



(51) International Patent Classification:

G06F 9/48 (2006.01) H04L 12/26 (2006.01)

(21) International Application Number:

PCT/SE2018/050583

(22) International Filing Date:

05 June 2018 (05.06.2018)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

62/516,427 07 June 2017 (07.06.2017) US

(71) Applicant: TELEFONAKTIEBOLAGET LM
ERICSSON (PUBL) [SE/SE]; 164 83 Stockholm (SE).(72) Inventors: FLINTA, Christofer; Sköldgatan 4, 118 63
Stockholm (SE). MORADI, Farnaz; Lindhagensterassen
27, 112 18 Stockholm (SE).(74) Agent: SJÖBERG, Mats; Ericsson AB, Patent Unit Kista,
RAN Implementation & Core (PU-KRIC), 164 80 Stock-
holm (SE).(81) Designated States (unless otherwise indicated, for every
kind of national protection available): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP,
KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME,
MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ,
OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,
SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.(84) Designated States (unless otherwise indicated, for every
kind of regional protection available): ARIPO (BW, GH,
GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ,
UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ,
TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK,
EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV,
MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,
TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,
KM, ML, MR, NE, SN, TD, TG).

Published:

- with international search report (Art. 21(3))
- in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE

(54) Title: METHOD AND NODE FOR DISTRIBUTED NETWORK PERFORMANCE MONITORING

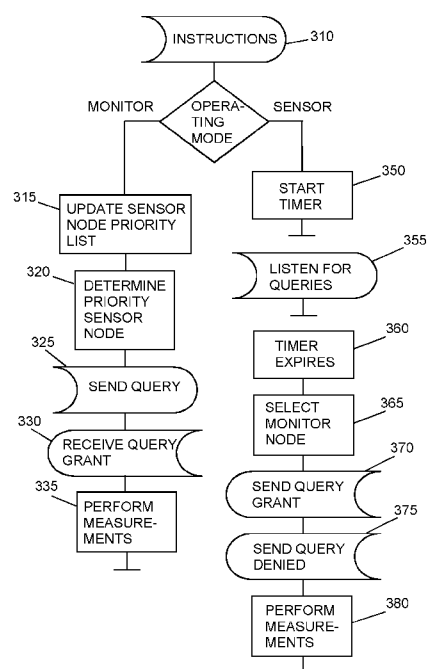


Figure 3

(57) Abstract: The application relates to a method and a node (110) for distributed performance monitoring in a communication network (100) comprising a set of nodes (110-160). Each node in the set is configured to operate alternatively as a monitor node or as a sensor node and wherein each monitor node in the network is configured to perform measurements together with a corresponding sensor node. The monitor node (110) determines (from a priority list) the sensor node (140) which has the highest priority and sends a first query (181) to that sensor node (140). The sensor node (140) on its side has a listening period for receiving queries. When the listening period expires, the sensor node (140) selects the monitor node (110) having the highest query priority and performs measurements 15 together with said selected monitor node (110).

METHOD AND NODE FOR DISTRIBUTED NETWORK PERFORMANCE MONITORING

TECHNICAL FIELD

The application relates to a method and a node for
5 distributed performance monitoring in a communication network.

BACKGROUND

Many enterprises are currently required to digitalize their business to reach customers, vendors, partners, essential
10 applications, etc. This digitalization is often performed by consuming services being offered by the cloud. Ever since, the amount of cloud services has grown in number while they are rapidly evolving, over time. Consequently, underlying infrastructure such as data centres and networks, must
15 synonymously evolve to sustain the increased demand of centralized computation. Thus, data centres and network infrastructure are increasing in both size and intricacy. As the dependence of cloud services are increasing, providers struggle to deliver certain metrics of the cloud, defined in
20 the Service Level Agreement (SLA). Due to the increasing complexity of the data centre infrastructures that are hosting cloud services, it has also become harder to monitor the data centre network. For instance, virtualization has enabled one physical machine to run multiple, separated
25 operating systems on the same host. Thus, adding another level of indirection by introducing a virtualization layer to monitoring.

New technological trends have emerged for the purpose of isolating and deploying applications. The trends are based on
30 a virtualization technique called container virtualization. Container-based virtualization can be described as lightweight virtualization, where only the kernel of the operating system is virtualized, instead of

virtualizing an entire machine. Container virtualization is gaining popularity due to the low overhead of resources. Container orchestrating platforms, such as Docker can also provide resource restriction and alleviates container
5 deployment. In addition to server virtualization, modern networks are transformed into virtualized networks. Using virtualized networks, enables the network to simply adapt and scale per current usage. This is done, namely by getting rid of proprietary hardware middleware boxes, which implements
10 one or more well defined functions, such as firewalls, intrusion detection systems and proxies. These middleware boxes are then implemented in software and connected to the network to reduce the overall complexity of the network, concurrently increasing the functionality and overview of the
15 network. Container orchestration platforms often require virtualized networks for internal and external communication.

A containerized distributed performance monitoring system called ConMon is described in the paper "*ConMon: an Automated Container Based Network Performance Monitoring System*" by
20 Moradi et al (IFIP/IEEE Symposium, May 2017). Its purpose is to monitor container resource consumption and end-to-end network performance from an application perspective. ConMon dynamically adapts to changes in application communication flows. The ConMon monitor can monitor container resource
25 utilization and perform both passive and active network monitoring.

ConMon is a distributed system and the monitoring is performed by deploying monitoring containers on physical servers, running containerized applications. By allowing the
30 monitoring containers to run adjacent to the applications, monitoring will be performed from an applications point of view, while still preserving application isolation.

Since the active monitoring system is distributed, the algorithm for scheduling the probing and the measurements

need to adhere to this intent. Implementing distributed schedulers requires the scheduling decisions to be based on less information compared to a centralized scheduling model. Suitable algorithms for this are for example Round Robin or
5 Controlled Random Scheduling, CRS.

Round Robin is a simple scheduling algorithm, often used in network schedulers such as DNS load balancers and best effort packet switches. Round Robin is a scheduling algorithm built around executing jobs in fixed slices of time units or work
10 cycles, called a Scheduling Quantum or just Quantum. During this quantum, only one job is performed whilst the rest of the queue must wait for its own turn. When one quanta reach its limit such as a time limit or one job completes, the next scheduled job will be executed. For finite scheduling this
15 process will continue until all the jobs complete, and for infinite scheduling the process will continue to schedule the upcoming job.

Due to Round Robins non-concurrent nature two things are guaranteed; No starvation of a process, since all the
20 processes get a fair amount of time executing, and no measurement conflicts since Round Robin is not concurrent by default.

Even though Round Robin fulfills the conditions of avoiding both measurement conflicts and starvation, it will not scale
25 well. This is due to its lack of concurrent execution. When the number of nodes to be monitored increases, the number of jobs to be scheduled will increase at scale. Since no concurrent measurements are done by default, the only way for a system with an increased number of nodes to reach full
30 monitoring coverage, that is when all machines have been monitored at least once, would then be to decrease the time it takes to monitor each machine.

Controlled Random Scheduling, or CRS, is a scheduling method proposed in the thesis work "*Scalable Network Tomography*

System" by Hoque, November 2009 and in the paper "*A Self-Organizing Scalable Network Tomography Control Protocol for Active Measurement Methods*" by Hoque et al, July 2010.

CRS is a distributed scheduling algorithm developed for
5 scheduling of active measurements in networks. The scheduler
is designed to reduce the total measurement time, by allowing
concurrent measurements. To avoid network congestions only a
specified amount of concurrent measurements is being
performed in the network. The number of desired concurrent
10 measurements in the network is set from the start by intents.
To run CRS in the cluster, each node must know how to reach
the other nodes at the given time of scheduling. CRS assumes
that each node can be in one of the two states; Measurement
state and the Sensor state at a given point of time.

15 By alternating between these states over time, each node can
either be in measurement state or in sensor state. Switching
between the states is performed by a Controlled Random
Function. The Controlled Random Function makes the decision
randomly by using pseudo randomizers to randomize a number,
20 and then comparing it against a certain threshold. If the
number exceeds this threshold, the node will become a Measure
node, otherwise it will become a Sensor node. By setting the
threshold, the desired ratio of Measure and Sensor nodes can
be expressed. However, the decision is still made randomly,
25 thus Controlled Random Scheduling. This decision is then
repeated periodically for all nodes in the cluster.

The CRS adheres to the following steps to perform
measurements without measurement interference randomly and
concurrently:

30 1. Role decision based on the controlled random function,
dividing the nodes into Measure and Sensor nodes.

2. If the node is a Measure node

- a. pick the first node randomly from the list of known nodes and send a monitoring request.
- b. If the node is a Sensor node and is free, start monitoring.
- 5 c. Else move on to the next node in the list and repeat 2.b
- d. When the time t expires, repeat from 1.
3. If the node is a Sensor node:
 - a. If free, accept incoming monitor request
 - b. Deny other measurement request whilst measuring
 - 10 c. Once the measurement is done, repeat 3.a
 - d. When the time t expires, repeat from 1.

Using the scheduling algorithm above allows the system to perform concurrent and distributed monitoring scheduling whilst still avoiding measurement conflicts by rejecting
15 conflicting requests.

Nevertheless, the CRS algorithm lacks the guarantee that a Measure - Sensor pair will be monitored once, also known as starvation. In the thesis "*Scalable Network Tomography System*" simulations of the algorithm show that the algorithm
20 never manages to measure all sensor/monitor pairs over the simulated timespan.

SUMMARY

With this background it is the object of the embodiments described below to obviate at least some of the disadvantages mentioned above.

5 The object is achieved by a node configured to operate alternatively as a monitor node or as a sensor node and wherein each monitor node in the network is configured to perform measurements together with a corresponding sensor node and that each monitor node is configured with a sensor
10 node priority list.

When the node is set to operate as a monitor node it is configured to:

- update the sensor node priority list from the last measurement,
- 15 - determine the sensor node which has the highest priority in the sensor node priority list,
- send a first query to the determined sensor node to perform measurements,
- if the query is granted, perform the measurements
20 together with the sensor node and report the result.

When the node is set to operate as a sensor node it is configured to:

- during a predetermined period of time listen for queries to perform measurements from at least one monitor node,
- 25 - when the predetermined period of time for listening has expired, select the monitor node for which the query has the highest priority, return a query grant to the selected monitor node, return a query denied to each one of the remaining querying monitor nodes and perform the
30 measurements together with the selected monitor node.

The object is further achieved by a method for performing measurements in a network comprising a set of nodes wherein each node in the set is configured to operate alternatively as a monitor node or as a sensor node and wherein each
5 monitor node is configured to perform measurements together with a corresponding sensor node. Each monitor node is further configured with a sensor node priority list.

When the node is set to operate as a monitor node the method comprises the following steps:

- 10 - updating the sensor node priority list from the last measurement,
- determining the sensor node which has the highest priority in the sensor node priority list,
- sending a query to the determined sensor node to perform
15 measurements,
- if the query is granted, performing the measurements together with the sensor node and reporting the result.

When the node is set to operate as a sensor node the method comprises the following steps:

- 20 - listening during a predetermined period of time for queries to perform measurements from at least one monitor node,
- selecting when the predetermined period of time for listening has expired, the monitor node for which the query
25 has the highest priority,
- returning a query grant to the selected monitor node,
- returning a query denied to each one of the remaining querying monitor nodes and
- performing the measurements together with the selected
30 monitor node.

Among the advantages is that the method shows a more consistent scheduling pattern than the CRS scheduler, and better scalability metrics in terms of the average time an application must wait for a monitoring event and the time
5 required to reach a full monitoring coverage.

Optionally, in the embodiments where the schedulers are implemented to run inside containers, where the purpose is to deploy the scheduling container on the same server as the running application, the monitoring gives a better
10 understanding of network performance from an application point of view.

The embodiments will now be described in more detail and referring to accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

15 Figure 1 is a block diagram illustrating a distributed network with a set of nodes wherein each node in the set is configured to operate alternatively as a monitor node or as a sensor node.

Figure 2 is a detailed block diagram illustrating a monitor
20 and a sensor node and a controller module.

Figure 3 is a flow charts illustrating an embodiment of the method for performing measurements in a distributed network.

DETAILED DESCRIPTION

Network Monitoring.

25 Network monitoring is the process where network metrics are measured to examine how the network behaves. Network monitoring is essential for large networks, where the different actors of the network have diverse interests of the network performance. For instance, service providers, can

measure the network to inspect what kind of services they can offer to consumers.

There are different ways to observe and quantify network behavior, when monitoring networks and the methods can work
5 on a microcosmic and a macrocosmic scale. In addition, networks can be measured passively or actively depending on measuring techniques. By measuring different aspects of the network, administrators and engineers can use the data for:

- Troubleshooting: Network diagnostics and fault
10 identification
- Performance Optimization: Identifying bottlenecks in the network and load balancing
- Network development and design: Finding needs for new network functions
- 15 • Planning and forecasting of current and coming network workloads
- Computer aided understanding of the network complexity

A summary of key aspects of network monitoring for the different actors has been listed in the paper "Active and
20 passive network measurements: a survey," by V. Mohan et al, Int. J. Comput. Sci. Inf. Technol., vol. 2, no. 4, pp. 1372-1385, 2011. The summary is reproduced in the table below:

Who	Goal	Measure
Internet Service Providers (ISP)	- Capacity Planning - Operations - Value-aided-services, such as customer reports - Usage based billing	- Bandwidth utilization - Packets per second - Round Trip Time RTT - RTT variance - Packet loss - Reachability - Circuit performance - Routing diagnostics
Users	- Monitor Performance - Plan Upgrades - Negotiate service contracts such as SLA - Optimize content delivery - Usage policing	- Bandwidth availability - Response time - Packet loss - Reachability - Connection rates - Service qualities - Host performance
Vendors	- Improve design and configuration of equipment - Implement real-time debugging and diagnostics of deployed network functions	- Trace samples - Log analytics

Network monitoring separates passive monitoring from active
5 monitoring depending on whether the monitoring method
generates probe packets which are injected into the network
or if the method uses the existing network data to provide
information. Passive monitoring monitors existing network
flows, where no probing is performed and thus it can measure
10 the network without changing the network behavior. Active
monitoring, on the other hand, injects data into the network
and observes the behavior of the injected data. Hence active
monitoring might affect the network and receiving nodes while
monitoring.

15 Active Monitoring.

Active monitoring measures the network by examining the
behavior of special data packets, called probe packets, that
are generated and injected into the network. The generated
packets can be of a variety of types, depending of what they
20 are supposed to measure. This could be a TCP packet with no
payload at all, or an UDP packet only containing a timestamp.
Active measuring tools often probe these packets since they
must be carefully constructed to represent actual network
traffic. These representations can vary from packet size to
25 the packets prioritizing in the router. Since active
measurements injects probe packets into the network to obtain
observations, it consumes network bandwidth, which can cause
network interference and measuring interference if two or
more measurements are performed simultaneously. The network
30 interference is directly derived from the amount of traffic
in the current network while measuring interference can be
caused by, not only the increased amount of traffic in the
network, but also the analyzing load on the targeted server.

It is important to understand that a busy server CPU can cause increased latency and TCP timeouts, interpreted as packet losses, which is not directly related to network issues. Thus, active monitoring often requires scheduling to prevent measurement interference.

Passive Monitoring.

Passive network monitoring gather network metrics from existing data flows in the network. It is often performed by listening to traffic, which is duplicated in the network with link splitters or hubs but could also be performed by analysis of router buffers. One common passive monitor is RMON, as described in RFC 1757 'Remote Network Monitoring Management Information Base' which allows remote passive monitoring from a central location where statistics and alarms can be generated by any time. One of the main benefits of using a passive monitor is that the passive monitor does not inject any probe packets into the network. Thus, measurement interference cannot occur when using a passive monitor. However, the passive monitor works through gathering statistics from aggregated data. For high speed networks and data centers the amount of data generated can cause problems for some systems, using several passive capturing points in the network. Modern passive monitors tend to optimize and reduce the amount of disk required to perform accurate analysis, through compression and removal and statistical sampling of data.

ConMon.

ConMon is a distributed, automated monitoring system for containerized environments. The system was developed foremost to adapt to the dynamic nature of containerized applications, where the monitoring adapts to accomplish accurate performance monitoring of both computer and network resources. The monitoring is performed by deploying monitoring containers on physical servers, running

containerized applications. By allowing the monitoring containers to run adjacent to the applications, monitoring will be performed from an applications point of view, while still preserving application isolation.

5 ConMon architecture.

The distributed monitoring system is composed of a variety of monitoring containers running adjacent to the application containers, residing on the same physical server. The two main monitoring agents are the Passive Monitor (PM) and the
10 Active Monitor (AM). To automate and enforce monitoring intents, the system uses an additional Monitor Controller (MC).

Monitoring Containers.

A monitor container is the running component of the
15 distributed system. The monitoring containers deploys a Monitor Controller Container together with additional monitoring functions adjacent to the application(s) to monitor. Each application container that is to be monitored, should have monitoring containers on the same server, where
20 the monitoring containers should be connected to the same virtual switch as the application container.

Monitor Controller Container.

The monitor controller is the core component of the ConMon system. On each physical server, running monitoring
25 containers, a monitor controller will be deployed. These monitor controllers will communicate with each other in a distributed fashion while allowing the system to communicate with other management layers. The monitoring controller, controls both the passive and active monitoring of the
30 network, though dynamic monitoring configurations. It can also receive new intents and requests through the management layer. Each server only need one monitoring controller.

Passive Monitoring Container.

The passive monitoring container is responsible for the passive monitoring of the network, see Passive Monitoring. The passive monitoring containers monitors the applications
5 network flows by analyzing the packets flowing through the virtual switch. This flow monitoring is performed through configuration of port mirroring or tapping in the virtual switches of the server. When monitoring containers are deployed in a server, the monitor controller requests the
10 switch to send a copy of the incoming packets to the passive monitor container. These packets will be used to evaluate flow matrices, perform passive network monitoring and to dynamically adapt monitoring by sending information to the Monitor Controller. A server running multiple application
15 containers only requires a single Passive Monitor, if the applications belongs to the same entity.

Active Monitor Container.

The active monitor container is responsible for the active monitoring of the network, see Active Monitoring. The active
20 monitor is connected to the same virtual switch as the application containers it is responsible to monitor. The active monitor performs end-to-end monitoring by sending probe packets to other active monitors around the network, thus the active monitors will act as both senders and
25 receivers of probe packets. As the active monitor is a separate entity from the application traffic, only one active monitor per server is adequate to perform precise active monitoring.

Collaboration of Monitoring Containers.

30 The monitoring containers running on the server are autonomous applications (such as Microservices) running inside isolated containers. These applications may communicate through web services to act as one distributed

system. Some of the key functions of the ConMon monitoring system and how the distributed monitoring containers communicate to accomplish accurate network monitoring, is described below:

5 Instantiation of Monitoring Containers

The monitoring containers are instantiated by the local server-specific monitoring controller. The monitor controller listens and acts on events triggered by the container management system or orchestrator, such as Docker. Once an application is deployed on the physical server, the monitor controller catches the event and triggers a request to deploy monitoring functions, from the container management system. When the monitoring containers are deployed, the monitoring controller attaches the newly deployed monitoring containers to the same virtual switch as the newly deployed application container is connected. The monitoring controller then configures the switch to perform packet tapping or packet mirroring to the passive monitoring container. The passive monitoring containers analyses the application containers packet flows and determines which remote servers to monitor actively. The application flow maps are then sent to the Monitoring Controller and can be used later for active monitoring scheduling and monitor discovery.

Discovery of Remote Monitors.

Most active measurements require the measurer and receiver of the measurements to identify each other in the network. The identification is also important for synchronization and latency measurements. If the information about the remote monitoring is not provided in advance, monitoring discovery is essential to find the remote monitors. The automatic remote monitor discovery is performed by passively gathering flows in the passive monitoring containers where the source and destination IP of the packets are used to identify the IP of the remote container. Once the remote application

containers have been identified the monitoring controller must find the corresponding remote monitoring controller for the remote application container. The query for accessing the IP of a remote monitoring controller can be implemented
5 through a variety of services such as distributed data bases or injected through monitoring intents.

The described embodiments below are based on a scheduler algorithm that is an improvement of the Controlled Random Scheduling CRS algorithm. The algorithm will henceforth be
10 referred to as the Controlled Priority-based Scheduler, CPS. CPS inherits the concurrent and distributed properties from CRS where each node is being allowed to switch between a monitor and a sensor node at a given random period. In addition, a static period can be set for the time the node
15 should spend in each state.

The CPS algorithm, however, uses a priority-based scheme to decide which monitor/sensor pairs to measure. The CPS algorithm will strive to achieve more consistency in measuring intervals, by always trying to measure the node
20 with the given highest priority. This will result in a cyclic-alike pattern where the monitor/sensor pairs that haven't been monitored for the longest period will be prioritized over the remaining possible monitor/sensor node pairs.

25 The priority in the described embodiments uses the current time difference between last monitoring event. The choice of time as priority is based on the evaluation of scalability and scheduler performance in addition to reducing the time between periodical measurements. Nevertheless, other types of
30 priority could be implemented.

The priority-based scheduler CPS is designed to prevent starvation and in addition, get a more consistent monitoring period, between all the monitor/sensor pairs. For the CRS scheduler on the other hand, it is possible to measure the

same monitor/sensor pair repeatedly in a short interval of time, whilst neglecting other monitor/sensor pairs during that period.

Controlled Priority Scheduler (CPS) Modules.

5 **Figure 2** illustrates two nodes 210, 240 each with the basic modules for the scheduling consisting of a monitor module 211, 241 and a sensor module 212, 242. Figure 2 also illustrates a controller module 290. The monitor and sensor modules are used in the two different states; the Monitoring
10 state and the Sensor state, while the controller module's 290 main responsibility is to handle the time each node should spend in each state.

A node, currently in the monitor state is referred to as a *monitor node*, while a node in the sensor state is referred to
15 as a *sensor node*.

Figure 1 illustrates a network 100 with a set of nodes 110-160 and a controller module 190 wherein each node in the set is configured to operate alternatively as a monitor node or as a sensor node according to the embodiments
20 described below.

When node 110 operates as a monitor node it sends queries 181 to a determined sensor node 140 and if the query is granted, the sensor node 140 returns a query grant 182 and the monitor node 110 and the sensor node 140 performs the
25 measurements together.

When node 110 operates as a sensor node it may receive queries 183 from several monitor nodes 160 etc. If the query is granted, the sensor node 110 returns a query grant 184 to the monitor node 160.

5 Sensor Module

The sensor module's 212, 242 main responsibility is to grant access to the measure request with the highest priority, sent from a monitor node. This monitor approval is implemented by letting the sensor node having a listening period for a predetermined period of time (eg a fixed number of seconds). During the listening period the sensor node will store the request and the corresponding priority, leaving the monitor node waiting. If another request arrives during the listening period, the sensor node will compare the priorities between the two requests and store the request with the highest priority, followed by sending a denying monitoring response to the node who sent the lower priority request. This process will repeat until the listening period expires. Then the sensor node will send a granting monitoring response to the stored monitor node with the highest priority.

After granting a monitoring request the sensor node will deny all incoming monitoring requests for a time period, enough for the monitoring event to complete. If the monitor event completes before the expiration time, the monitor node will reset the sensor node. This expiration time prevents the sensor nodes from blocking incoming requests if the monitor node would fail to unlock the sensor node during a monitoring event. Once the monitor node has completed the monitoring event, the sensor node will go back to the listening state.

30

Monitor Module

The monitor module 211, 241 is responsible for obtaining the first listening sensor node with the highest priority. This feature is implemented by referencing all the host endpoints in a sorted a list, containing a reference to the host endpoints and their corresponding priorities. The list is sorted based on priority in a descending order. The monitor node will then send a monitoring request to the first host in the list. Due to the sensor node's listening period, the monitor node will have an expiration time on the request, that is set to for example one second longer than the sensor's listening period. If the request waiting period exceeds the expiration time, the monitor node will remove the node from the list of potential sensor hosts. This expiration could be due to a faulty sensor node where the expired connection will be reported as an error. However, if the sensor node is functioning correctly, the monitor will receive a monitoring response, containing information about the sensor node. The response will tell the monitor if the host is in the sensor state, if the node is busy or not and if the request has been granted.

On a granted request the monitor node will open monitoring servers on the sensor node and perform the monitoring event. Once completed the servers will be closed and the monitor node will report the results. If, however, the monitor event contains errors the monitor will not close the servers, depending on the error type. If the port is busy, this could for instance be a user triggered event and thus the servers should remain open. On a successful monitoring event, the monitor module will restart its process by re-referencing the host endpoints and priorities in a sorted list.

If the monitoring request is denied however, the monitor node will remove the host from the sorted list and try the second highest priority in the list. If the requests for all nodes in the list are denied, the monitor node will wait a period

for the system to change state, and then repeating by re-referencing the host endpoints in a sorted list.

Controller Module

5 The controller module's 190, 290 main responsibilities are initiation of the node state, adding healthy monitoring host endpoints, removing unreachable host endpoints, randomizing the time the node should spend in the sensor and monitoring mode, and switching modes. The controller module initializes
10 by randomizing the node state, to a monitor or a sensor state. Once the initial state is set, the controller module will then randomize the time the node should spend in the sensor and monitor state. The minimum and maximum time will scale according to the number of nodes in the cluster. The
15 controller module will give the monitor and sensor mode the same amount of time based on: If the node has performed monitoring for a long period, it should receive monitoring for a long period.

If the node has performed monitoring for a short period, it
20 should receive monitoring for a short period. It is possible to set a static scheduled time for the controller module, to decide how long each node should spend in the monitoring and sensor state.

The sensor mode is always followed by the monitoring mode,
25 before the controller module randomizes a new sensor/monitor interval for the mode. Between the switching between the sensor mode and the monitoring mode, the controller module allows all measurement events to complete by waiting a fixed amount of time.

The table below shows an overview of the different components' main responsibilities:

Sensor Module	Controller Module	Monitor Module
Accept and respond to Monitoring Requests: • on initiation • after a successful monitoring event • after a time-out	Add new nodes to the monitor system	Obtain the sensor node with the highest priority that is listening to monitor requests
Deny Monitoring Requests during a monitoring event	Remove nodes that are not longer responding from the monitor system	Send monitor requests
Keeping track of the current request with the highest priority during a listening event	Setting the time the node will spend in the sensor/monitor mode. Can be static or random	Perform the monitoring event, on an approved monitoring request
Approving monitoring for the request with the highest priority	Switching the node between the states	Starting and stopping remote monitoring servers
Denying requests with priorities lower than the current highest priority	Initiation of states	Reporting the result

5

Optionally, a sensor node can grant measurement permission to different monitoring requests that do not conflict with other measurements. For example, an ICMP ping can be executed
10 concurrently with other measurements such as bandwidth measurements.

An example on how to perform the distributed network measurements according to one embodiment is illustrated by the flow chart in Figure 3 and described below.

15 Before starting the actual measurement, the controller module 290 in this example has in step 310 instructed the nodes

210,240 to operate as a monitor node and as a sensor node respectively.

Monitor node 210 updates its priority list of sensor nodes to query in step 315. In step 320 the monitor node 210
5 determines the sensor node in the priority list that has the highest priority.

As said above, the priority in the described embodiments is a function of the current time difference between the last monitoring event. That is, the monitor/sensor pairs that
10 haven't been monitored for the longest period will be prioritized over the remaining possible monitor/sensor node pairs. But again, other types of priorities could be implemented. For example, the priority of a node can dynamically be increased if the network latency between two
15 application containers has to be monitored more frequently since a time-critical task is being executed. After the task is finished, the monitoring priority can be reset.

In the meantime, sensor node 240 has started a timer in step 350 for listen for queries (step 355) from several monitor
20 nodes during a predefined time period.

In step 325 the monitor node 210 sends a query to the determined sensor node 340.

When the timer for the predefined time period to listen for queries in the sensor node 240 expires in step 360, the
25 sensor node 240 selects in step 365 the monitor node 210 with the query having the highest priority (in this case, the query that was received first during the listening period). The sensor node 240 sends in step 370 a query grant to the monitor node 210 which receives the grant in step 330. If the
30 sensor node has received queries with lower priorities from other monitor nodes during the listening period, it sends a query denied to these nodes in step 375.

When the monitor node 210 has received the query grant in step 330 it starts to perform the measurements together with the determined sensor node 240 in steps 335,380. An embodiment of the method described in pseudocode is found below:

```

1. Get list of nodes
2. Receive start event
3. Optional: choose role selecting method(random/scheduled)
10 4. Loop
    1. Select role (monitor/sensor) according to role selection
       method
    2. Update list of nodes
    3. If monitor role
15        1. Loop:
            1. Update priorities based on time since last
               measurement event
            2. Sort list of nodes based on priority in descending
               order
20            3. Select first node in list (node with highest
               priority)
            4. Loop:
                1. Send measurement query to selected node
                2. If permission not granted or no response is
25                   received during time T1
                    1. If selected node is not last node in
                       list
                        1. Select next node in list
                        2. Else (no more items in list)
30                            1. Wait during time T2
                    3. Else (permission is granted)
                        1. Lock the selected node for time T3 to
                           reject all incoming monitor queries,
                           starting/stopping of servers *)
35                        2. Commit measurement to selected node
                           during time T4 and store results
                        3. Unlock selected node

    4. If sensor role
40        1. Loop:
            1. Listen to measurement queries from monitors during
               a time T5
                1. If incoming query
45                    1. If no stored query
                        1. Store incoming query
                    2. Else (there is a stored query)
                        1. Compare priority of the new
                           query to the stored query
                        2. Reject the query with lower
50                           priority and update storage to save
                           query with highest priority
                2. If there is a stored query
                    1. Grant permission to query
                    2. Listen to measurement queries from monitors
55                       during a time T4 from granting
                        1. If incoming query
                            1. Reject the query
                3. If sensor locked after time T3 from granting
                    1. Unlock sensor

```

*) The server could for example be an Iperf tool. Iperf is a commonly used tool for estimating the end-to-end throughput, latency, jitter, and packet loss rate across a path. Iperf is implemented to use the client-server model where measurements are performed by generating UDP or TCP flows in the client. The generated packets are then injected into the network and transmitted across a path to until it reaches the Iperf destination server. The packets are then analysed, and the results are returned to the client when the stream completes. Both the Iperf server and clients can be run in parallel by defining session specific ports for listening and sending.

An example of the method in Java code is shown below:

```

SensorNode
How the sensor handles incoming request during a listening
period.
5  public MonitorResponse nodeToGrantMeasure(MonitorRequest request){
    if(!listening() || isBusy()){
        return new MonitorResponse(false);
    }
    //else - Randomize if priorities are equal
10  if(highest == null){
        highest = request;
    }else if(request.getRequestToMeasure().getPriority() >
        highest.getRequestToMeasure().getPriority()){
        highest = request;
15  }else if(request.getRequestToMeasure().getPriority() <
        highest.getRequestToMeasure().getPriority()){
        return new MonitorResponse(false);
    }else{
        highest = (ThreadLocalRandom.current().nextBoolean()) ? highest :
20  request;
    }
    //Wait for listenTo - 0.1 time to expire prevent data races
    while((listenTo-100) > System.currentTimeMillis()){
        try {
25  if(highest != request){
            return new MonitorResponse(false);
        }
        sleep(100);
    } catch (InterruptedException e) {
30  e.printStackTrace();
    }
    highest = null;
    return new MonitorResponse(true);
35  }

Sensor Node lock mechanism
public void setBusy(){
    this.isBusy = true;
    this.timeout = System.currentTimeMillis() + this.errorTimeout;
40  }
    //If the sensor have not been unlocked for a maximum time due to failer
    from a monitornode
    //the sensor will unlock itself
    public boolean isBusy(){
45  if(timeout == 0){
        return isBusy;
    }else{
        isBusy = timeout > System.currentTimeMillis();
        return isBusy;
50  }
    }
    Sensor Node unlock mechanism
    public void setNotBusy(){
        super.setNotBusy();
55  listenTo = System.currentTimeMillis() + this.listeningPeriod;
    }

```

```

In the superclass
public void setNotBusy(){
isBusy = false;
timeout = 0;
5   }

```

Monitor Mode

Find a the most prioritized node that grants the measurement request

```

10  private MeasureNodeModel obtainMeasureNode(MeasureNodes reference) {
    if (reference.isEmpty()) {
        try {
            sleep(2000);
        } catch (InterruptedException e) {
15      e.printStackTrace();
        }
        return null;
    }
    MeasureNodeModel node = reference.getHighestPriorityNode();
20  if (node.getEndpoint().contains(properties.getMyIP())) {
        node.resetPriority();
        reference.remove(node);
        return obtainMeasureNode(reference);
    }
25  try {
        if (!sendMonitorRequest(node)) {
            reference.remove(node);
            return obtainMeasureNode(reference);
        }
30  } catch (JsonProcessingException e1) {
        reportToObserver("ERROR", "Parse error: " +
            e1.getMessage(), uriEndpoints.getUsers());
    }
    return node;
35  }

```

Controller

Switching between states

```

public void expiredTime(long starttime, long timeout){
isSensorNode = (System.currentTimeMillis() - starttime > timeout) ?
40  !isSensorNode :
    isSensorNode;
}
public void monitorMode(long startTime, long timeout){
45  while(!isSensorNode){
        try {
            monitorMode.priorityMonitor(nodes);
            expiredTime(startTime, timeout);
        } catch (Exception e){
            e.printStackTrace();
50  System.out.println(e.getMessage());
        }
    }
}
public void sensorMode(long startTime, long timeout){

```

55

```

        while(isSensorNode){
            expiredTime(startTime, timeout);
            try {
                sleep(1000);
5         } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
10    public void allowMeasurementsToComplete(){
        try {
            sleep(10000);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
15         }
    }
    public long getTimeout(){
        if(staticTimeout != 0){
            return staticTimeout;
20         }
        return ThreadLocalRandom.current().nextInt(20000, 60000 +
            2*nodes.size()*1000);
    }
    public void tick() throws JsonProcessingException {
25         long startTime = System.currentTimeMillis();
        long timeout = getTimeout();
        this.updateNodes();
        monitorMode(startTime, timeout);
        startTime = System.currentTimeMillis();
30         sensorMode(startTime, timeout);
        allowMeasurementsToComplete();
        tick();
    }

    Initiation and update
35    public Controller(GetNetworkProperties properties, UriEndpoints
        uriEndpoints, CXSMonitorMode
        monitorMode){
        this.properties = properties;
        this.uriEndpoints = uriEndpoints;
40         nodes = new MeasureNodes();
        this.updateNodes();
        isSensorNode = ThreadLocalRandom.current().nextBoolean();
        this.monitorMode = monitorMode;
    }
45    public void updateNodes(){
        EndpointHandler endpointHandler = new EndpointHandler(properties);
        nodes.addNewNodes(endpointHandler.refreshEndpoints());
        endpointHandler.writeEndpoints(nodes);
        properties.reportToObserver(nodes.removeBadNodes(), "Removed bad nodes
50         " ,
        uriEndpoints.getUsers());
    }

```

The CPS scheduler has been compared with other decentralized schedulers based on Round Robin scheduling and the Controlled Random Scheduler (CRS). The schedulers were implemented to run inside containers, where the purpose was to deploy the scheduling container on the same host as the running application.

Scheduler Performance

Full coverage is defined as when all nodes have monitored all nodes, in other words when all possible monitoring pairs have been monitored at least once. As the amount of monitoring pairs is quadratic to the number of nodes in the cluster, a cluster containing N nodes will have: $N \cdot (N-1) = N^2 - N$ monitor pairs.

The tables below summarize the estimation of the time to reach full coverage as the cluster grows and the average waiting period for monitoring (x = number of nodes).

Scheduler	Time to reach full coverage [Minutes]
Round Robin	$T = 0,2202x^2 - 0,0652x - 1,3742$
CRS	$T = 3,5462x - 10,503$
CPS	$T = 1,1158x + 0,6904$

Scheduler	Average Waiting Period for Monitoring [Minutes]
CRS	$T = 0,3004x - 0,6042$
CPS	$T = 0,407x - 0,7604$
Theoretical	$T = 0,2242x$

In the evaluations of the CPS scheduler compared with other decentralized schedulers such as Round Robin scheduling or Random Scheduler (CRS), the CPS showed a more consistent scheduling pattern than the CRS scheduler, and better scalability metrics in terms of the average time an application has to wait for a monitoring event and the time required to reach a full monitoring coverage, when compared to Round Robin and CRS scheduling. The active monitoring

scheduler gives a more reliable insight in how applications utilizes network performance, and how to perform concurrent monitoring, while avoiding conflicting monitoring events.

Returning to Figure 2, the block diagram also illustrates a detailed but non-limiting example of how the two nodes 210,240 (here operating as a monitor node and as a sensor node respectively) may be structured to bring about the above-described solution and embodiments thereof. The node 210,240 may be configured to operate according to any of the examples and embodiments for employing the solution as described herein. The node 210,240 is shown to comprise a processor unit 213,243 and a memory unit 214, 244, said memory unit comprising instructions executable by said processor unit 213,243 whereby the nodes are operable as described herein. The node 210,240 also comprises a communication circuit (I/O unit) 215,245 for receiving and sending information in the manner described herein.

Abbreviations:

CPS Controlled Priority Scheduling

20 CRS Controlled Random Scheduling

DNS Domain Name Server

ICMP Internet Control Message Protocol

OS Operating System

SLA Service Level Agreement

25

CLAIMS

1. A node (110) belonging to a set of nodes (110-160) in a network (100) wherein each node in the set is configured to operate alternatively as a monitor node or as a sensor node and wherein each monitor node in the network is configured to perform measurements together with a corresponding sensor node, the node *characterised in that* each monitor node is configured with a sensor node priority list and
- 10 when said node is set to operate as a monitor node (110) it is configured to:
- update the sensor node priority list from the last measurement;
 - determine the sensor node (140) which has the highest priority in the sensor node priority list;
 - 15 - send a first query (181) to the determined sensor node (140) to perform measurements;
 - if the query is granted (182), perform the measurements together with the sensor node (140) and report the result;
- 20 when said node is set to operate as a sensor node (110) it is configured to:
- during a predetermined period of time listen for queries (183) to perform measurements from at least one monitor node (160);
 - 25 - when the predetermined period of time for listening has expired, select the monitor node for which the query has the highest priority, return a query grant (184) to the selected monitor node (160), return a query denied to each of the remaining querying monitor nodes and perform the
 - 30 measurements together with the selected monitor node (160).

2. A node (110) according to claim 1, characterised in that when the node is set to operate as a monitor node it is further configured to:

- if the first query is denied by the determined sensor node (140), determine a second sensor node (150) that has the second highest priority;
- send a second query to the determined second sensor node (150) to perform measurements.

3. A node according to claim 2, further configured to when the node is set to operate as a monitor node, repeat the steps to determine a further sensor node as long as the queries are denied.

4. A node according to claim 3, further configured to when the node is set to operate as a monitor node and if no sensor node has granted any of the queries, repeat the query steps after a predetermined period of time.

5. A node according to any of the preceding claims, further configured to when the node is set to operate as a sensor node (110) to deny further queries from other monitor nodes as long as the sensor node performs measurements together with the selected monitor node (160).

6. A node according to any of the preceding claims further configured to receive instructions from a controller module (290) to initiate the step to operate as a monitor node or as a sensor node.

7. A node according to claim 6 further configured to receive from the controller module (290) an instruction to alternate between to operate as a monitor node or as a sensor node.

8. A node according to any of the preceding claims wherein the measurement priority of the sensor nodes is a

function of the time since the last measurement was performed by the monitor node together with the sensor node and wherein the measurement priority of the queries from the monitor node is a function of the time of reception by the sensor node.

9. A node (210) according to any of the preceding claims being a physical server wherein the server comprises a monitor module (211) and a sensor module (212) each module implemented as software containers configured to run on said server and further configured so that the server can alternatively operate as a monitor node or as a sensor node respectively.

10. A system comprising a set of nodes (110,160) as in any of the preceding claims and a controller module (190) configured to communicate with the nodes and further configured for any or a combination of the following:

- adding or removing nodes to or from the set of nodes;
- instructing the node to initiate the state of to operate as a monitor node or as a sensor node;
- instructing the node to alternate between to operate as a monitor node or as a sensor node.

11. A method for performing measurements in a network (100) comprising a set of nodes (110-160) wherein each node in the set is configured to operate alternatively as a monitor node or as a sensor node and wherein each monitor node is configured to perform measurements together with a corresponding sensor node, the method **characterized in that** each monitor node is configured with a sensor node priority list and

when said node is set to operate as a monitor node (110):

- updating (315) the sensor node priority list from the last measurement;
- determining (320) the sensor node (140) which has the highest priority in the sensor node priority list;
- 5 - sending (325) a first query to the determined sensor node 140 to perform measurements;
- if the query is granted (330), performing (335) the measurements together with the sensor node (140) and reporting the result;
- 10 when said node is set to operate as a sensor node (110):
 - listening (355) during a predetermined period of time for queries to perform measurements from at least one monitor node (160);
 - selecting (365) when the predetermined period of time for
15 listening has expired (360), the monitor node for which the query has the highest priority;
 - returning (370) a query grant to the selected monitor node (160);
 - returning (375) a query denied to each of the remaining
20 querying monitor nodes and
 - performing (380) the measurements together with the selected monitor node (160).
- 12. A method as in claim 11 when the node is set to operate as a monitor node further comprising the steps of:
 - 25 - determining a second sensor node (150) if the first query is denied by the determined sensor node (140), that has the second highest priority;

- sending a second query to the determined second sensor node (150) to perform measurements.

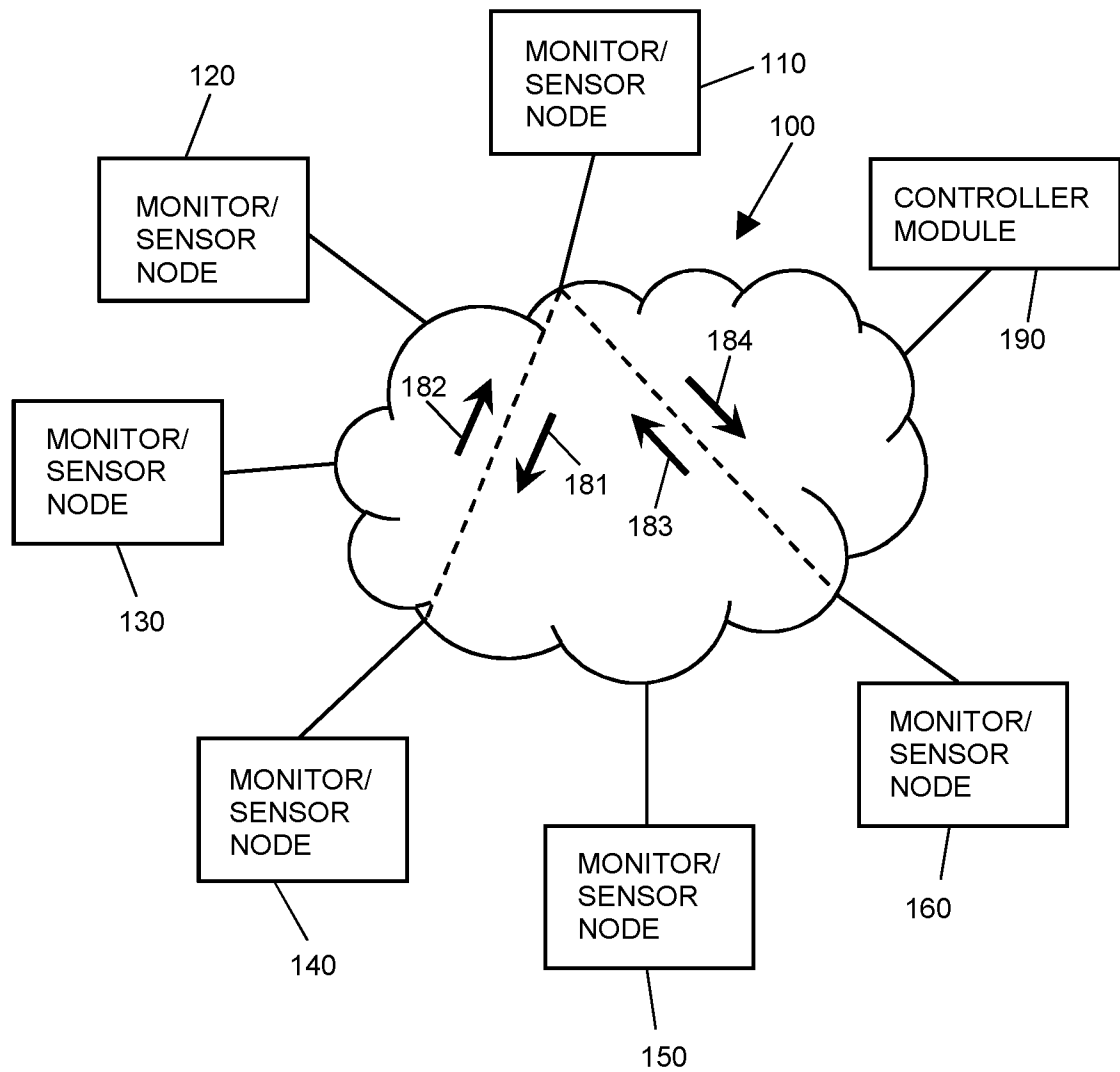
13. A method as in claim 12 when the node is set to operate as a monitor node, further comprising repeating the
5 steps to determine a further sensor node as long as the queries are denied.

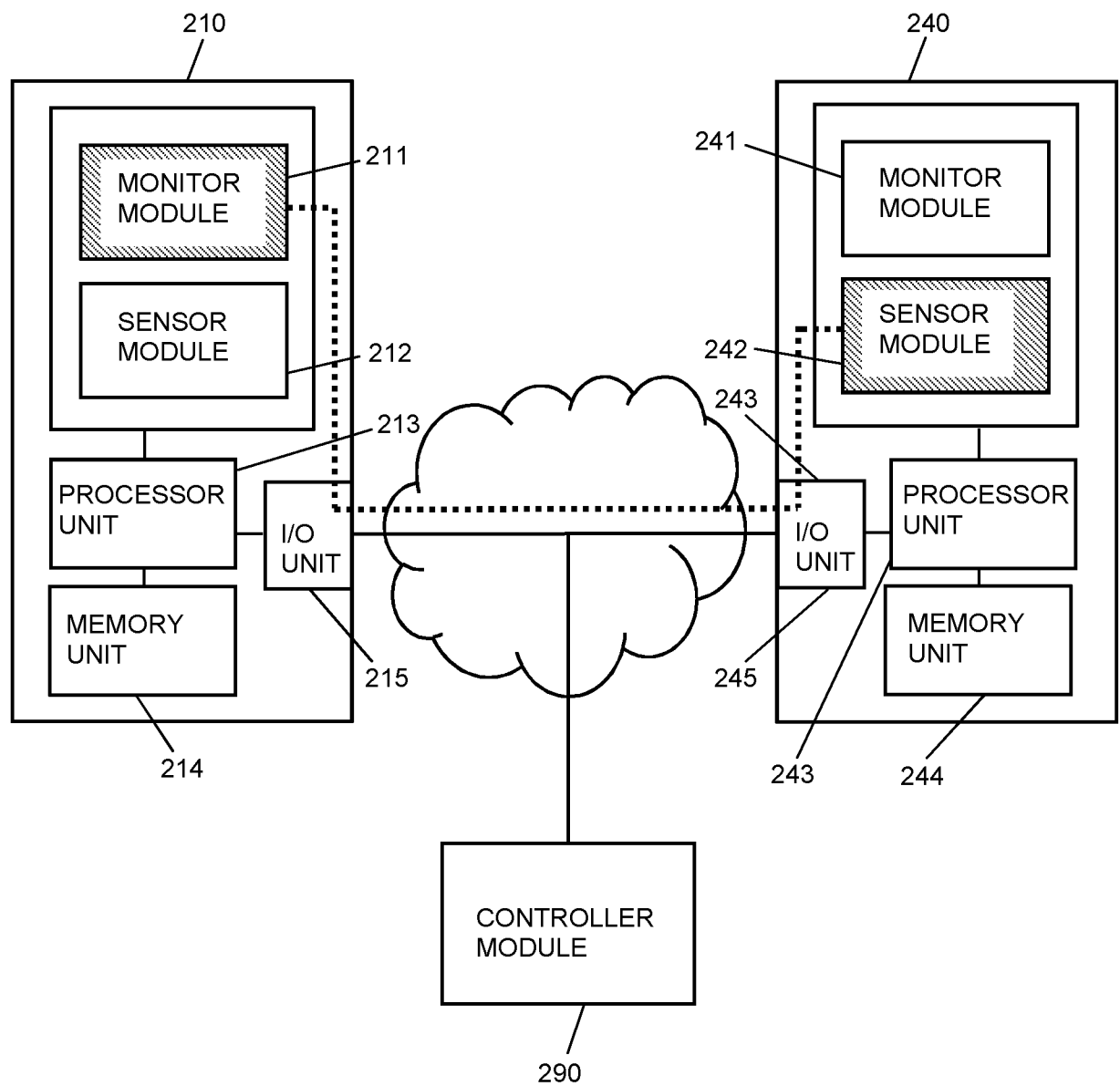
14. A method as in any of the claims 11-13 when the node is set to operate as a sensor node (110) further comprising the step of denying further queries from other
10 monitor nodes as long as the sensor node performs measurements together with the selected monitor node (160).

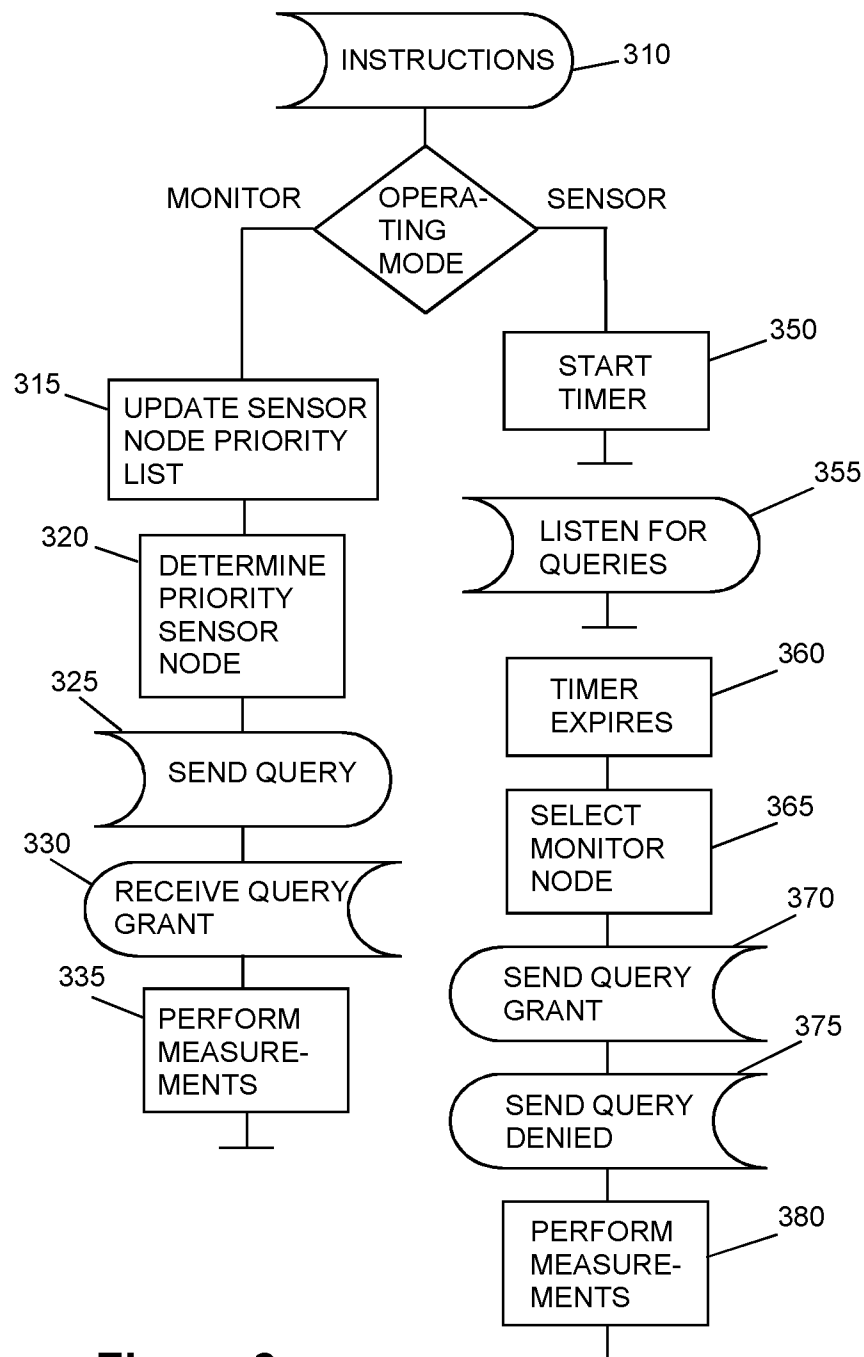
15. A method as in any of the claims 11-14 further comprising the step of receiving (310) from a controller module (190), instructions to initiate the state to operate
15 as a monitor node or as a sensor node.

16. A method as in claim 15 further comprising the step of receiving (310) from the controller module (190), instructions to alternate between to operate as a monitor node or as a sensor node.

20 17. A method as in any of the claims 11-16 wherein the priority of the sensor nodes is a function of the time since the last measurement was performed by the monitor node together with the sensor node and wherein the priority of the queries from the monitor node is a function of the
25 time of reception by the sensor node.

**Figure 1**

**Figure 2**

**Figure 3**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE2018/050583

A. CLASSIFICATION OF SUBJECT MATTER

IPC: see extra sheet

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: G06F, H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE, DK, FI, NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, PAJ, WPI data, COMPENDEX, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Rezaul Hoque: "Scalable network tomography system", Master of Science Thesis, Stockholm, Sweden, Publication date: 2010-03-09, Retrieved from: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A302704&dswid=3385 ; abstract; pages 15-29 --	1-17
Y	US 20140165070 A1 (PERSIKOV SERGEY ET AL), 12 June 2014 (2014-06-12); abstract; paragraph [0018] --	1-17

☒ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier application or patent but published on or after the international filing date	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&" document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 21-09-2018	Date of mailing of the international search report 21-09-2018
Name and mailing address of the ISA/SE Patent- och registreringsverket Box 5055 S-102 42 STOCKHOLM Facsimile No. + 46 8 666 02 86	Authorized officer Ralf Boström Telephone No. + 46 8 782 28 00

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE2018/050583

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Rezaul Hoque et al.: "A self-organizing scalable network tomography control protocol for active measurement methods", Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2010 International Symposium on, Publication date: 2010-07-11.; Chapter IV --	1-17
A	Farnaz Moradi et al.: "ConMon: An automated container based network performance monitoring", 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)system", Publication date: 2017-05-08.; abstract; Chapter III --	1-17
P, X	Mathew Clegg: "Scheduling network performance monitoring in the cloud", Uppsala Universitet, Publication date: 2017-08-28, Retrieved from: http://uu.diva-portal.org/smash/record.jsf?pid=diva2%3A1136475&dswid=-2097 ; abstract; Section 6.3 -- -----	1-17

Continuation of: second sheet

International Patent Classification (IPC)

G06F 9/48 (2006.01)

H04L 12/26 (2006.01)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/SE2018/050583

US 20140165070 A1 12/06/2014 US 9400682 B2 26/07/2016