

Operation of a distributed deterministic network

Technical Field

5

The present invention pertains to a method for operating a distributed network, the distributed network comprising a plurality of subnets. Each subnet comprises a plurality of nodes.

10

Further aspects relate to a corresponding distributed network, a node of a distributed network, a corresponding computer program product and a software architecture encoded on a non-transitory medium.

15

Background Art

In distributed networks a plurality of nodes are arranged in a distributed fashion. In distributed networks computing, software and data are spread out across the plurality of nodes. The nodes establish computing resources and the distributed networks may use distributed computing techniques.

An example of distributed networks are blockchain networks. Blockchain networks are consensus-based, electronic ledgers based on blocks. Each block comprises transactions and other information. Furthermore, each block contains a hash of the previous block so that blocks become chained together to create a permanent, unalterable record of all transactions which have been written to the blockchain. Transactions may contain small programs known e.g. as smart contracts.

In order for a transaction to be written to the blockchain, it must be "validated" by the network. In other words, the network nodes have to gain consent on blocks to be written to the blockchain. Such consent may be achieved by various consensus protocols.

One type of consensus protocols are proof-of-work consensus protocols. A proof-of-work consensus protocol generally requires some work from the parties that participate in the consensus protocol, usually corresponding to processing time by a computer. Proof-of-work-based cryptocurrency systems such as Bitcoin involve the solving of computationally intensive puzzles to validate transactions and to create new blocks.

Another type of consensus protocols are proof-of-stake-consensus protocols. Such proof-of-stake protocols have the advantage that they do not require time-consuming and energy-intensive computing. In proof-of-stake based blockchain networks e.g. the creator of the next block is chosen via combinations of random selection as well as the stake of the respective node in the network.

Apart from cryptocurrencies, distributed networks may be used for various other applications. In particular, they may be used for providing decentralized and distributed computing capabilities and services.

Accordingly, there is a need for distributed networks with enhanced functionalities.

Disclosure of the Invention

One object of an aspect of the invention is to provide a method for operating a distributed network with enhanced functionalities, in particular in terms of security, efficiency and/or fairness.

According to an embodiment of a first aspect of the invention, a computer-implemented method for operating a distributed network is provided. The distributed network comprises a plurality of nodes and one or more subnets. The method comprises steps of running a set of computational units and assigning each of the computational units to one of the one or more subnets

according to a subnet-assignment, thereby creating an assigned subset of the set of computational units for each of the subnets. The method further comprises running on each node of the one or more subnets the assigned subset
5 of the computational units, wherein the computational units are configured to execute computations in a deterministic manner, thereby replicating the assigned subsets of the computational units on replicas across the respective subnet. The method further comprises running a consensus
10 component, the consensus component being configured to reach a consensus on input blocks to be processed by the replicas of a respective subnet. The method further comprises measuring, by each of the replicas of a subnet, replica-local data and reporting, by each of the replicas
15 of a subnet, the replica-local data to the consensus component of the subnet.

Such an embodied method provides advantageous network operations. In particular, such a method allows to measure replica-local data by the nodes of the subnet and
20 to use this replica-local data for an analysis of the computations executed across the respective subnet.

According to embodiments, replica-local data may be defined as any local data of a replica, in particular any data on which the replicas of a respective subnet do
25 not have reached a consensus on, i.e. which has not run through the consensus component of the subnet.

According to such an embodied method the replicas of the subnet itself are used to measure the replica-local data and report these data to the consensus
30 component. Then the replica-local data may be used for an analysis of the subnet.

This method can be used to observe the replicas in order to detect malicious, poor or passive behaviour, e.g. a freeriding of replicas. Free-riding may be described
35 as optimistically following, but short-cutting, the protocol with the goal of saving resources and/or expenses. It is not actively malicious.

Furthermore, such an embodied method may be used to observe the computational units running on the replicas in order to detect below-average performance of certain computational units. The below-average performance
5 can be caused by poorly programmed or maliciously programmed computational units or by maliciously crafted input from a user to the computational units. In particular, such an embodied method may be used to detect and eliminate resource exhaustion attacks by computational
10 units or users.

In addition, methods according to embodiments of the invention may be used to increase fairness in charging the computational units for the use of network resources despite non-deterministic effects.

15 According to embodiments the replica-local data is data pertaining to the computations being locally executed by the corresponding replica.

While the results of the computations, namely the execution states of the computational units, are the same on all replicas, the efficiency and performance, e.g. in terms of the use of processing resources of the nodes such as memory bus and caches, may differ from replica to replica. Hence according to an embodiment the replica-local data may be in particular data pertaining to the
25 computations being locally executed by the corresponding replica. This data may encompass in particular non-deterministic data, in particular non-deterministic measurands of the executed computations.

According to embodiments the replica-local
30 data is data comprising numeric or enumerable quantities. Such numeric or enumerable data includes boolean data.

Measuring numeric or enumerable data facilitates an efficient analysis of the data. More particularly, it allows an efficient comparison of the data
35 and further facilitates an efficient implementation of an action logic to take appropriate actions to improve the operation of the subnet.

For example, one measurand may pertain to the wall clock time that executions for a given computational unit take, measured by the replica. More precisely, the replica may measure the ratio of the replica-local wall
5 clock time to the logical time. Here, the logical time required for a given execution is deterministic and the same on all replicas, whereas the wall-clock time is non-deterministic and depends on replica-local effects such as the use of the memory bus and the state of caches on the
10 replica.

According to an embodiment, the method further comprises utilizing the replica-local data for an analysis, in particular a statistical analysis. According to
15 embodiments the analysis is performed of computations executed across the respective subnet.

According to embodiments the method further comprises taking one or more actions in dependence on the analysis of the replica-local data. Such actions may be generally any measures to improve the operation of the
20 subnet. The actions may be performed according to embodiments by a system computational unit of the subnet. Such a system computational unit receives the reported replica-local data, preferably in an analysed form, and may decide in dependence on this data on the actions to be
25 taken.

According to embodiments, the actions may comprise e.g. adjusting bounds and/or prices for one or more of the computational units of the subnet. Such bounds may be e.g. the logical time allocated to the computational
30 units. The price may be a price which is charged to the computational units by the network for the use of processing and network resources of the network. Such price may be set in a local currency of the network and is sometimes referred to as gas.

35 In the same or a similar manner the subnet may also adjust bounds and/or prices for users of the

distributed network, in particular for users that uses the computational units which are run on the respective subnet.

Furthermore, according to embodiments misbehaving or poorly behaving replicas may be banned or
5 in other words eliminated from the network.

According to an embodiment, the method may comprise a step of computing, by the respective subnet, a set of geometric median values of the replica-local data across the subnet.

10 More particularly, each of the replicas of the subnet may provide the replica-local data and then the median value of the respective replica local-data is computed across the subnet. Taking the median value has the advantage that malicious or misbehaving replicas cannot
15 influence the final value that is used for the subsequent analysis of the subnet operation. In this respect embodiments of the invention provide the advantage that the final value cannot be skewed so much by providing extremely large or small values by a malicious node or a
20 malicious computational unit.

According to an embodiment, a statistical analysis of the set of median values may then be performed and one or more actions may be taken in dependence on the result of the statistical analysis.

25 According to an embodiment, the statistical analysis may involve the application of methods configured to ignore outliers, in particular a method that comprises the computation of a geometric median.

According to an embodiment the statistical
30 analysis may be performed robustly in the sense that its outcome is not influenced by the outliers of reported replica-local measurands. A statistical analysis that starts with taking median values is one exemplary embodiment of a robust statistical analysis.

35 According to an embodiment, it is assumed/expected that up to f malicious replicas are present in a subnet. The statistical analysis may then only

proceed if at least $2f+1$ reports from different replicas are present. The statistical analysis may then discard the f lowest and f highest values and only take into account the remaining values.

5 According to embodiments the statistical analysis may involve computing a deviation of the set of median values from an expected value and taking the one or more actions in dependence on the deviation. As an example, computational units which have a resource consumption that
10 deviates more than a predefined percentage from the expected value may be subject to an action such as a reduction of the allocated logical time.

 In addition to an analysis of the median values, a statistical analysis of further data may be
15 performed. The further data may be in particular data which is publicly verifiable blockchain data, i.e. data that is available to everyone and which hence does not need to be reported (anymore) to the consensus component. According to other embodiments the further data may come from a
20 trusted source or trusted provider.

 According to an embodiment the components which are responsible for the analysis and the action may take the one or more actions in dependence on the statistical analysis of the replica-local data, the set of median
25 values and/or the further data.

 According to an embodiment, the subnet may provide a time window for the reporting of the replica-local data by the replicas of the subnet. Such a time window may be predefined by the subnet and implemented e.g.
30 as code in the subnet client. As an example, the time window may be specified e.g. in terms of a predefined number of input blocks. Such a time window facilitates a regular analysis of the behaviour of the subnet components.

 According to an embodiment, the replica-local
35 data may be data on the resource consumption of the computational units on the replica, in particular the wall clock time of execution and/or the maximum memory allocated

during execution. This is non-deterministic data which may differ from replica to replica.

According to an embodiment, the replica-local data may be data on the resource consumption of users on the replica, in particular the logical time and/or logical time of execution performed on behalf of the user and/or the maximum memory allocated during execution performed on behalf of the user.

According to an embodiment, the replica-local data may be data on measurements of other replicas that the replica communicates with. A replica that the replica communicates with may be on the same subnet or on a different subnet. Furthermore, it may mean that the two replicas have established a permanent communication link, e.g. according to an overlay topology, or it may mean that the two replicas communicate only sporadically over a non-permanent communication link. Furthermore, it may refer to a case in which the two replicas are part of a gossip network.

According to an embodiment, for each other replica that the replica communicates with, the replica-local data may be data

on measurements of first parameters, in particular on communication parameters of the other replica with the replica, in particular latency, response time, down time and/or reliability. Such communication parameters may be also denoted as low-level communication parameters.

According to an embodiment, for each other replica that the replica communicates with, the replica-local data may be data on measurements of second parameters, in particular on semantic communication parameters of the other replica with the replica, such as correctness according to a protocol specification, e.g. a subnet protocol specification. Such communication parameters may be denoted as high-level communication parameters.

According to an embodiment, for each other replica on the subnet, the replica-local data may be data on measurements about the activity of the other replica in the consensus protocol as observed by the replica, such as
5 the number of proposals of input blocks and/or the number of signature shares produced by the other replica. In this embodiment it is not necessary that the replica has directly communicated with the other replica on which it took the measurement.

10 According to an embodiment, the further data may be data from a trusted source, in particular a blockchain. According to such an embodiment, the further data may be the number of blocks in a given section of that blockchain that were proposed by each replica. Blocks that
15 are part of an agreed upon blockchain may be called finalized blocks. Moreover, according to an embodiment, the further data may be the number of times that each replica has participated in a multi-signature for all multi-signatures that appear in a block of a given section
20 of the blockchain.

According to embodiments, the replicas of the subnet may process non-replicated executions on behalf of users. Such non-replicated executions may not be intended to go through consensus, run on any other replica or alter
25 the replicated state. Such non-replicated executions may be called query calls. In such an embodiment the replica-local data may be data on the resource consumption of query calls for each computational unit or for each user. Such an embodied method provides an advantageous way to charge
30 computational units and/or users for query charges, e.g. in the form of gas. In particular, a computational unit may be charged an amount proportional to the median of the gas costs that it incurred across all replicas.

In this respect, gas may be defined as a
35 complementary accounting unit that is used by the distributed network to charge for the use of its resources and is separate from the native currency of the network.

Gas may also be denoted as fuel or cycles. Providing a complementary accounting unit provides the advantage that the costs for the use of the resources can be kept stable and independent from market fluctuations of the native
5 currency. As an analogy to a car, where fuel or gas is needed to run a car, the gas according to embodiments of the invention is needed to run the resources of the distributed network.

According to embodiments, the subnets may also
10 be denoted as replicated computing clusters. In such replicated computing cluster the computational units that have been assigned to a respective subnet are run on each node of the subnet and are hence replicated across the subnet, thereby traversing the same chain of execution
15 states.

According to an embodiment of another aspect of the invention, a distributed network is provided which is configured to perform the method steps of the method aspects of the invention.

20 According to an embodiment of another aspect of the invention, a node of a distributed network is provided which is configured to perform and/or to participate in corresponding method steps of the method aspects of the invention.

25 According to an embodiment of another aspect of the invention, a computer program product for operating a distributed network is provided. The computer program product comprises a computer readable storage medium having program instructions embodied therewith, the program
30 instructions executable by one or more of a plurality of nodes of the distributed network to cause the one or more of the plurality of nodes to perform steps of the method aspects of the invention.

According to an embodiment of another aspect
35 of the invention, a computer program product for operating a node of a distributed network is provided.

According to an embodiment of another aspect of the invention, a software architecture encoded on a non-transitory computer readable medium is provided. The software architecture is configured to operate one or more nodes of a distributed network. The encoded software architecture comprises program instructions executable by one or more of the plurality of nodes to cause the one or more of the plurality of nodes to perform a method comprising steps of the method aspects of the invention.

Features and advantages of one aspect of the invention may be applied to the other aspects of the invention as appropriate.

Other advantageous embodiments are listed in the dependent claims as well as in the description below.

15

Brief Description of the Drawings

The invention will be better understood and objects other than those set forth above will become apparent from the following detailed description thereof. Such description makes reference to the annexed drawings, wherein:

FIG. 1 shows an exemplary block diagram of a distributed network according to an embodiment of the invention;

FIG. 2 illustrates in a more detailed way computational units running on a node of the network and forming a replica of a subnet;

FIG. 3 illustrates steps of a computer-implemented method according to an embodiment of the invention;

FIG. 4 illustrates a message/data flow in a distributed network according to an embodiment of the invention;

FIG. 5 illustrates main processes which are run on each node of the network according to an embodiment of the invention;

FIG. 6 shows a schematic block diagram of protocol components of a subnet protocol client;

Fig. 7 shows an exemplary visualization of a workflow of a messaging protocol and a consensus protocol
5 and the associated components;

FIG. 8 shows a layer model illustrating main layers which are involved in the exchange of inter-subnet and intra-subnet messages;

FIG. 9 illustrates the creation of input blocks
10 by a consensus component according to an exemplary embodiment of the invention;

FIG. 10 shows a more detailed illustration of a computational unit;

FIG. 11 shows a flow chart comprising method
15 steps of a computer-implemented method for running a distributed network;

FIG. 12 shows a flow chart comprising further method steps of a computer-implemented method for operating a distributed network; and

20 FIG. 13 shows an exemplary embodiment of a node according to an embodiment of the invention.

Modes for Carrying Out the Invention

25 At first, some general aspects and terms of embodiments of the invention will be introduced.

According to embodiments, a distributed network comprises a plurality of nodes that are arranged in a distributed fashion. In such a distributed network
30 computing, software and data is distributed across the plurality of nodes. The nodes establish computing resources and the distributed network may use in particular distributed computing techniques.

According to embodiments, distributed networks
35 may be in particular embodied as blockchain networks. The term "blockchain" shall include all forms of electronic, computer-based, distributed ledgers. According to some

embodiments, the blockchain network may be embodied as proof-of-work blockchain network. According to other embodiments, the blockchain network may be embodied as proof-of-stake blockchain network.

5 A computational unit may be defined as a piece of software that is running on a node of the distributed network and which has its own unit state. The unit state may also be denoted as execution state.

Each of the subnets is configured to replicate
10 the set of computational units, in particular the states of the computational units, across the subnet. As a result, the computational units of a respective subnet traverse always the same chain of unit states/execution states, provided they behave honestly. A computational unit
15 comprises the code of the computational unit and the unit state/execution state of the computational unit.

An artefact may be any information that is exchanged between the nodes of a subnet.

A replica is formed by a set of computational
20 units that runs on a node and are assigned to the same subnet.

A messaging protocol may be defined as a protocol that manages the exchange of messages within the network, in particular of unit-to-unit messages. In
25 particular, the messaging protocol may be configured to route the unit-to-unit messages from a sending subnet to a receiving subnet. For this, the messaging protocol uses the respective subnet-assignment. The subnet-assignment indicates to the messaging protocol the respective
30 location/subnet of the computational units of the respective communication.

According to embodiments, a unit state shall be understood as all the data or information that is used by the computational unit, in particular the data that the
35 computational unit stores in variables, but also data the computational units get from remote calls. The unit state may represent in particular storage locations in the

respective memory locations of the respective node. The contents of these memory locations, at any given point in the execution of the computational units, is called the unit state according to embodiments. The computational units may be in particular embodied as stateful computational units, i.e. the computational units are designed according to embodiments to remember preceding events or user interactions.

FIG. 1 shows an exemplary block diagram of a distributed network 100 according to an embodiment of the invention.

The distributed network 100 comprises a plurality of nodes 10, which may also be denoted as network nodes 10. The plurality of nodes 10 are distributed over a plurality of subnets 11. In the example of FIG. 1, four subnets 11 denoted with SNA, SNB, SNC and SND are provided.

Each of the plurality of subnets 11 is configured to run a set of computational units on each node 10 of the respective subnet 11. According to embodiments a computational unit shall be understood as a piece of software, in particular as a piece of software that comprises or has its own unit state or in other words execution state.

The network 100 comprises communication links 12 for intra-subnet communication within the respective subnet 11, in particular for intra-subnet unit-to-unit messages to be exchanged between computational units assigned to the same subnet and for intra-subnet messages comprising replica-local data for reporting.

Furthermore, the network 100 comprises communication links 13 for inter-subnet communication between different ones of the subnets 11, in particular for inter-subnet unit-to-unit messages to be exchanged between computational units assigned to different subnets.

Accordingly, the communication links 12 may also be denoted as intra-subnet or Peer-to-Peer (P2P) communications links and the communication links 13 may

also be denoted as inter-subnet or Subnet-to-Subnet (SN2SN) communications links.

According to embodiments of the invention the subnets 11 are configured to replicate the set of computational units across the respective subnet 11. More particularly, the subnets 11 are configured to replicate the unit state of the computational units across the respective subnet 11. The set of computational units of a subnet that runs on a node forms a replica of the subnet.

The network 100 may be in particular a proof-of-stake blockchain network.

Proof-of-stake (PoS) describes a method by which a blockchain network reaches distributed consensus about which node is allowed to create the next block of the blockchain. PoS-methods may use a weighted random selection, whereby the weights of the individual nodes may be determined in particular in dependence on the assets (the "stake") of the respective node.

FIG. 2 illustrates in a more detailed way computational units 15 running on nodes 10 of the network 100. The network 100 is configured to assign each of the computational units which are running on the network 100 to one of the plurality of subnets, in this example to one of the subnets SNA, SNB, SNC or SND according to a subnet-assignment. The subnet-assignment of the distributed network 100 creates an assigned subset of the whole set of computational units for each of the subnets SNA, SNB, SNC and SND.

More particularly, FIG. 2 shows a node 10 of the subnet SNA of FIG. 1. The subnet assignment of the distributed network 100 has assigned a subset of four computational units 15 to the subnet SNA more particularly the subset of computational units CU_{A1} , CU_{A2} , CU_{A3} and CU_{A4} . The assigned subset of computational units CU_{A1} , CU_{A2} , CU_{A3} and CU_{A4} runs on each node 10 of the subnet SNA. Furthermore, the assigned subset of computational units CU_{A1} , CU_{A2} , CU_{A3} and CU_{A4} is replicated across the whole subnet SNA such that

each of the computational units CU_{A1} , CU_{A2} , CU_{A3} and CU_{A4} traverses the same chain of unit states and forms on each node 10 of the subnet a replica of the subnet. This may be implemented in particular by performing an active replication in space of the unit state of the computational units CU_{A1} , CU_{A2} , CU_{A3} and CU_{A4} on each of the nodes 10 of the subnet SNA. The computational units CU_{A1} , CU_{A2} , CU_{A3} and CU_{A4} are embodied as user computational units. In addition, a subnet may also comprise system computational units which are not shown in FIG. 2.

As shown in FIG. 1, the distributed network 100 comprises a central control unit CCU, 20. The central control unit 20 may comprise a central registry 21 to provide network control information to the nodes of the network.

According to embodiments, the distributed network may be in particular configured to exchange inter-subnet messages 16 between the subnets SNA, SNB, SNC and SND via a messaging protocol. The inter-subnet messages 16 may be in particular embodied as inter-subnet unit-to-unit messages 16a to be exchanged between computational units that have been assigned to different subnets according to the subnet-assignment. As an example, the distributed network 100 may be configured to exchange a unit-to-unit message M1, 16a between the computational unit CU_{A1} as sending computational unit running on the subnet SNA and the computational unit CU_{B2} as receiving computational unit running on the subnet SNB. In addition, the inter-subnet messages 16 may be embodied as signalling messages 16b. The signalling messages 16b may encompass acknowledgement messages (ACK) adapted to acknowledge an acceptance or receipt of the unit-to-unit messages or non-acknowledgement messages (NACK) adapted to not-acknowledge an acceptance (corresponding to a rejection) of the unit-to-unit messages, e.g. to indicate a transmission failure.

The network 100 may be in particular configured to store the subnet-assignment of the computational units

10 as network configuration data. This information may also be stored in the central registry.

According to further embodiments, the network 100 may be configured to exchange the inter-subnet messages 16 via a messaging protocol and a consensus protocol. The consensus protocol may be configured to reach a consensus on the selection and/or processing order of the inter-subnet messages 16 at the respective receiving subnet.

Referring e.g. to the subnet SNB, it receives inter-subnet messages 16 from the subnets SNA, SNC and SND. The consensus protocol receives and processes these inter-subnet messages 16 and performs a predefined consensus algorithm or consensus mechanism to reach a consensus on the selection and/or processing order of the received inter-subnet messages 16.

Referring now to FIG. 3, a computer-implemented method for operating a distributed network according to embodiments of the invention will be explained.

According to this embodiment it is assumed that the subnet SNA comprises three nodes. On each of the three nodes there are run replicas 1, 2 and 3 respectively, wherein each of the replicas 1, 2 and 3 comprises the set of computational units CU_{A1} , CU_{A2} , CU_{A3} and CU_{A4} . Each of the replicas 1, 2 and 3 measures during its operation replica-local data and reports this replica-local data to a consensus component 63. The consensus component 63 is configured to reach a consensus on input blocks which are later on to be processed by the replicas 1, 2 and 3 of the subnet SNA. Apart from the replica-local data the consensus component 63 may receive e.g. messages from users or other subnets which will be explained in more detail further below. The subnet SNA utilizes then the replica-local data for an analysis of the computations which have been executed by the computational units CU_{A1} , CU_{A2} , CU_{A3} and CU_{A4} across the subnet. The analysis may be done by a statistical analysis component 62a. Then the subnet may perform or initiate one or more actions based on the

analysis. The actions may be performed on a replica level or on a subnet level. The actions may be initiated by an action component 62b.

Referring now to FIG. 4, a message/data flow in a distributed network according to an embodiment of the invention is illustrated. More particularly, FIG. 4 illustrates how input blocks are created which are then processed by the replicas of a subnet.

The illustration is again done with respect to the subnet SNA which comprises the three replicas 1, 2 and 3 on which the 4 computational units CU_{A1} , CU_{A2} , CU_{A3} and CU_{A4} are run.

Each of the replicas 1, 2 and 3 execute the same sequence of input blocks. The input blocks are created by the consensus component 63. The consensus component 63 receives a plurality of input data from a plurality of sources.

At first, the consensus component 63 receives data from its own subnet SNA. The own subnet SNA comprises a reporting component 41 and a messaging component 61. The reporting component 41 reports the replica-local data which has been measured by the respective replicas 1, 2 or 3 of the subnet SNA to the consensus component 63. In addition, the messaging component 61 of the subnet SNA provides intra-subnet messages denoted as SN-data sent by one of the computational units CU_{A1} , CU_{A2} , CU_{A3} or CU_{A4} to one of the other computational unit CU_{A1} , CU_{A2} , CU_{A3} or CU_{A4} .

It should be noted that the reporting component 41 is per replica, whereas the messaging component 61 is per subnet, at least logically. While the messaging component 61 is deterministic, the reporting component 41 may produce different output on each replica.

Furthermore, the consensus component 63 receives further SN-data from messaging components 61 of the other subnets SNB, SNC and SND.

In addition, the consensus component 63 receives user data from users of the distributed network.

Such incoming user data may also be denoted as ingress messages.

FIG. 5 illustrates main processes which may be run on each node 10 of the network 100 according to an embodiment of the invention. A network client of networks according to embodiments of the invention is the set of protocol components that are necessary for a node 10 to participate in the network. According to embodiments, each node 10 is a member of a mainnet. Furthermore, each node may be a member of one or more subnets.

A node manager 50 is configured to start, re-start and update a mainnet protocol client 51, a subnet protocol client 52 and a security application 53. According to other embodiments, the central control unit 20 may be used instead of the mainnet protocol client (see FIG. 1). According to embodiments several subnet protocol clients may be used, thereby implementing several replicas.

According to embodiments, each of the plurality of subnets 11 is configured to run a separate subnet protocol client 52 on its corresponding nodes 10. The mainnet protocol client 51 is in particular configured to distribute configuration data to and between the plurality of subnets 11. The mainnet protocol client 51 may be in particular configured to run only system computational units, but not any user-provided computational units. The mainnet protocol client 51 is the local client of the mainnet and the subnet protocol client 52 is the local client of the subnet. The subnet protocol client 52 comprises a registry 52a and the mainnet protocol client a registry 51a.

The security application 53 stores secret keys of the nodes 10 and performs corresponding operations with them.

The node manager 50 may monitor e.g. the registry 21 of the control unit 20, it may instruct the nodes to participate in a subnet, and/or it may instruct the nodes to stop participation in a subnet.

FIG. 6 shows a schematic block diagram of protocol components 600 of a subnet protocol client, e.g. of the subnet protocol client 52 of FIG. 5.

5 The protocol components 600 comprise a messaging component 61 which is configured to run a messaging protocol and an execution component 62 configured to run an execution protocol for executing execution messages, in particular for executing unit-to-unit
10 messages and/or ingress messages. The protocol components 600 further comprise a consensus component 63 configured to run a consensus protocol, a networking component 64 configured to run a networking protocol, a state manager component 65 configured to run a state manager protocol,
15 an X-Net component 66 configured to run a cross-subnet transfer protocol and an ingress message handler component 67 configured to handle ingress message received from an external user of the network. The protocol components 600 comprise in addition a crypto-component 68. The crypto-
20 component 68 co-operates with a security component 611, which may be e.g. embodied as the security application 53 as described with reference to FIG. 5. Furthermore, the subnet-protocol client 52 may cooperate with a reader component 610, which may be a part of the mainnet protocol
25 client 51 as described with reference to FIG. 5. The reader component 610 may provide information that is stored and distributed by the mainnet to the respective subnet protocol client 52. This includes the assignment of nodes to subnets, node public keys, assignment of computational
30 units to subnets etc.

 The messaging component 61 and the execution component 62 are configured such that all computation, data and state in these components is identically replicated across all nodes of the respective subnet, more
35 particularly all honest nodes of the respective subnet. This is indicated by the wave-pattern background of these components.

Such an identical replication is achieved according to embodiments on the one hand by virtue of the consensus component 63 that ensures that the stream of inputs to the messaging component 61 is agreed upon by the
5 respective subnet and thus identical for all nodes, more particularly by all honest nodes. On the other hand, this is achieved by the fact that the messaging component 61 and the execution component 62 are configured to perform a deterministic and replicated computation.

10 The X-Net Transfer component 66 sends message streams to other subnets and receives message streams from other subnets.

Most components may access the crypto component 68 to execute cryptographic algorithms and the mainnet
15 reader 70 for reading configuration information.

The execution component 62 receives from the messaging component 61 a unit state of the computational unit and an incoming message for the computational unit, and returns an outgoing message and the updated unit state
20 of the computational unit. While performing the execution, it may also measure a resource consumption (gas consumption) of the processed message (query).

The messaging component 61 is clocked by the input blocks received from the consensus component 63. That
25 is, for each input block, the messaging component 61 performs steps as follows. It parses the respective input blocks to obtain the messages for its computational units. Furthermore, it routes the messages to the respective input queues of the different computational units and schedules
30 messages to be executed according to the capacity each computational unit got assigned. Then it uses the execution component 62 to process a message by the corresponding computational unit, resulting in messages to be sent being added to an output queue of the respective computational
35 unit. However, when the message is destined to a computational unit on the same subnet it may be put directly in the input queue of the corresponding

computational unit. The messaging component 61 finally routes the messages of the output queues of the computational units into message streams for subnets on which the receiving computational units are located and
5 forwards these message streams to the state manager component 65 to be certified, i.e., signed by the respective subnet.

The state manager component 65 comprises a
10 certification component 65a. The certification component 65a is configured to certify the output streams of the respective subnet. This may be performed e.g. by a threshold-signature, a multi-signature or a collection of individual signatures of the computational units of the
15 respective subnet.

As mentioned the execution component 62 performs a deterministic and replicated execution of the input blocks. This execution is performed on the respective node 10 and uses resources of the node 10. However, the
20 execution of the input blocks uses resources of the respective node, in particular physical resources such as memory, processors(s), network bandwidth etc. The use of these resources is non-deterministic and may differ between the replicas of a subnet. Accordingly, the non-
25 deterministic part of the execution of the input blocks is represented in FIG. 6 with an execution component 69. The non-deterministic execution component 69 comprises a measurement component 69a. The measurement component 69a measures the replica-local data of the execution, in
30 particular the non-deterministic data of the execution.

The execution component 69 sends the measured replica-local data to a reporting component 41. The reporting component 41 reports the replica-local data to the networking component 64 which forwards/routes the
35 replica-local data to the consensus component 63.

The networking component 64 also forwards the replica-local data to the other replicas on the subnet via

a broadcast protocol, e.g. a gossip protocol. Conversely, the network component 64 also accepts the replica-local data produced by the other replicas and forwards that to the consensus component 63 as well.

5 The reporting component 41 comprises a certification component 41a. The certification component 41a may certify the received replica-local data before forwarding it to the networking component 64. The certification component 41a may issue in particular a
10 signature of the replica/node on the replica-local data.

 The replica-local data is then processed by the consensus component 63. The consensus component 63 runs a consensus protocol and forms input blocks to be processed by the replica.

15 The replica-local data is then processed by the messaging component 61 which forwards/routes the replica-local data to the execution component 62.

 The execution component 62 comprises as sub-components a statistical analysis component 62a and an
20 action component 62b.

 The statistical analysis component 62a is configured to perform a statistical analysis of the replica-local data.

 The action component 62b is configured to take
25 or initiate one or more actions in dependence on the analysis of the replica-local data.

 Both the statistical analysis component 62a and the action component 62b are implemented as part of the respective subnet protocol client 52.

30 According to embodiments the statistical analysis component 62a and the action component 62b may be implemented as system computational units.

 Fig. 7 shows an exemplary visualization of a workflow 700 of the messaging protocol and the consensus
35 protocol and the associated components, e.g. of the messaging component 61 and the consensus component 63 of Fig. 6.

More particularly, Fig. 7 visualizes the workflow of inter-subnet messages exchanged between a subnet SNB and subnets SNA and SNC. Furthermore, the subnet SNB exchanges ingress messages with a plurality of users U. Furthermore, the subnet SNB measures, reports and analyses replica-local data (RL-data).

Starting from the bottom right of FIG. 7, a plurality of input streams 701, 702, 703 and 704 is received by a consensus component 63. The consensus component 63 is a subnet consensus component that is run by a subnet client of the subnet SNB. The input stream 701 comprises inter-subnet messages 711 from the subnet SNA to the Subnet SNB. The input stream 702 comprises inter-subnet messages 712 from the subnet SNC to the Subnet SNB. The input stream 703 comprises ingress messages 713 from the plurality of users U to the subnet SNB. The input stream 704 comprises replica-local data 714 of the replicas of the subnet SNB.

The inter-subnet messages 711 and 712 comprise inter-subnet unit-to-unit messages to be exchanged between the computational units of the different subnets as well as signalling messages. The signalling messages are used to acknowledge or not acknowledge an acceptance of unit-to-unit messages. The messaging component 61 is configured to send the signalling messages from a receiving subnet to a corresponding sending subnet, i.e. in this example from the subnet SNB to the subnets SNA and SNC. The messaging component 61 is according to this example configured to store the sent inter-subnet unit-to-unit messages until an acknowledgement message has been received for the respective unit-to-unit message. This provides a guaranteed delivery.

The consensus component 63 is configured to receive and process the inter-subnet messages 711, 712 of the subnets SNA, SNC, the ingress messages 713 of the users U and the replica-local data 714 of the replicas from SNB and to generate a queue of input blocks 720 from the inter-

subnet messages 711, 712, the ingress messages 713 and the replica-local data 714 according to a predefined consensus mechanism that is executed by the corresponding consensus protocol. Each input block 720 produced by consensus
5 contains a set of ingress messages 713, a set of inter-subnet messages 711, 712, replica-local data 714 and execution parameters 715, EP. The execution parameters 715, EP may include in particular a random seed, a designated execution time and/or a height index. The consensus
10 component 63 may also vary the number of messages in every input block based on the current load of the subnet.

The consensus component 63 provides the queue of input blocks 720 then to the messaging component 61 which is configured to execute the messaging protocol and
15 to process the input blocks 720.

The messaging protocol and the messaging component 61 are clocked by the input blocks 720 received from the consensus component 63.

Before processing the received input blocks,
20 the messaging component 61 may perform one or more pre-processing steps including one or more input checks. The input checks may be performed by an input check component 740.

If the input checks have been passed
25 successfully, the messages of the respective input block 720 may be further processed by the messaging component 61 and the corresponding messages may be appended to a corresponding queue in an induction pool of an induction pool component 731. The induction pool component 731 of
30 the messaging component 61 receives input blocks and input messages that have been successfully passed the input check component 740 and have accordingly been accepted by the messaging component 61 for further processing.

In general, the messaging component 61 pre-
35 processes the input blocks 720 by placing ingress messages, signalling messages, inter-subnet messages and replica-local data into the induction pool component 731 as

appropriate. Signalling messages in the subnet streams are treated as acknowledgements of messages of the output queues which can be purged.

In this example, the induction pool component
5 731 comprises subnet-to-unit queues SNA-B1, SNC-B1, SNA-B2 and SNC-B2 as well as user-to-unit queues U-B1 and U-B2 and the replica-local data for the statistical analysis component 62a and the action component 62b.

Following these pre-processing steps, the
10 messaging component 61 invokes the execution component 62 (see FIG. 6) to execute as much of the induction pool as is feasible during a single execution cycle, providing the designated execution time and the random seed as additional inputs. Following the execution cycle, a resulting output
15 queue of messages, which may also be denoted as output messages, is fed to an output queue component 733. Initially the output queue component 733 comprises unit-to-unit and unit-to-user output queues, in this example the unit-to-unit output queues B1-A1, B1-C2, B2-A2 and B2-
20 C3 and the unit-to-user output queues B1-U1 and B2-U4. As an example, the messages B1-A1 denote output messages from the computational unit B1 of subnet SNB to the computational unit A1 of subnet SNA. As another example, the messages B1-U1 denote output messages from the
25 computational unit B1 of subnet SNB to the user U1. Furthermore, the output queue comprises action messages 716 comprising actions that have been taken by the action component 62b. The action messages are addressed to the registries 52a of the subnet protocol client or to a
30 separate system computational unit.

The output queue component 733 post-processes the resulting output queue of the output messages by forming a set of per-subnet output streams to be certified, e.g. by the certification component 65a as shown in FIG.
35 6, and disseminated by other components. In this example, the per-subnet output streams SNB-SNA, SNB-SNC and SNB-U are provided.

Hence the messaging component 61 further comprises a state storage component 732 that is configured to store the state/unit state of the computational units of the respective subnet, in this example the states of the computational units B1 and B2 of the subnet SNB. The
5 corresponding unit state is the working memory of each computational unit.

The messaging component 61 revolves around mutating certain pieces of system state deterministically.
10 In each round, the execution component 61 will execute certain messages from the induction pool by reading and updating the state of the respective computational unit and return any outgoing messages the executed computational unit wants to send. These outgoing messages or in other
15 words output messages go into the output queue component 733, which initially contains unit-to unit messages between computational units of the network. While intra-subnet messages between computational units of the same subnet may be routed and distributed internally within the
20 respective subnet, inter-subnet messages are routed into output streams sorted by subnet-destinations.

In addition, two pieces of state may be maintained according to embodiments to inform the rest of the system about which messages have been processed. A
25 first piece may be maintained for inter-subnet messages and a second piece of state for ingress messages.

In the following the interactions between the mainnet protocol clients 51 and the subnet protocol clients 52 is described in more detail (see FIG. 5). The mainnet
30 protocol clients 51 manages a number of registries that contain configuration information for the subnets. These registries are implemented by computational units on the mainnet. As mentioned according to other embodiments the central registry may be used instead of the mainnet.
35 Furthermore, each of the subnet protocol clients 52 maintains a registry 52a. This registry 52a comprises control and/or system information for locally managing the

subnet operation. This may be in particular used to allocate the resources of the replica/node to the individual computational units. As an example, the registry 52a may specify a maximum logical time allocated to the computational units of the replica. In this respect the action messages could comprise e.g. an instruction to the registry 52a that the logical time allocated to a misbehaving computational unit of the subnet shall be reduced.

10 This may in particular also be used to ban a misbehaving replica. As an example, the registry 52a may specify a list of replicas that are assigned to the subnet. In this respect the action messages could comprise e.g. an instruction to the registry 52a that the misbehaving
15 replica is removed or replaced.

FIG. 8 shows a layer model 800 illustrating main layers which are involved in the exchange of inter-subnet and intra-subnet messages. The layer model 800 comprises a messaging layer 81 which is configured to serve
20 as an upper layer for the inter-subnet communication. More particularly, the messaging layer 81 is configured to route inter subnet messages between computational units of different subnets. Furthermore, the messaging layer 81 is configured to route ingress messages from users of the
25 network to computational units of the network.

The layer model 800 further comprises a plurality of consensus layers 82 which are configured to receive inter-subnet messages from different subnets as well as ingress messages and to organize them, in
30 particular by agreeing on a processing order, in a sequence of input blocks which are then further processed by the respective subnet. In addition, the layer model 800 comprises a peer-to-peer (P2P) layer that is configured to organize and drive communication between the nodes of a
35 single subnet.

According to embodiments, the network may comprise a plurality of further layers, in particular an

execution layer which is configured to execute execution messages on the computational units of the network.

Referring now to FIG. 9, the creation of blocks
5 in distributed networks according to embodiments of the invention is illustrated. The blocks may be in particular the input blocks 720 shown in FIG. 7 which are created by the consensus component 63 that runs the consensus protocol, in particular a local subnet consensus protocol.

10 In this exemplary embodiment three input blocks 901, 902 and 903 are illustrated. Block 901 comprises a plurality of transactions, namely the transactions tx1.1, tx1.2 and possibly further transactions indicated with dots. Block 902 comprises also a plurality of transactions,
15 namely the transactions tx2.1, tx2.2 and possibly further transactions indicated with dots. Block 903 also comprises a plurality of transactions, namely the transactions tx3.1, tx3.2 and possibly further transactions indicated with dots. The input blocks 901, 902 and 903 are chained
20 together. More particularly, each of the blocks comprises a block hash of the previous block. This cryptographically ties the current block to the previous block(s).

According to embodiments the transactions may be inter-subnet messages, ingress messages and signalling
25 messages.

According to embodiments, the input blocks 901, 902 and 903 may be created by a proof-of-stake consensus-protocol.

However, it should be noted that the input
30 blocks generated by the consensus component do not need to be chained together according to embodiments. Rather any consensus protocol that reaches some kind of consensus between the nodes of a subnet on the processing order of received messages may be used according to embodiments.

35 FIG. 10 shows a more detailed illustration of a computational unit 1000 according to an embodiment of the invention.

The computational unit 1000 comprises an input queue 1001, an output queue 1002, an application state 1003 and a system state 1004.

The computational unit 1000 generally
5 comprises the code of the computational unit and the unit state/execution state of the computational unit.

FIG. 11 shows a flow chart 1100 comprising method steps of a computer-implemented method for running a distributed network comprising a plurality of subnets
10 according to embodiments of the invention. The distributed network may be e.g. embodied as the network 100 as shown in FIG. 1.

At a step 1110, each subnet of the plurality of subnets runs a set of computational units on its nodes,
15 wherein each of the computational units comprises its own unit state.

At a step 1120, the network replicates the set of computational units across the respective subnet.

FIG. 12 shows a flow chart 1200 comprising
20 method steps of a computer-implemented method according to an embodiment of the invention. The distributed network may be e.g. embodied as the network 100 as shown in FIG. 1.

At a step 1210, the replicas/nodes of a subnet
25 measure replica-local data of the respective replica.

At a step 1220, the replicas/nodes report the replica-local data to a consensus component, e.g. to the consensus component 63 as described above.

In one or more following steps, following
30 consensus, the replica-local data is utilized for an analysis of the computations executed across the respective subnet.

More particularly, at a step 1230, the respective subnet computes a set of median values of the
35 replica-local data across the subnet.

As an example, assuming a subnet has e.g. 30 replicas and each of the 30 replicas has 100 computational

units. Further assuming that each of the replicas measures 2 replica local values for each of the 100 computational units as replica-local data, e.g. the number of cache misses and the ratio of wall clock time to logical time of the executions done by the respective computational unit. Then, the subnet computes two sets of median values, namely for each of the 100 computational units a median value for the number of cache misses and a median value for the ratio of wall clock time to logical time. As a result, in this example the subnet has computed two sets of median values, each set comprising 100 median values.

As a second example, assuming a subnet has e.g. 30 replicas. Further assuming that each replica maintains a direct communication link with 10 other replicas from the subnet, thereby forming an overlay topology. Further assuming that each of the replicas measures one replica local value for each of the 10 other replicas that it communicates with as replica-local data, e.g. the average download speed at which the protocol artifacts are received from the other replica. Then the subnet computes one set of median values, namely for each of the 30 replicas a median value for the download speed provided by that replica. As a result, in this example the subnet has computed one set of median values, the set comprising 30 median values, each median value calculated from 10 reported values.

As a third example, assuming a subnet has e.g. 30 replicas. Further assuming that each of the replicas measures one replica local value for each of the 29 other replicas as replica-local data, e.g. the number of times that the replica observed, within a defined time window, a signature share from the other replica for a threshold signature that was to be created by the subnet. Then the subnet computes one set of median values, namely for each of the 30 replicas a median value for the number of observed signature shares. As a result, in this example the subnet has computed one set of median values, the set comprising

30 median values, each median value calculated from 29 reported values.

Furthermore, at a step 1240, further data, in particular publicly verifiable data such as blockchain data, may be provided.

Then, at a step 1250, the subnet performs an analysis, more particularly a statistical analysis of the set of median values and the further data. The statistical analysis may be used e.g. to identify computational units that produce significantly more cache misses than the other computational units.

Next, at a step 1260, the subnet takes one or more actions in dependence on the statistical analysis.

As an example, the logical time allocated to computational units which produce a lot of cache misses may be reduced.

Referring now to Fig. 13, a more detailed block diagram of a network node 10 according to embodiments of the invention is shown, e.g. of the network 100 of FIG. 1. The network node 10 establishes a computing node that may perform computing functions and may hence be generally embodied as computing system or computer. The network node 10 may be e.g. a server computer. The network node 10 may be operational with numerous other general purpose or special purpose computing system environments or configurations.

The network node 10 may be described in the general context of computer system-executable instructions, such as program modules, being executed by a computer system. Generally, program modules may include routines, programs, objects, components, logic, data structures, and so on that perform particular tasks or implement particular abstract data types. The network node 10 is shown in the form of a general-purpose computing device. The components of network node 10 may include, but are not limited to, one or more processors or processing units 1315, a system memory 1320, and a bus 1316 that

couples various system components including system memory 1320 to processor 1315.

Bus 1316 represents one or more of any of several types of bus structures.

5 Network node 10 typically includes a variety of computer system readable media.

 System memory 1320 can include computer system readable media in the form of volatile memory, such as random access memory (RAM) 1321 and/or cache memory 1322.
10 Network node 1310 may further include other removable/non-removable, volatile/non-volatile computer system storage media. By way of example only, storage system 1323 can be provided for reading from and writing to a non-removable, non-volatile magnetic media (not shown and typically called
15 a "hard drive"). As will be further depicted and described below, memory 1320 may include at least one computer program product having a set (e.g., at least one) of program modules that are configured to carry out the functions of embodiments of the invention.

20 Program/utility 1330, having a set (at least one) of program modules 1331, may be stored in memory 1320 by way of example, and not limitation, as well as an operating system, one or more application programs, other program modules, and program data. Each of the operating
25 system, one or more application programs, other program modules, and program data or some combination thereof, may include an implementation of a networking environment. Program modules 1331 generally carry out the functions and/or methodologies of embodiments of the invention as
30 described herein. Program modules 1331 may carry out in particular one or more steps of a computer-implemented method for operating a distributed network, e.g. of one or more steps of the methods as described above.

 Network node 10 may also communicate with one
35 or more external devices 1317 such as a keyboard or a pointing device as well as a display 1318. Such communication can occur via Input/Output (I/O) interfaces

1319. Still yet, network node 10 can communicate with one or more networks 1340 such as a local area network (LAN), a general wide area network (WAN), and/or a public network (e.g., the Internet) via network adapter 1341. According
5 to embodiments the network 1340 may be in particular a distributed network comprising a plurality of network nodes 10, e.g. the network 100 as shown in FIG. 1.

Aspects of the present invention may be embodied as a system, in particular a distributed network
10 comprising a plurality of subnets, a method, and/or a computer program product. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present
15 invention.

The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device. The computer readable storage medium may be, for example, but is not
20 limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A computer readable storage medium, as used herein, is not
25 to be construed as being transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide or other transmission media (e.g., light pulses passing through a fiber-optic cable), or electrical signals
30 transmitted through a wire.

Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external
35 storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network. The network may comprise copper transmission

cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and/or edge servers.

Computer readable program instructions for
5 carrying out operations of the present invention may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or either source code or object code written
10 in any combination of one or more programming languages, including an object oriented programming language such as Smalltalk, C++ or the like, and conventional procedural programming languages, such as the "C" programming language or similar programming languages.

15 Aspects of the present invention are described herein with reference to flowchart illustrations and/or block diagrams of methods, networks, apparatus (systems), and computer program products according to embodiments of the invention.

20 Computer readable program instructions according to embodiments of the invention may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions,
25 which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer readable program instructions may also be stored in a
30 computer readable storage medium that can direct a computer, a programmable data processing apparatus, and/or other devices to function in a particular manner, such that the computer readable storage medium having instructions stored therein comprises an article of manufacture
35 including instructions which implement aspects of the function/act specified in the flowchart and/or block diagram block or blocks.

The computer readable program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other device to cause a series of operational steps to be performed on the computer, other
5 programmable apparatus or other device to produce a computer implemented process, such that the instructions which execute on the computer, other programmable apparatus, or other device implement the functions/acts specified in the flowchart and/or block diagram block or
10 blocks.

The flowchart and block diagrams in the figures illustrate the architecture, functionality, and operation of possible implementations of networks, systems, methods, and computer program products according to various
15 embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some
20 alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon
25 the functionality involved.

While there are shown and described presently preferred embodiments of the invention, it is to be distinctly understood that the invention is not limited thereto but may be otherwise variously embodied and
30 practiced within the scope of the following claims.

Claims

1. A computer-implemented method for operating a distributed network, the distributed network comprising a plurality of nodes, the network comprising one or more subnets, the method comprising
- 5 running a set of computational units;
- assigning each of the computational units to one of the one or more subnets according to a subnet-assignment, thereby creating an assigned subset of the set of computational units for each of the subnets;
- 10 running on each node of the one or more subnets the assigned subset of the computational units, wherein the computational units are configured to execute computations in a deterministic manner, thereby replicating the assigned subsets of the computational units on replicas across the respective subnet;
- running a consensus component, the consensus component being configured to reach a consensus on input blocks to be processed by the replicas of a respective subnet;
- 20 measuring, by each of the replicas of a subnet, replica-local data; and
- reporting, by each of the replicas of a subnet, the replica-local data to the consensus component.
- 25
2. A computer-implemented method according to claim 1, wherein the replica-local data is data pertaining to the computations being locally executed by the corresponding replica.
- 30
3. A computer-implemented method according to claim 1 or claim 2, wherein the replica-local data is non-deterministic data.
- 35
4. A computer-implemented method according to any of the preceding claims, wherein the replica-local data

comprises non-deterministic measurands of the executed computations.

5 5. A computer-implemented method according to any of the preceding claims, wherein the replica-local data is data comprising numeric or enumerable quantities.

6. A computer-implemented method according to any of the preceding claims; further comprising
10 utilizing the replica-local data for an analysis, in particular a statistical analysis, in particular a statistical analysis of computations executed across the respective subnet.

15 7. A computer-implemented method according to claim 6, the method further comprising taking one or more actions in dependence on the analysis of the replica-local data.

20 8. A computer-implemented method according to any of the preceding claims, further comprising computing, by the respective subnet, a set of geometric median values of the replica-local data across the subnet.

25 9. A computer-implemented method according to claim 8, the method further comprising performing a statistical analysis of the set of geometric median values; and taking one or more actions in dependence on the
30 result of the statistical analysis of the set of geometric median values.

10. A computer-implemented method according to any of the preceding claims 6-9, the analysis applying
35 methods configured to ignore outliers, in particular a method that comprises the computation of a geometric median.

11. A computer-implemented method according to
any of the preceding claims 8-10, further comprising
computing a deviation of the set of geometric
5 median values from an expected value; and
taking the one or more actions in dependence
on the deviation.

12. A computer-implemented method according to
10 any of the preceding claims 6 to 11, further comprising
performing a statistical analysis of further
data, the further data being in particular data which is
publicly verifiable blockchain data; and
taking the one or more actions in dependence
15 on the statistical analysis of the replica-local data, the
set of geometric median values and/or the further data.

13. A computer-implemented method according to
any of the preceding claims, further comprising
20 providing a time window for the reporting of
the replica-local data by the replicas of the subnet.

14. A computer-implemented method according to
any of the preceding claims 6-13, wherein the utilizing of
25 the replica-local data for an analysis is performed by a
system computational unit of the subnet.

15. A computer-implemented method according to
any of the preceding claims 7 to 14, wherein the taking of
30 the one or more actions is performed by a system
computational unit of the subnet.

16. A computer-implemented method according to
any of the preceding claims, wherein the replica-local data
35 is selected from the group consisting of:
data on the resource consumption of the
computational units on a respective replica, in particular

the wall clock time of execution and/or the maximum memory allocated during execution;

data on the resource consumption of users on a respective replica, in particular wall clock time and/or
5 logical time of execution performed on behalf of the user, the maximum memory allocated during execution performed on behalf of the user;

for each other replica that a respective replica communicates with, data on measurements of
10 communication parameters of the other replica with the respective replica, in particular latency, response time, down time and/or reliability;

for each other replica that a respective replica communicates with, data on measurements of semantic
15 communication parameters of the other replica with the respective replica, in particular correctness according to a protocol specification;

for each other replica on a respective subnet, data on measurements about the activity of the other
20 replica in the consensus protocol as observed by a respective replica, in particular a number of proposals of input blocks and/or a number of signature shares produced by the other replica.

25 17. A computer-implemented method according to any of the preceding claims 12 - 16, wherein the further data is selected from the group consisting of:

the number of finalized blocks from each replica; and

30 the number of times that each replica has participated in a multi-signature.

18. A computer-implemented method according to any of the preceding claims 7 to 17, wherein the one or
35 more actions are selected from the group consisting of:

adjusting bounds and/or prices for one or more of the computational units of the subnet;

adjusting bounds and/or prices for users of the distributed network; and
banning of replicas.

5 19. A computer-implemented method according to any of the preceding claims, further comprising
receiving, by the consensus component of a subnet,
inter-subnet messages from other subnets;
10 intra-subnet messages from replicas of the same subnet;
messages comprising reports of the replica-local data; and
external user messages of users of the network;
15 generating a queue of input blocks from the received messages according to a predefined consensus mechanism; and
executing the queue of input blocks by the replicas of the subnet.

20 20. A computer-implemented method according to any of the preceding claims, the method further comprising
executing on each node non-replicated computations of the computational units, and
25 wherein the replica-local data comprises measurands of the non-replicated executed computations.

21. A distributed network, the distributed network comprising one or more subnets, wherein each of
30 the one or more subnets comprises a plurality of assigned nodes, wherein the distributed network is configured to perform the steps of a method according to any of the preceding claims.

35 22. A node for a distributed network according to claim 21.

23. A computer program product for operating a distributed network, the distributed network comprising one or more subnets, the computer program product comprising a computer readable storage medium having
5 program instructions embodied therewith, the program instructions executable by one or more of the plurality of nodes to cause the one or more of the plurality of nodes to perform a method according to any of the preceding claims 1 to 20.