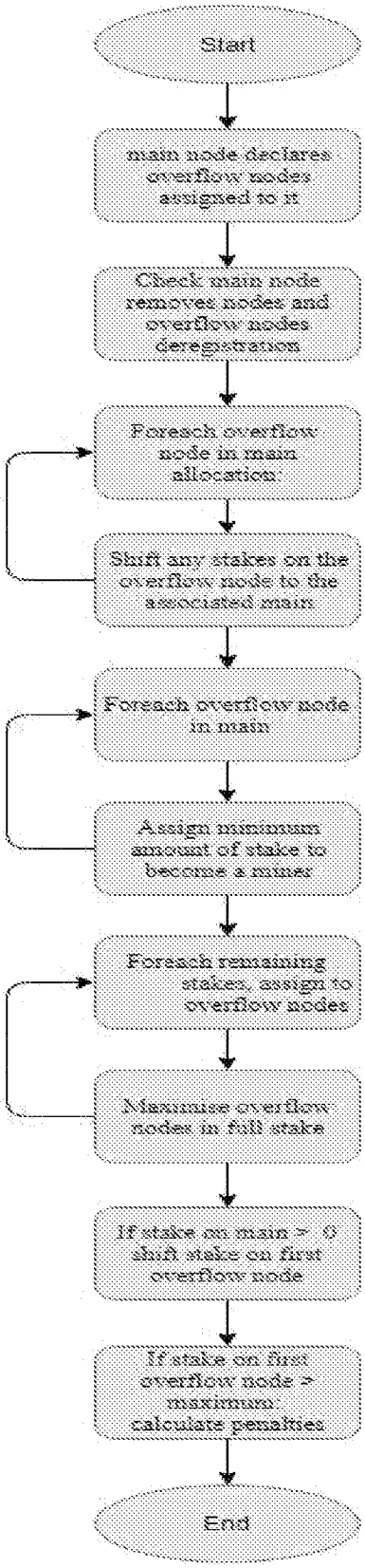
Start
main node declares overflow nodes assigned to it
Check main node removes nodes and overflow nodes deregistration
Foreach overflow node in main allocation:
Shift any stakes on the overflow node to the associated main
Foreach overflow node in main
Assign minimum amount of stake to become a miner
Foreach remaining stakes, assign to overflow nodes
Maximise overflow nodes in full stake
If stake on main > 0 shift stake on first overflow node
If stake on first overflow node > maximum: calculate penalties
End

Fig. 1

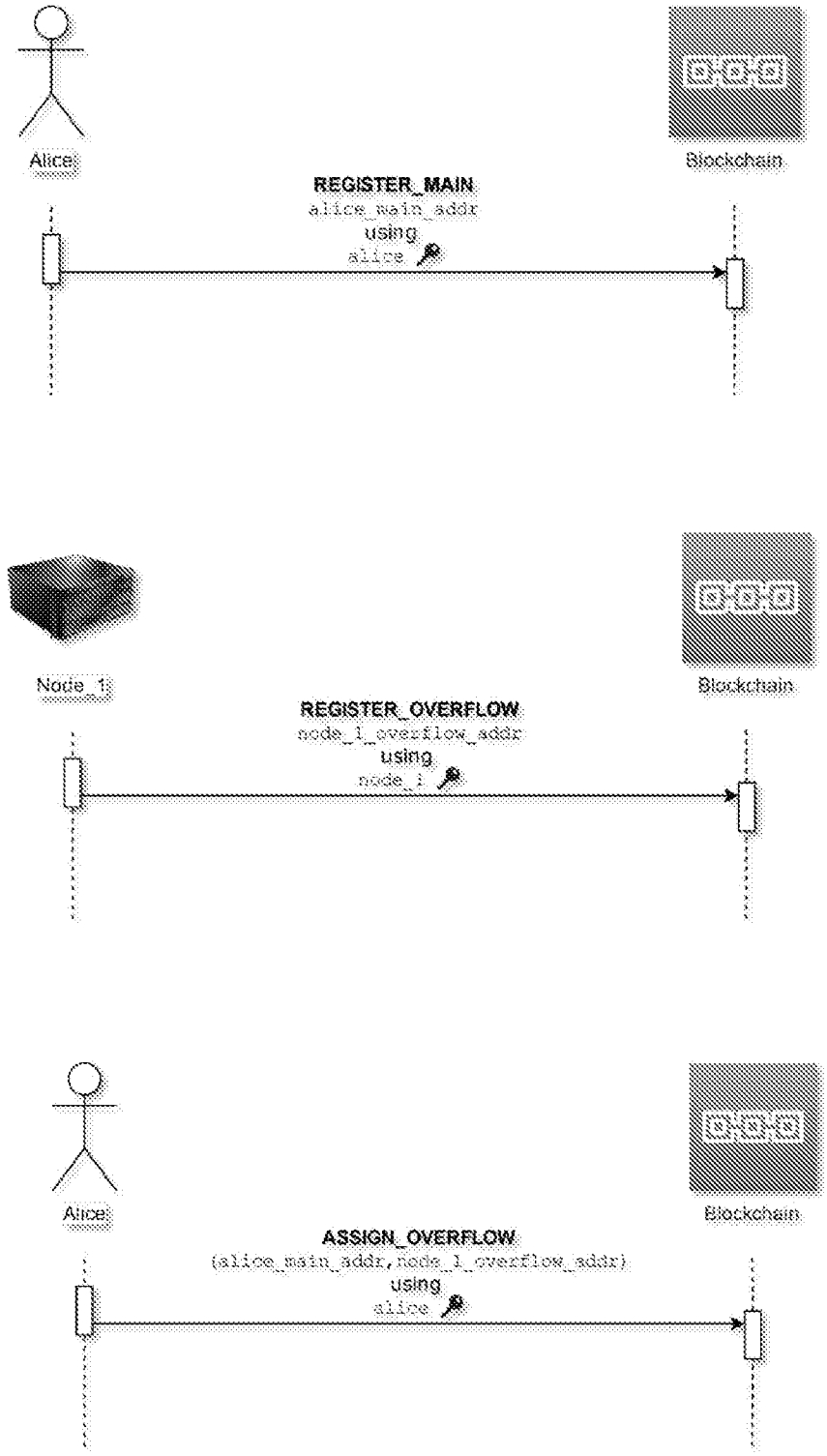
Alice
Blockchain
REGISTER_MAIN
alice_main_addr
using
alice
Node_1
Blockchain
REGISTER_OVERFLOW
node_1_overflow_addr
using
node_1
Alice
Blockchain
ASSIGN_OVERFLOW
(alice_main_addr,node_1_overflow_addr)
using
alice

Fig. 2

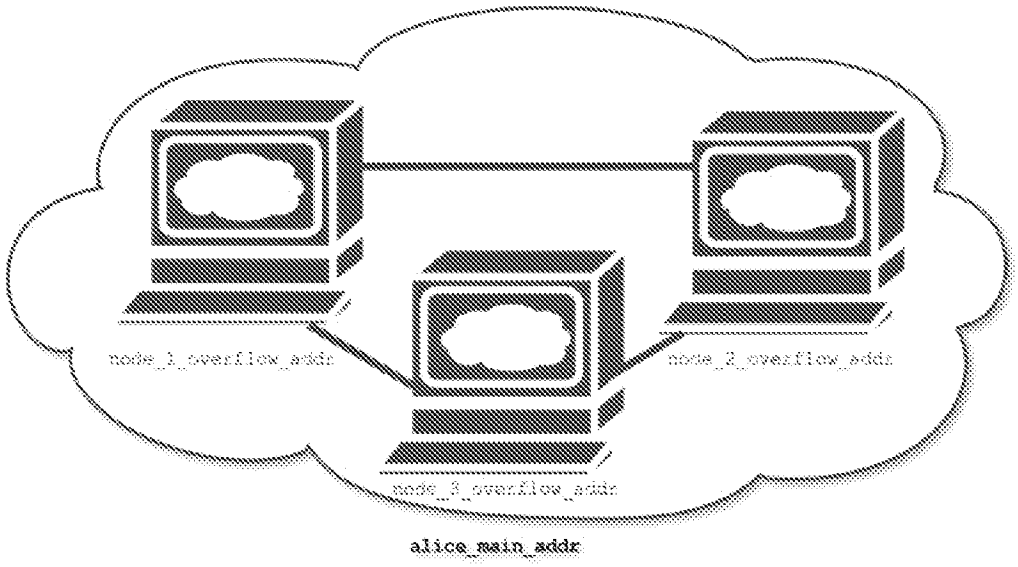
node_1_overflow_addr
node_2_overflow_addr
node_3_overflow_addr
alice_main_addr

Fig. 3

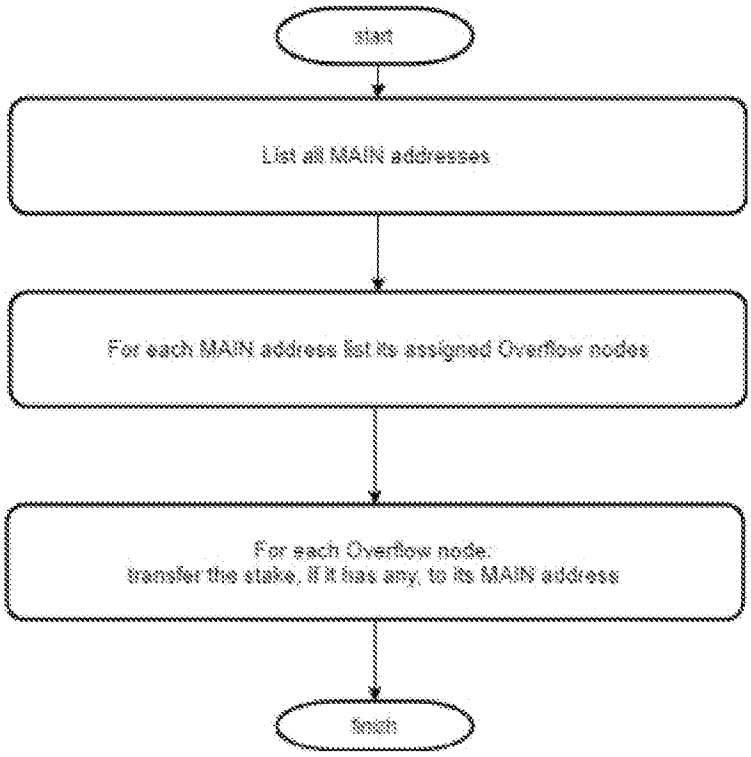
start
List all MAIN addresses
For each MAIN address list its assigned Overflow nodes
For each Overflow node:
transfer the stake, if it has any, to its MAIN address
finish

Fig. 4

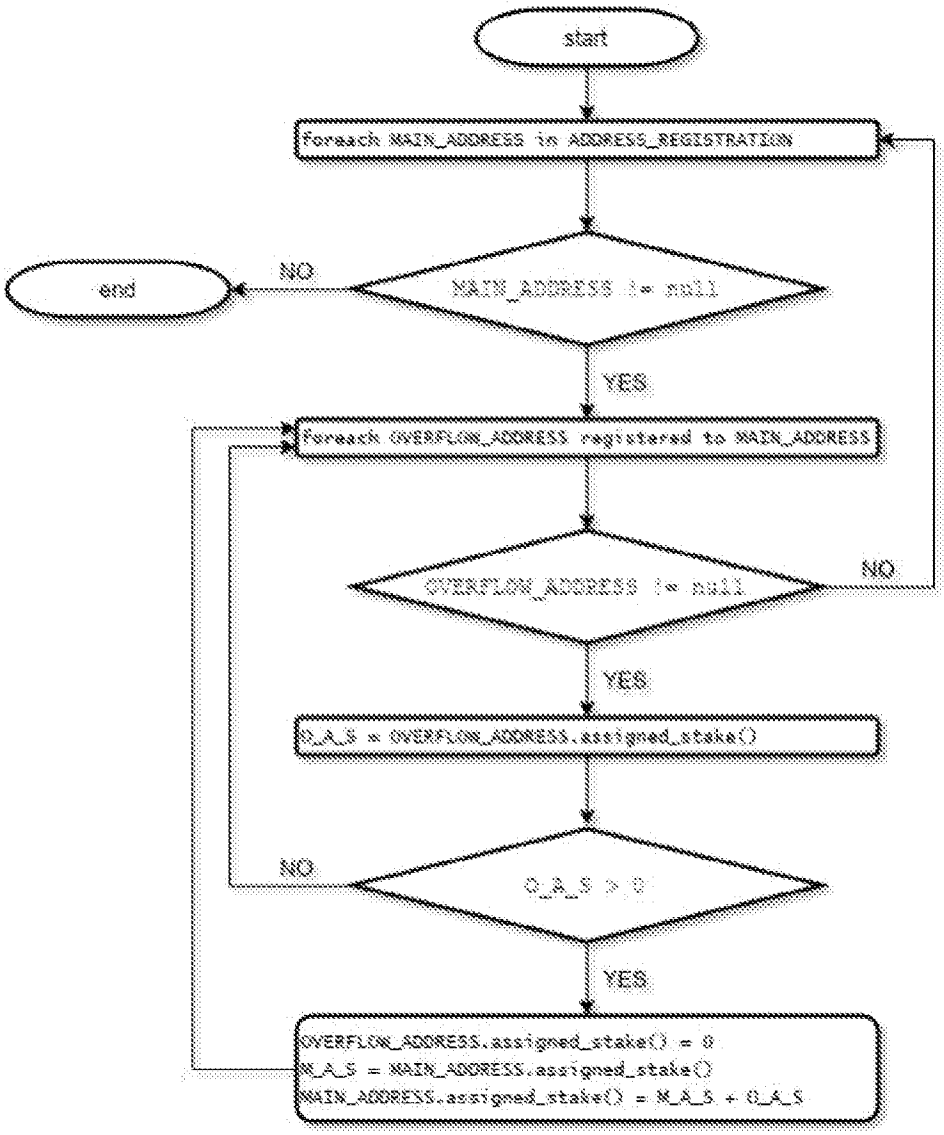

Fig. 5

	INITIAL STATE							
	not miner	full pay						
MAIN	S_min_VAL	S_min_VAL	S_min_VAL	S_min_VAL	S_min_VAL	S_min_VAL	S_min_VAL	RND
OVERFLOW	0							
OVERFLOW	0							
OVERFLOW	0							

Fig. 6

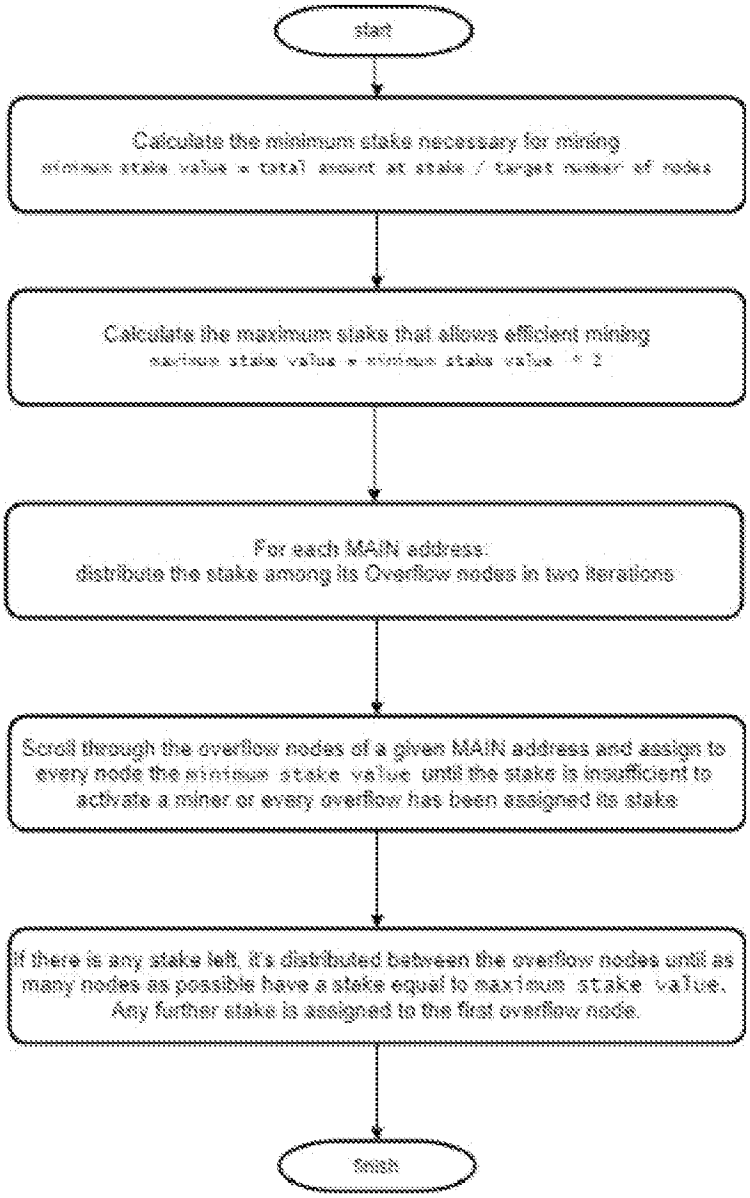
start
Calculate the minimum stake necessary for mining
minimum stake value = total amount at stake / target number of nodes
Calculate the maximum stake that allows efficient mining
maximum stake value = minimum stake value * 2
For each MAIN address:
distribute the stake among its Overflow nodes in two iterations
Scroll through the overflow nodes of a given MAIN address and assign to every node the minimum stake value until the stake is insufficient to activate a miner or every overflow has been assigned its stake
If there is any stake left, it's distributed between the overflow nodes until as many nodes as possible have a stake equal to maximum stake value. Any further stake is assigned to the first overflow node.
finish

Fig. 7

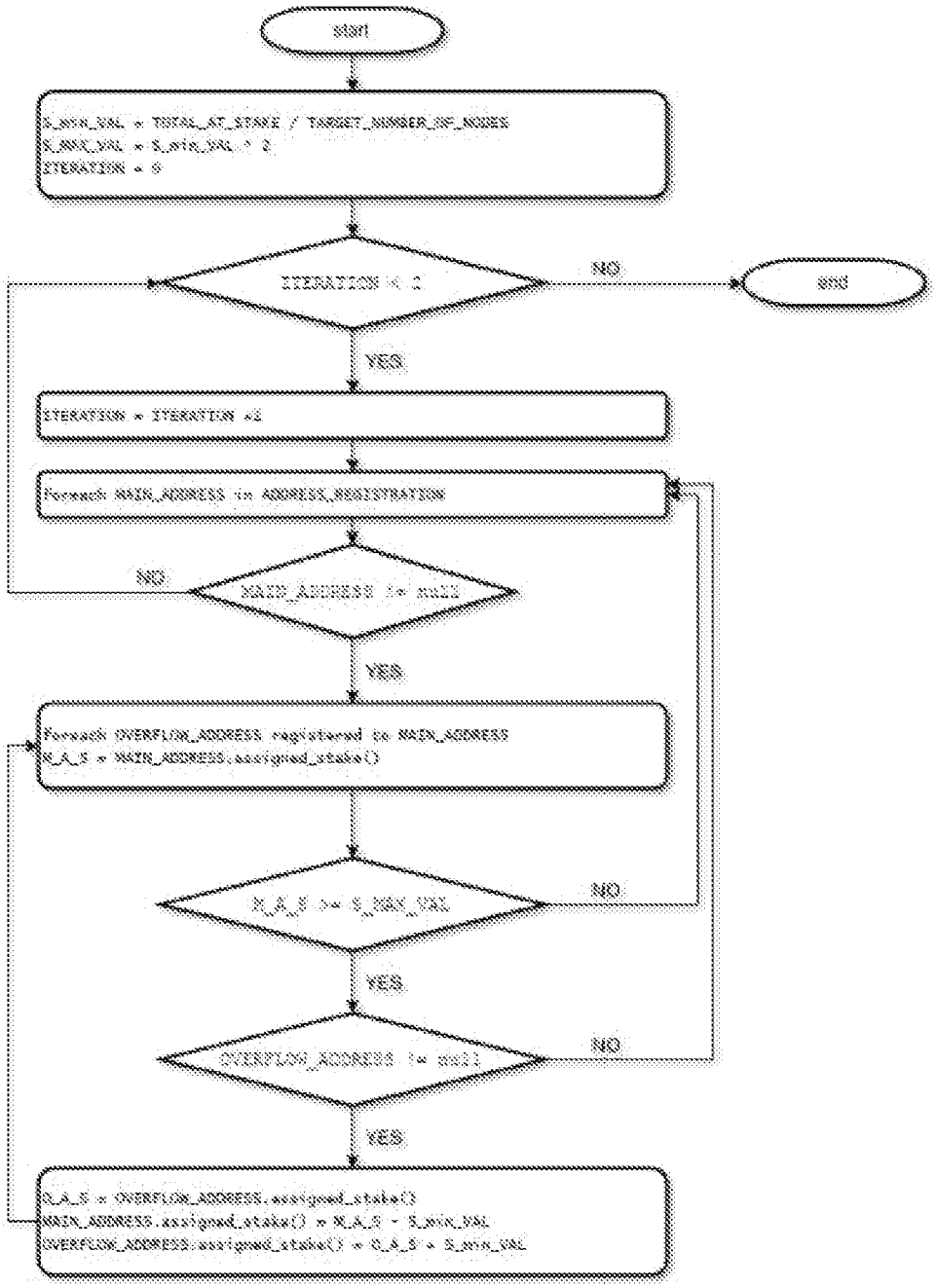
start
S_min_VAL = TOTAL_AT_STAKE / TARGET_NUMBER_OF_NODES
S_max_VAL = S_min_VAL * 2
ITERATION = 0
ITERATION < 2
NO
end
YES
ITERATION = ITERATION +1
Foreach MAIN_ADDRESS in ADDRESS_REGISTRATION
NO
MAIN_ADDRESS != null
YES
Foreach OVERFLOW_ADDRESS registered to MAIN_ADDRESS
O_A_S = MAIN_ADDRESS.assigned_stake()
O_A_S >= S_max_VAL
NO
YES
OVERFLOW_ADDRESS != null
NO
YES
O_A_S = OVERFLOW_ADDRESS.assigned_stake()
MAIN_ADDRESS.assigned_stake() = M_A_S - S_min_VAL
OVERFLOW_ADDRESS.assigned_stake() = O_A_S + S_min_VAL

Fig. 8

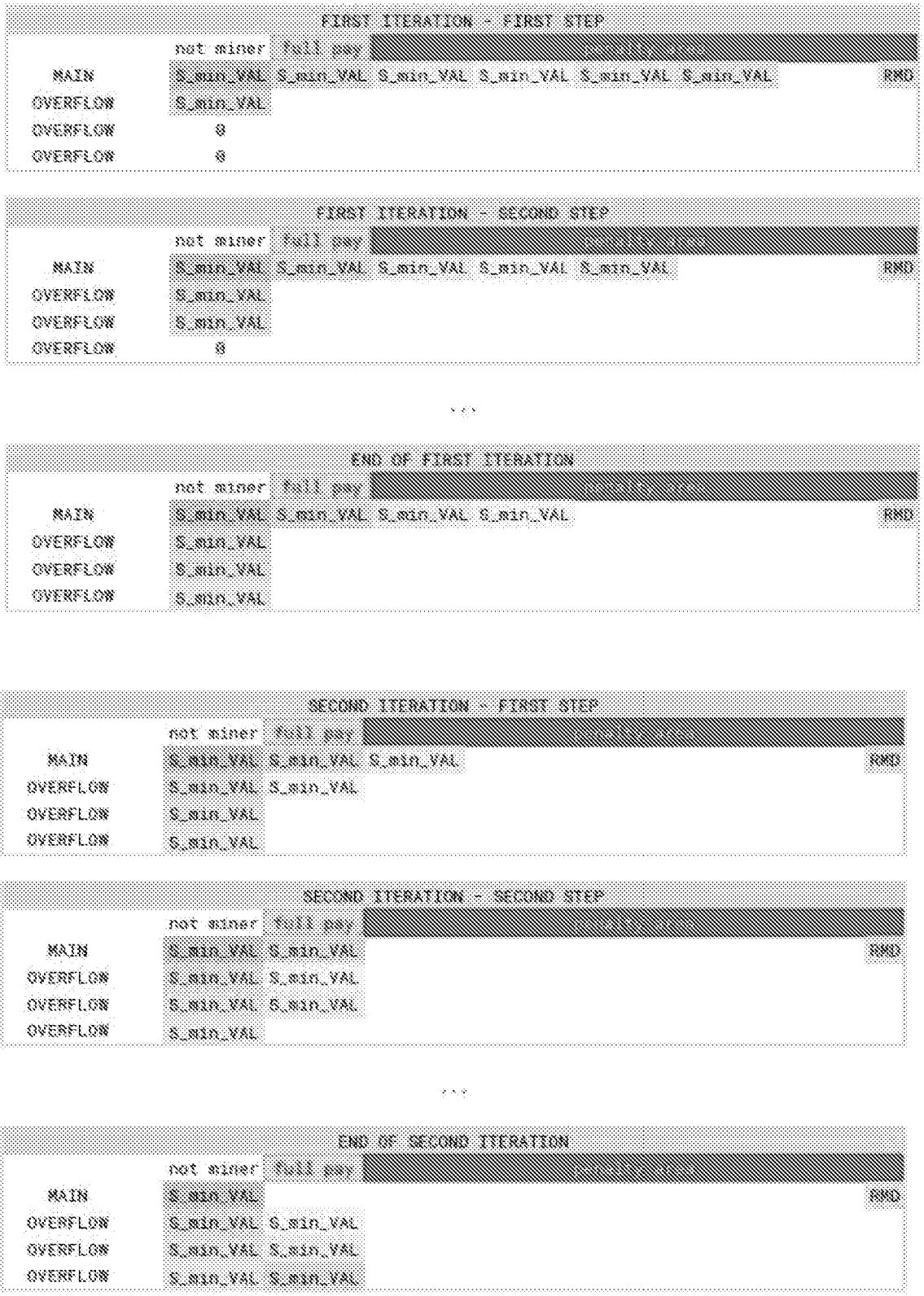
FIRST ITERATION - FIRST STEP
not miner full gas
MAIN S_min_VAL S_min_VAL S_min_VAL S_min_VAL S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL
OVERFLOW 0
OVERFLOW 0
FIRST ITERATION - SECOND STEP
not miner full gas
MAIN S_min_VAL S_min_VAL S_min_VAL S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL
OVERFLOW S_min_VAL
OVERFLOW 0
. . .
END OF FIRST ITERATION
not miner full gas
MAIN S_min_VAL S_min_VAL S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL
OVERFLOW S_min_VAL
OVERFLOW S_min_VAL
SECOND ITERATION - FIRST STEP
not miner full gas
MAIN S_min_VAL S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL
OVERFLOW S_min_VAL
SECOND ITERATION - SECOND STEP
not miner full gas
MAIN S_min_VAL S_min_VAL RND
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL
. . .
END OF SECOND ITERATION
not miner full gas
MAIN S_min_VAL RND
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL S_min_VAL
OVERFLOW S_min_VAL S_min_VAL

Fig. 9

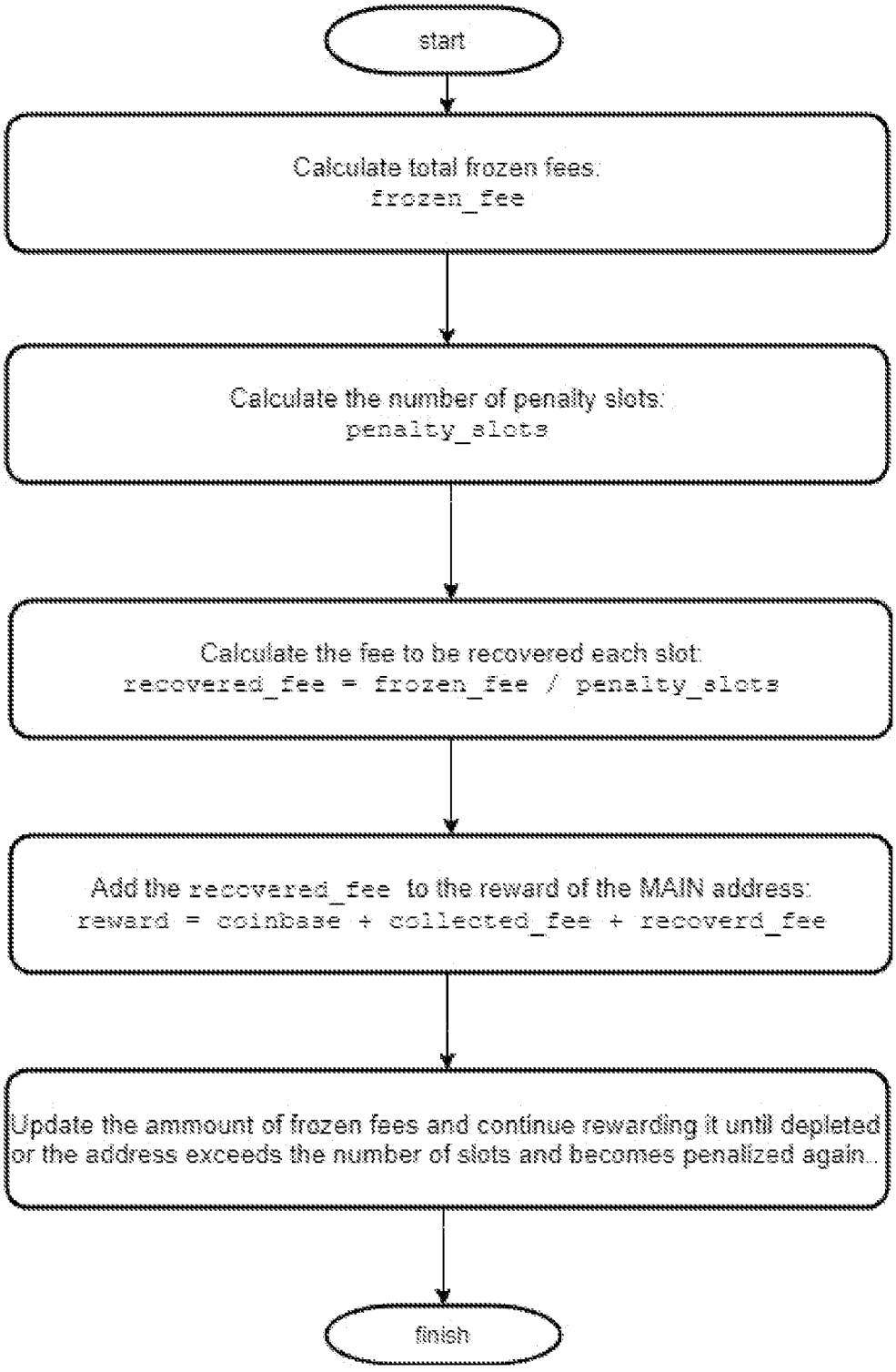
start
Calculate total frozen fees:
frozen_fee
Calculate the number of penalty slots:
penalty_slots
Calculate the fee to be recovered each slot:
recovered_fee = frozen_fee / penalty_slots
Add the recovered_fee to the reward of the MAIN address:
reward = coinbase + collected_fee + recoverd_fee
Update the ammount of frozen fees and continue rewarding it until depleted or the address exceeds the number of slots and becomes penalized again..
finish

Fig. 10

N1	ID1	2
N2	ID2	1
N3	ID3	0
N4	ID4	3
N5	ID5	0

Fig. 11

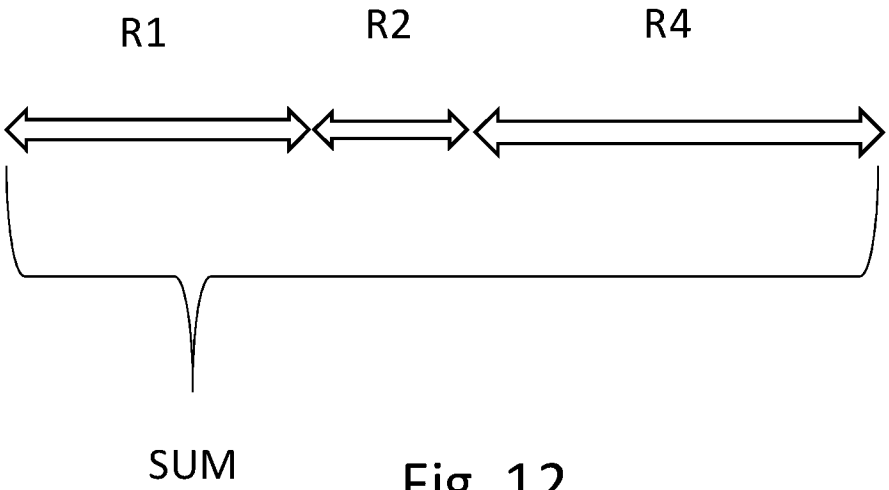

Fig. 12

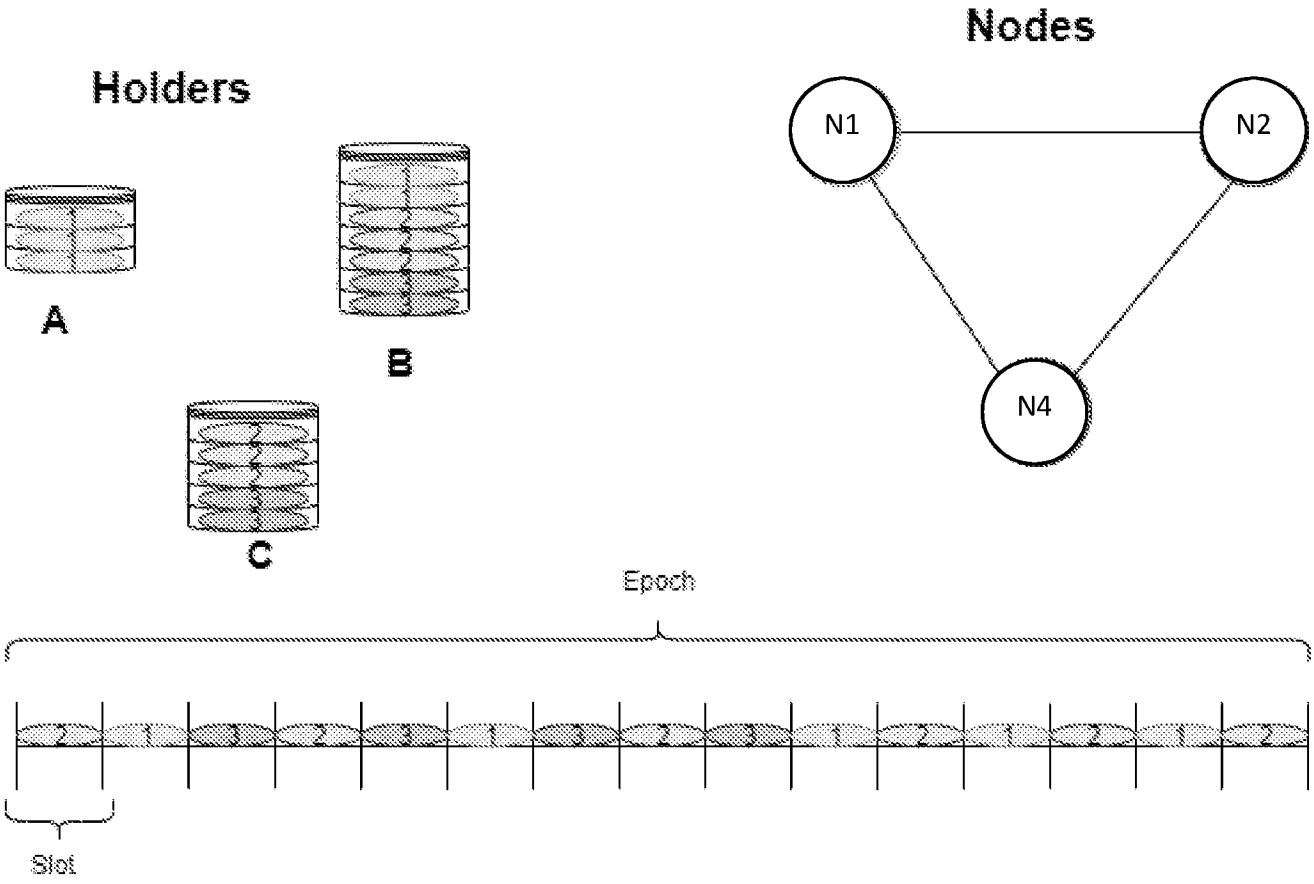

Fig. 13

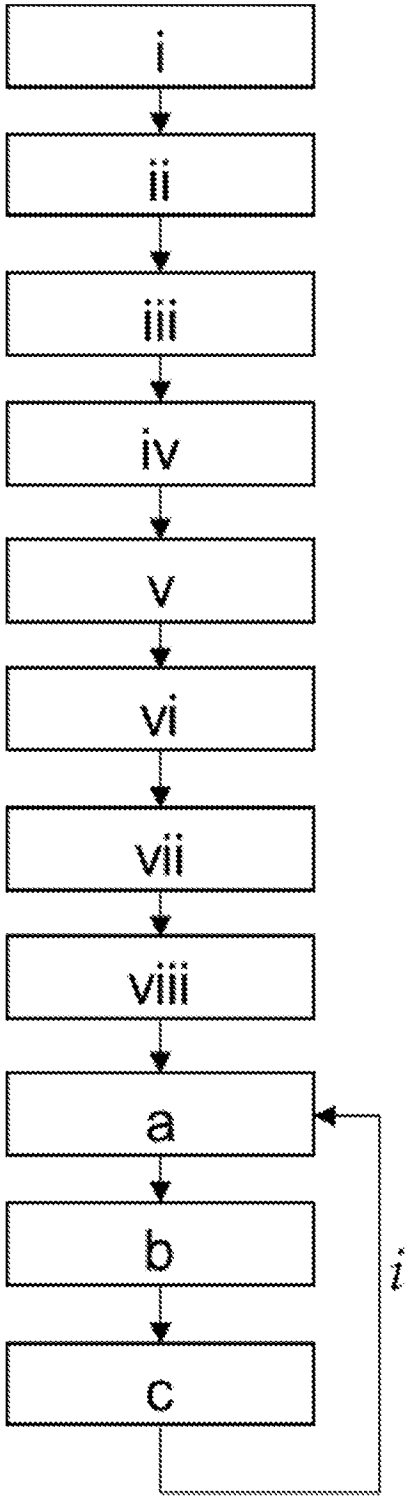

Fig. 14