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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 January 2016 (07.01.2016)

(10) International Publication Number
WO 2016/003398 A1

(51) International Patent Classification:
H04L 12/801 (2013.01)

(21) International Application Number:
PCT/US2014/044811

(22) International Filing Date:
30 June 2014 (30.06.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445 Compaq Center Drive West, Houston, Texas 77070 (US).

(72) Inventor: **MOZOLEWSKI, Mark B.**; Hewlett-Packard Company, 8000 Foothills Blvd., Roseville, California 95747 (US).

(74) Agents: **GARDINER, Austin W.** et al.; Hewlett-Packard Company, Intellectual Property Administration, 3404 East Harmony Road, Mail stop 35, Fort Collins, Colorado 80528 (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, 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, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:
— *as to the identity of the inventor (Rule 4.17(i))*
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:
— *with international search report (Art. 21(3))*

(54) Title: FLOW TABLE COMPARISON

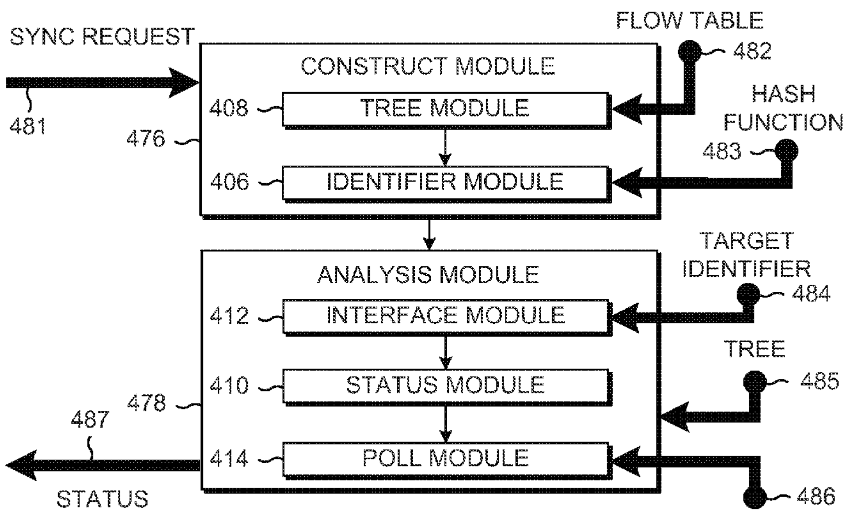

FIG. 4B

(57) Abstract: In one implementation, a tree is maintained based on a flow table and maintains a plurality of identifiers of the flow table. By comparing an identifier of the tree to a targer identifier, a status of the flow table is identified. And a system polls a mis-matched area of the flow table based on the status of the flow table, as it searches the identifier of the tree that is different from a target identifier.

WO 2016/003398 A1

Flow Table Comparison

BACKGROUND

[0001] Computers commonly communicate via a network of devices. For example, a communication from a source computer to a destination computer can hop through multiple intermediary devices, including routers and gateways, before reaching the destination. Software-defined networking ("SDN") generally separates the physical plane of a network and the control plane of the network. For example, an SDN-enabled controller can identify a communication path and provide a rule to each device of the path. An SDN-enabled device maintains a flow table of forwarding rules.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Figures 1 and 2 are block diagrams depicting example network management systems.

[0003] Figure 3 depicts an example environment in which various network management systems can be implemented.

[0004] Figure 4A depicts an example network management system according to one implementation.

[0005] Figures 4B-4D depict example modules of an example network management system according to one implementation.

[0006] Figures 4E and 4F depict example binary tree representations maintained by an example network management system according to one implementation.

[0007] Figures 5 and 6 are flow diagrams depicting example methods for tracking a flow table.

DETAILED DESCRIPTION

[0008] In the following description and figures, some example implementations of network management systems, apparatus, and/or methods of tracking a flow table are described. A flow table is a table of entries to forward traffic, such as forward a network packet from one device of the network to another. The entries create “flows” of information through the network based on the table entries. A flow represents the communication of information via a path of the network. Communications, such as transmissions related to a service offered by a device of the network, can follow a flow provided by the network based on a flow table. SDN-compatible networks may provide a service or multiple services to devices or other networks. As used herein, a service is any appropriate supplying of communication, transmissions, software, or any other product, resource, or activity that may be capable of executing on a network of electronic devices.

[0009] SDN-compatible networks can abstract the hardware of the system from the services being provided. For example, an SDN network can decouple the traffic control decisions from the physical systems that forward network traffic. An SDN network may allow the service to be provided without regard to the underlying physical hardware. For example, a first network device may become latent from becoming overprovisioned and the SDN network may initiate the service to follow a different traffic flow through a second network device. As another example, the network device or a port of the network device may be malfunctioning and traffic may be rerouted. In both examples, the customer may not notice a change in service because the SDN controller may make the network routing decisions autonomously.

[0010] An entry is a representation of the instructions in memory, such as a ternary content-addressable memory (“TCAM”), to manage a communication of the network. Each entry can include a match field to match against network packets, a counter to update for matching packets, and an instruction, such as an instruction to modify the action set or pipeline processing. An entry can automatically expire and a flow removal instruction is an optional feature of some SDN frameworks, such as described by the OPENFLOW networking standard. Thus, in some SDN networking environments, a state of the flow table of a device managed by an SDN controller may not be synchronized with the SDN controller’s state of the flow table. The difference in state between the SDN controller and the managed device is commonly resolved by polling each of the entries in the flow table to ensure the tables are coordinating.

[0011] Various examples described below relate to tracking a flow table based on a tree representation of identifiers. By maintaining a tree of identifiers representing the state of a flow table, flow tables can be compared for synchronization using a single node comparison and tracking a flow table's state can be maintained without polling the entire flow table.

[0012] The terms "include," "have," and variations thereof, as used herein, mean the same as the term "comprise" or appropriate variation thereof. Furthermore, the term "based on," as used herein, means "based at least in part on." Thus, a feature that is described as based on some stimulus can be based only on the stimulus or a combination of stimuli including the stimulus. Furthermore, the term "maintain" (and variations thereof) as used herein means "to create, delete, add, remove, access, update, and/or modify."

[0013] Figures 1 and 2 are block diagrams depicting example network management systems. Referring to figure 1, the example network management system 100 of figure 1 generally includes a data store 102, a table engine 104, an identifier engine 106, a tree engine 108, and a status engine 110. In general, the status engine 110 can identify the status of a flow table based on a tree of identifiers provided by the tree engine 108 and the identifier engine 106 as derived from the flow table maintained by the table engine 104. The example network management system 100 can include an interface engine 112 to receive a reference to identify and/or compare to an identifier of the tree and a poll engine 114 to retrieve information associated with a flow table entry.

[0014] The table engine 104 represents any appropriate circuitry or combination of circuitry and executable instructions to maintain a flow table. For example, the table engine 104 can comprise controller circuitry to maintain a TCAM. The flow table can be managed based on flow modification instructions to maintain an entry of the flow table. A flow table commonly includes a plurality of flow table entries. In an SDN network, a single table may include hundreds or thousands of flow table entries.

[0015] The identifier engine 106 represents any appropriate circuitry or combination of circuitry and executable instructions to maintain a plurality of identifiers.

The plurality of identifiers is associated with the plurality of flow table entries. For example, one of the plurality of identifiers can map to one of the plurality of flow table entries.

[0016] An identifier represents any character, number, value, state, label, category, or other representation capable of indicating an association with an entry. For example, the identifier for a flow table entry can be a unique hexadecimal value based on the flow modification instruction and match field. The identifier can be the result of a function. An example function is a hash function, such as a cryptographic hash function. Example cryptographic hash functions include secure hash algorithms (e.g. SHA-3) and message-digest algorithms (e.g. MD5). A "hash function" as used herein refers to any appropriate function that can map data of an arbitrary size to data of a fixed size. For example, an identifier of a flow table entry can be a hash value resulting from inputting the flow table entry (or a representation thereof) through a hash function to represent the flow table entry as a 2048 character string. The identifier can represent multiple flow table entries and/or identifiers. In other words, the function to produce the identifier can receive multiple inputs for a single output. For example, the identifier can result from a computation based on hashes of multiple flow table entries and/or identifiers.

[0017] The tree engine 108 represents any appropriate circuitry or combination of circuitry and executable instructions to maintain a tree of nodes based on the plurality of identifiers. The tree can be constructed with leaf nodes (i.e. nodes with no children) consisting of the flow table entries and non-leaf nodes (i.e. nodes with a child) to contain an identifier associated with the children of the non-leaf node. The node identifiers (i.e. identifiers associated with nodes of the tree) can represent the identifiers (or entries) of the nodes below them (i.e. the children of the nodes). For example, each parent node has a parent identifier that is associated with child identifiers of the child nodes of the parent node. In this manner, a parent can represent a combination of the child nodes and, thereby, represent the state of the descendent flow table entries associated with the child nodes. More details regarding construction and maintenance of the tree are provided in the description associated with figures 4E and 4F.

[0018] The tree engine 108 can organize the nodes of the tree based on an application associated with the flow table entry. For example, a flow table may include entries applicable to specific applications based on IP address, and the flow table can arrange the entries associated with a first application together and the entries associated with a second application together. Thus, a tree that represents the entire flow table can include sub-trees that represent the flow table entries associated with each application associated with the flow table, where each sub-tree represents the flow table entries of one of the applications. As used herein, the term “application” refers to a set of executable instructions that provide a service when executed by a processor resource.

[0019] The status engine 110 represents any appropriate circuitry or combination of circuitry and executable instructions to identify a status of a flow table based on a comparison of an identifier associated with a node of the tree to a target identifier. A status can be any character, number, value, label, category, or other representation capable of describing the state of the flow table based on a comparison of identifiers. A target identifier is an identifier from a reference source. Identifiers can be compared for strict equality to produce the status. For example, the identifiers can be made from a cryptographic hash function that produces a large difference in output based on a small variation on input, and identifiers that are not exact may not represent the same input (e.g. identifiers resulting from a cryptographic hash function that have a relatively small number of characters that are different could potentially represent very different entries in the flow table).

[0020] The target identifier represents an identifier to reference to achieve coordination. For example, the target identifier can be an identifier associated with a root node of the SDN controller and the network device can compare the target identifier to the root node of the tree built by the tree engine 106. When the comparison between the root node of the source (e.g. network device) and the root node of the target (e.g. SDN controller) are the same, the status of the flow table can represent a match, up-to-date status, or the like. In response to a match, the status engine 110 can initiate relay of a synchronization confirmation message to the device or user requesting the comparison. The synchronization confirmation message can confirm the flow table state of the source device is the same as the flow table state of the target device.

[0021] A difference between the source node and the target node can indicate that an entry of the flow table in the source (e.g. the network device when synchronizing with the SDN controller) is mismatched because the identifier of the source node represents the flow table entries of the mismatched area. The target node is a node of a target tree associated with a target identifier. An appropriate message can be relayed based on the status derived from the mismatch.

[0022] The mismatched flow table entry can be discovered (and then updated) using the tree maintained by the tree engine 108. The status engine 110 can search the tree for a node having a node identifier different from a target identifier. For example, in a binary tree of identifiers with a node identified as mismatched has two children, the two children of the target identifier can be retrieved via the interface engine 112, discussed herein, and the child nodes can be compared for equality. In that example, a first source child may represent the first half of the table and the second source child may represent the second half of the flow table, and when a first source child matches the first target child and the second source child mismatches the second target child, then the mismatch is located in the second half of the flow table. The pattern may be repeated until the identifier in direct relation to the mismatched entry is identified. The identified entry can then be updated or otherwise synchronized to match the reference flow table.

[0023] The interface engine 112 represents any appropriate circuitry or combination of circuitry and executable instructions to receive the target identifier from a node of a second tree. For example, the second tree can be the flow table of the SDN controller to which a network device should synchronize. The second tree can be a reference for synchronization and may be maintained by a master flow table, such as a flow table maintained by the SDN controller managing a set of devices that may each have their own flow table. The identifiers being compared should be from the equivalent level and/or position in the tree. For example, the interface engine 112 should retrieve an identifier of a non-leaf node from a target tree that is at the expected level of the non-leaf node of a source tree for the comparison to appropriately represent the status of the non-leaf node.

[0024] The poll engine 114 represents any appropriate circuitry or combination of circuitry and executable instructions to poll an area of the flow table. For example, the poll engine 112 can poll an area of the flow table determined as mismatched based on the status identified by the status engine 110. The poll engine 114 can poll an entry associated with the section of the reference flow table when the status of the flow table is identified as mismatched. Thus the replacement flow table entry (e.g. replacement flow modification instruction and match fields) can be placed in the mismatched flow table to bring the state of the flow table from a mismatched status to a matched status.

[0025] The data store 102 can contain information utilized by the engines 104, 106, 108, 110, 112, and 114. For example, the data store 102 can store a flow table entry, a hash function, an identifier, and a root node.

[0026] Figure 2 depicts the example network management system 200 can be implemented on a memory resource 220 operatively coupled to a processor resource 222. The processor resource 222 can be operatively coupled to a data store 202. The data store 202 can be the same as the data store 102 of figure 1.

[0027] Referring to figure 2, the memory resource 220 can contain a set of instructions that are executable by the processor resource 222. The set of instructions can implement the system 200 when executed by the processor resource 222. The set of instructions stored on the memory resource 220 can be represented as a table module 204, an identifier module 206, a tree module 208, a status module 210, interface module 212, and a poll engine 214. The processor resource 222 can carry out a set of instructions to execute the modules 204, 206, 208, 210, 212, 214, and/or any other appropriate operations among and/or associated with the modules of the system 200. For example, the processor resource 222 can carry out a set of instructions to maintain a flow table, maintain a first tree based on a plurality of entries of the flow table, compare an identifier of the first tree to an identifier of a second tree, and identify a status of a section of the flow table as compared to the second tree. The table module 204, the identifier module 206, the tree module 208, the status module 210, the interface module 212, and the poll module 214 represent program instructions that when executed function as the table engine 104, the identifier engine 106, the tree engine 108, the status engine 110, the interface engine 112, and the poll engine 114 of figure 1, respectively.

[0028] The processor resource 222 can be any appropriate circuitry capable of processing (e.g. compute) instructions. For example, the processor resource 222 can be a central processing unit ("CPU") that enables management of a network by fetching, decoding, and executing modules 204, 206, 208, 210, 212, and 214. Example processor resources 222 include CPUs, semiconductor-based microprocessors, application specific integrated circuits ("ASIC"), a field-programmable gate array ("FPGA"). The processor resource 222 can be one or multiple processing elements capable of retrieving instructions from the memory resource 220 and executing those instructions. Such multiple processing elements can be integrated in a single device or distributed across devices. The processor resource 222 can process the instructions serially, concurrently, or in partial concurrence.

[0029] The memory resource 220 and the data store 202 represent a medium to store data utilized and/or produced by the system 200. The medium can be any non-transitory medium or combination of non-transitory mediums able to electronically store data, such as modules of the system 200 and/or data used by the system 200. For example, the medium can be a storage medium, which is distinct from a transitory transmission medium, such as a signal. The medium can be machine-readable, such as computer-readable. The memory resource 220 can be said to store program instructions that when executed by the processor resource 222 implements the system 200 of figure 2. The memory resource 220 can be integrated in the same device as the processor resource 222 or it can be separate but accessible to that device and the processor resource 222. The memory resource 220 can be distributed across devices. The memory resource 220 and the data store 202 can represent the same physical medium or separate physical mediums. The data of the data store 202 can include representations of data and/or information mentioned herein.

[0030] In the discussion herein, the engines 104, 106, 108, 110, 112, and 114 of figure 1 and the modules 204, 206, 208, 210, 212, and 214 of figure 2 have been described as a combination of circuitry and executable instructions. Such components can be implemented in a number of fashions. Looking at figure 2, the executable instructions can be processor executable instructions, such as program instructions, stored on the memory resource 220, which is a tangible, non-transitory computer-readable storage medium, and the circuitry can be electronic circuitry, such as processor resource 222, for executing those instructions. The instructions residing on the memory resource 220 can comprise any set of instructions to be executed directly (such as machine code) or indirectly (such as a script) by the processor resource 222.

[0031] In one example, the executable instructions can be part of an installation package that when installed can be executed by the processor resource 222 to implement the system 200. In that example, the memory resource 220 can be a portable medium such as a compact disc, a digital video disc, a flash drive, or memory maintained by a computer device, such as a network device 318 of figure 3, from which the installation package can be downloaded and installed. In another example, the executable instructions can be part of an application or applications already installed. The memory resource 220 can be a non-volatile memory resource such as read only memory ("ROM"), a volatile memory resource such as random access memory ("RAM"), a storage device, or a combination thereof. Example forms of a memory resource 220 include static RAM ("SRAM"), dynamic RAM ("DRAM"), electrically erasable programmable ROM ("EEPROM"), content-addressable memory ("CAM"), flash memory, or the like. The memory resource 220 can include integrated memory such as a hard drive ("HD"), a solid state drive ("SSD"), or an optical drive.

[0032] Figure 3 depicts an example environment in which various example network management systems 300 and 301 can be implemented. The example environment 390 is shown to include example systems 300 and 301 for tracking a flow table. The systems 300 and 301 (described herein with respect to figures 1 and 2) can represent generally any combination of circuitry and executable instructions to track a flow table. The system 300 can include a table engine 304, an identifier engine 306, a tree engine 308, a status engine 310, an interface engine 312, and a poll engine 314 that are the same as the table engine 104, the identifier engine 106, the tree engine 108, the status engine 110, the interface engine 112, and the poll engine 114 of figure 1, respectively, and the associated descriptions are not repeated for brevity. The system 301 can include a table module 324, an identifier module 326, a tree module 328, a status module 330, an interface module 332, and a poll module 334 that are the same as the table module 204, the identifier module 206, the tree module 208, the status module 210, the interface module 212, and the poll module 214 of figure 2, respectively, and the associated descriptions are not repeated for brevity. As shown in figure 3, the engines 304, 306, 308, 310, 312, and 314 and/or modules 324, 326, 328, 330, 332 and 334 can be integrated into a compute device, such as an SDN-enabled router, or a management controller, such as SDN controller 316. The engines 304, 306, 308, 310, 312, and 314 and/or modules 324, 326, 328, 330, 332, and 334 can be integrated via circuitry or as installed instructions into a memory resource of a compute device, such as network device 318, or a management controller, such as SDN controller 316.

[0033] The example environment 390 can include network devices 318. The network devices 318 represent generally any compute devices, whether virtual or real, to transmit and respond to other network devices 318. For example, the network device 318 can operate a combination of circuitry and executable instructions to provide a network packet in response to a request for a page or functionality of an application. Example network devices 318 include switches, routers, gateways, web servers, application servers, data servers, and the like. The network devices 318 can represent any compute devices with a combination of circuitry and executable instructions to communicate a network request and receive and/or process the corresponding responses. For example, the network device 318 can represent a compute device having SDN-capability of forwarding traffic within a network, such network 336. Though sometimes not depicted in figures, devices described herein may include computing components, such as input/output ports, communications controllers, buses, peripherals, and management controllers, such as SDN controller 316.

[0034] The compute devices can be located on separate networks 336 or part of the same network 336. The example environment 390 can include any appropriate number of networks 336 and any number of the networks 336 can include a cloud compute environment. For example, networks 336 can be distributed networks comprising virtual computing resources or "clouds." Any appropriate combination of the systems 300 and 301 and network devices 318 can be a virtual instance of a resource of a virtual shared pool of resources. The engines and/or modules of the system 300 herein can reside and/or execute "on the cloud" (e.g. reside and/or execute on a virtual shared pool of resources).

[0035] A link 338 generally represents one or a combination of a cable, wireless connection, fiber optic connection, or remote connections via a telecommunications link, an infrared link, a radio frequency link, or any other connectors of systems that provide electronic communication. The link 338 can include, at least in part, intranet, the Internet, or a combination of both. The link 338 can also include intermediate proxies, routers, switches, load balancers, and the like.

[0036] Referring to figures 1-3, the engines 104, 106, 108, 110, 112, and 114 of figure 1 and/or the modules 204, 206, 208, 210, 212, and 214 of figure 2 can be distributed across devices 316 and 318, or a combination thereof. The engine and/or modules can complete or assist completion of operations performed in describing another engine and/or module. For example, the status engine 310 of figure 3 can request, complete, or perform the methods or operations described with the status engine 110 of figure 1 as well as the table engine 104, the identifier engine 106, the tree engine 108, the interface engine 112, and the poll engine 114 of figure 1. Thus, although the various engines and modules are shown as separate engines in figures 1 and 2, in other implementations, the functionality of multiple engines and/or modules may be implemented as a single engine and/or module or divided in a variety of engines and/or modules. The example engines of the system 300 and/or example modules of the system 301 can perform example methods described in connection with figures 4A-4F and 5-6.

[0037] Figures 4A-4F depict example implementations and example functionalities of a network management system according to one implementation. Referring to figure 4A, a SDN controller 416 includes a processor resource 422 and a memory resource 420 including a table module 404, a construct module 476, an analysis module 478, and flow table entries 441-448. The processor resource 422 and the memory resource 420 can be described the same as the processor resource 322 and memory resource 320 of figure 3, respectively. A SDN network device 418 includes flow table entries 441-448. The representation of the flow table can be arranged by application. For example, entries 441-444 can be associated with a first application and entries 445-448 can be associated with a second application. In this manner, the network management system can make flow compatibility checks and flow table synchronizations based on an application and/or a service.

[0038] Figure 4B depicts example modules used to implement example network management systems. The example modules of figure 4B generally include a construct module 476 and an analysis module 478. The example modules of figure 4B can be implemented on an SDN controller 416 (as shown in figure 4A) and/or a SDN network device 418.

[0039] As shown in figure 4B, a synchronization request 481 is made to the network management system. The construct module 476 can retrieve the flow table 482 from the table module 404 to build and maintain a tree data structure to represent the entries of the flow table 482. The construct module 476 can include a tree module 408 and an identifier module 406 that are the same as the tree module 208 and the identifier module 206 of figure 2, respectively. The synchronization request 481 can identify the source to use as a synchronization reference. For example, the SDN controller's flow table can be selected or otherwise identified as the flow table to match (i.e. the flow table to use as a reference for synchronization).

[0040] The identifier module 406 can receive a hash function 483 to create the identifiers of the nodes of the tree 485. The identifier module 406 can configure a hash module 480 with the hash function 483 to calculate hash values, as shown in figures 4C and 4D. The hash function can be polymorphic or otherwise capable of receiving various size and quantity of inputs. For example as shown in figure 4C, the hash module 480 can receive a flow table entry, such as flow table entry A 451 and compute a hash value, such as identifier A 491, that is associated with the input entry 451 based on the hash function 483. For another example as shown in figure 4D, the hash module 480 can receive multiple identifiers and compute hash value that associates with the inputs. For example, identifier A 491 and identifier B 492 can be combined via the hash module 480 to create identifier AB 493.

[0041] In this manner, a tree, such as the binary tree 485 depicted in figure 4E, can be created with nodes having identifiers based on hash values of the nodes' children. For example, entry A 451 can be hashed to compute a hash value of hash A to represent entry A and is placed in a first node 461, entry B 452 can be hashed to compute a hash value represented as hash B to associate with a second node 462, the hash values of the first node 461 and the second node 462 can be hashed together to compute a parent identifier associated with a third node 470 represented as hash AB in figure 4E. In that example, an identifier of a fourth node 472 (i.e. parent of third node 470) is computed and represented as hash ABC as well as the identifier of a fifth node which is the root node 474 represented as a combination of all the active flow table entries 451-453 and 455-457 and labeled hash ABCEFG in figure 4E. The nodes 461-467 are non-leaf parent nodes that have hash values based on the associated flow table entries and nodes 470-475 are parent nodes that have hash value identifiers based on the hash values of their children. The tree 485 can be built in this manner from the leaves up to a single root node 474 that represents the state of the flow table and the root node 474 can be used for comparison analysis between flow tables. Thus, the root identifier represents the descendant leaves of the root node and can encompass the state of the entire flow table in a single identifier. A descendant node is a node that is a direct or indirect child of a node. For example, the leaves of a tree are all descendants of the root of the tree. An ancestor node is a node that is a direct or indirect parent of a node. For example, the root of a tree is an ancestor to all leaves of the tree.

[0042] Referring to figure 4B, analysis can be performed on the flow table via the analysis module 478 once the tree 485 is constructed or otherwise available. The analysis module 478 can include program instructions represented as an interface module 414, a status module 410, and a poll module 414 that are the same as the interface module 212, the status module 210, and the poll module 214 of figure 2, respectively. The analysis module can receive the tree 485 created by the construct module 476 and the target identifier 484 received via the interface module 412. The target identifier 484, as shown in figure 4E, is the root node of the SDN controller's flow table and is represented as hash ABCDFG. The identifier of the root node 474 of the tree 485 (represented as the hash value of hash ABCEFG) is compared to the target identifier 484. The status module 410 can compare the values of hash ABCEFG and hash ABCDFG and identify that those identifiers are not equivalent. The status module

410 can notify the SDN controller 416 and/or the SDN network device 418 that the flow tables are mismatched.

[0043] The tree 485 can be organized based on application. For example in figure 4A, flow table entries A-D can be designated for a first application and the flow table entries E-H can be designated for a second application. In that example, node 472 of figure 4E can represent state of the flow table entries associated with the first application and the node 473 of figure 4E can represent the state of the flow table entries associated with the second application. Organizing the tree based on application and/or service allows for the flow table to be synchronized by request of the application to ensure the service can continue to operate as expected. The comparison, searching, and polling of the section of the flow table entries would be similar to that of comparing the entire flow table, but would operate with the sub-tree associated with the application rather than the entire binary tree 485 of figure 4E; in other words, the comparison of the first application in the previous example would start as if the root node was node 472.

[0044] Referring to figures 4A and 4E, the flow table of the SDN network device 418 can be synchronized with the flow table of the SDN controller 416 using the tree 485 and the target tree of the SDN controller (not shown, but comparable to the resulting tree 485 in figure 4F). The status module 484 can search the nodes of the target tree to identify the nodes that are mismatched based on the identifiers at the equivalent level of the tree 485. For example looking at figure 4E, the hash EFG would not be equivalent to a node of the target tree, however, the node 471 associated with hash FG would exist and a comparison of the identifiers of equivalent node in the target tree would produce a match, thus the mismatch would be in entry E 455 because hash FG exists in the target node. A further comparison between hash E and the target tree would confirm a mismatch. Similarly, the hash ABCD and hash CD produce a conflict between the tree of the SDN network device 418 and the SDN controller 416, but hash C would not produce a conflict and therefore, entry D 454 would be accorded with a mismatch. Once the entries and/or parent hashes of the entries are identified, the poll module 414 can poll the SDN controller's flow table to find an active entry at entry D 444 and that entry E 445 has been deactivated. The flow table of the SDN network device 418 can be updated with information polled by the poll module 414. The nodes of the tree can be recalculated for new identifiers based on the polled information and the tree can be maintained accordingly, resulting in tree 485 of figure 4F where entry D 454 has been added to the tree 485 and entry E 455 has been removed from the tree 485.

[0045] Figures 5 and 6 are flow diagrams depicting example methods for tracking a flow table. Referring to figure 5, example methods for tracking a flow table can generally comprise maintaining a binary tree based on a flow table, performing a comparison of an identifier, and identifying the status of the flow table based on the comparison.

[0046] At block 502, a binary tree is maintained based on a flow table. Each parent node of the binary tree is maintained with a parent identifier based on child identifiers of the child nodes of the parent node. The leaves of the binary tree are associated with the entries of the flow table. The tree can be built from the leaves up where each parent is based on the child below. For example, a flow table entry can be hashed to create an entry identifier, two adjacent entry identifiers can be hashed to one parent identifier to represent two adjacent flow table entries, two adjacent parent identifiers can be hashed to one grandparent identifier to represent four adjacent flow table entries, and continuing until a root identifier is reached that represents all of the flow table entries requested for comparison and/or synchronization. At block 504, a comparison of a parent identifier to a target identifier is performed. For example, a string compare function can be performed on the parent identifier and target identifier. At block 506, a status of the flow table is identified based on the comparison at block 504. For example, the result of the comparison (such as true or false) can determine whether the state of the flow table is matched with a reference table based on the target identifier.

[0047] Figure 6 includes blocks similar to blocks of figure 5 and provides additional blocks and details. In particular, figure 6 depicts additional blocks and details generally regarding searching the binary tree for a mismatched area, polling the flow table, maintaining a node based on the status, and recalculating a hash of a parent node. Blocks 602/604, 606, and 610 are similar to blocks 502, 504, and 506 of figure 5 and, for brevity, their respective descriptions are not been repeated.

[0048] At block 608, the binary tree is searched for a mismatched area based on the comparison at block 606. For example, the hash of a node of the binary tree is compared to the target hash of an equivalent target node from a target tree, and then the child nodes of the mismatched parent node are compared to the child nodes of the target node of the target tree. Those children are examined via similar comparisons with identifiers of expected nodes and so forth down the tree from the root until an entry is identified as mismatched. For example, a child identifier of a child node of the binary tree is compared with a sub-target identifier from a target tree (e.g. the sub-target identifier is associated with a child node of a node of the target tree associated with the target identifier). The target tree represents a reference of flow table entries to replicate in order to synchronize the flow tables between a controller and a device of the network.

[0049] The binary tree can be searched based on an application. For example, a branch of the binary tree can be selected to compare for synchronization based on a position of the entries of the flow table associated with the application.

[0050] At block 610, the entry of the flow table is polled. In particular, the entry of the flow table at an area identified as inconsistent (e.g. mismatched) is polled from a source flow table. The flow table is updated with the entry from the reference flow table and a node of the tree is maintained accordingly based on the status at block 614. For example, a leaf can be added to a binary tree associated with an active flow table entry that was not previously located in the flow table. In that example, a hash calculation is performed for an entry identifier of the parent node of the leaf and the hash for the parent identifier (and grandparents and so forth) are recalculated based on the hash calculation for the new entry identifier at block 616. For another example, a removal of a leaf of the binary tree is performed when the leaf is associated with an inactive flow table entry. In that example, the parent identifiers of parent nodes that have the removed leaf as a decedent are recalculated at block 616.

[0051] Although the flow diagrams of figures 4-6 illustrate specific orders of execution, the order of execution may differ from that which is illustrated. For example, the order of execution of the blocks may be scrambled relative to the order shown. Also, the blocks shown in succession may be executed concurrently or with partial concurrence. All such variations are within the scope of the present description.

[0052] The present description has been shown and described with reference to the foregoing examples. It is understood, however, that other forms, details, and examples may be made without departing from the spirit and scope of the following claims.

CLAIMS

What is claimed is:

1. A system comprising:

a table engine to maintain a flow table, a flow table having a plurality of flow table entries;

an identifier engine to maintain a plurality of identifiers, an identifier of a plurality of identifiers associated with an entry of the plurality of flow table entries;

a tree engine to maintain a tree of nodes based on the plurality of identifiers, a parent node having a parent identifier that is associated with child identifiers of child nodes of the parent node; and

a status engine to identify a status of the flow table based on a comparison of an identifier associated with a node of the tree to a target identifier.

2. The system of claim 1, comprising:

a poll engine to poll a mismatched area of the flow table based on the status of the flow table;

wherein the status engine is to search the tree for a node having an identifier different from a target identifier, the difference between the identifier and the target identifier to indicate the entry of flow table is located within the mismatched area.

3. The system of claim 1, wherein the tree engine is to:

organize nodes of the tree based on an application associated with an entry of the plurality of flow table entries.

4. The system of claim 1, comprising:

an interface engine to receive the target identifier from a node of a second tree, the second tree being identified as a synchronization reference.

5. The system of claim 1, wherein the comparison is between a root node of a first tree and a root node of a second tree and the status engine identifies the status as

updated when the identifier of the root node of the first tree matches the identifier of the root node of the second tree.

6. A computer readable storage medium comprising a set of instructions executable by a processor resource to:

maintain a flow table, the flow table to contain a plurality of entries;

maintain a first tree based on the plurality of entries, the first tree having a root and a non-leaf node of the first tree to contain an identifier associated with children of the non-leaf node of the first tree;

compare the identifier of the first tree to an identifier of a second tree, the identifier of a second tree associated with a non-leaf node of the second tree at an expected level of the non-leaf node of the first tree; and

identify a status of a section of the flow table associated with the non-leaf node.

7. The medium of claim 6, wherein the set of instructions is executable by the processor resource to:

poll an entry associated with the section of the flow table when the status of the flow table is identified as mismatched.

8. The medium of claim 6, wherein the set of instructions is executable by the processor resource to:

provide a synchronization confirmation message when the status indicates a match between the identifier of the root of the first tree and the identifier of a root of the second tree.

9. The medium of claim 6, wherein the set of instructions is executable by the processor resource to:

compute a hash for the non-leaf node of the first tree, wherein the hash is the identifier based on hashes of child nodes of the non-leaf node of the first tree.

10. The medium of claim 9, wherein an identifier of a parent node is a result of a cryptographic hash function with identifiers of the child nodes of the parent node as input to cryptographic hash function.

11. A method of tracking a flow table comprising:
maintaining each parent node of a binary tree with a parent identifier based on child identifiers of child nodes of the parent node, the binary tree having a first leaf associated with an entry of the flow table;
performing a comparison of the parent identifier to a target identifier; and
identifying a status of the flow table based on the comparison.

12. The method of claim 11, comprising:
searching the binary tree for a mismatched area based on the comparison;
polling the entry of the flow table at an area identified as inconsistent.

13. The method of claim 12, wherein:
searching the binary tree comprises:
selecting a branch of the binary tree to compare for synchronization based on an application associated with the entry of the flow table; and
performing the comparison of the parent identifier to the target identifier comprises:
comparing a child identifier of one of the child nodes of the binary tree with a sub-target identifier from a target tree, the sub-target identifier associated with a child node of a target node of the target tree, the target node associated with the target identifier and the target tree to represent a reference of flow table entries to replicate.

14. The method of claim 11, wherein maintaining the binary tree comprises:
adding a second leaf to the binary tree associated with an active flow table entry;

performing a hash calculation for an entry identifier of a parent node of the second leaf; and

recalculating the parent identifier based on the hash calculation, the second leaf is a descendant of the parent node associated with the parent identifier and the parent identifier to be a hash value.

15. The method of claim 11, wherein maintaining the binary tree comprises:

performing a removal of the first leaf of the binary tree when the first leaf is associated with an inactive flow table entry; and

recalculating the parent identifier when the first leaf was a child node of the parent node associated with the parent identifier, the parent identifier to be a hash value.

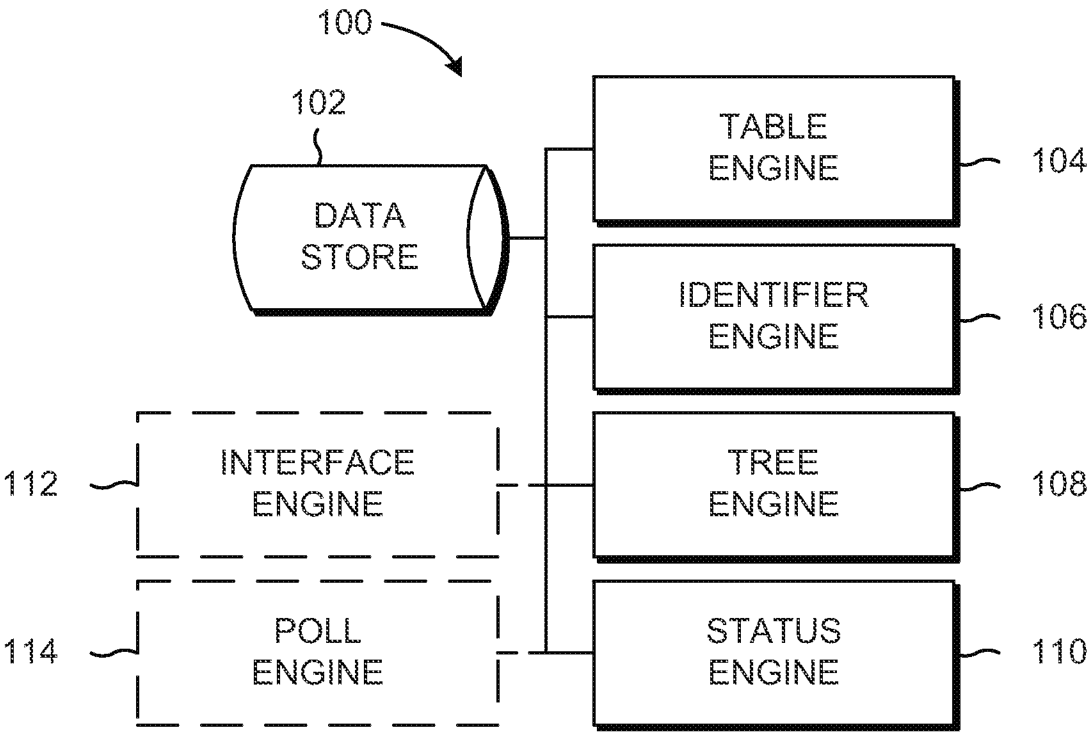
100
102
DATA STORE
TABLE ENGINE
104
IDENTIFIER ENGINE
106
112
INTERFACE ENGINE
TREE ENGINE
108
114
POLL ENGINE
STATUS ENGINE
110

FIG. 1

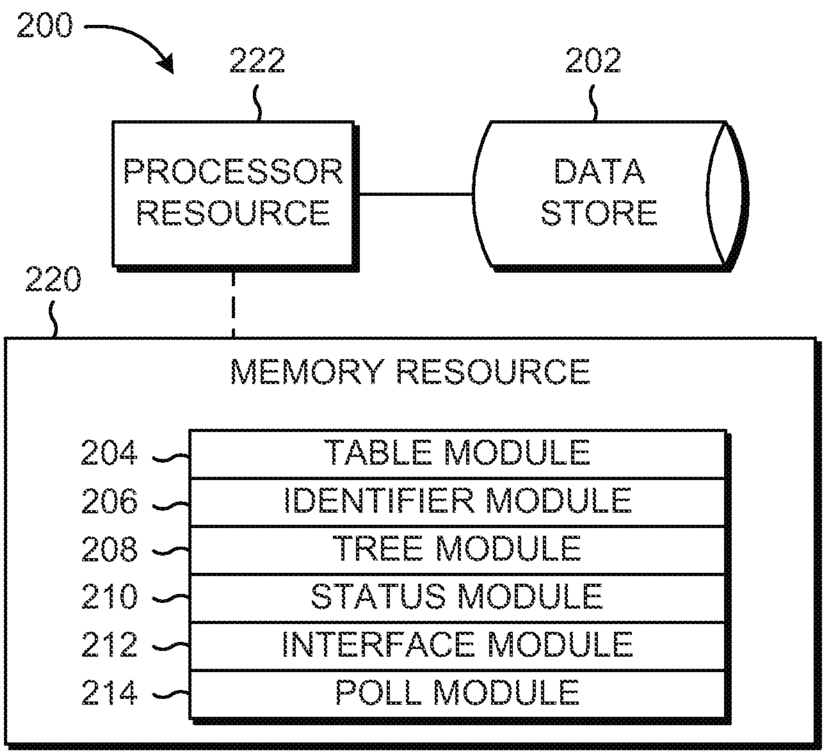
200
222
202
PROCESSOR RESOURCE
DATA STORE
220
MEMORY RESOURCE
204
TABLE MODULE
206
IDENTIFIER MODULE
208
TREE MODULE
210
STATUS MODULE
212
INTERFACE MODULE
214
POLL MODULE

FIG. 2

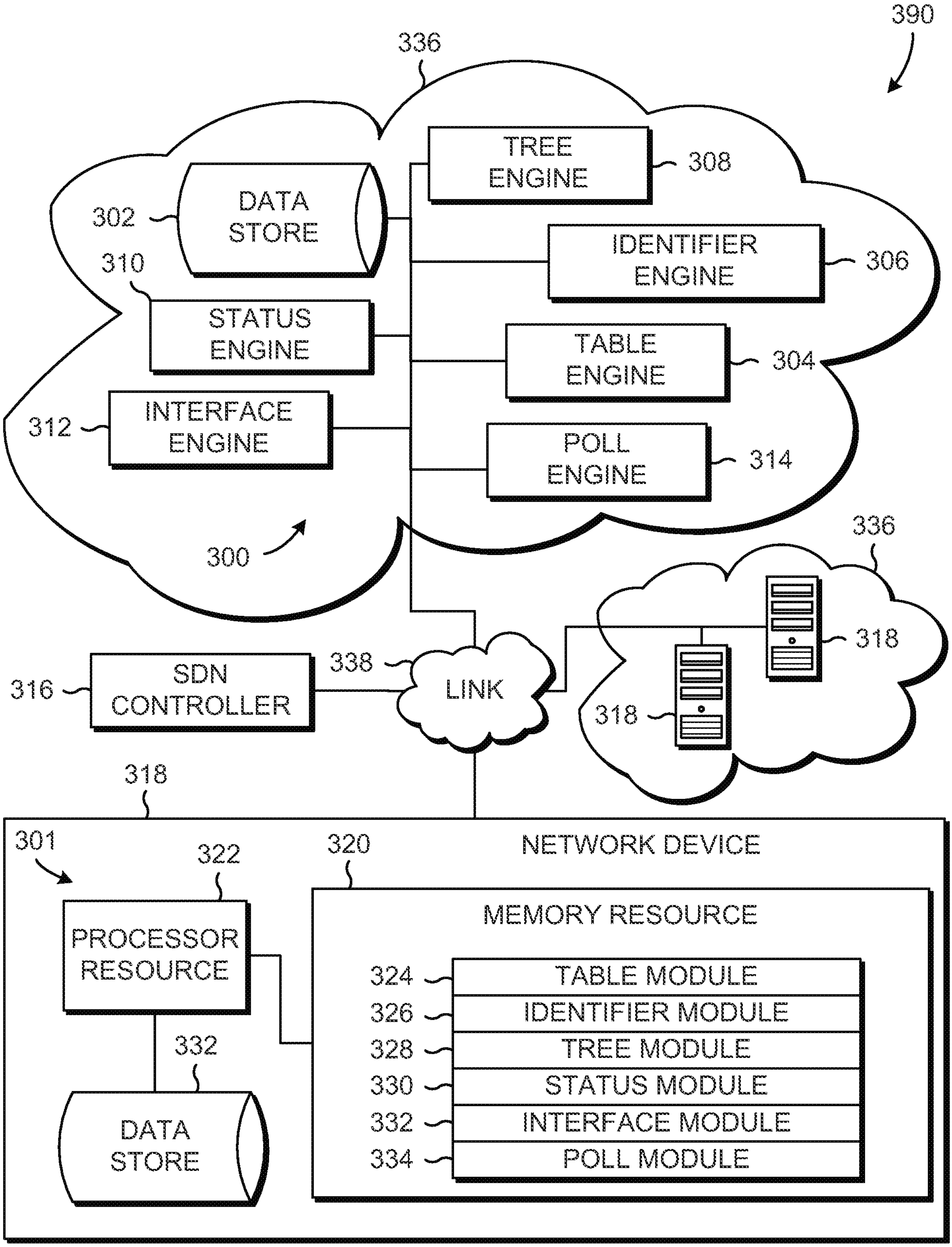
390
336
TREE ENGINE
308
DATA STORE
302
IDENTIFIER ENGINE
306
310
STATUS ENGINE
TABLE ENGINE
304
312
INTERFACE ENGINE
POLL ENGINE
314
300
336
318
316
SDN CONTROLLER
338
LINK
318
318
301
322
320
NETWORK DEVICE
PROCESSOR RESOURCE
MEMORY RESOURCE
324
TABLE MODULE
326
IDENTIFIER MODULE
328
TREE MODULE
330
STATUS MODULE
332
INTERFACE MODULE
334
POLL MODULE
332
DATA STORE

FIG. 3

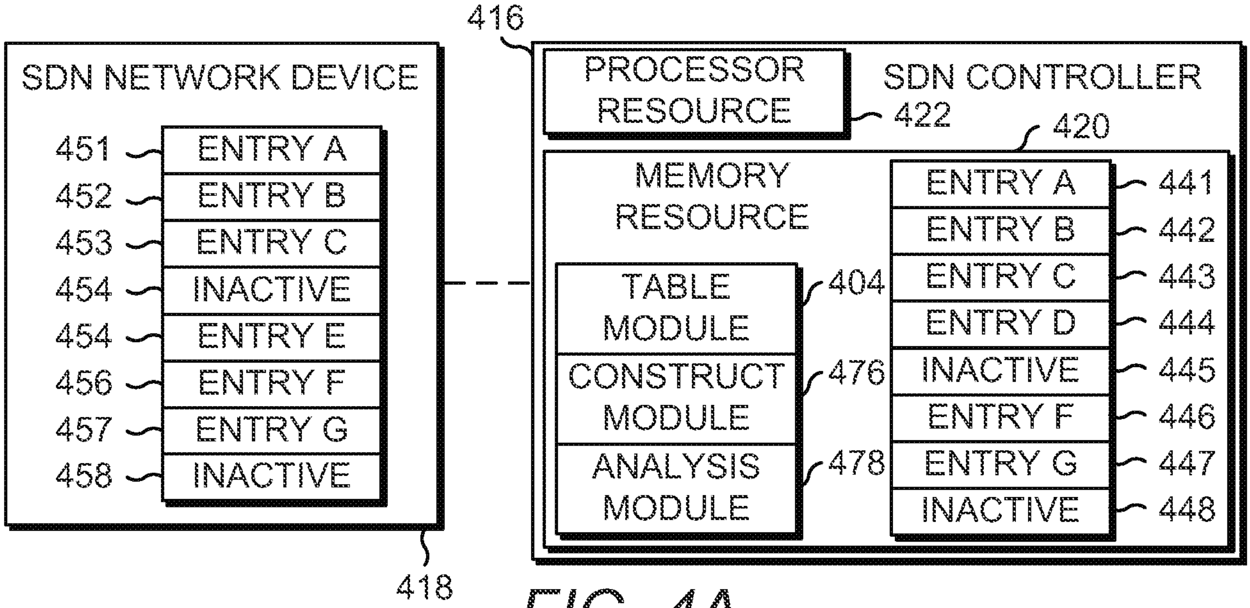

FIG. 4A

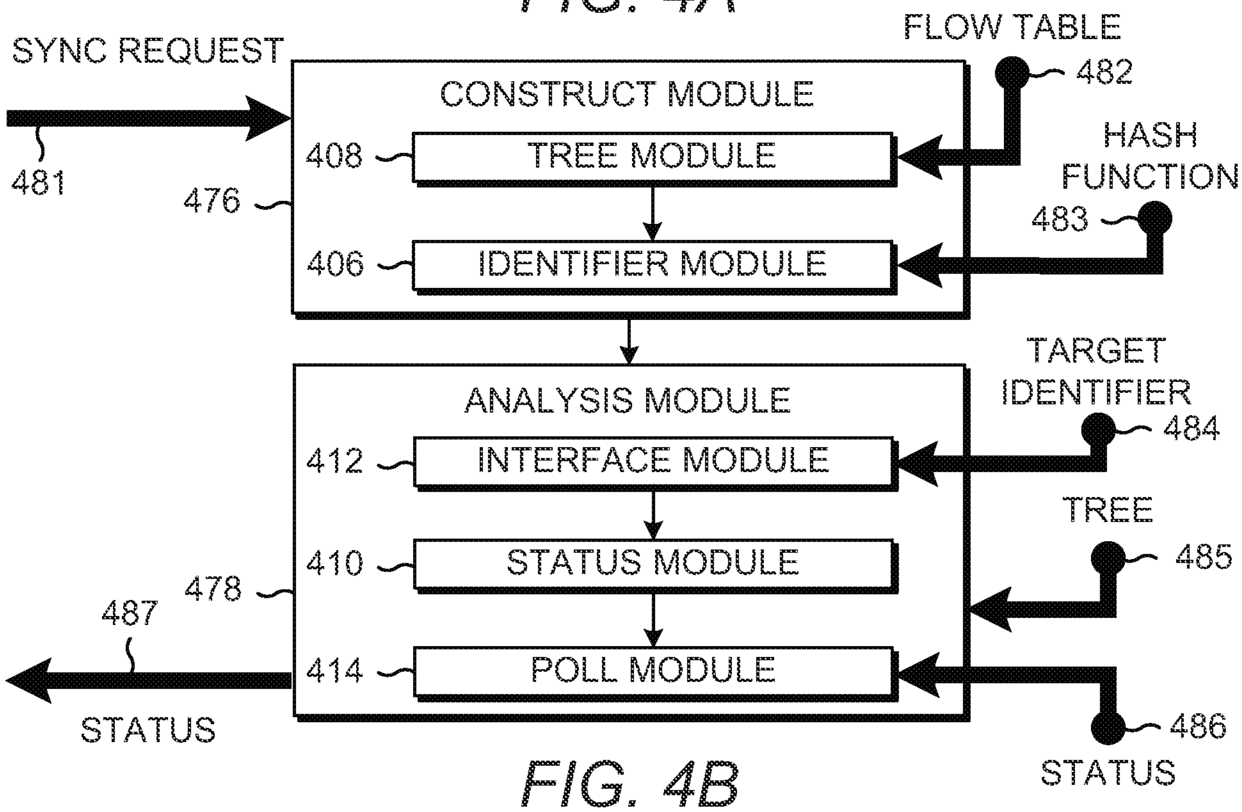

FIG. 4B

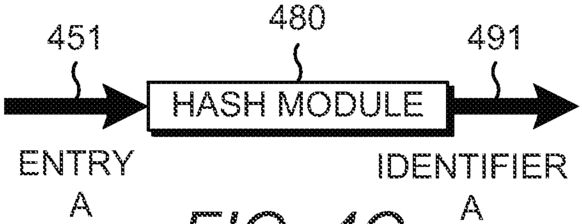

FIG. 4C

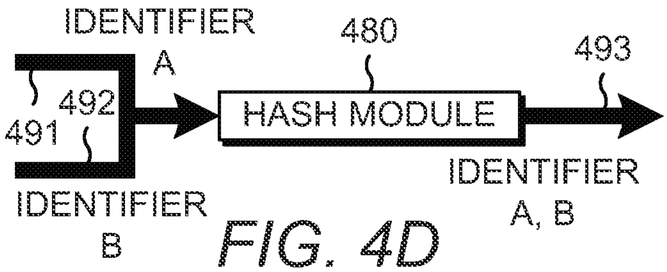

FIG. 4D

4/5

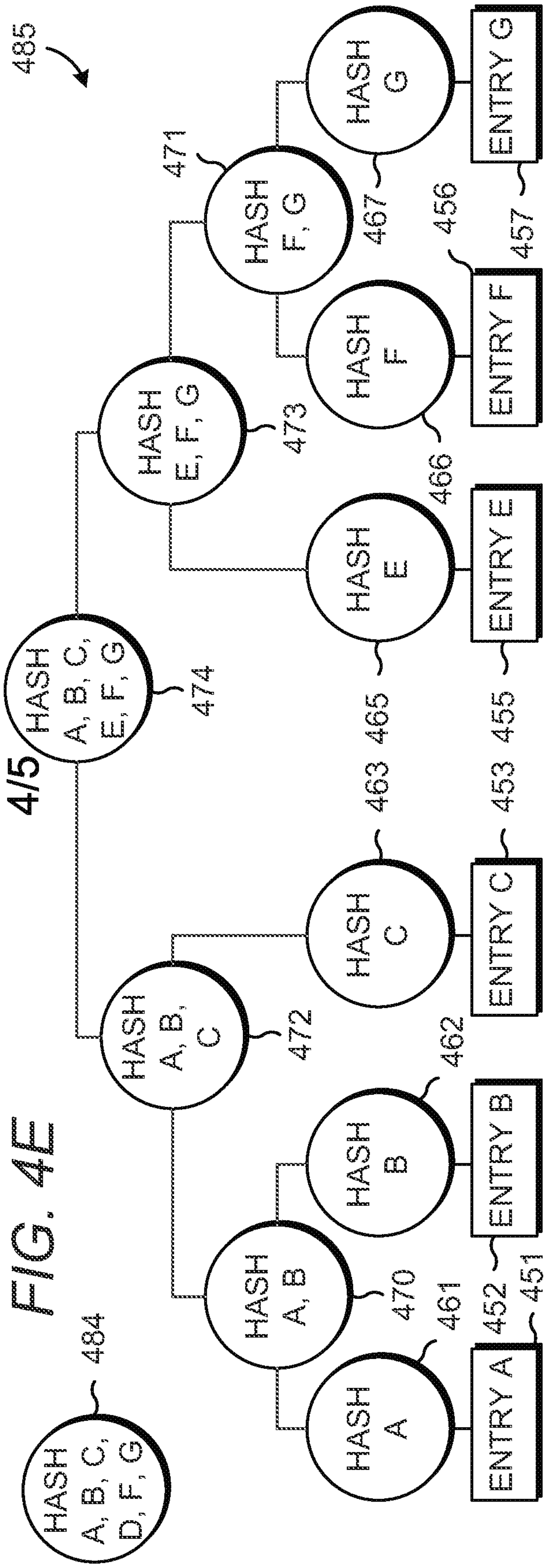

FIG. 4E

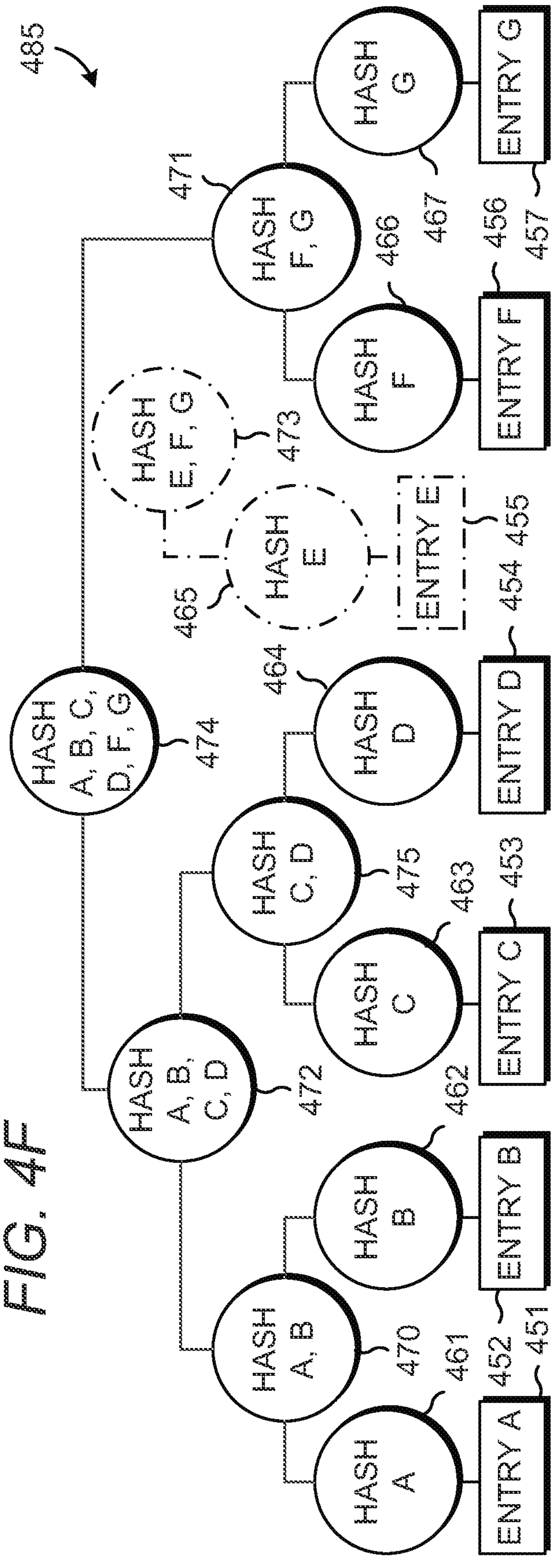

FIG. 4F

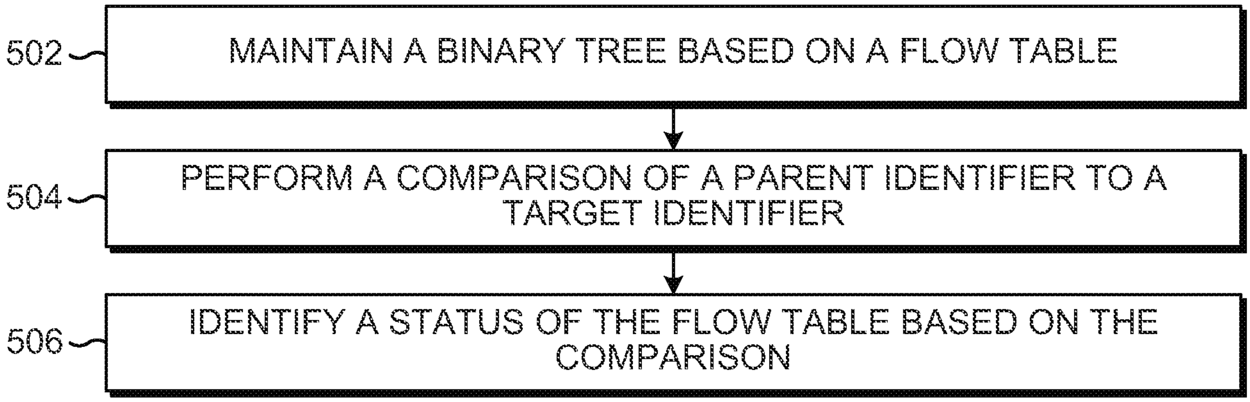
502
MAINTAIN A BINARY TREE BASED ON A FLOW TABLE
504
PERFORM A COMPARISON OF A PARENT IDENTIFIER TO A TARGET IDENTIFIER
506
IDENTIFY A STATUS OF THE FLOW TABLE BASED ON THE COMPARISON

FIG. 5

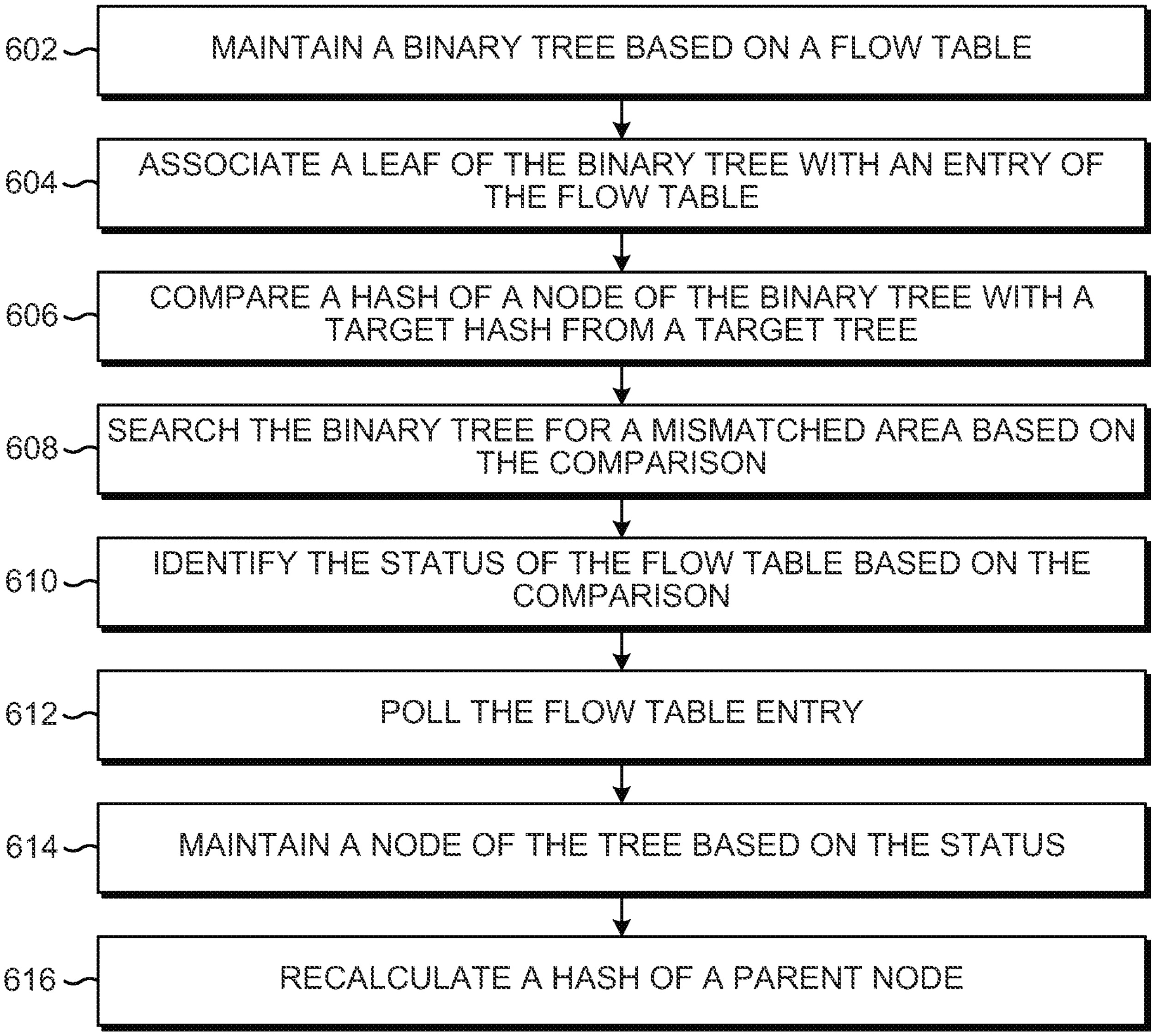
602
MAINTAIN A BINARY TREE BASED ON A FLOW TABLE
604
ASSOCIATE A LEAF OF THE BINARY TREE WITH AN ENTRY OF THE FLOW TABLE
606
COMPARE A HASH OF A NODE OF THE BINARY TREE WITH A TARGET HASH FROM A TARGET TREE
608
SEARCH THE BINARY TREE FOR A MISMATCHED AREA BASED ON THE COMPARISON
610
IDENTIFY THE STATUS OF THE FLOW TABLE BASED ON THE COMPARISON
612
POLL THE FLOW TABLE ENTRY
614
MAINTAIN A NODE OF THE TREE BASED ON THE STATUS
616
RECALCULATE A HASH OF A PARENT NODE

FIG. 6

INTERNATIONAL SEARCH REPORT	International application No. **PCT/US2014/044811**

A. CLASSIFICATION OF SUBJECT MATTER

H04L 12/801(2013.01)i

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 12/801; G06F 15/173; H04L 12/24; H04L 12/56

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: flow table, identifier, tree, comparison, entry

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 7478156 B1 (PRATAP PEREIRA) 13 January 2009 See column 5, lines 55-57; column 6, lines 10-14; column 8, lines 34-35; and figures 2 and 5A.	1-15
A	OPEN NETWORKING FOUNDATION, `OpenFlow Switch Specification version 1.3.0 (Wire protocol 0x04)`, 25 June 2012 See page 23, lines 1-2; page 24, lines 14-17; and figure 1. (http://web.archive.org/web/20130923062502/https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-scec-v1.3.0.pdf)	1-15
A	ADAM ZAREK, `OpenFlow Timeouts Demystified`, Univ. of Toronto, 18 October 2012 See page 9, lines 1-10 and figure 3. (http://web.archive.org/web/*/http://www.eecg.toronto.edu/~lie/papers/zarek_mscthesis.pdf)	1-15
A	US 2012-0213074 A1 (EITHAN GOLDFARB et al.) 23 August 2012 See paragraphs [0039]-[0043]; and figure 2.	1-15
A	US 2011-0310734 A1 (YASUHIRO MIZUKOSHI) 22 December 2011 See paragraphs [0023]-[0028]; and figure 1.	1-15

☒ Further documents are listed in the continuation of Box C. ☒ 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	Date of mailing of the international search report
25 February 2015 (25.02.2015)	**25 February 2015 (25.02.2015)**

Name and mailing address of the ISA/KR	Authorized officer
International Application Division Korean Intellectual Property Office 189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea	AHN, Jeong Hwan
Facsimile No. ++82 42 472 3473	Telephone No. +82-42-481-8440

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2014/044811

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 7478156 B1	13/01/2009	None	
US 2012-0213074 A1	23/08/2012	EP 2482495 A1	01/08/2012
		IL 210897 D0	28/04/2011
		US 8767551 B2	01/07/2014
US 2011-0310734 A1	22/12/2011	EP 2493128 A1	29/08/2012
		US 8837286 B2	16/09/2014
		WO 2011-049019 A1	28/04/2011