



- (51) International Patent Classification:
H04L 12/891 (2013.01) *H04L 12/937* (2013.01)
- (21) International Application Number:
PCT/US2015/026234
- (22) International Filing Date:
16 April 2015 (16.04.2015)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
996/CHE/2015 2 March 2015 (02.03.2015) IN
- (71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).
- (72) Inventors: **CHANDRAN, Sugesh**; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN). **MATHEW, Subin Cyriac**; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN). **KANIAMPADY SEBASTIAN, Celestian**; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN).
- (74) Agents: **FERGUSON, Christopher W.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 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, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.
- (84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

[Continued on next page]

- (54) Title: CONTROLLING AN UNKNOWN FLOW INFLOW TO AN SDN CONTROLLER IN A SOFTWARE DEFINED NETWORK (SDN)

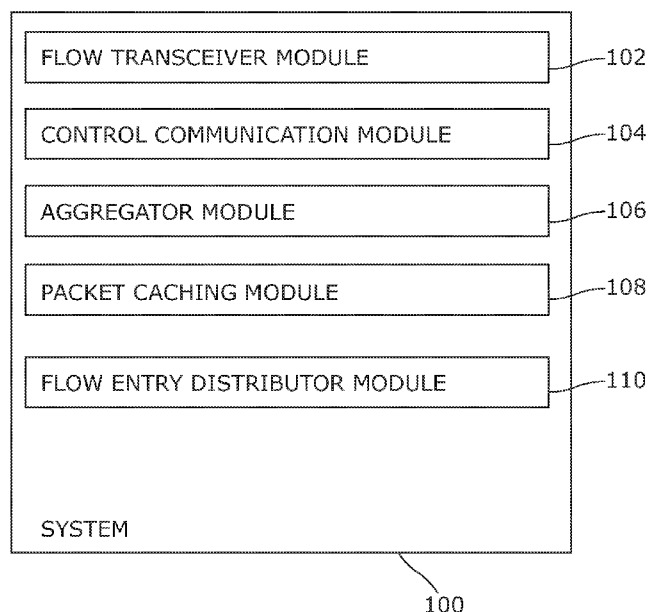


Fig. 1

(57) Abstract: Examples disclosed herein relate to controlling an unknown flow inflow to an SDN controller in a software defined network (SDN). In an example, an optimizer may be provided, between a switch and an SDN controller, to intercept an unknown flow from the switch to the SDN controller, in a software defined network. A portion of a data packet from each data packet in a plurality of data packets from the unknown flow may be aggregated at the optimizer. Only the aggregated portion of the data packet from each data packet may be sent, from the optimizer to the SDN controller, in a single package.

**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))*

CONTROLLING AN UNKNOWN FLOW INFLOW TO AN SDN CONTROLLER IN A SOFTWARE DEFINED NETWORK (SDN)

Background

[001] A new approach in networking includes a routing architecture in which data and control planes are decoupled. This new split-architecture framework that focuses on splitting of control plane from forwarding and data plane is the basis of software defined networking (SDN). In a software defined network (SDN), the control plane is implemented in an SDN controller and the data plane is implemented in the networking infrastructure (e.g., switches and routers). Data forwarding on a network device is controlled through flow table entries populated by the SDN controller that manages the control plane for that network. A network device that receives packets on its interfaces looks up its flow table to check the actions that need to be taken on a received packet.

Brief Description of the Drawings

[002] For a better understanding of the solution, embodiments will now be described, purely by way of example, with reference to the accompanying drawings, in which:

[003] FIG. 1 is a block diagram of an example system for controlling an unknown flow inflow to an SDN controller in a software defined network;

[004] FIG. 2 is a diagram of an example network system for controlling an unknown flow inflow to an SDN controller in a software defined network;

[005] FIG. 3 is a block diagram of an example method for controlling an unknown flow inflow to an SDN controller in a software defined network; and

[006] FIG. 4 is a block diagram of an example system for controlling an unknown flow inflow to an SDN controller in a software defined network.

Detailed Description

[007] Software defined networking (SDN) is an approach to networking in which control is decoupled from networking equipment and given to a device called a controller (or SDN controller). The controller is aware of all the devices and their points of interconnection in a SDN network and may perform various functions such as routing, policy implementation, receiving unknown flow packets, path resolution, flow programming, etc. Each new or missed flow through the network is routed via the controller that decides the network path for a flow and adds an entry for that flow in a flow table, in each of the network devices along the path. A SDN enabled device consults a flow table(s) for forwarding packets in the data plane. Each forwarding rule (flow entry) includes an action that dictates how traffic that matches the rule is to be handled. Thus, in a software defined network, a network administrator may shape traffic from a centralized control console (i.e. an SDN controller) instead of directly interacting with individual switches.

[008] There may be a scenario where the number of unknown flows in a software defined network may increase. For instance, in case there's a sudden spike of new flows in a part of the network, or if there's a DoS (Denial of Service) attack on the network. In such events, the load on a SDN controller may increase tremendously since every unknown packet received by a network device on the network may be forwarded to the controller. Thus, the controller may get flooded with unwanted packets. In fact, the load on a controller may become so heavy that a livelock situation may arise wherein the controller may end up servicing only packet interrupts and nothing else. Needless to say, this is not desirable from the perspective of a network user.

[009] To address such issues, the present disclosure describes various examples for controlling an unknown flow inflow to an SDN controller in a software defined network (SDN). In an example, an optimizer may be provided, between a switch and an SDN controller, to intercept an unknown flow from the switch to the SDN controller in a software defined network. The optimizer may aggregate a portion of a data packet from each data packet in a plurality of data packets from the unknown flow, wherein selection of the portion of the data packet is based upon a predefined criterion between the optimizer and the controller. Upon aggregation, the optimizer may send only the aggregated portion of the data packet from each data packet to the SDN controller, and retain the original data packets at the optimizer. The proposed solution proposes placing one or more intelligent controller channel optimizers in the paths between a central SDN controller and various network devices to efficiently regulate the unknown flow inflow to the controller.

[0010] FIG. 1 is a block diagram of an example system for controlling an unknown flow inflow to an SDN controller in a software defined network. In an example, system 100 may represent any type of computing system capable of reading machine-executable instructions. Examples of computing device 100 may include, without limitation, a server, a desktop computer, a notebook computer, a tablet computer, a thin client, a mobile device, a personal digital assistant (PDA), a phablet, and the like.

[0011] In another example, system 100 may be a network device such as, but not limited to, a network switch, a network router, a virtual switch, and a virtual router. In a further example, system 100 may be an SDN enabled device or an Open-Flow enabled device. In a yet another example, system 100 may be a computer application (machine-executable instructions).

[0012] In an example, system 100 may be called as an “optimizer”. In an example, an optimizer 100 may be placed between an SDN controller and a

network device (for example, a switch) in a software defined network. Among other tasks, optimizer 100 may intercept an unknown flow from the network device to the SDN controller. In another instance, network devices present in a software defined network may be divided into a number of plurality of groups, and each group may be called as a “cluster”. In such case, an exclusive optimizer may be assigned for each cluster of network devices. A software defined network, thus, may include a plurality of optimizers. For each cluster, an assigned optimizer may act as a mediator between a central SDN controller and a network device present in the cluster. In an instance, an assigned optimizer may intercept an unknown flow from a network device of the cluster to the SDN controller.

[0013] In the example of FIG. 1, system (or optimizer) 100 may include a flow transceiver module 102, a control communication module 104, an aggregator module 106, a packet caching module 108, and a flow entry distributor module 110. The term “module” may refer to a software component (machine readable instructions), a hardware component or a combination thereof. A module may include, by way of example, components, such as software components, processes, tasks, co-routines, functions, attributes, procedures, drivers, firmware, data, databases, data structures, Application Specific Integrated Circuits (ASIC) and other computing devices. A module may reside on a volatile or non-volatile storage medium and configured to interact with a processor of a system (e.g. 100).

[0014] Flow transceiver module 102 may receive an unknown flow from a network device of a software defined network. In an instance, the unknown flow is meant for an SDN controller in the software defined network. In an example, flow transceiver module 102 may intercept an unknown flow to an SDN controller from a network device of a “cluster” in a software defined network. In an instance, flow transceiver module 102 may intercept unknown

flows only from network devices of a particular cluster in a software defined network.

[0015] Controller communication module 104 may inform or advertise to an SDN controller a capability of the optimizer. In an example, the capability may include an aggregation capability of the optimizer (for instance, how many headers packets may be aggregated by an optimizer for sharing with an SDN controller, what size or portion of an unknown data packet may be used by an optimizer during an aggregation, etc.). In another example, the capability may include a buffering capability of the optimizer (for instance, how many original data packets from an unknown flow may be buffered by an optimizer). In another example, the capability may include metrics such as, but not limited to, the maximum load an optimizer may handle, current load handled by an optimizer. In a further example, controller communication module 104 may share details related to different profiles or modes in which an optimizer may operate, to an SDN controller. Controller communication module 104 may also provide packet priority information to an SDN controller in case a load threshold is exceeded. An SDN controller may generate and register a profile of an optimizer based on the information shared by controller communication module 104 of an optimizer. In an instance, depending on the information received from controller communication module 104 of an optimizer, an SDN controller may respond to the optimizer by providing the flow entry tuples that the SDN controller may be interested in receiving from the optimizer. Such information may help the optimizer to plan an aggregation or buffering criterion. For instance, if the controller is interested only in an IP subnet, the optimizer may send just the L3 header to the controller. In an instance, controller communication module 104 may receive various types of information from an SDN controller. Such information may relate to, by way of examples, the current load of an SDN controller and data packet prioritization information.

[0016] Aggregator module 106 may aggregate a portion of a data packet from each data packet in a plurality of data packets from an unknown flow, into a single package. In an instance, aggregate module 106 may select a “portion” of the data packet for aggregation based upon a predefined criterion between the optimizer and the controller. For example, if the agreement between the optimizer and the controller includes that the optimizer would share the first 40 bytes of a data packet from an unknown flow with the controller; aggregate module 106 may select such portion (i.e. the first 40 bytes from a data packet) accordingly. In an instance, the “portion” may include the header portion of a data packet from an unknown flow. In such case, aggregate module 106 may aggregate multiple packet headers into a single packet or request. Once a portion of a data packet from each data packet in a plurality of data packets from an unknown flow is aggregated into a single package, aggregate module 106 may send the package to an SDN controller. In an example, there may be a prioritization of the packets to be sent to the controller, depending on the load on the SDN controller. High latency traffic may be sent immediately, and low priority traffic may be deferred at the optimizer. In an example, in response to receiving the aggregated portion of the data packets from the unknown flow, the SDN controller may determine a flow path for the unknown flow in the software defined network.

[0017] Packet caching module 108 may cache or buffer packets from an unknown flow, which may be received by the optimizer 100 from a network device in a software defined network. In an instance, packet caching module 108 may buffer a plurality of data packets received from a network device that belongs to a cluster of network devices in a software defined network, based on an aggregation criteria. Packet caching module 108 may cache data packets from an unknown flow till the time an SDN controller informs the optimizer about the flow entries that are to be programmed on appropriate network devices for that flow. In other words, packet caching module 108 may not release buffered data packets into a software defined

network until it determines that the SDN controller has decided a flow path for the unknown flow.

[0018] In an example, once an SDN controller determines a flow path for an unknown flow in a software defined network, in response to receiving the aggregated portion of data packets from the unknown flow, flow entry distributor module 110 may configure the flow path in appropriate network devices of the software defined network. The flow entries for the unknown flow may be communicated to the optimizer by the SDN controller. In an example, flow entry distributor module 110 may program the flow entries for an unknown flow on appropriate network devices of a cluster assigned to the optimizer 100.

[0019] FIG. 2 is a diagram of an example network system 200 for controlling an unknown flow inflow to an SDN controller in a software defined network. Network system 100 may include an SDN controller 202, an optimizer “A” 204, an optimizer “B” 212, and a plurality of network switches 206, 208, 210, 214, and 216. Although only one SDN controller 202, two optimizers, and five switches are shown in FIG. 2, other examples of this disclosure may include more than one SDN controller, and more or less number of optimizers and network switches. In an example, network system 200 may include one or more computer systems or an end user device(s) (not shown) that may be the source or destination of packet flows into network system 200. In an example, network system 200 may be based on software-defined networking (SDN) architecture.

[0020] SDN controller 202 may be any server, computing device, or the like. In an example, SDN controller 202 may be a computer application (machine-executable instructions). SDN controller 202 may define the data flow that occurs in network system 200. In other words, SDN controller 202 may determine how packets should flow through the network devices 206, 208, 210, 214, and 216 of network system 200. SDN controller 202 may

communicate with network devices 206, 208, 210, 214, and 216 via a standardized protocol (example, OpenFlow) or a suitable API.

[0021] SDN controller 202 may communicate with optimizers (204, 212) and switches 206, 208, 210, 214, and 216 over a computer network. Computer network may be a wireless or wired network. Computer network may include, for example, a Local Area Network (LAN), a Wireless Local Area Network (WAN), a Metropolitan Area Network (MAN), a Storage Area Network (SAN), a Campus Area Network (CAN), or the like. Further, computer network may be a public network (for example, the Internet) or a private network (for example, an intranet).

[0022] Switches 206, 208, 210, 214, and 216 may each include a physical network switch or a virtual switch. In an example, at least one of the network switches 206, 208, 210, 214, and 216 may be an SDN enabled device or an Open-Flow enabled device. Switches 206, 208, 210, 214, and 216 may each include one or more flow tables (not shown). Each flow table in switches 206, 208, 210, 214, and 216 may contain a flow entry (or flow entries) 206. SDN controller 202 may add, update, and delete flow entries in flow tables both reactively (in response to packets) and proactively. Switches 206, 208, 210, 214, and 216 may each communicate with SDN controller 202 via a standardized protocol such as OpenFlow. Switches 206, 208, 210, 214, and 216 may each accept directions from SDN controller 202 to change values in a flow table.

[0023] A flow table matches an incoming packet to a particular flow and specifies the function that may be performed on the packet. If a flow entry matching with a flow is found in a flow table, instructions associated with the specific flow entry may be executed. A packet matches a flow table entry if the values in the packet match fields used for the lookup match those defined in the flow table entry. If no match is found in a flow table (such cases may be termed as “flow table misses”), and the flow may be termed

as an “unknown flow”. In such case, the packet may be forwarded to SDN controller 202. In an example, at least one of the switches 206, 208, and 210 may generate an unknown flow for the SDN controller 202 that may be intercepted by the optimizer “A” 204. Likewise, at least one of the switches 214 and 216 may generate an unknown flow for the SDN controller 202 that may be intercepted by the optimizer “B” 212.

[0024] In an example, switches 206, 208, 210, 214, and 216 present in network system 200 may be divided into a plurality of groups. Each group may be called as a “cluster”. In such case, an exclusive optimizer may be assigned for each cluster of switches. For each cluster, an assigned optimizer may act as a mediator between a SDN controller and a switch present in the cluster. In an instance, an assigned optimizer may intercept an unknown flow from a cluster switch to an SDN controller. Referring to FIG. 2, switches 206, 208, and 210, may be grouped together to form a cluster, and optimizer 204 “A” may be assigned to the cluster. In such case, optimizer “A” 204 may intercept an unknown flow for SDN controller 202 from one or more switches (206, 208, or 210) of the cluster. Likewise, switches 214 and 216 may be grouped together to form another cluster, and optimizer “B” 212 may be assigned to the cluster. In such case, optimizer “B” 212 may intercept an unknown flow for SDN controller 202 from one of the switches (214 and 216) of the cluster.

[0025] In an example, optimizers “A” 204 and “B” 212 may each be analogous to system 100, in which like reference numerals correspond to the same or similar, though perhaps not identical, components. For the sake of brevity, components or reference numerals of FIG. 2 having a same or similarly described function in FIG. 1 are not being described in detail in connection with FIG. 2. Said components or reference numerals may be considered alike. Optimizers “A” 204 and “B” 212 may each include a flow transceiver module, a control communication module, an aggregator module, a packet caching module, and a flow entry distributor module. In an example,

aforementioned modules may perform functionalities similar to those described for flow transceiver module 102, control communication module 104, aggregator module 106, packet caching module 108, and flow entry distributor module 110 of FIG. 1, respectively. In an example, the flow transceiver module of optimizer 204 may intercept an unknown flow from switch 206 to SDN controller 202 in network system 200. The aggregate module may aggregate a portion of a data packet from each data packet in a plurality of data packets from the unknown flow, wherein the selection of the portion of the data packet may be based upon a predefined agreement with the SDN controller 202. The aggregate module may then send only the aggregated portions of the data packets to the SDN controller 202, in a single packet. The original data packets from the unknown data flow that are intercepted by the flow transceiver module may not be shared with the SDN controller 202. The packet caching module may buffer the plurality of data packets from the unknown flow. Once the SDN controller 202 receives the package containing the aggregated portions of the data packets, it may determine a flow path for the unknown flow, and communicate the flow path to the optimizer 204. Based on the received flow path, the flow entry distributor module may configure a flow entry for the flow path in an appropriate switch of the network system. In an example, the SDN controller 202 may configure a flow entry for the flow path in an appropriate switch of the network system 200.

[0026] FIG. 3 is a block diagram of an example method 300 for controlling an unknown flow inflow to an SDN controller in a software defined network. The method 300, which is described below, may be executed on a computing device such as system of FIG. 1 or optimizers 214, 212 of FIG. 2. However, other suitable computing devices may execute method 300 as well. At block 302, an optimizer may be provided, between a switch and an SDN controller, to intercept an unknown flow from the switch to the SDN controller, in a software defined network. At block 304, a portion of a data packet from each data packet in a plurality of data packets from the unknown flow may be

aggregated at the optimizer. In an example, selection of the portion of the data packet may be based upon a predefined criterion between the optimizer and the SDN controller. At block 306, only the aggregated portion of each data packet may be sent from the optimizer to the SDN controller, in a single package.

[0027] FIG. 4 is a block diagram of an example system 400 for controlling an unknown flow inflow to an SDN controller in a software defined network. System 400 includes a processor 402 and a machine-readable storage medium 404 communicatively coupled through a system bus. In an example, system 400 may be analogous to SDN controller 102 of FIG. 1. Processor 402 may be any type of Central Processing Unit (CPU), microprocessor, or processing logic that interprets and executes machine-readable instructions stored in machine-readable storage medium 404. Machine-readable storage medium 404 may be a random access memory (RAM) or another type of dynamic storage device that may store information and machine-readable instructions that may be executed by processor 402. For example, machine-readable storage medium 404 may be Synchronous DRAM (SDRAM), Double Data Rate (DDR), Rambus DRAM (RDRAM), Rambus RAM, etc. or storage memory media such as a floppy disk, a hard disk, a CD-ROM, a DVD, a pen drive, and the like. In an example, machine-readable storage medium may be a non-transitory machine-readable medium. Machine-readable storage medium 404 may store instructions 406, 408, and 410. In an example, instructions 406 may be executed by processor 402 to intercept an unknown flow from a network device to an SDN controller in a software defined network. Instructions 408 may be executed by processor 402 to aggregate a header portion of a data packet from each data packet in a plurality of data packets from the unknown flow. Instructions 410 may be executed by processor 402 to send, in a single package, only the aggregated header portion of the data packet to the SDN controller.

[0028] For the purpose of simplicity of explanation, the example method of FIG. 3 is shown as executing serially, however it is to be understood and appreciated that the present and other examples are not limited by the illustrated order. The example systems of FIGS. 1, 2 and 4, and method of FIG. 3 may be implemented in the form of a computer program product including computer-executable instructions, such as program code, which may be run on any suitable computing device in conjunction with a suitable operating system (for example, Microsoft Windows, Linux, UNIX, and the like). Embodiments within the scope of the present solution may also include program products comprising non-transitory computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer. By way of example, such computer-readable media can comprise RAM, ROM, EPROM, EEPROM, CD-ROM, magnetic disk storage or other storage devices, or any other medium which can be used to carry or store desired program code in the form of computer-executable instructions and which can be accessed by a general purpose or special purpose computer. The computer readable instructions can also be accessed from memory and executed by a processor.

[0029] It should be noted that the above-described examples of the present solution is for the purpose of illustration only. Although the solution has been described in conjunction with a specific embodiment thereof, numerous modifications may be possible without materially departing from the teachings and advantages of the subject matter described herein. Other substitutions, modifications and changes may be made without departing from the spirit of the present solution. All of the features disclosed in this specification (including any accompanying claims, abstract and drawings), and/or all of the steps of any method or process so disclosed, may be combined in any combination, except combinations where at least some of such features and/or steps are mutually exclusive.

Claims:

1. A method of controlling an unknown flow inflow to an SDN controller in a software defined network (SDN), comprising:

providing an optimizer, between a switch and an SDN controller, to intercept an unknown flow from the switch to the SDN controller, in a software defined network;

aggregating, at the optimizer, a portion of a data packet from each data packet in a plurality of data packets from the unknown flow, wherein selection of the portion of the data packet is based upon a predefined criterion between the optimizer and the SDN controller; and

sending, from the optimizer to the SDN controller, only the aggregated portion of the data packet from each data packet, in a single package.

2. The method of claim 1, further comprising buffering the plurality of data packets from the unknown flow at the optimizer.

3. The method of claim 2, further comprising releasing, by the optimizer, the buffered data packets into the software defined network, in response to determining by the optimizer that the SDN controller has decided a flow path for the unknown flow.

4. The method of claim 1, wherein a flow path for the unknown flow is determined by the SDN controller, in response to receiving the aggregated portion of each data packet.

5. The method of claim 4, wherein the flow path for the unknown flow is configured in a network device by the SDN controller.

6. A system to control an unknown flow inflow to an SDN controller in a software defined network (SDN), comprising:

a flow transceiver module to intercept an unknown flow from a switch to an SDN controller, in a software defined network; and

an aggregator module to:

aggregate a portion of a data packet from each data packet in a plurality of data packets from the unknown flow, wherein selection of the portion of the data packet is based upon a predefined agreement with the SDN controller; and send, in a single packet, only the aggregated portion of the data packet from each data packet to the SDN controller.

7. The system of claim 6, further comprising a packet caching module to buffer the plurality of data packets from the unknown flow.

8. The system of claim 6, further comprising a flow entry distributor module to receive a flow path for the unknown flow from the SDN controller.

9. The system of claim 8, wherein the flow entry distributor module to configure a flow entry in a network device based on the received flow path.

10. The system of claim 6, wherein the portion includes a header portion of the data packet.

11. A non-transitory machine-readable storage medium comprising instructions to control an unknown flow inflow to an SDN controller in a software defined network, the instructions executable by a processor to:

intercept an unknown flow from a network device to an SDN controller in a software defined network;

aggregate a header portion of a data packet from each data packet in a plurality of data packets from the unknown flow; and

send, in a single package, only the aggregated header portion of the data packet from each data packet to the SDN controller.

12. The storage medium of claim 11, further comprising instructions to buffer the plurality of data packets from the unknown flow.

13. The storage medium of claim 11, further comprising instructions to release the buffered data packets into the software defined network in response to a determination that the SDN controller has decided a flow path for the unknown flow.

14. The storage medium of claim 11, further comprising instructions to receive, from the SDN controller, a flow path for the unknown flow in the software defined network.

15. The storage medium of claim 14, further comprising instructions to configure the flow path for the unknown flow in a network device of the software defined network.

1/4

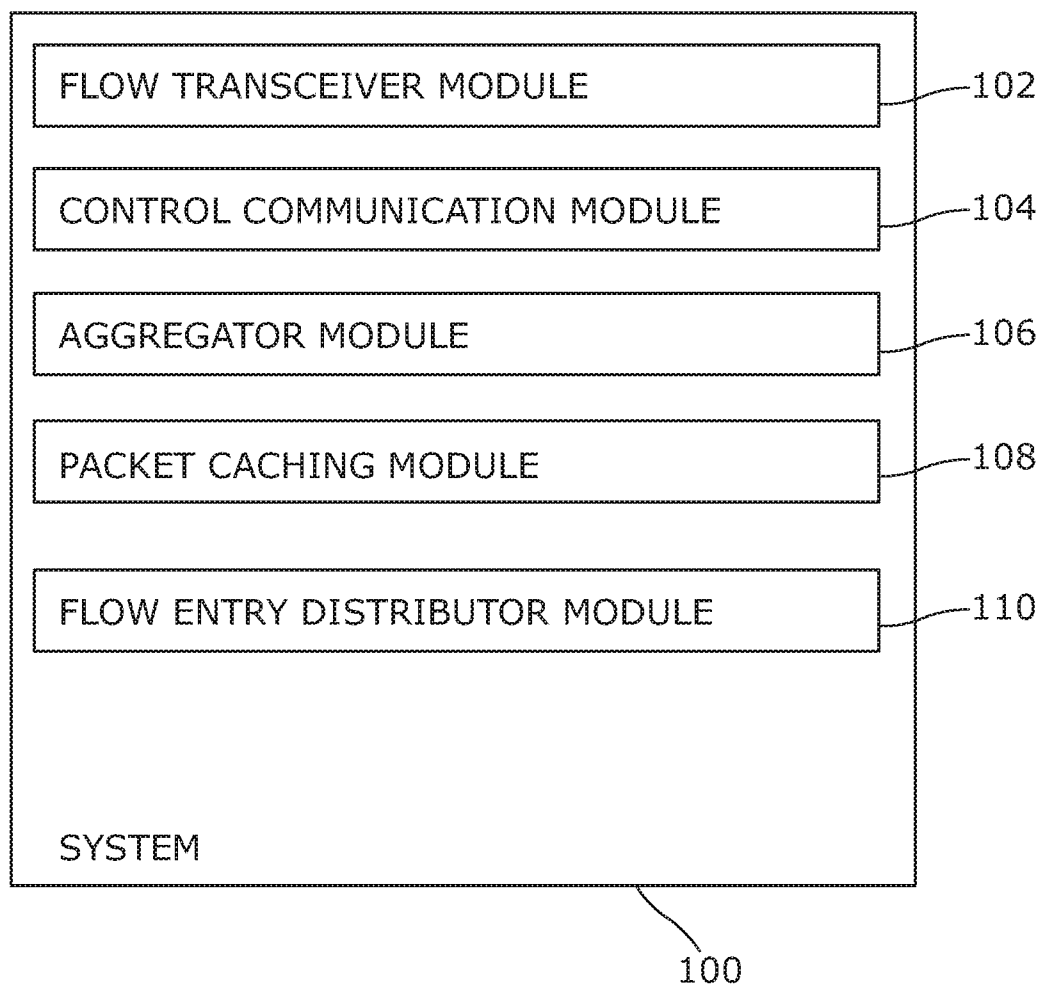


Fig. 1

2/4

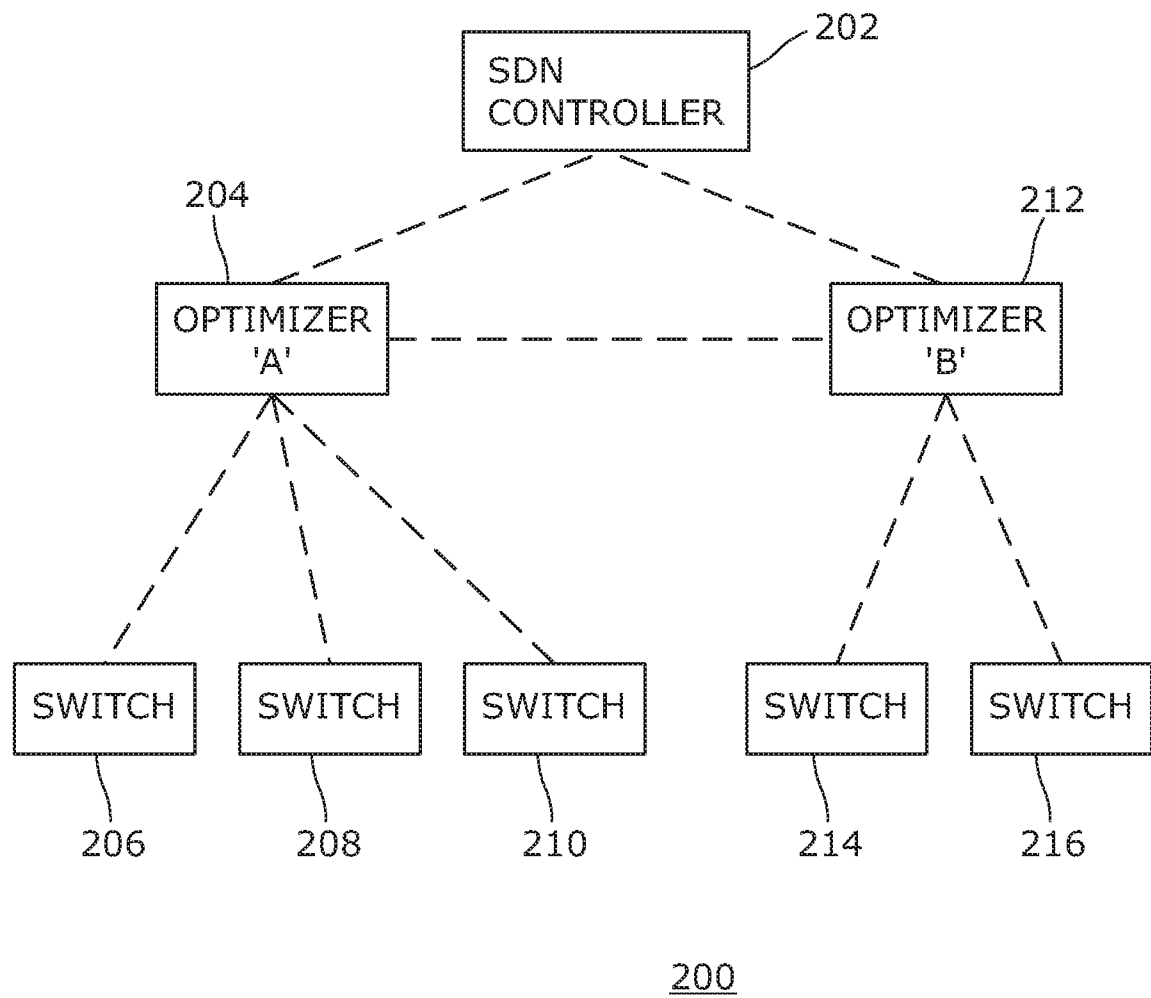


Fig. 2

3/4

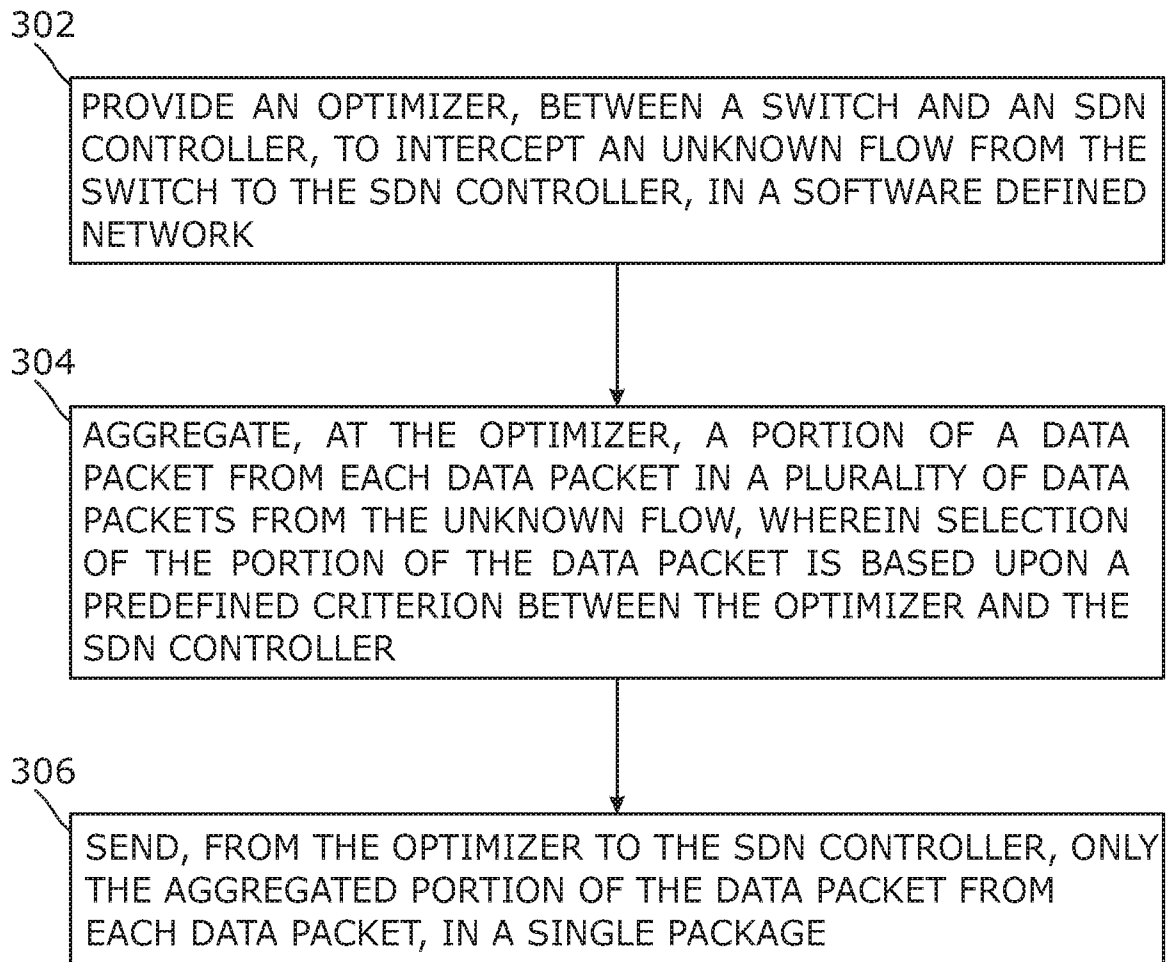
300

Fig. 3

4/4

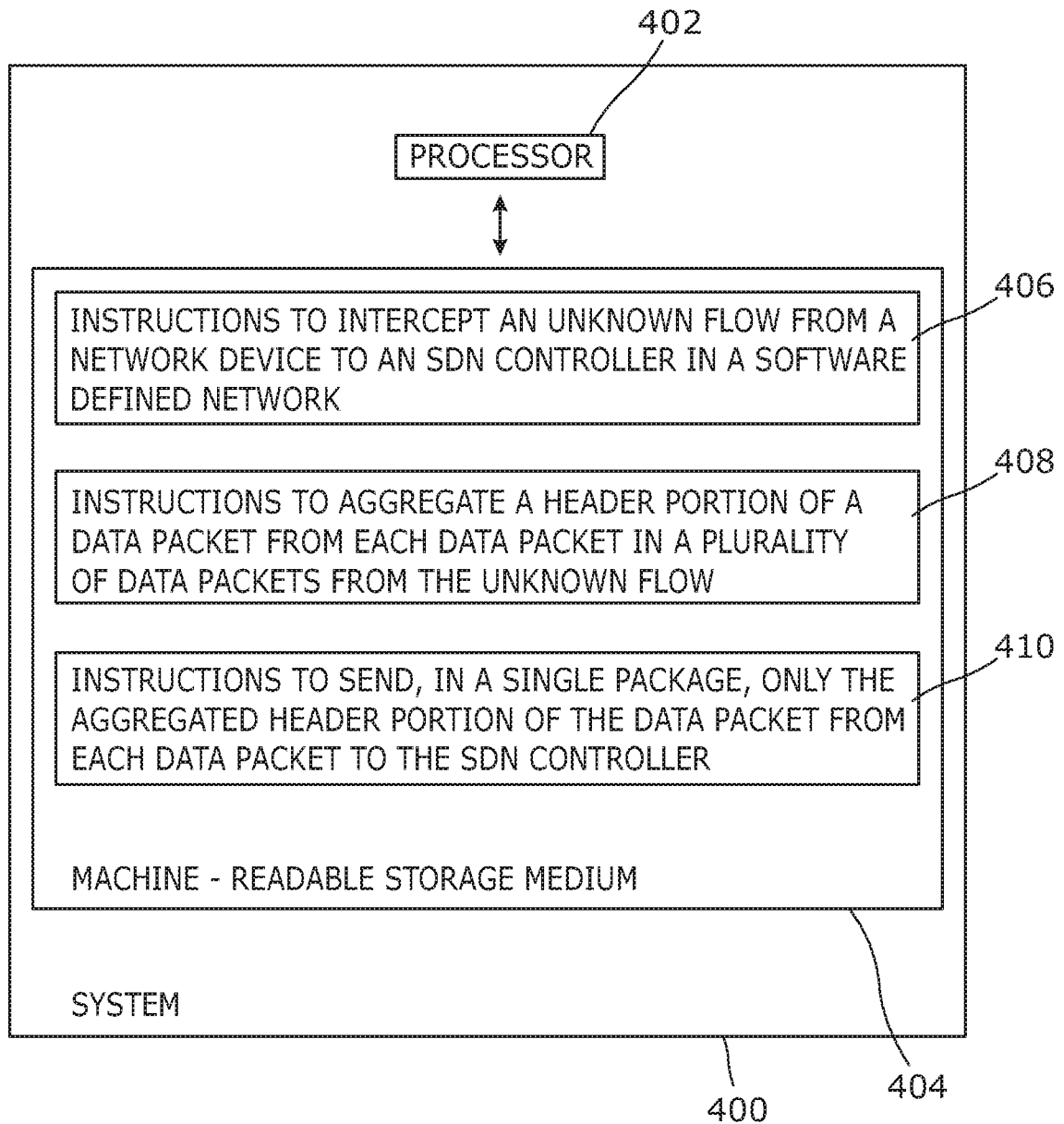


Fig. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/026234**A. CLASSIFICATION OF SUBJECT MATTER****H04L 12/891(2013.01)i, H04L 12/937(2013.01)i**

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

B. FIELDS SEARCHEDMinimum documentation searched (classification system followed by classification symbols)
H04L 12/891; G06F 15/173; H04L 12/26; H04L 12/28; H04L 12/56; H04L 12/937Documentation 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 modelsElectronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: SDN, network, controller, optimizer, data, packet, aggregate, header, unknown, flow, criteria, selection**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2013-0010803 A1 (SYUHEI YAMAGUCHI) 10 January 2013 See paragraphs [0003]-[0004], [0015], [0089]-[0090]; and figure 12.	1-15
Y	US 2013-0114466 A1 (NICIRA, INC.) 09 May 2013 See paragraphs [0127]-[0128], [0205], [0344]; and figures 10, 23.	1-15
A	US 2013-0259044 A1 (YOSUKE TANABE) 03 October 2013 See paragraphs [0003]-[0004], [0012]-[0013], [0025], [0048]; and figures 3-4, 7.	1-15
A	US 2013-0223452 A1 (HIDEYUKI SHIMONISHI et al.) 29 August 2013 See paragraphs [0016]-[0017], [0039], [0056]; and figure 2.	1-15
A	US 2013-0103817 A1 (TEEMU KOPONEN et al.) 25 April 2013 See paragraphs [0003], [0126], [0141], [0527]; 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

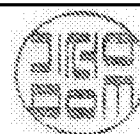
Date of the actual completion of the international search
25 November 2015 (25.11.2015)Date of mailing of the international search report
02 December 2015 (02.12.2015)Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 35208,
Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

KIM, Seong Woo

Telephone No. +82-42-481-3348



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/026234

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2013-0010803 A1	10/01/2013	CN 102783097 A CN 102783097 B EP 2552059 A1 EP 2552059 A4 EP 2552059 B1 JP 5440691 B2 WO 2011-118566 A1	14/11/2012 08/04/2015 30/01/2013 27/11/2013 03/12/2014 12/03/2014 29/09/2011
US 2013-0114466 A1	09/05/2013	None	
US 2013-0259044 A1	03/10/2013	CN 103262479 A EP 2654251 A1 JP 2014-161098 A JP 5534033 B2 JP 5725236 B2 KR 10-1500251 B1 KR 10-2013-0099199 A WO 2012-081721 A1	21/08/2013 23/10/2013 04/09/2014 25/06/2014 27/05/2015 06/03/2015 05/09/2013 21/06/2012
US 2013-0223452 A1	29/08/2013	WO 2012-060403 A1	10/05/2012
US 2013-0103817 A1	25/04/2013	AU 2013-249152 A1 CA 2849930 A1 CN 103891209 A CN 104081734 A CN 104170334 A EP 2748706 A1 EP 2748977 A1 EP 2748990 A1 EP 2748993 A2 EP 2748994 A1 JP 2014-535213 A JP 2015-501109 A JP 2015-507448 A KR 10-2014-0066781 A US 2013-0103818 A1 US 2013-208623 A1 US 2013-211549 A1 US 2013-212148 A1 US 2013-212235 A1 US 2013-212243 A1 US 2013-212244 A1 US 2013-212245 A1 US 2013-212246 A1 US 2013-219037 A1 US 2013-219078 A1 US 2014-0247753 A1 US 2014-0348161 A1 US 2014-0351432 A1	24/10/2013 02/05/2013 25/06/2014 01/10/2014 26/11/2014 02/07/2014 02/07/2014 02/07/2014 02/07/2014 02/07/2014 25/12/2014 08/01/2015 05/03/2015 02/06/2014 25/04/2013 15/08/2013 15/08/2013 15/08/2013 15/08/2013 15/08/2013 15/08/2013 15/08/2013 15/08/2013 22/08/2013 22/08/2013 04/09/2014 27/11/2014 27/11/2014

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/026234

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		WO 2013-063329 A1	02/05/2013
		WO 2013-063330 A1	02/05/2013
		WO 2013-063332 A1	02/05/2013
		WO 2013-158917 A2	24/10/2013
		WO 2013-158917 A3	05/06/2014
		WO 2013-158918 A1	24/10/2013
		WO 2013-158918 A4	16/01/2014
		WO 2013-158920 A1	24/10/2013