(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 June 2018 (28.06.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/115934 A1

(51) International Patent Classification:
H04L 12/26 (2006.01) *H04L 29/06* (2006.01)

(21) International Application Number:
PCT/IB2016/057904

(22) International Filing Date:
21 December 2016 (21.12.2016)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON (PUBL) [SE/SE]; SE-164 83, Stockholm (SE).

(72) Inventors: **BISHT, Ashutosh**; A-403, Tower3, Adarsh Palm Retreat, Bellandur, Bangalore 560103 (IN). **K, Faseela**; Fern Icon, Survey No 28 and 36/5 Doddanakundi Village, Bangalore 560037 (IN).

(74) Agent: **DE VOS, Daniel M.**; Nicholson De Vos Webster & Elliott LLP, 99 Almaden Boulevard, Suite 710, San Jose, CA 95113 (US).

(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, 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,

(54) Title: PACKET TIMESTAMPING IN SOFTWARE DEFINED NETWORKING NETWORKS

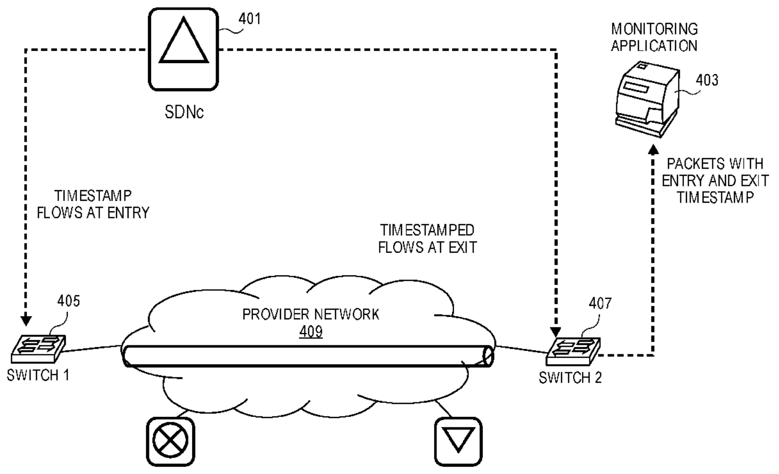


FIG. 4

(57) Abstract: A method is implemented by a network device for the insertion of a timestamp into a packet header. The insertion of the timestamp into the packet header enables performance metrics in a software defined networking (SDN) network. The method includes receiving a packet on an ingress port of the network device, determining whether the received packet matches timestamp criteria, upon determining that the received packet matches the timestamp criteria, copying a value in a global timestamp packet register in a flow control pipeline into a header of the received packet, and forwarding the received packet toward a destination.

WO 2018/115934 A1

[Continued on next page]

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))*

PACKET TIMESTAMPING IN SOFTWARE DEFINED NETWORKING NETWORKS

TECHNICAL FIELD

[0001] Embodiments of the invention relate to the field of network metrics for software defined networking (SDN) networks; and more specifically, to the monitoring of traffic using timestamps where the timestamps are added to packet headers from a global timestamp packet register.

BACKGROUND

[0002] Software Defined Networking (SDN) is a networking paradigm that has gained a lot of interest in the recent times. Under SDN, the data plane and the control plane are separated. The data path nodes are simple forwarding engines that are programmed by a controller using a set of flow control protocol rules. The most popular flow control protocol is OpenFlow (OF) developed by the Open Networking Foundation (ONF). Such forwarding engines are called OF switches. The central controller is responsible for exposing the programming interface to the user via standard north bound interfaces. The benefits of Software Defined Networking include remote administration, reduced costs for switches, enhanced configurability upgradability and similar advantages.

[0003] Administration of networks including SDN networks seek to manage network performance, which includes monitoring metrics of performance. Such metrics include network delay. Network delay is defined as the time taken by a set of related packets, referred to as a flow, as it traverses a set of network elements all managed by a single administrative entity. One can estimate it by measuring the time when the packets ingress the network and measuring the time again when the packets egress the network. The difference between the two times would give the estimate of the network delay for that packet. To minimize measurement errors, the average of a number of such measurements is provided as a network delay estimate.

[0004] Network delay can be defined for individual flows, aggregate of flows or on a per application basis. Network delay is an important metric of network performance. Often delay and jitter (delay variance) measurement values are tied to service level agreements (SLAs), which are contracts between network operators and customers for specific performance levels. Therefore, it is necessary for network operators and administrators to develop tools that can measure network delay accurately.

[0005] In many networks, there are a series of services that are applied to most traffic that traverses the network. The services can be referred to as a service chain, since they form a sequence of services that may be applied is a sequence or 'chain.' Metrics for these service chains are important for network administrators and operators. Network delay on service chains is the total delay experienced by a packet on the service chain. The network delay for the service chains is the time spent by a packet between the service plane entry point, which is the point at which the packet ingresses the service chain, and the service plane exit point, which is the point at which the packet egresses the service chain. In a topology where network service header (NSH) is used with a service function chain the network delay may be between a first service function (s1) and a second service function (s2) and NSH may be used with encapsulated traffic. The measurement of the network delay for SLA verification and network trouble shooting is essential as in the case of traditional networks.

[0006] Additionally, in case of service chaining networks residing in the cloud where network elements are controlled by a centralized SDN controller, the measured information on a live network can be used to measure information that can be fed into the SDN controller which can use it for service chain path load balancing. This would not be possible in traditional networks where packet path selection is decided by distributed routing protocols. In a cloud environment, the measured information can be used for determining the virtual machine load and behavior analysis and this analysis can help in virtual network function (VNF) placements elasticity (e.g., spawning new virtual machines (VMs), shutting down VMs, VM migration and similar functions). Thus, generating accurate service chain delay measurements is an important feature in service chaining solutions.

SUMMARY

[0007] The embodiments include a method implemented by a network device. The method provides for the insertion of a timestamp into a packet header to enable performance metrics in a software defined networking (SDN) network. The method of inserting the timestamp includes receiving a packet on an ingress port of the network device, determining whether the received packet matches timestamp criteria, upon determining that the received packet matches the timestamp criteria, copying a value in a global timestamp packet register in a flow control pipeline into a header of the received packet, and forwarding the received packet toward a destination.

[0008] In other embodiments, the network device is configured to implement the method for insertion of a timestamp into a packet header to enable performance metrics in the SDN network and includes at least one real-time clock and a flow control pipeline. The flow control pipeline includes a global timestamp packet register. The flow control pipeline is communicatively coupled to the at least one real-time clock. The flow control pipeline is also configured to

receive a packet on an ingress port of the network device, to determine whether the received packet matches timestamp criteria, upon determining that the received packet matches the timestamp criteria, to copy the value in the global timestamp packet register into a header of the received packet, and to forward the received packet toward a destination.

[0009] In a further embodiment, a computing device is in communication with the network device. The computing device is configured to execute a plurality of virtual machines for implementing network function virtualization (NFV). The computing device is configured to implement the method for insertion of the timestamp into a packet header to enable performance metrics in an SDN network. The computing device includes at least one real-time clock, a non-transitory computer-readable medium having stored therein a flow control pipeline module, and a processor. The processor is coupled to the at least one real-time clock and the non-transitory computer-readable medium. The processor is configured to execute one of the plurality of virtual machines. The virtual machine is configured to execute the flow control pipeline that includes a global timestamp packet register. The flow control pipeline is configured to receive a packet on an ingress port of the network device, to determine whether the received packet matches timestamp criteria, upon determining that the received packet matches the timestamp criteria, to copy the value in the global timestamp packet register into a header of the received packet, and to forward the received packet toward a destination.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The invention may best be understood by referring to the following description and accompanying drawings that are used to illustrate embodiments of the invention. In the drawings:

[0011] Figure 1 is a diagram of one embodiment of a switch in a software defined networking (SDN) network that implements the improved network delay computation.

[0012] Figure 2A is a flowchart of one embodiment of a process for a switch to handle received traffic to determine the traffic to be timestamped.

[0013] Figure 2B is a flowchart of one embodiment of a time manager process to update a global timestamp packet register.

[0014] Figure 3 is a diagram of one embodiment of a process for a controller to configure a switch to be utilized with the timestamping system.

[0015] Figure 4 is a diagram of an example application of the timestamping process that provides more accurate timestamps and thereby more accurate network delay calculations.

[0016] Figure 5A illustrates connectivity between network devices (NDs) within an exemplary network, as well as three exemplary implementations of the NDs, according to some embodiments of the invention.

[0017] Figure 5B illustrates an exemplary way to implement a special-purpose network device according to some embodiments of the invention.

[0018] Figure 5C illustrates various exemplary ways in which virtual network elements (VNEs) may be coupled according to some embodiments of the invention.

[0019] Figure 5D illustrates a network with a single network element (NE) on each of the NDs, and within this straight forward approach contrasts a traditional distributed approach (commonly used by traditional routers) with a centralized approach for maintaining reachability and forwarding information (also called network control), according to some embodiments of the invention.

[0020] Figure 5E illustrates the simple case of where each of the NDs implements a single NE, but a centralized control plane has abstracted multiple of the NEs in different NDs into (to represent) a single NE in one of the virtual network(s), according to some embodiments of the invention.

[0021] Figure 5F illustrates a case where multiple VNEs are implemented on different NDs and are coupled to each other, and where a centralized control plane has abstracted these multiple VNEs such that they appear as a single VNE within one of the virtual networks, according to some embodiments of the invention.

[0022] Figure 6 illustrates a general purpose control plane device with centralized control plane (CCP) software 650), according to some embodiments of the invention.

DETAILED DESCRIPTION

[0023] The following description describes methods and apparatus for improving the accuracy of network performance metrics for a software defined networking (SDN) network. Specifically, the embodiments provide a method and apparatus for improving the accuracy and efficiency of timestamping packets at the switches. The embodiments introduce a global timestamp packet register in the flow control pipeline that enables copy instructions to efficiently insert the timestamp into a header of a packet. The global timestamp packet register can be updated regularly with a timestamp value derived from a real-time clock (RTC) or similar component of the switch.

[0024] In the following description, numerous specific details such as logic implementations, opcodes, means to specify operands, resource partitioning/sharing/duplication implementations, types and interrelationships of system components, and logic partitioning/integration choices are set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the invention may be practiced without such specific details. In other instances, control structures, gate level circuits and full software instruction sequences have not been shown in detail in order not to obscure the invention. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate functionality without undue experimentation.

[0025] References in the specification to "one embodiment," "an embodiment," "an example embodiment," etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to affect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0026] Bracketed text and blocks with dashed borders (e.g., large dashes, small dashes, dot-dash, and dots) may be used herein to illustrate optional operations that add additional features to embodiments of the invention. However, such notation should not be taken to mean that these are the only options or optional operations, and/or that blocks with solid borders are not optional in certain embodiments of the invention.

[0027] In the following description and claims, the terms "coupled" and "connected," along with their derivatives, may be used. It should be understood that these terms are not intended as synonyms for each other. "Coupled" is used to indicate that two or more elements, which may or may not be in direct physical or electrical contact with each other, co-operate or interact with each other. "Connected" is used to indicate the establishment of communication between two or more elements that are coupled with each other.

[0028] An electronic device stores and transmits (internally and/or with other electronic devices over a network) code (which is composed of software instructions and which is sometimes referred to as computer program code or a computer program) and/or data using machine-readable media (also called computer-readable media), such as machine-readable storage media (e.g., magnetic disks, optical disks, read only memory (ROM), flash memory devices, phase change memory) and machine-readable transmission media (also called a carrier) (e.g., electrical, optical, radio, acoustical or other form of propagated signals – such as carrier waves, infrared signals). Thus, an electronic device (e.g., a computer) includes hardware and software, such as a set of one or more processors coupled to one or more machine-readable storage media to store code for execution on the set of processors and/or to store data. A 'set,' as used herein, refers to any positive whole number of items including one item. For instance, an electronic device may include non-volatile memory containing the code since the non-volatile memory can persist code/data even when the electronic device is turned off (when power is removed), and while the electronic device is turned on that part of the code that is to be executed by the processor(s) of that electronic device is typically copied from the slower non-volatile memory into volatile memory (e.g., dynamic random access memory (DRAM), static random access memory (SRAM)) of that electronic device. Typical electronic devices also include a set or one or more physical network interface(s) to establish network connections (to transmit and/or receive code and/or data using propagating signals) with other electronic devices. One or more parts of an embodiment of the invention may be implemented using different combinations of software, firmware, and/or hardware.

[0029] A network device (ND) is an electronic device that communicatively interconnects other electronic devices on the network (e.g., other network devices, end-user devices). Some network devices are "multiple services network devices" that provide support for multiple networking functions (e.g., routing, bridging, switching, Layer 2 aggregation, session border control, Quality of Service, and/or subscriber management), and/or provide support for multiple application services (e.g., data, voice, and video).

[0030] Network delay measurements such as two-way active measurement protocol (TWAMP) are the most commonly used path delay measurement protocols in traditional routers and switches. TWAMP can be used for both one-way and two-way delay determination. A test stream of UDP based request-response messages are generated and timestamped at the two ends of the path whose delay is to be measured. TWAMP relies on the two endpoints being time-synced using network time protocol (NTP) or precision time protocol (PTP), where PTP provides greater (microsecond) accuracy. The timestamp differences convey the path delay. TWAMP does not measure the delay experienced by the real traffic on the path, it measures the delay experienced by a test stream, which is a set of test packets sent between the two endpoints. The assumption is that this measurement is close to the delay experienced by the real traffic. This might be a valid assumption in the case of traditional network paths which typically consist of packet forwarding nodes that are application unaware. However, in other scenarios real traffic may be forwarded significantly differently from the test packet stream.

[0031] Service chain delay measurements using network service header (NSH) is another network metric collection mechanism. The NSH is a protocol developed specifically for service chains. A NSH is added to traffic that is traversing a service chain. The NSH is added in the packet header to provide a separate service plane independent of the transport protocol. The NSH defines a sequence of service functions that the corresponding packet must traverse prior to reaching its ultimate destination. The NSH is inserted by a switch at the ingress into the network or service chain. The NSH may be inserted between the original packet and any encapsulation (e.g., tunneling or label swap protocol encapsulation). The embodiments work with network devices and service functions that understand the NSH.

[0032] In the embodiments, the packet entry-point adds the timestamps to the traffic at the time of entry and this information is carried in the NSH (e.g., in the service platform context field of NSH). Similarly, the egress-point of the service chain also adds the timestamps at the time of exit in the NSH. The switch at the exit point of the service chain or network transmits the information from the NSH including the timestamp at the entry point (which is obtained from the NSH) and the timestamp at the exit point, to the controller. The controller or other monitoring application then calculates the difference to identify the network delay. Since a spare field in NSH is used for carrying the timestamp, the original packet is not modified.

[0033] The embodiments utilize entry and exit points to the service chain or network that are time synced using any process or protocol like NTP or PTP to ensure the synchronization. The embodiments also use the timestamping capability that is available in the data plane of the switches. In some embodiments, special measurement packets may be sent along the service plane.

[0034] The embodiments overcome the problems with existing solutions. Current SDN based solutions such as the ones based on OpenFlow protocol, do not provide a method for an SDN controller to program the SDN switches for timestamping the packets. There is no method to make the system time on an OpenFlow switch available over a generic interface to the packet pipeline processing.

[0035] The embodiments overcome these limitations of prior solutions. The embodiments provide a global timestamp packet register in a flow control switch (e.g., an OpenFlow switch), that provides current time. This global timestamp packet register, as with other packet registers, is available in packet processing pipeline.

[0036] The embodiments provide a method and system that is compatible with SDN networks and switches in SDN networks. In particular, the method and system are compatible with network performance monitoring in service chains in SDN networks. These SDN networks can be implemented using any type of protocol including any type of flow control protocol. The examples provided herein discuss implementation examples related to the OpenFlow protocol by way of example and not limitation. One skilled in the art would understand that the methods and systems can also be applied and used in conjunction with other similar protocols.

[0037] In an OpenFlow implementation, the packet register fields are used to store temporary values and information alongside the packet through pipeline processing. The packet registers cannot be matched in tables, i.e., they usually cannot be used in the flow entry match structure. They can be used with the set-field and copy-field actions of the pipeline.

[0038] In the embodiments, the current time is stored in packet headers (i.e., the NSH) of received traffic by copying the current value from the global timestamp packet register into the packet. This may be done using flow control protocol (e.g., OpenFlow) copy-instructions. The copy-instructions support and enable copying a value from any packet register to any packet header field, including from the global timestamp packet register to a NSH. An SDN controller uses this copy-instruction with the global timestamp packet register, as required, to update timestamp in desired packet headers by creating flow table entries that match for desired packet criteria and as an action copy the value from the global timestamp packet register into the header.

[0039] Unlike a typical packet register, this global timestamp packet register cannot be modified (set-to) by the packet processing pipeline. The value of global timestamp packet register is updated only by a switch internal time management mechanism (e.g., a time manager outside the flow control packet processing pipeline). The switch (e.g., the time manager) uses an underlying set of RTCs (Real-time Clocks), to derive and set a value of the global timestamp packet register. In some embodiments, a RTC in the architecture has a value that is readily usable as a timestamp to be inserted into the global timestamp packet register. However, in other embodiments, time may be tracked in multiple RTCs and/or may require some computation to derive from the available RTCs to make it suitable for use in the global timestamp packet. For example, in some systems a first RTC tracks a time that a device is activated, while a second RTC tracks the time elapsed since the activation. In such a case, the process may combine these values to derive a value to be used in the global timestamp packet register.

[0040] Figure 1 is a diagram of one embodiment of a SDN network including a switch configured to support the timestamp insertion process. The process and mechanism for the timestamp insertion process are implemented in the example embodiment in a switch 105 in an SDN network 103. The switch 105 may be in communication with a controller 101 in the SDN network 103. The controller 101 configures the flow table 125 of the switch 101 for processing of the traffic received by the switch 105. The configuration of the flow table 125 can include adding a flow table entry to define the criteria for adding a timestamp into a packet. The flow table entry includes the matching criteria 127B and the copy instructions 127A for identifying the packets. Any set of packets and packet criteria can be set including adding the timestamp to all received traffic or all traffic associated with a service chain.

[0041] The copy instruction 127A in the flow table entry takes the value from the global timestamp packet register 115 and adds it to the header of a received packet 113A to generate an updated packet 113B with the timestamp value. The received packet 113A may be received through any ingress port 107 of the switch 105 and may be forwarded via any egress port 109.

[0042] The global timestamp packet register 115 is a protected register from a set of packet registers in the flow control pipeline 111. The global timestamp packet register 115 can only be modified or updated by a time manager 117. The time manager 117 can be a process that is internal or external to the flow control pipeline 111. The time manager 117 may call a system or operating system application programming interface (API) 119 or similar interface to obtain current time values from a set of local real-time clocks (RTCs) 121 or similar time sources. In some embodiments, the time manager 117 can poll the RTC at set intervals, such as every millisecond or similar time increment to ensure a current time value is in the global timestamp packet register at any time. In other embodiments, the value to be placed in the global timestamp packet register must be derived from the available RTC values depending on the mechanism of the available RTCs. The time manager 117 can be configured to handle any type of RTC data to generate a value for the global timestamp packet register 115 that represents a current time to indicate a time of arrival or processing of a packet at a network device.

[0043] In one embodiment, the switch 105 is an OpenFlow switch that supports a global timestamp packet register 115. This OpenFlow packet register is a reserved register for exposing the real-time clock. In this example, the RTC is a register within the switch that is part of the chipset and available via operating system calls. The RTC is thereby used to provide the time value to the global timestamp packet register. Since the time is available via OpenFlow packet register, and OpenFlow supports copying the value from packet registers to packet header fields, this becomes a generic interface available for an OpenFlow controller to store the current time in packet header fields by creating a flow table entry.

[0044] In one example embodiment, each packet register in the flow control pipeline 111 is 64 bit wide and maskable. A switch 105 may optionally implement any number of packet registers. The embodiments reserve one packet-register as the global timestamp packet register 115. The value of this global timestamp packet register 115 can be copied to any of the other registers or other packet header fields as specified in a copy instruction. In some embodiments, the timestamp packet register may be associated with each packet (similar to other packet registers), but can also be described as the global timestamp packet register that is common to all packet processing pipelines. This is because, unlike other packet registers that can be individually set during pipeline processing, this packet register is not set during packet pipeline processing. This packet register is set using the switch 105 internal mechanism of the time manager 117.

[0045] The embodiments thus provide advantages over the existing solutions. The SDN controller and network applications (running on top of controller) can add a timestamp to packet headers in programmatic manner. This allows for development and improvement in the accuracy of SDN applications for network delay measurement, path optimization VNF placements in cloud environments. The addition of timestamps to packet headers can be done as a regular packet processing instruction supported by a flow control protocol such as OpenFlow, by the SDN controller. No additional hardware is required in the switch, since the underlying hardware and API support such a configuration in many switch architectures and operating systems.

[0046] The operations in the flow diagrams will be described with reference to the exemplary embodiments of the other figures. However, it should be understood that the operations of the flow diagrams can be performed by embodiments of the invention other than those discussed with reference to the other figures, and the embodiments of the invention discussed with reference to these other figures can perform operations different than those discussed with reference to the flow diagrams.

[0047] Figures 2A and 2B are flowcharts of the processes implemented on the switch to insert timestamps and improve network performance monitoring. The process of Figure 2A is implemented by the flow control pipeline as configured for this purpose by the SDN controller. The process is initiated when the switch receives a packet on an ingress port (Block 201). The pipeline determines whether the packet meets the matching criteria of a flow table entry (Block 203). If the received packet does not meet the matching criteria of the flow table entry related to the timestamp insertion process, then the packet may be further processed by the flow control pipeline and then forwarded by the switch via the appropriate egress port toward its destination (Block 207).

[0048] If the received packet does match the criteria for timestamp insertion, then the process executes the associated action which is to copy the current timestamp value of the global timestamp packet register into the packet header (Block 205). The matching criteria can be any matching criteria including identifying specific service chains or identifying all received packets. Once the global timestamp packet register value is inserted into the packet header, then the packet can be forwarded toward its destination via the appropriate egress port (Block 207). Any additional flow control pipeline processing can be performed either before or after the processing related to the timestamp insertion process and before the forwarding of the packet.

[0049] In some embodiments, the process may further send a copy of the timestamp or the packet with the timestamp to a SDN controller or a monitoring application to enable the SDN controller or the monitoring application to monitor the performance metrics of the SDN network. The implementing network device may be at the edge of the SDN network and be handling the ingress or egress of traffic in the network. In some embodiments, the network device may be internal to the SDN network and the timestamps can be used for subsections of the SDN network.

[0050] Figure 2B is a flowchart of the process of the time manager in maintaining the value of the global timestamp packet register. The time manager can be configured to query or poll the RTC to get or derive an updated value at any interval based on the desired level of accuracy in the timestamp value. The time manager checks to determine whether the defined update interval has expired (Block 251). If the interval has not expired, then the process continues to wait until the interval expires. If the time interval has expired, then the time manager calls an API of the switch or a system call of the operating system to obtain a current RTC value or set of RTC values (Block 253). In some embodiments, where the calls for the current RTC value(s) need to be further processed, a computation of a timestamp value is performed (Block 255). The computed timestamp value can then be written to the global timestamp packet register thereby overwriting and replacing the prior value (Block 257). The global timestamp packet register may have write privileges restricted to the time manager to prevent other processes from modifying it.

[0051] Figure 3 is a diagram of one embodiment of the process for configuring the switches in an SDN network for network performance monitoring and for collecting the results. The process may be initiated where an administrator wants to enable or start network performance monitoring, in particular network delay monitoring. The controller or component thereof generates a configuration message to be sent to a particular switch in the SDN network or a set of switches (Block 301). The configuration message may be sent to a switch that is an entry point and/or a switch that is an exit point such that each inserts and/or reports timestamps associated with traffic traversing the network and in some cases specific service chains in the network.

[0052] The process inserts a set of matching criteria for timestamp insertion operations into the configuration message (Block 303). Any matching criteria can be specified. The configuration message can also include a specified action of a copy instruction that will copy the contents of a local global timestamp packet register into a specified header portion (e.g., a sub field of the NSH) of matching packets received at the switch being configured. The constructed configuration message is then forwarded to the entry point and/or exit point switch to be configured (Block 305). When the switches are configured, then they will modify received packets by inserting timestamps into their header information such as in the NSH. The controller and/or a monitoring application operating on the controller or remotely from the controller may then receive copies of the packets or the header information after they are modified, for example upon exit from a service chain or exit from the network (Block 307). The monitoring application or controller can then compare the entry point timestamp with the exit point timestamp to determine latency or network delay (Block 309).

[0053] Figure 4 is a diagram of one example application of the timestamp insertion process in an SDN network. In this example implementation, an SDN controller 401 is in communication with a switch 1 405 that is an entry point for an entry for the provider network 409, which may include a service chain in some cases. The SDN controller 401 is also in communication with a switch 2 407 that is an exit point for the provider network. The SDN controller 401 adds a flow table entry at switch 1 405 and switch 2 407. The switch 2 407 sends a copy of the timestamp information to a monitoring application 403 to determine network performance metrics, specifically network delay across the provider network 409.

[0054] Further embodiments are discussed below to provide sample flow rules using the global timestamp packet register, to achieve a delay analysis application for Service Chains. In one example, the flow rules may use the existing open virtual switch (OVS) Nicira extensions for network service header (NSH) operations to demonstrate the solution.

[0055] At the service plane entry point 405, a timestamp can be copied to a specific NSH field.

```
ovs-ofctl add-flow br-int "table=0, priority=260, in_port=1
actions=load:0x800001->NXM_NX_NSP[],
load:0xFF->NXM_NX_NSI[],
load:2->NXM_NX_NSH_MDTYPE[],
load:nxm_reg_1->NXM_NX_NSH_METADATA_1[],
push_nsh,output:2
```

[0056] This rule causes a flow to be set up in table 0 (at priority of 260). The match rule is all packets coming from input port1. The action associated with the packet is to update a NSH header with service path 0x800001, update a NSH header with service index 0xff, and MD_type as 2, update first portion of metadata field of NSH header with the value from global timestamp packet register (register-1), and send out the packet from output port2.

[0057] At the service plane exit point, timestamp at the exit switch can be copied to another distinct field of NSH:

```
ovs-ofctl add-flow br-int "table=0, priority=260, in_port=5
actions=load:0x800001->NXM_NX_NSP[],
load:0xFF->NXM_NX_NSI[],
load:2->NXM_NX_NSH_MDTYPE[],
load:nxm_reg_1->NXM_NX_NSH_METADATA_2[],
push_nsh,output:6
```

[0058] This rule causes a flow to be set up in table 0 (at priority of 260). The match rule is all packets coming from input port5. The action associated with the packet is to update a NSH header with service path 0x800001, update a NSH header with service index 0xff and MD-type as 2, update second portion of metadata field of NSH header with the value from global timestamp packet register (register-1), and send out the packet from output port6. The packet at the exit can be copied and fed to a delay calculation application which can reside within or outside the controller 401, which can analyze the timestamps and calculate accurate delays.

[0059] For ease of illustration, the solution is described in terms of timestamping all packets coming from a port. This can however be made more granular using granular match actions supported by OF protocols (such as timestamping packets coming from specific source-ip, source-port, etc.).

[0060] The embodiments as described provide a method and system that have benefits over existing solutions. Existing delay measurements such as TWAMP are unsuited for delay measurements in service chained networks due to their dependence on generated packets. The embodiments make use of OpenFlow constructs to timestamp a packet. This innovation measures delay of actual packets, without requiring any special headers or OpenFlow extensions.

[0061] The embodiments provide timestamping capability using a packet register construct via a flow control protocol such as OpenFlow. This packet register, in turn, gets the current time of day from underlying hardware such as the RTC register. Such RTC registers are common in many hardware. Such RTC registers are also made available by operating system calls. Thus, the solution can be used in a wide array of systems having varying operating systems and hardware configurations.

[0062] Figure 5A illustrates connectivity between network devices (NDs) within an exemplary network, as well as three exemplary implementations of the NDs, according to some embodiments of the invention. Figure 5A shows NDs 500A-H, and their connectivity by way of lines between 500A-500B, 500B-500C, 500C-500D, 500D-500E, 500E-500F, 500F-500G, and 500A-500G, as well as between 500H and each of 500A, 500C, 500D, and 500G. These NDs are physical devices, and the connectivity between these NDs can be wireless or wired (often referred to as a link). An additional line extending from NDs 500A, 500E, and 500F illustrates that these NDs act as ingress and egress points for the network (and thus, these NDs are sometimes referred to as edge NDs; while the other NDs may be called core NDs).

[0063] Two of the exemplary ND implementations in Figure 5A are: 1) a special-purpose network device 502 that uses custom application–specific integrated–circuits (ASICs) and a special-purpose operating system (OS); and 2) a general purpose network device 504 that uses common off-the-shelf (COTS) processors and a standard OS.

[0064] The special-purpose network device 502 includes networking hardware 510 comprising compute resource(s) 512 (which typically include a set of one or more processors), forwarding resource(s) 514 (which typically include one or more ASICs and/or network processors), and physical network interfaces (NIs) 516 (sometimes called physical ports), as well as non-transitory machine readable storage media 518 having stored therein networking software 520. A physical NI is hardware in a ND through which a network connection (e.g., wirelessly through a wireless network interface controller (WNIC) or through plugging in a cable to a physical port connected to a network interface controller (NIC)) is made, such as those shown by the connectivity between NDs 500A-H. During operation, the networking software 520 may be executed by the networking hardware 510 to instantiate a set of one or more networking software instance(s) 522. Each of the networking software instance(s) 522, and that part of the networking hardware 510 that executes that network software instance (be it hardware dedicated to that networking software instance and/or time slices of hardware temporally shared by that networking software instance with others of the networking software instance(s) 522), form a separate virtual network element 530A-R. Each of the virtual network element(s) (VNEs) 530A-R includes a control communication and configuration module 532A-R (sometimes referred to as a local control module or control communication module) and forwarding table(s) 534A-R, such that a given virtual network element (e.g., 530A) includes the control communication and configuration module (e.g., 532A), a set of one or more forwarding table(s) (e.g., 534A), and that portion of the networking hardware 510 that executes the virtual network element (e.g., 530A). The flow control pipeline 535A-R and time manager 537A-R may implement the functions described herein including the updating of packets with timestamps using the global timestamp packet register by the flow control pipeline 535A-R and the updating of this register by the time manager 537A-R.

[0065] The special-purpose network device 502 is often physically and/or logically considered to include: 1) a ND control plane 524 (sometimes referred to as a control plane) comprising the compute resource(s) 512 that execute the control communication and configuration module(s) 532A-R; and 2) a ND forwarding plane 526 (sometimes referred to as a forwarding plane, a data plane, or a media plane) comprising the forwarding resource(s) 514 that utilize the forwarding table(s) 534A-R and the physical NIs 516. By way of example, where the ND is a router (or is implementing routing functionality), the ND control plane 524 (the compute resource(s) 512 executing the control communication and configuration module(s) 532A-R) is typically responsible for participating in controlling how data (e.g., packets) is to be routed (e.g., the next hop for the data and the outgoing physical NI for that data) and storing that routing information in the forwarding table(s) 534A-R, and the ND forwarding plane 526 is responsible for receiving that data on the physical NIs 516 and forwarding that data out the appropriate ones of the physical NIs 516 based on the forwarding table(s) 534A-R.

[0066] Figure 5B illustrates an exemplary way to implement the special-purpose network device 502 according to some embodiments of the invention. Figure 5B shows a special-purpose network device including cards 538 (typically hot pluggable). While in some embodiments the cards 538 are of two types (one or more that operate as the ND forwarding plane 526 (sometimes called line cards), and one or more that operate to implement the ND control plane 524 (sometimes called control cards)), alternative embodiments may combine functionality onto a single card and/or include additional card types (e.g., one additional type of card is called a service card, resource card, or multi-application card). A service card can provide specialized processing (e.g., Layer 4 to Layer 7 services (e.g., firewall, Internet Protocol Security (IPsec), Secure Sockets Layer (SSL) / Transport Layer Security (TLS), Intrusion Detection System (IDS), peer-to-peer (P2P), Voice over IP (VoIP) Session Border Controller, Mobile Wireless Gateways (Gateway General Packet Radio Service (GPRS) Support Node (GGSN), Evolved Packet Core (EPC) Gateway)). By way of example, a service card may be used to terminate IPsec tunnels and execute the attendant authentication and encryption algorithms. These cards are coupled together through one or more interconnect mechanisms illustrated as backplane 536 (e.g., a first full mesh coupling the line cards and a second full mesh coupling all of the cards).

[0067] Returning to Figure 5A, the general purpose network device 504 includes hardware 540 comprising a set of one or more processor(s) 542 (which are often COTS processors) and network interface controller(s) 544 (NICs; also known as network interface cards) (which include physical NIs 546), as well as non-transitory machine readable storage media 548 having stored therein software 550. During operation, the processor(s) 542 execute the software 550 to instantiate one or more sets of one or more applications 564A-R. While one embodiment does not implement virtualization, alternative embodiments may use different forms of virtualization. For example, in one such alternative embodiment the virtualization layer 554 represents the kernel of an operating system (or a shim executing on a base operating system) that allows for the creation of multiple instances 562A-R called software containers that may each be used to execute one (or more) of the sets of applications 564A-R; where the multiple software containers (also called virtualization engines, virtual private servers, or jails) are user spaces (typically a virtual memory space) that are separate from each other and separate from the kernel space in which the operating system is run; and where the set of applications running in a given user space, unless explicitly allowed, cannot access the memory of the other processes. In another such alternative embodiment the virtualization layer 554 represents a hypervisor (sometimes referred to as a virtual machine monitor (VMM)) or a hypervisor executing on top of a host operating system, and each of the sets of applications 564A-R is run on top of a guest operating system within an instance 562A-R called a virtual machine (which may in some cases be considered a tightly isolated form of software container) that is run on top of the hypervisor - the guest operating system and application may not know they are running on a virtual machine as opposed to running on a "bare metal" host electronic device, or through para-virtualization the operating system and/or application may be aware of the presence of virtualization for optimization purposes. In yet other alternative embodiments, one, some or all of the applications are implemented as unikernel(s), which can be generated by compiling directly with an application only a limited set of libraries (e.g., from a library operating system (LibOS) including drivers/libraries of OS services) that provide the particular OS services needed by the application. As a unikernel can be implemented to run directly on hardware 540, directly on a hypervisor (in which case the unikernel is sometimes described as running within a LibOS virtual machine), or in a software container, embodiments can be implemented fully with unikernels running directly on a hypervisor represented by virtualization layer 554, unikernels running within software containers represented by instances 562A-R, or as a combination of unikernels and the above-described techniques (e.g., unikernels and virtual machines both run directly on a hypervisor, unikernels and sets of applications that are run in different software containers).

[0068] The instantiation of the one or more sets of one or more applications 564A-R, as well as virtualization if implemented, are collectively referred to as software instance(s) 552. Each set of applications 564A-R, corresponding virtualization construct (e.g., instance 562A-R) if implemented, and that part of the hardware 540 that executes them (be it hardware dedicated to that execution and/or time slices of hardware temporally shared), forms a separate virtual network element(s) 560A-R. The flow control pipeline 565A-R and time manager 567A-R may implement the functions described herein including the updating of packets with timestamps using the global timestamp packet register by the flow control pipeline 565A-R and the updating of this register by the time manager 567A-R.

[0069] The virtual network element(s) 560A-R perform similar functionality to the virtual network element(s) 530A-R - e.g., similar to the control communication and configuration module(s) 532A and forwarding table(s) 534A (this virtualization of the hardware 540 is sometimes referred to as network function virtualization (NFV)). Thus, NFV may be used to consolidate many network equipment types onto industry standard high volume server hardware, physical switches, and physical storage, which could be located in Data centers, NDs, and customer premise equipment (CPE). While embodiments of the invention are illustrated with each instance 562A-R corresponding to one VNE 560A-R, alternative embodiments may implement this correspondence at a finer level granularity (e.g., line card virtual machines virtualize line cards, control card virtual machine virtualize control cards, etc.); it should be understood that the techniques described herein with reference to a correspondence of instances 562A-R to VNEs also apply to embodiments where such a finer level of granularity and/or unikernels are used.

[0070] In certain embodiments, the virtualization layer 554 includes a virtual switch that provides similar forwarding services as a physical Ethernet switch. Specifically, this virtual switch forwards traffic between instances 562A-R and the NIC(s) 544, as well as optionally between the instances 562A-R; in addition, this virtual switch may enforce network isolation between the VNEs 560A-R that by policy are not permitted to communicate with each other (e.g., by honoring virtual local area networks (VLANs)).

[0071] The third exemplary ND implementation in Figure 5A is a hybrid network device 506, which includes both custom ASICs/special-purpose OS and COTS processors/standard OS in a single ND or a single card within an ND. In certain embodiments of such a hybrid network device, a platform VM (i.e., a VM that that implements the functionality of the special-purpose network device 502) could provide for para-virtualization to the networking hardware present in the hybrid network device 506.

[0072] Regardless of the above exemplary implementations of an ND, when a single one of multiple VNEs implemented by an ND is being considered (e.g., only one of the VNEs is part of a given virtual network) or where only a single VNE is currently being implemented by an ND, the shortened term network element (NE) is sometimes used to refer to that VNE. Also in all of the above exemplary implementations, each of the VNEs (e.g., VNE(s) 530A-R, VNEs 560A-R, and those in the hybrid network device 506) receives data on the physical NIs (e.g., 516, 546) and forwards that data out the appropriate ones of the physical NIs (e.g., 516, 546). For example, a VNE implementing IP router functionality forwards IP packets on the basis of some of the IP header information in the IP packet; where IP header information includes source IP address, destination IP address, source port, destination port (where “source port” and “destination port” refer herein to protocol ports, as opposed to physical ports of a ND), transport protocol (e.g., user datagram protocol (UDP), Transmission Control Protocol (TCP), and differentiated services code point (DSCP) values.

[0073] Figure 5C illustrates various exemplary ways in which VNEs may be coupled according to some embodiments of the invention. Figure 5C shows VNEs 570A.1-570A.P (and optionally VNEs 570A.Q-570A.R) implemented in ND 500A and VNE 570H.1 in ND 500H. In Figure 5C, VNEs 570A.1-P are separate from each other in the sense that they can receive packets from outside ND 500A and forward packets outside of ND 500A; VNE 570A.1 is coupled with VNE 570H.1, and thus they communicate packets between their respective NDs; VNE 570A.2-570A.3 may optionally forward packets between themselves without forwarding them outside of the ND 500A; and VNE 570A.P may optionally be the first in a chain of VNEs that includes VNE 570A.Q followed by VNE 570A.R (this is sometimes referred to as dynamic service chaining, where each of the VNEs in the series of VNEs provides a different service – e.g., one or more layer 4-7 network services). While Figure 5C illustrates various exemplary relationships between the VNEs, alternative embodiments may support other relationships (e.g., more/fewer VNEs, more/fewer dynamic service chains, multiple different dynamic service chains with some common VNEs and some different VNEs).

[0074] The NDs of Figure 5A, for example, may form part of the Internet or a private network; and other electronic devices (not shown; such as end user devices including workstations, laptops, netbooks, tablets, palm tops, mobile phones, smartphones, phablets, multimedia phones, Voice Over Internet Protocol (VOIP) phones, terminals, portable media players, GPS units, wearable devices, gaming systems, set-top boxes, Internet enabled household appliances) may be coupled to the network (directly or through other networks such as access networks) to communicate over the network (e.g., the Internet or virtual private networks (VPNs) overlaid on (e.g., tunneled through) the Internet) with each other (directly or through servers) and/or access content and/or services. Such content and/or services are typically provided by one or more servers (not shown) belonging to a service/content provider or one or more end user devices (not shown) participating in a peer-to-peer (P2P) service, and may include, for example, public webpages (e.g., free content, store fronts, search services), private webpages (e.g., username/password accessed webpages providing email services), and/or corporate networks over VPNs. For instance, end user devices may be coupled (e.g., through customer premise equipment coupled to an access network (wired or wirelessly)) to edge NDs, which are coupled (e.g., through one or more core NDs) to other edge NDs, which are coupled to electronic devices acting as servers. However, through compute and storage virtualization, one or more of the electronic devices operating as the NDs in Figure 5A may also host one or more such servers (e.g., in the case of the general purpose network device 504, one or more of the software instances 562A-R may operate as servers; the same would be true for the hybrid network device 506; in the case of the special-purpose network device 502, one or more such servers could also be run on a virtualization layer executed by the compute resource(s) 512); in which case the servers are said to be co-located with the VNEs of that ND.

[0075] A virtual network is a logical abstraction of a physical network (such as that in Figure 5A) that provides network services (e.g., L2 and/or L3 services). A virtual network can be implemented as an overlay network (sometimes referred to as a network virtualization overlay) that provides network services (e.g., layer 2 (L2, data link layer) and/or layer 3 (L3, network layer) services) over an underlay network (e.g., an L3 network, such as an Internet Protocol (IP) network that uses tunnels (e.g., generic routing encapsulation (GRE), layer 2 tunneling protocol (L2TP), IPSec) to create the overlay network).

[0076] A network virtualization edge (NVE) sits at the edge of the underlay network and participates in implementing the network virtualization; the network-facing side of the NVE uses the underlay network to tunnel frames to and from other NVEs; the outward-facing side of the NVE sends and receives data to and from systems outside the network. A virtual network instance (VNI) is a specific instance of a virtual network on a NVE (e.g., a NE/VNE on an ND, a part of a NE/VNE on a ND where that NE/VNE is divided into multiple VNEs through emulation); one or more VNIs can be instantiated on an NVE (e.g., as different VNEs on an ND). A virtual access point (VAP) is a logical connection point on the NVE for connecting external systems to a virtual network; a VAP can be physical or virtual ports identified through logical interface identifiers (e.g., a VLAN ID).

[0077] Examples of network services include: 1) an Ethernet LAN emulation service (an Ethernet-based multipoint service similar to an Internet Engineering Task Force (IETF) Multiprotocol Label Switching (MPLS) or Ethernet VPN (EVPN) service) in which external systems are interconnected across the network by a LAN environment over the underlay network (e.g., an NVE provides separate L2 VNIs (virtual switching instances) for different such virtual networks, and L3 (e.g., IP/MPLS) tunneling encapsulation across the underlay network); and 2) a virtualized IP forwarding service (similar to IETF IP VPN (e.g., Border Gateway Protocol (BGP)/MPLS IPVPN) from a service definition perspective) in which external systems are interconnected across the network by an L3 environment over the underlay network (e.g., an NVE provides separate L3 VNIs (forwarding and routing instances) for different such virtual networks, and L3 (e.g., IP/MPLS) tunneling encapsulation across the underlay network)). Network services may also include quality of service capabilities (e.g., traffic classification marking, traffic conditioning and scheduling), security capabilities (e.g., filters to protect customer premises from network – originated attacks, to avoid malformed route announcements), and management capabilities (e.g., full detection and processing).

[0078] Fig. 5D illustrates a network with a single network element on each of the NDs of Figure 5A, and within this straight forward approach contrasts a traditional distributed approach (commonly used by traditional routers) with a centralized approach for maintaining reachability and forwarding information (also called network control), according to some embodiments of the invention. Specifically, Figure 5D illustrates network elements (NEs) 570A-H with the same connectivity as the NDs 500A-H of Figure 5A.

[0079] Figure 5D illustrates that the distributed approach 572 distributes responsibility for generating the reachability and forwarding information across the NEs 570A-H; in other words, the process of neighbor discovery and topology discovery is distributed.

[0080] For example, where the special-purpose network device 502 is used, the control communication and configuration module(s) 532A-R of the ND control plane 524 typically include a reachability and forwarding information module to implement one or more routing protocols (e.g., an exterior gateway protocol such as Border Gateway Protocol (BGP), Interior Gateway Protocol(s) (IGP) (e.g., Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), Routing Information Protocol (RIP), Label Distribution Protocol (LDP), Resource Reservation Protocol (RSVP) (including RSVP-Traffic Engineering (TE): Extensions to RSVP for LSP Tunnels and Generalized Multi-Protocol Label Switching (GMPLS) Signaling RSVP-TE)) that communicate with other NEs to exchange routes, and then selects those routes based on one or more routing metrics. Thus, the NEs 570A-H (e.g., the compute resource(s) 512 executing the control communication and configuration module(s) 532A-R) perform their responsibility for participating in controlling how data (e.g., packets) is to be routed (e.g., the next hop for the data and the outgoing physical NI for that data) by distributively determining the reachability within the network and calculating their respective forwarding information. Routes and adjacencies are stored in one or more routing structures (e.g., Routing Information Base (RIB), Label Information Base (LIB), one or more adjacency structures) on the ND control plane 524. The ND control plane 524 programs the ND forwarding plane 526 with information (e.g., adjacency and route information) based on the routing structure(s). For example, the ND control plane 524 programs the adjacency and route information into one or more forwarding table(s) 534A-R (e.g., Forwarding Information Base (FIB), Label Forwarding Information Base (LFIB), and one or more adjacency structures) on the ND forwarding plane 526. For layer 2 forwarding, the ND can store one or more bridging tables that are used to forward data based on the layer 2 information in that data. While the above example uses the special-purpose network device 502, the same distributed approach 572 can be implemented on the general purpose network device 504 and the hybrid network device 506.

[0081] Figure 5D illustrates that a centralized approach 574 (also known as software defined networking (SDN)) that decouples the system that makes decisions about where traffic is sent from the underlying systems that forwards traffic to the selected destination. The illustrated centralized approach 574 has the responsibility for the generation of reachability and forwarding information in a centralized control plane 576 (sometimes referred to as a SDN control module, controller, network controller, OpenFlow controller, SDN controller, control plane node, network virtualization authority, or management control entity), and thus the process of neighbor discovery and topology discovery is centralized. The centralized control plane 576 has a south bound interface 582 with a data plane 580 (sometime referred to the infrastructure layer, network forwarding plane, or forwarding plane (which should not be confused with a ND forwarding plane)) that includes the NEs 570A-H (sometimes referred to as switches, forwarding elements, data plane elements, or nodes). The centralized control plane 576 includes a network controller 578, which includes a centralized reachability and forwarding information module 579 that determines the reachability within the network and distributes the forwarding information to the NEs 570A-H of the data plane 580 over the south bound interface 582 (which may use the OpenFlow protocol). Thus, the network intelligence is centralized in the centralized control plane 576 executing on electronic devices that are typically separate from the NDs.

[0082] For example, where the special-purpose network device 502 is used in the data plane 580, each of the control communication and configuration module(s) 532A-R of the ND control plane 524 typically include a control agent that provides the VNE side of the south bound interface 582. In this case, the ND control plane 524 (the compute resource(s) 512 executing the control communication and configuration module(s) 532A-R) performs its responsibility for participating in controlling how data (e.g., packets) is to be routed (e.g., the next hop for the data and the outgoing physical NI for that data) through the control agent communicating with the centralized control plane 576 to receive the forwarding information (and in some cases, the reachability information) from the centralized reachability and forwarding information module 579 (it should be understood that in some embodiments of the invention, the control communication and configuration module(s) 532A-R, in addition to communicating with the centralized control plane 576, may also play some role in determining reachability and/or calculating forwarding information – albeit less so than in the case of a distributed approach; such embodiments are generally considered to fall under the centralized approach 574, but may also be considered a hybrid approach).

[0083] While the above example uses the special-purpose network device 502, the same centralized approach 574 can be implemented with the general purpose network device 504 (e.g., each of the VNE 560A-R performs its responsibility for controlling how data (e.g., packets) is to be routed (e.g., the next hop for the data and the outgoing physical NI for that data) by communicating with the centralized control plane 576 to receive the forwarding information (and in some cases, the reachability information) from the centralized reachability and forwarding information module 579; it should be understood that in some embodiments of the invention, the VNEs 560A-R, in addition to communicating with the centralized control plane 576, may also play some role in determining reachability and/or calculating forwarding information – albeit less so than in the case of a distributed approach) and the hybrid network device 506. In fact, the use of SDN techniques can enhance the NFV techniques typically used in the general purpose network device 504 or hybrid network device 506 implementations as NFV is able to support SDN by providing an infrastructure upon which the SDN software can be run, and NFV and SDN both aim to make use of commodity server hardware and physical switches.

[0084] Figure 5D also shows that the centralized control plane 576 has a north bound interface 584 to an application layer 586, in which resides application(s) 588. The centralized control plane 576 has the ability to form virtual networks 592 (sometimes referred to as a logical forwarding plane, network services, or overlay networks (with the NEs 570A-H of the data plane 580 being the underlay network)) for the application(s) 588. Thus, the centralized control plane 576 maintains a global view of all NDs and configured NEs/VNEs, and it maps the virtual networks to the underlying NDs efficiently (including maintaining these mappings as the physical network changes either through hardware (ND, link, or ND component) failure, addition, or removal). The Application(s) 588 may include a timestamp configuration 581 that communicates with the switches implementing the timestamping process. The timestamp configuration 581 sends the configuration information including the matching criteria and instructions for inserting timestamps using the flow control pipeline.

[0085] While Figure 5D shows the distributed approach 572 separate from the centralized approach 574, the effort of network control may be distributed differently or the two combined in certain embodiments of the invention. For example: 1) embodiments may generally use the centralized approach (SDN) 574, but have certain functions delegated to the NEs (e.g., the distributed approach may be used to implement one or more of fault monitoring, performance monitoring, protection switching, and primitives for neighbor and/or topology discovery); or 2) embodiments of the invention may perform neighbor discovery and topology discovery via both the centralized control plane and the distributed protocols, and the results compared to raise exceptions where they do not agree. Such embodiments are generally considered to fall under the centralized approach 574, but may also be considered a hybrid approach.

[0086] While Figure 5D illustrates the simple case where each of the NDs 500A-H implements a single NE 570A-H, it should be understood that the network control approaches described with reference to Figure 5D also work for networks where one or more of the NDs 500A-H implement multiple VNEs (e.g., VNEs 530A-R, VNEs 560A-R, those in the hybrid network device 506). Alternatively or in addition, the network controller 578 may also emulate the implementation of multiple VNEs in a single ND. Specifically, instead of (or in addition to) implementing multiple VNEs in a single ND, the network controller 578 may present the implementation of a VNE/NE in a single ND as multiple VNEs in the virtual networks 592 (all in the same one of the virtual network(s) 592, each in different ones of the virtual network(s) 592, or some combination). For example, the network controller 578 may cause an ND to implement a single VNE (a NE) in the underlay network, and then logically divide up the resources of that NE within the centralized control plane 576 to present different VNEs in the virtual network(s) 592 (where these different VNEs in the overlay networks are sharing the resources of the single VNE/NE implementation on the ND in the underlay network).

[0087] On the other hand, Figures 5E and 5F respectively illustrate exemplary abstractions of NEs and VNEs that the network controller 578 may present as part of different ones of the virtual networks 592. Figure 5E illustrates the simple case of where each of the NDs 500A-H implements a single NE 570A-H (see Figure 5D), but the centralized control plane 576 has abstracted multiple of the NEs in different NDs (the NEs 570A-C and G-H) into (to represent) a single NE 570I in one of the virtual network(s) 592 of Figure 5D, according to some embodiments of the invention. Figure 5E shows that in this virtual network, the NE 570I is coupled to NE 570D and 570F, which are both still coupled to NE 570E.

[0088] Figure 5F illustrates a case where multiple VNEs (VNE 570A.1 and VNE 570H.1) are implemented on different NDs (ND 500A and ND 500H) and are coupled to each other, and where the centralized control plane 576 has abstracted these multiple VNEs such that they appear as a single VNE 570T within one of the virtual networks 592 of Figure 5D, according to some embodiments of the invention. Thus, the abstraction of a NE or VNE can span multiple NDs.

[0089] While some embodiments of the invention implement the centralized control plane 576 as a single entity (e.g., a single instance of software running on a single electronic device), alternative embodiments may spread the functionality across multiple entities for redundancy and/or scalability purposes (e.g., multiple instances of software running on different electronic devices).

[0090] Similar to the network device implementations, the electronic device(s) running the centralized control plane 576, and thus the network controller 578 including the centralized reachability and forwarding information module 579, may be implemented a variety of ways (e.g., a special purpose device, a general-purpose (e.g., COTS) device, or hybrid device). These electronic device(s) would similarly include compute resource(s), a set or one or more physical NICs, and a non-transitory machine-readable storage medium having stored thereon the centralized control plane software. For instance, Figure 6 illustrates, a general purpose control plane device 604 including hardware 640 comprising a set of one or more processor(s) 642 (which are often COTS processors) and network interface controller(s) 644 (NICs; also known as network interface cards) (which include physical NIs 646), as well as non-transitory machine readable storage media 648 having stored therein centralized control plane (CCP) software 650.

[0091] In embodiments that use compute virtualization, the processor(s) 642 typically execute software to instantiate a virtualization layer 654 (e.g., in one embodiment the virtualization layer 654 represents the kernel of an operating system (or a shim executing on a base operating system) that allows for the creation of multiple instances 662A-R called software containers (representing separate user spaces and also called virtualization engines, virtual private servers, or jails) that may each be used to execute a set of one or more applications; in another embodiment the virtualization layer 654 represents a hypervisor (sometimes referred to as a virtual machine monitor (VMM)) or a hypervisor executing on top of a host operating system, and an application is run on top of a guest operating system within an instance 662A-R called a virtual machine (which in some cases may be considered a tightly isolated form of software container) that is run by the hypervisor ; in another embodiment, an application is implemented as a unikernel, which can be generated by compiling directly with an application only a limited set of libraries (e.g., from a library operating system (LibOS) including drivers/libraries of OS services) that provide the particular OS services needed by the application, and the unikernel can run directly on hardware 640, directly on a hypervisor represented by virtualization layer 654 (in which case the unikernel is sometimes described as running within a LibOS virtual machine), or in a software container represented by one of instances 662A-R). Again, in embodiments where compute virtualization is used, during operation an instance of the CCP software 650 (illustrated as CCP instance 676A) is executed (e.g., within the instance 662A) on the virtualization layer 654. In embodiments where compute virtualization is not used, the CCP instance 676A is executed, as a unikernel or on top of a host operating system, on the "bare metal" general purpose control plane device 604. The instantiation of the CCP instance 676A, as well as the virtualization layer 654 and instances 662A-R if implemented, are collectively referred to as software instance(s) 652.

[0092] In some embodiments, the CCP instance 676A includes a network controller instance 678. The network controller instance 678 includes a centralized reachability and forwarding information module instance 679 (which is a middleware layer providing the context of the network controller 578 to the operating system and communicating with the various NEs), and an CCP application layer 680 (sometimes referred to as an application layer) over the middleware layer (providing the intelligence required for various network operations such as protocols, network situational awareness, and user – interfaces). At a more abstract level, this

CCP application layer 680 within the centralized control plane 576 works with virtual network view(s) (logical view(s) of the network) and the middleware layer provides the conversion from the virtual networks to the physical view. The CCP application layer 680 may include a timestamp configuration 681 that communicates with the switches implementing the timestamping process. The timestamp configuration 681 sends the configuration information including the matching criteria and instructions for inserting timestamps using the flow control pipeline.

[0093] The centralized control plane 576 transmits relevant messages to the data plane 580 based on CCP application layer 680 calculations and middleware layer mapping for each flow. A flow may be defined as a set of packets whose headers match a given pattern of bits; in this sense, traditional IP forwarding is also flow–based forwarding where the flows are defined by the destination IP address for example; however, in other implementations, the given pattern of bits used for a flow definition may include more fields (e.g., 10 or more) in the packet headers. Different NDs/NEs/VNEs of the data plane 580 may receive different messages, and thus different forwarding information. The data plane 580 processes these messages and programs the appropriate flow information and corresponding actions in the forwarding tables (sometime referred to as flow tables) of the appropriate NE/VNEs, and then the NEs/VNEs map incoming packets to flows represented in the forwarding tables and forward packets based on the matches in the forwarding tables.

[0094] Standards such as OpenFlow define the protocols used for the messages, as well as a model for processing the packets. The model for processing packets includes header parsing, packet classification, and making forwarding decisions. Header parsing describes how to interpret a packet based upon a well-known set of protocols. Some protocol fields are used to build a match structure (or key) that will be used in packet classification (e.g., a first key field could be a source media access control (MAC) address, and a second key field could be a destination MAC address).

[0095] Packet classification involves executing a lookup in memory to classify the packet by determining which entry (also referred to as a forwarding table entry or flow entry) in the forwarding tables best matches the packet based upon the match structure, or key, of the forwarding table entries. It is possible that many flows represented in the forwarding table entries can correspond/match to a packet; in this case the system is typically configured to determine one forwarding table entry from the many according to a defined scheme (e.g., selecting a first forwarding table entry that is matched). Forwarding table entries include both a specific set of match criteria (a set of values or wildcards, or an indication of what portions of a packet should be compared to a particular value/values/wildcards, as defined by the matching capabilities – for specific fields in the packet header, or for some other packet content; this may be any flow control protocol (e.g., OpenFlow) supported matching criteria that correlates with the actions in the table), and a set of one or more actions for the data plane to take on receiving a matching packet. For example, an action may be to push a header onto the packet, for the packet using a particular port, flood the packet, or simply drop the packet. Thus, a forwarding table entry for IPv4/IPv6 packets with a particular transmission control protocol (TCP) destination port could contain an action specifying that these packets should be dropped.

[0096] Making forwarding decisions and performing actions occurs, based upon the forwarding table entry identified during packet classification, by executing the set of actions identified in the matched forwarding table entry on the packet.

[0097] However, when an unknown packet (for example, a "missed packet" or a "match-miss" as used in OpenFlow parlance) arrives at the data plane 580, the packet (or a subset of the packet header and content) is typically forwarded to the centralized control plane 576. The centralized control plane 576 will then program forwarding table entries into the data plane 580 to accommodate packets belonging to the flow of the unknown packet. Once a specific forwarding table entry has been programmed into the data plane 580 by the centralized control plane 576, the next packet with matching credentials will match that forwarding table entry and take the set of actions associated with that matched entry.

[0098] A network interface (NI) may be physical or virtual; and in the context of IP, an interface address is an IP address assigned to a NI, be it a physical NI or virtual NI. A virtual NI may be associated with a physical NI, with another virtual interface, or stand on its own (e.g., a loopback interface, a point-to-point protocol interface). A NI (physical or virtual) may be numbered (a NI with an IP address) or unnumbered (a NI without an IP address). A loopback interface (and its loopback address) is a specific type of virtual NI (and IP address) of a NE/VNE (physical or virtual) often used for management purposes; where such an IP address is referred to as the nodal loopback address. The IP address(es) assigned to the NI(s) of a ND are referred to as IP addresses of that ND; at a more granular level, the IP address(es) assigned to NI(s) assigned to a NE/VNE implemented on a ND can be referred to as IP addresses of that NE/VNE.

[0099] Next hop selection by the routing system for a given destination may resolve to one path (that is, a routing protocol may generate one next hop on a shortest path); but if the routing system determines there are multiple viable next hops (that is, the routing protocol generated forwarding solution offers more than one next hop on a shortest path – multiple equal cost next hops), some additional criteria is used - for instance, in a connectionless network, Equal Cost Multi Path (ECMP) (also known as Equal Cost Multi Pathing, multipath forwarding and IP multipath) may be used (e.g., typical implementations use as the criteria particular header fields to ensure that the packets of a particular packet flow are always forwarded on the same next hop to preserve packet flow ordering). For purposes of multipath forwarding, a packet flow is defined as a set of packets that share an ordering constraint. As an example, the set of packets in a particular TCP transfer sequence need to arrive in order, else the TCP logic will interpret the out of order delivery as congestion and slow the TCP transfer rate down.

[00100] While the invention has been described in terms of several embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments described, can be practiced with modification and alteration within the spirit and scope of the appended claims. The description is thus to be regarded as illustrative instead of limiting.

CLAIMS

What is claimed is:

1. A method implemented by a network device, the method for insertion of a timestamp into a packet header to enable performance metrics in a software defined networking (SDN) network, the method comprising:

    receiving a packet on an ingress port of the network device;

    determining whether the received packet matches timestamp criteria;

    upon determining that the received packet matches the timestamp criteria, copying a value in a global timestamp packet register in a flow control pipeline into a header of the received packet; and

    forwarding the received packet toward a destination.

2. The method of claim 1, wherein the value in the global timestamp packet register is copied into a network service header field in the received packet.

3. The method of claim 1, further comprising:

    updating the value in the global timestamp packet register with a timestamp value derived from a real-time clock of the network device.

4. The method of claim 1, further comprising:

    receiving the timestamp criteria from a configuration message from a controller of the SDN network.

5. The method of claim 1, further comprising:

    sending a copy of timestamp information for the received packet to a monitoring application for network performance evaluation.

6. The method of claim 3, further comprising:

    computing the timestamp value using a set of values received from a call to obtain values for the real-time clock.

7. The method of claim 1, wherein the global timestamp packet register is an OpenFlow packet register.

8. A network device configured to implement a method for insertion of a timestamp into a packet header to enable performance metrics in a software defined networking (SDN) network, the network device comprising:

at least one real-time clock; and

a flow control pipeline including a global timestamp packet register, the flow control pipeline communicatively coupled to the at least one real-time clock, the flow control pipeline configured to receive a packet on an ingress port of the network device, to determine whether the received packet matches timestamp criteria, upon determining that the received packet matches the timestamp criteria, to copy a value in the global timestamp packet register into a header of the received packet, and to forward the received packet toward a destination.

9. The network device of claim 8, wherein the flow control pipeline is further configured to copy the value in the global timestamp packet register into a network service header field in the received packet.

10. The network device of claim 8, further comprising:

a time manager coupled to the flow control pipeline, the time manager to update the value in the global timestamp packet register with a timestamp value derived from the at least one real-time clock of the network device.

11. The network device of claim 8, wherein the flow control pipeline is further configured to receive the timestamp criteria from a configuration message from a controller of the SDN network.

12. The network device of claim 8, wherein the flow control pipeline is further configured to send a copy of timestamp information for the received packet to a monitoring application for network performance evaluation.

13. The network device of claim 10, wherein the time manager is further configured to compute the timestamp value using a set of values received from a call to obtain values for the at least one real-time clock.

14. The network device of claim 8, wherein the global timestamp packet register is an OpenFlow packet register.

15. A computing device in communication with a network device, the computing device to execute a plurality of virtual machines for implementing network function virtualization (NFV),

the computing device configured to implement a method for insertion of a timestamp into a packet header to enable performance metrics in a software defined networking (SDN) network, the computing device comprising:

at least one real-time clock;

a non-transitory computer-readable medium having stored therein a flow control pipeline module; and

a processor coupled to the at least one real-time clock and the non-transitory computer-readable medium, the processor configured to execute one of the plurality of virtual machines, the virtual machine configured to execute the flow control pipeline that includes a global timestamp packet register, the flow control pipeline configured to receive a packet on an ingress port of the network device, to determine whether the received packet matches timestamp criteria, upon determining that the received packet matches the timestamp criteria, to copy a value in the global timestamp packet register into a header of the received packet, and to forward the received packet toward a destination.

16. The computing device of claim 15, wherein the flow control pipeline is further configured to copy the value in the global timestamp packet register into a network service header field in the received packet.

17. The computing device of claim 15, wherein the processor is further configured to execute a time manager coupled to the flow control pipeline, the time manager to update the value in the global timestamp packet register with a timestamp value derived from the at least one real-time clock of a computing device.

18. The computing device of claim 15, wherein the flow control pipeline is further configured to receive the timestamp criteria from a configuration message from a controller of the SDN network.

19. The computing device of claim 15, wherein the flow control pipeline is further configured to send a copy of timestamp information for the received packet to a monitoring application for network performance evaluation.

20. The computing device of claim 17, wherein the time manager is further configured to compute the timestamp value using a set of values received from a call to obtain values for the at least one real-time clock.

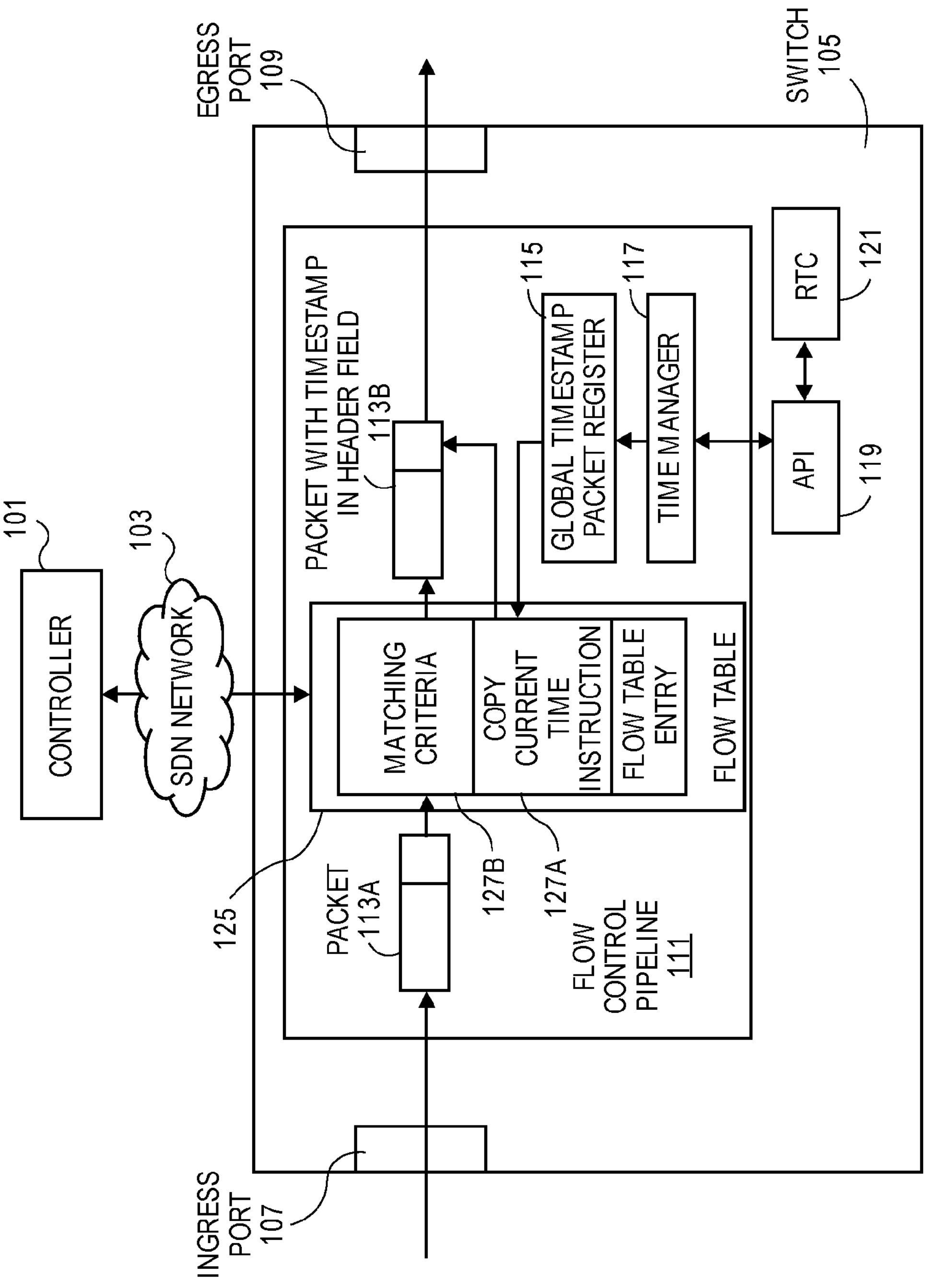

CONTROLLER
101
SDN NETWORK
103
EGRESS PORT 109
INGRESS PORT 107
SWITCH 105
PACKET WITH TIMESTAMP IN HEADER FIELD
113B
PACKET
113A
MATCHING CRITERIA
COPY CURRENT TIME INSTRUCTION
FLOW TABLE ENTRY
FLOW TABLE
125
127B
127A
FLOW CONTROL PIPELINE 111
GLOBAL TIMESTAMP PACKET REGISTER
115
TIME MANAGER
117
API
119
RTC
121

FIG. 1

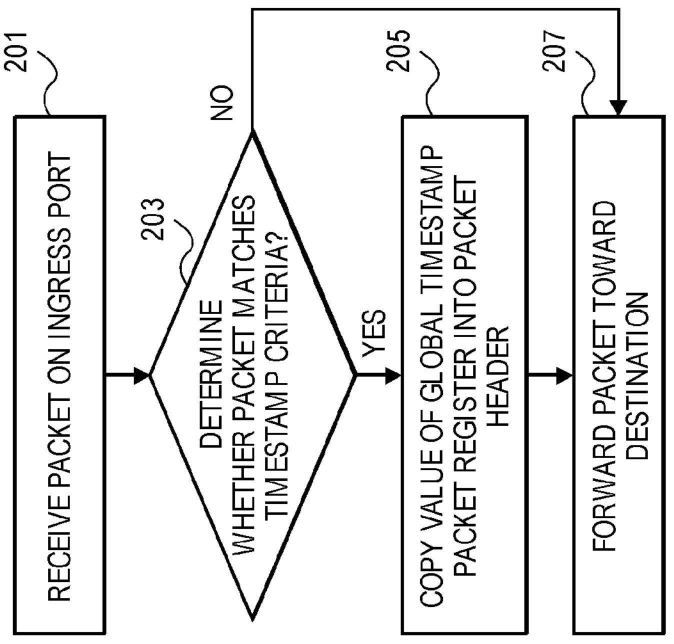

201
RECEIVE PACKET ON INGRESS PORT
203
DETERMINE WHETHER PACKET MATCHES TIMESTAMP CRITERIA?
NO
YES
205
COPY VALUE OF GLOBAL TIMESTAMP PACKET REGISTER INTO PACKET HEADER
207
FORWARD PACKET TOWARD DESTINATION

FIG. 2A

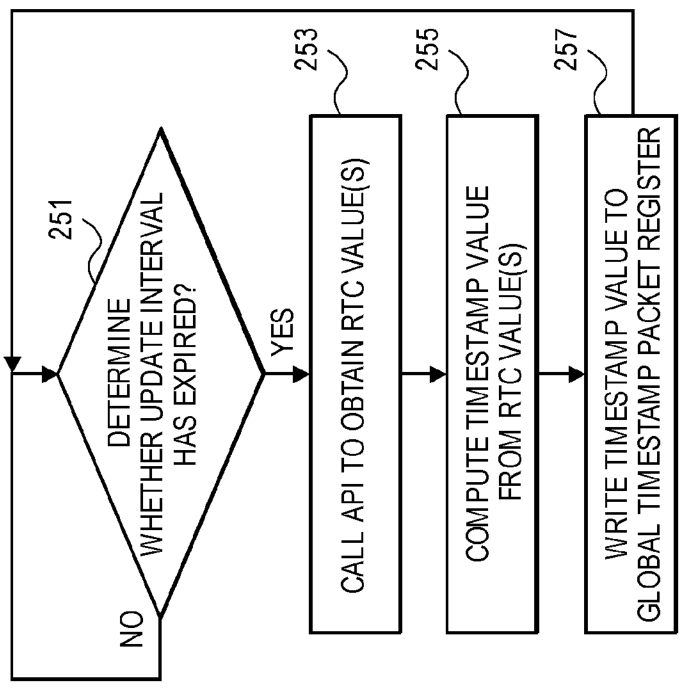

251
DETERMINE WHETHER UPDATE INTERVAL HAS EXPIRED?
NO
YES
253
CALL API TO OBTAIN RTC VALUE(S)
255
COMPUTE TIMESTAMP VALUE FROM RTC VALUE(S)
257
WRITE TIMESTAMP VALUE TO GLOBAL TIMESTAMP PACKET REGISTER

FIG. 2B

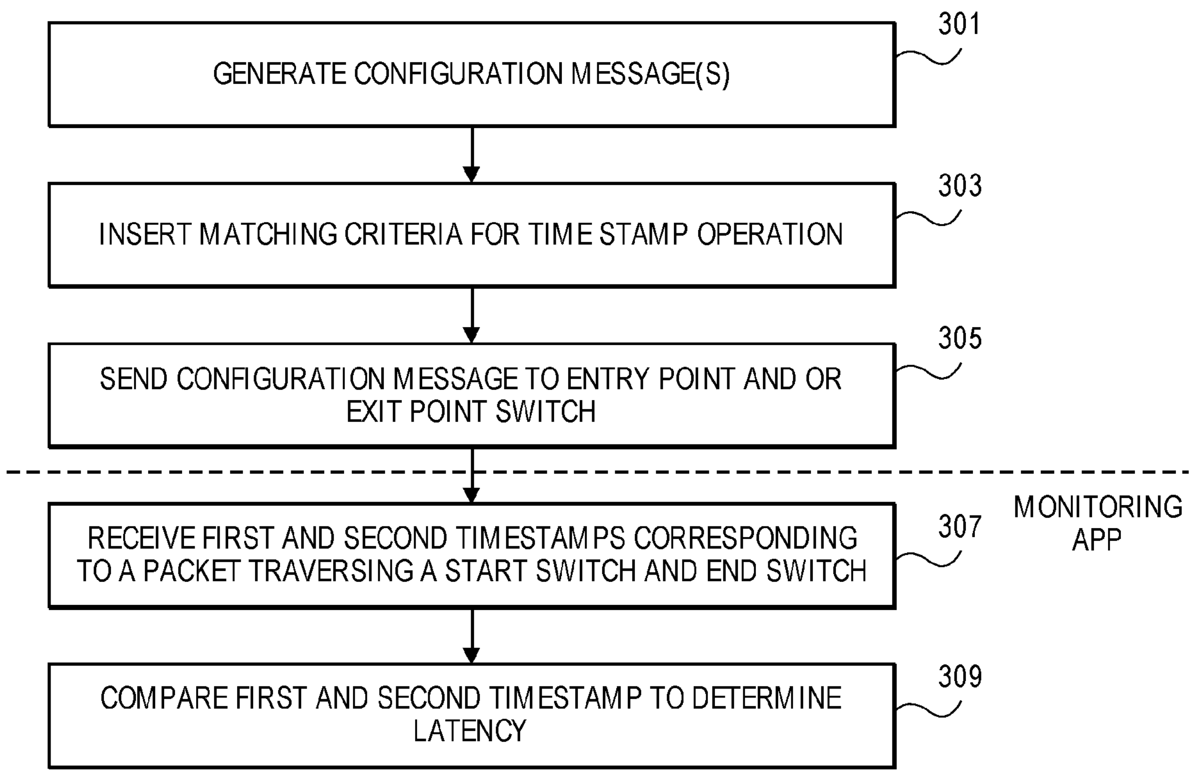

GENERATE CONFIGURATION MESSAGE(S)
301
INSERT MATCHING CRITERIA FOR TIME STAMP OPERATION
303
SEND CONFIGURATION MESSAGE TO ENTRY POINT AND OR EXIT POINT SWITCH
305
MONITORING APP
RECEIVE FIRST AND SECOND TIMESTAMPS CORRESPONDING TO A PACKET TRAVERSING A START SWITCH AND END SWITCH
307
COMPARE FIRST AND SECOND TIMESTAMP TO DETERMINE LATENCY
309

FIG. 3

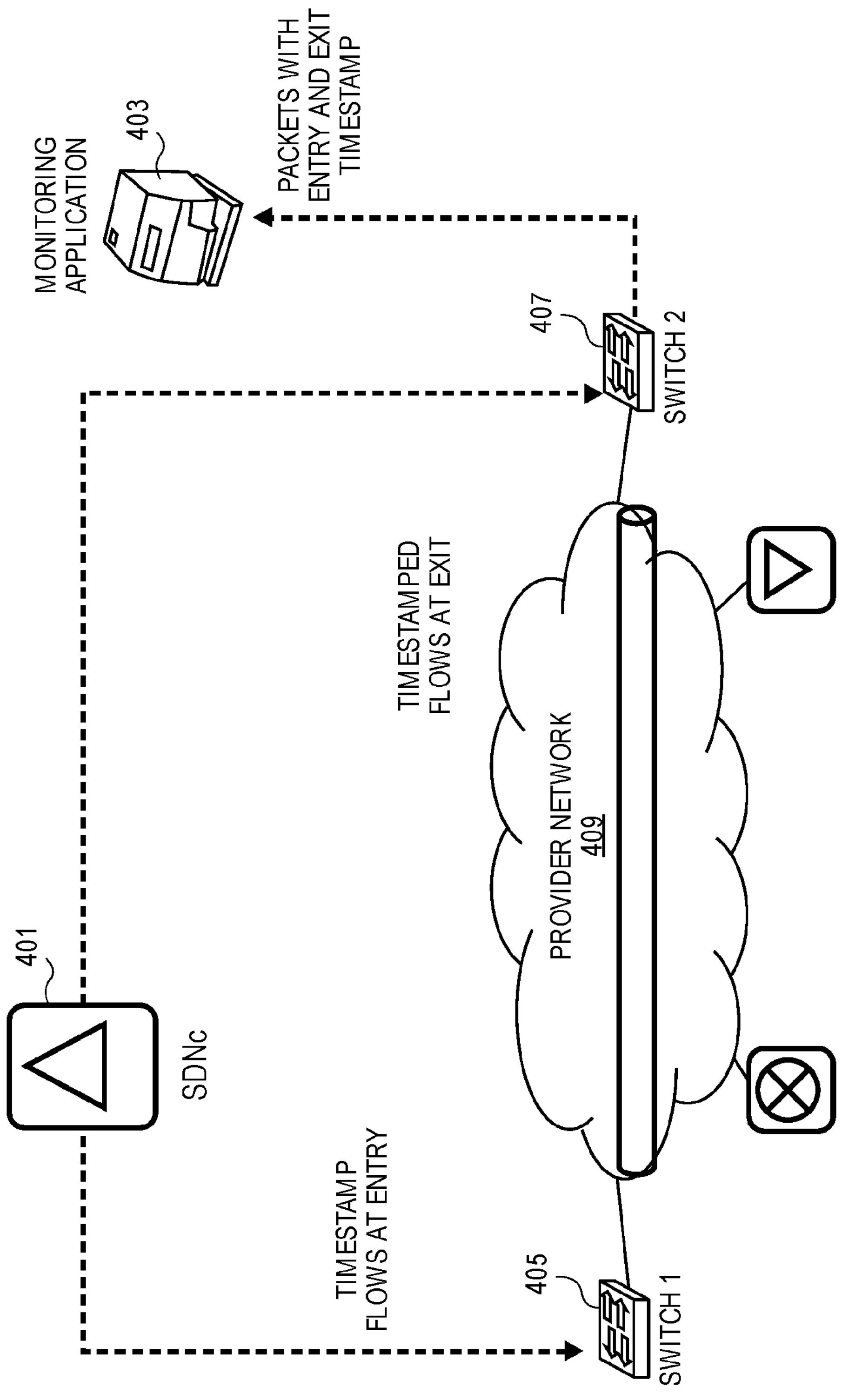
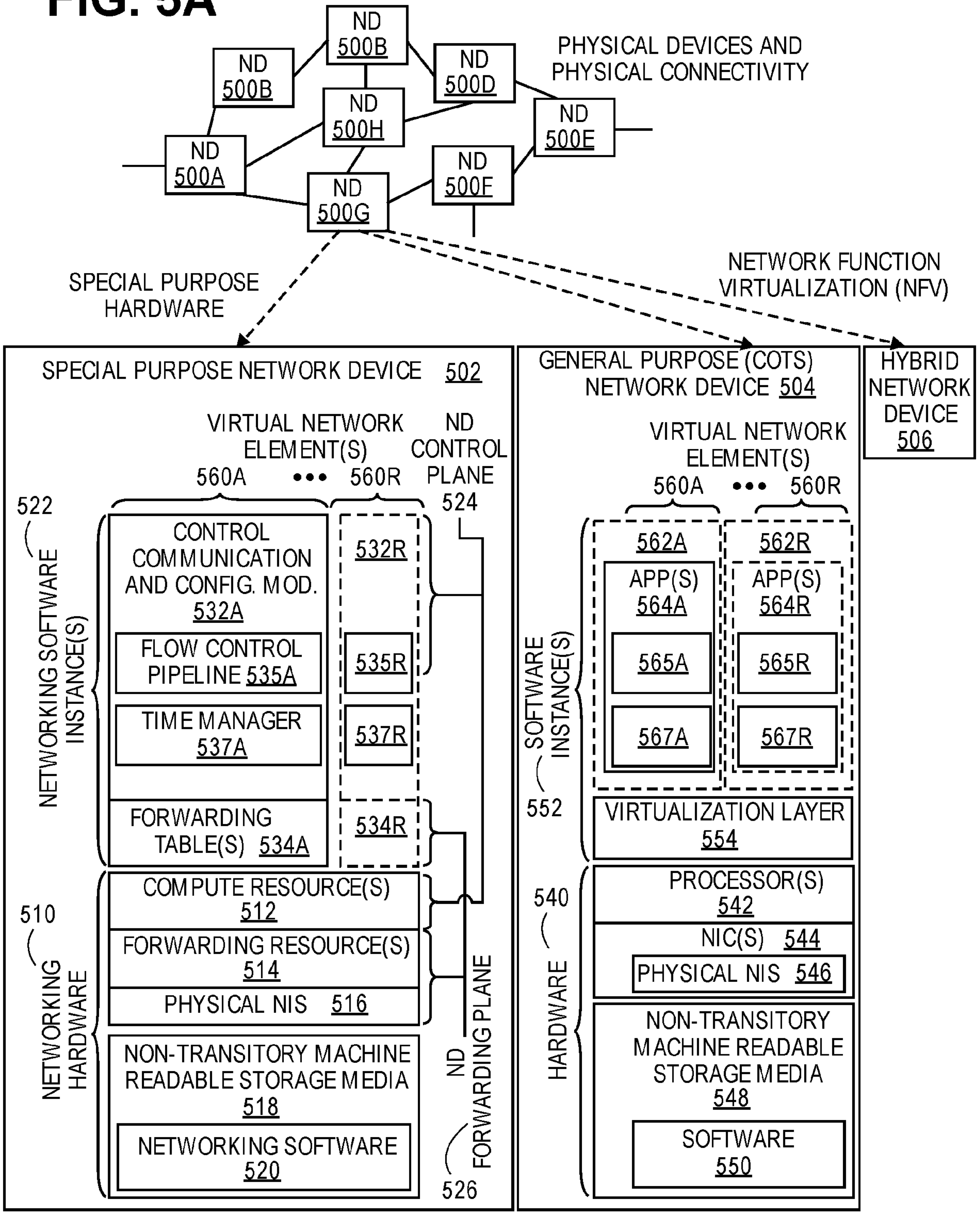

MONITORING APPLICATION
403
PACKETS WITH ENTRY AND EXIT TIMESTAMP
401
SDNc
407
SWITCH 2
TIMESTAMPED FLOWS AT EXIT
PROVIDER NETWORK
409
TIMESTAMP FLOWS AT ENTRY
405
SWITCH 1

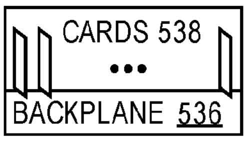
FIG. 4

FIG. 5A



ND 500B
ND 500B
ND 500D
PHYSICAL DEVICES AND PHYSICAL CONNECTIVITY
ND 500H
ND 500E
ND 500A
ND 500F
ND 500G
SPECIAL PURPOSE HARDWARE
NETWORK FUNCTION VIRTUALIZATION (NFV)
SPECIAL PURPOSE NETWORK DEVICE 502
VIRTUAL NETWORK ELEMENT(S)
560A ••• 560R
ND CONTROL PLANE 524
522
NETWORKING SOFTWARE INSTANCE(S)
CONTROL COMMUNICATION AND CONFIG. MOD. 532A
532R
FLOW CONTROL PIPELINE 535A
535R
TIME MANAGER 537A
537R
FORWARDING TABLE(S) 534A
534R
510
NETWORKING HARDWARE
COMPUTE RESOURCE(S) 512
FORWARDING RESOURCE(S) 514
PHYSICAL NIS 516
NON-TRANSITORY MACHINE READABLE STORAGE MEDIA 518
NETWORKING SOFTWARE 520
ND FORWARDING PLANE
526
GENERAL PURPOSE (COTS) NETWORK DEVICE 504
VIRTUAL NETWORK ELEMENT(S)
560A ••• 560R
562A
562R
APP(S) 564A
APP(S) 564R
565A
565R
567A
567R
SOFTWARE INSTANCE(S)
552
VIRTUALIZATION LAYER 554
540
HARDWARE
PROCESSOR(S) 542
NIC(S) 544
PHYSICAL NIS 546
NON-TRANSITORY MACHINE READABLE STORAGE MEDIA 548
SOFTWARE 550
HYBRID NETWORK DEVICE 506

FIG. 5B



CARDS 538
•••
BACKPLANE 536

FIG. 5C

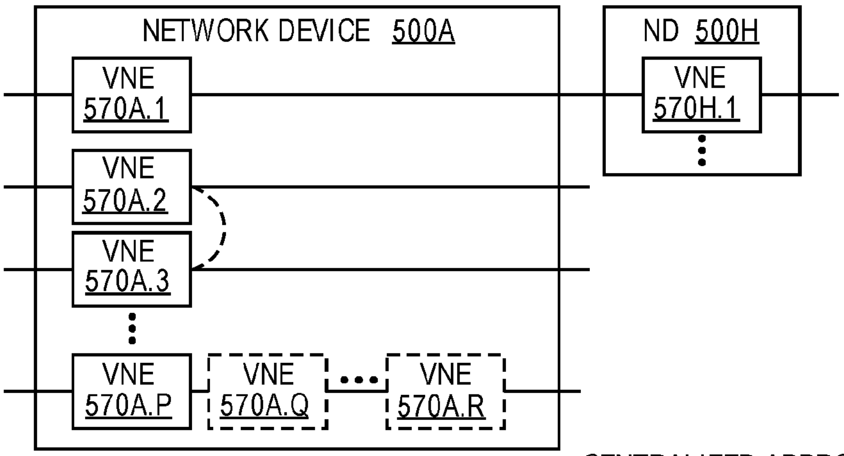


FIG. 5D

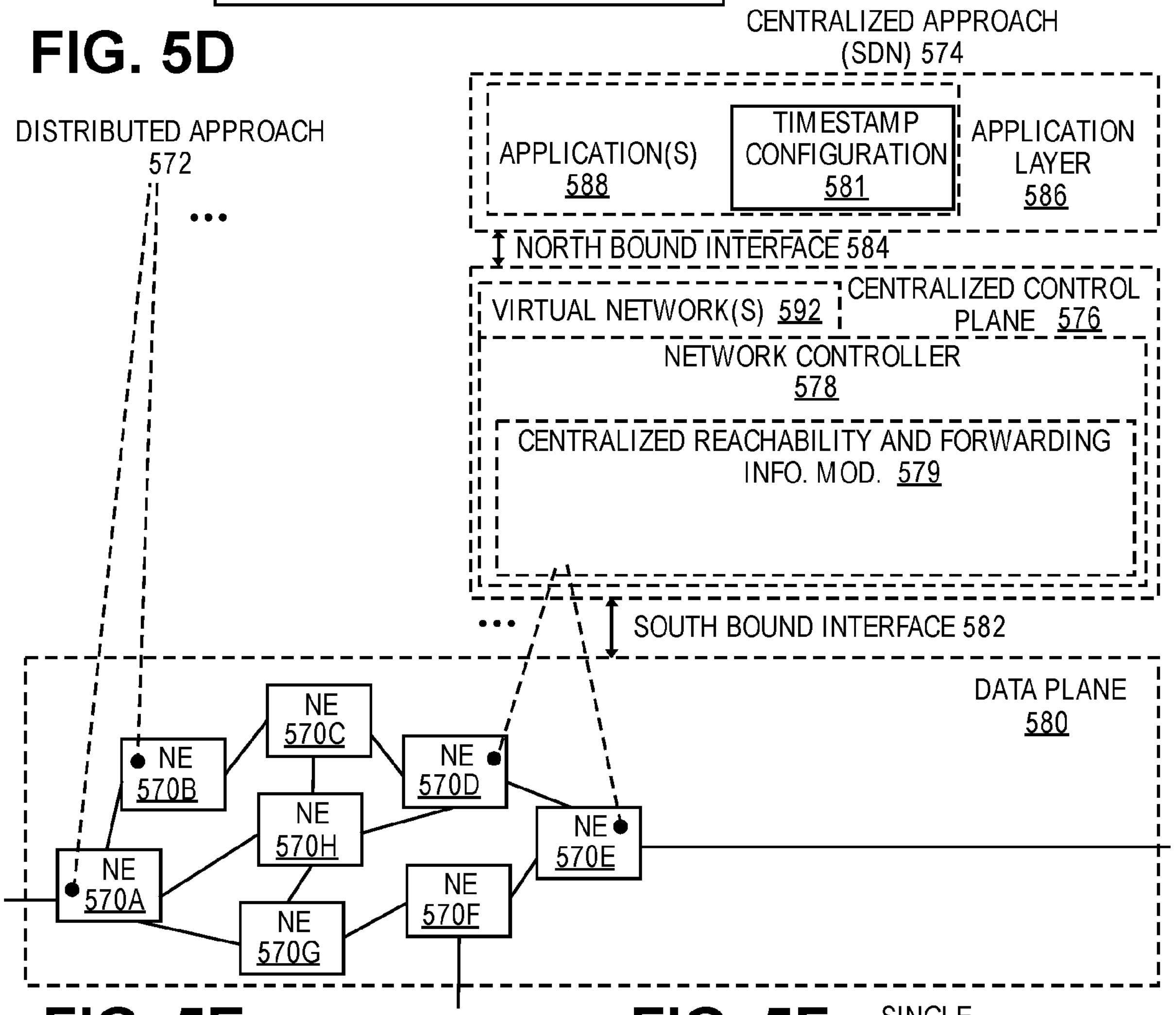

CENTRALIZED APPROACH (SDN) 574
DISTRIBUTED APPROACH 572
APPLICATION(S) 588
TIMESTAMP CONFIGURATION 581
APPLICATION LAYER 586
NORTH BOUND INTERFACE 584
VIRTUAL NETWORK(S) 592
CENTRALIZED CONTROL PLANE 576
NETWORK CONTROLLER 578
CENTRALIZED REACHABILITY AND FORWARDING INFO. MOD. 579
SOUTH BOUND INTERFACE 582
DATA PLANE 580
NE 570A
NE 570B
NE 570C
NE 570D
NE 570E
NE 570F
NE 570G
NE 570H

FIG. 5E

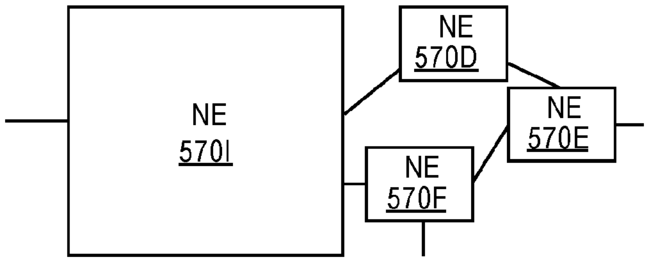


FIG. 5F

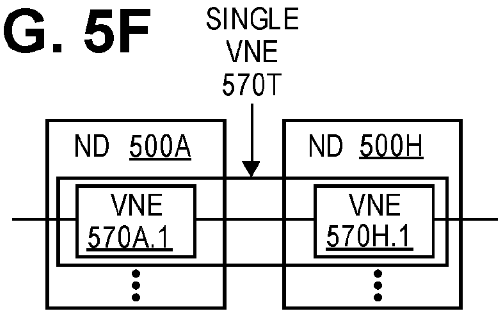
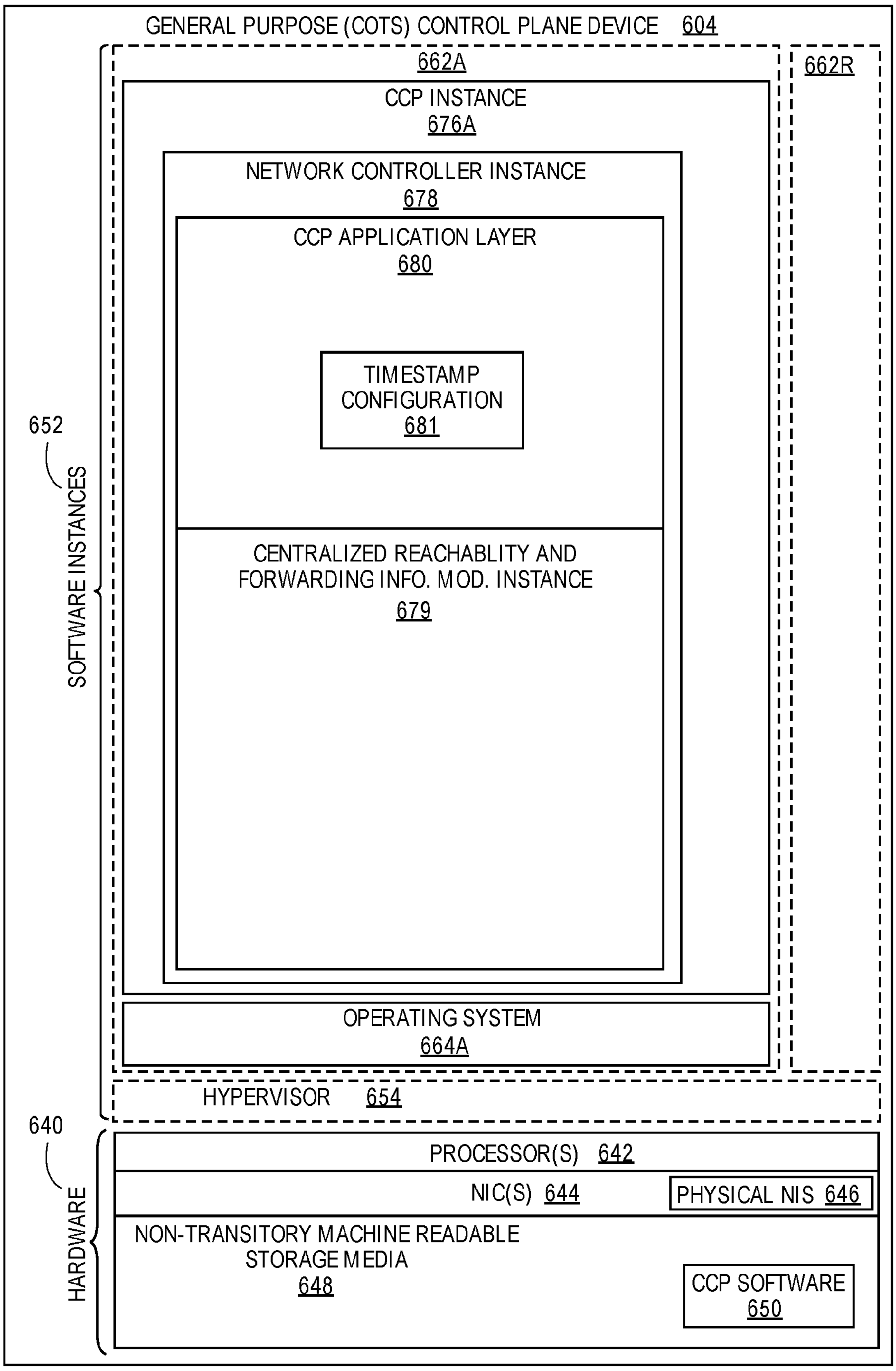


FIG. 6


GENERAL PURPOSE (COTS) CONTROL PLANE DEVICE 604
662A
662R
CCP INSTANCE 676A
NETWORK CONTROLLER INSTANCE 678
CCP APPLICATION LAYER 680
TIMESTAMP CONFIGURATION 681
CENTRALIZED REACHABLITY AND FORWARDING INFO. MOD. INSTANCE 679
652
SOFTWARE INSTANCES
OPERATING SYSTEM 664A
HYPERVISOR 654
640
HARDWARE
PROCESSOR(S) 642
NIC(S) 644
PHYSICAL NIS 646
NON-TRANSITORY MACHINE READABLE STORAGE MEDIA 648
CCP SOFTWARE 650

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2016/057904

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L12/26 H04L29/06
ADD.

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)
H04L

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

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

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2016/178134 A1 (TELEFONAKTIEBOLAGET LM ERICSSON [SE]) 10 November 2016 (2016-11-10) abstract page 8, line 19 - page 10, line 32 page 15, lines 1-32 page 17, line 5 - page 18, line 8 page 20, line 21 - page 21, line 11; figures 1, 2, 3, 8B -----	1-20
A	WO 2015/040624 A1 (HEWLETT PACKARD DEVELOPMENT CO [US]; APATHOTHARANAN RAMASAMY [IN]; DEV) 26 March 2015 (2015-03-26) abstract; figures 1, 2, 3 paragraphs [0026] - [0036] -----	1-20

☐ Further documents are listed in the continuation of Box C.

[✓ X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"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)

"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

"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

"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

"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

"&" document member of the same patent family

Date of the actual completion of the international search 3 July 2017	Date of mailing of the international search report 11/07/2017
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Authorized officer Fantacone, Vincenzo

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/IB2016/057904

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2016178134 A1	10-11-2016	US 2016330111 A1	10-11-2016
		WO 2016178134 A1	10-11-2016
WO 2015040624 A1	26-03-2015	US 2016226742 A1	04-08-2016
		WO 2015040624 A1	26-03-2015