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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 September 2014 (25.09.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/150059 A1

(51) International Patent Classification:
H04W 4/00 (2009.01) *H04W 4/20* (2009.01)

(21) International Application Number:
PCT/US2014/022055

(22) International Filing Date:
7 March 2014 (07.03.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/838,719 15 March 2013 (15.03.2013) US

(71) Applicant: CISCO TECHNOLOGY, INC. [US/US]; 170 West Tasman Drive, San Jose, California 95134 (US).

(72) Inventors: PFEIFER, Jason; 170 West Tasman Drive, San Jose, California 95134 (US). PRUSS, Richard M.; 170 West Tasman Drive, San Jose, California 95134 (US). MCDOWALL, John E.; 170 West Tasman Drive, San Jose, California 95134 (US).

(74) Agents: MCCLELLAN, Gero G. et al.; Patterson & Sheridan, LLP, 24 Greenway Plaza, Suite 1600, Houston, TX 77046 (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).

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

(54) Title: SUPPORTING PROGRAMMABILITY FOR ARBITRARY EVENTS IN A SOFTWARE DEFINED NETWORKING ENVIRONMENT

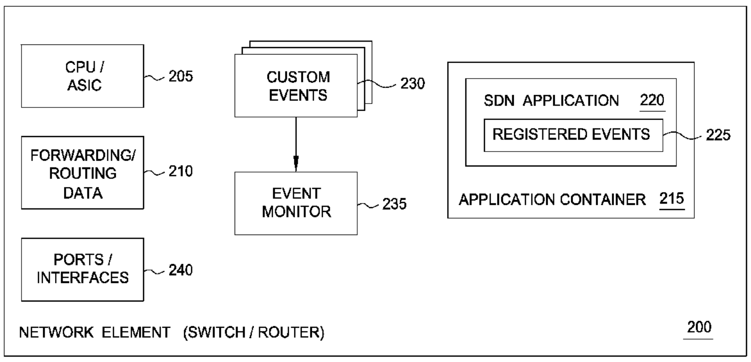


FIG. 2

(57) Abstract: Techniques are disclosed for using arbitrary criteria to define events occurring within a network infrastructure, as well and techniques for detecting and responding to the occurrence of such custom events. Doing so allows a collection of networking elements (switches, routers, etc.) to perform a variety of distributed functions from within the network itself to respond to custom events. Further, because custom events are published across the network, multiple network elements can communicate and respond to the same event. Thus, unlike currently available event management systems, custom events (and responding applications) can be used to create and coordinate software defined networking within a common network infrastructure.

WO 2014/150059 A1

SUPPORTING PROGRAMMABILITY FOR ARBITRARY EVENTS IN A SOFTWARE DEFINED NETWORKING ENVIRONMENT

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims benefit of United States patent application serial number 13/838,719, filed March 15, 2013, which is herein incorporated by reference.

TECHNICAL FIELD

[0002] Embodiments of the invention generally relate to techniques for managing computer network infrastructure. More, techniques are disclosed to support programmability for arbitrary events in a software defined networking environment. BACKGROUND

[0003] Networking hardware, Switches and routers, have not traditionally been programmable entities. Instead, network hardware is configured using command line interfaces or external applications and a console connection. Some programming frameworks running on connected hosts or appliances are can also be used to manage network hardware. These approaches put applications managing the network on systems external to the network itself.

[0004] While most network management occurs via command line interfaces or configuration tools, some network devices may be configured to respond to specific events that occur on a given network device. For example, embedded event manager (EEM) is a feature included by Cisco's IOS operating system (and some other Cisco operating systems such as IOS-XR, IOS-XE, and NX-OS). EEM allows the behavior of a Cisco network device (e.g., a switch or router) to adapt to some user requirements by supporting a limited set of events and actions performed in response to such events within the network device itself. Using EEM, some problems can be identified and resolved automatically in advance by setting event triggers (called event detectors) to watch for specific types of situations or thresholds, or run a set of actions periodically.

[0005] At the same time EEM, and other network management frameworks, does not support the handling of custom or arbitrary events. That is, while EEM can be used to configure a switch or router to respond to a limited set of events, the range of such events is limited to whatever is pre-configured for a given switch. Thus, EEM does not allow network elements to respond to arbitrary user-specified events, custom protocols, or other compound or complex events. Further, the event processing provided by EEM (and other similar technologies) is limited to processing events local to the switch itself.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] So that the manner in which the above recited aspects are attained and can be understood in detail, a more particular description of embodiments of the invention, briefly summarized above, may be had by reference to the appended drawings.

[0007] It is to be noted, however, that the appended drawings illustrate only typical embodiments of this invention and are therefore not to be considered limiting of its scope, for the invention may admit to other equally effective embodiments.

[0008] Figure 1 illustrates an example computer networking infrastructure configured to support software defined networking, according to one embodiment.

[0009] Figure 2 further illustrates a network element configured to support programmability for arbitrary events in a software defined networking environment, according to one embodiment.

[0010] Figure 3 illustrates a method for a network device to respond to arbitrary events within a software defined networking environment, according to one embodiment.

[0011] Figure 4 illustrates a method for a network device to publish arbitrary events within a software defined networking environment, according to one embodiment.

[0012] Figure 5 illustrates multiple network elements configured to respond to user-defined events, within a software defined networking environment, according to one embodiment.

[0013] Figure 6 illustrates an example computing system configured with network management application used to configure a software defined networking environment, according to one embodiment.

DESCRIPTION

OVERVIEW

[0014] Embodiments presented herein include a method for processing network events on a network device in a common network infrastructure. This method may generally include launching, in a container independent from a traffic management component of the network device, an application. This method may also include registering the application to receive notifications of at least a first network even and receiving, by the application, a notification that an instance of the first network event has occurred. The notification indicates an event type and includes metadata associated with the instance of the network event. In response, the application executes one or more functions on the network device from within the container.

[0015] Other embodiments include, without limitation, a computer-readable medium that includes instructions that enable a processing unit to implement one or more aspects of the disclosed methods as well as a system having a processor, memory, and application programs configured to implement one or more aspects of the disclosed methods.

DESCRIPTION OF EXAMPLE EMBODIMENTS

[0016] Embodiments presented herein provide techniques for supporting programmability for arbitrary events in a software defined networking environment. More specifically, embodiments presented herein allow developers to define custom events monitored for within a network infrastructure. Once defined, an event-matching application may monitor for occurrences of the custom event. The event matching application can run as a process on a switch or router, on a blade in the same chassis as the switch, or in a separate network-connected computing system. The event matching application has programmatic access to read and modify any events already generated by a network device (e.g., embedded event manager (EEM) events on a Cisco switch or router). The event matching application can also access policies, flows, packets, and other information associated with any traffic coming into the network element. More generally, any capability that the switch possesses may be exposed as a possible information source for the event matching application, including functionality such as deep packet inspection, quality of service (QoS), system logs, interface states, forwarding rules, etc.

[0017] When a custom event occurs, the event matching application publishes the event to registered subscribers. In turn, any network device or element (including the network element observing the event) which has registered to receive notification that the custom event has occurred can respond by executing custom applications. For example, in response to an interface going down on a switch, a custom event indicating that this has occurred could be published to neighboring switches. In response, an application running on the neighboring switches could update routing information regarding the link state of the interface. Further, in response to the custom event, the switch with the failed interface may invoke applications configured to evaluate whether the link state can be repaired programmatically, and if not send a message to an administrator, all from within the switch itself.

[0018] As another example, assume an enterprise wishes to monitor all off-site email communication to identify whether email traffic is being sent out in an unauthorized manner. To do this, the network administrator could create a custom "insecure email" event as 1) an SMTP connection 2) with a receiver outside the organization 3) including an attachment 4) where a portion of the attachment includes the string "confidential" or "secret" in it. When an “insecure email” event occurs, the switch could forward the suspect email to the appropriate administrator.

[0019] In one embodiment, applications on the network elements execute in a container. The container may provide an execution space on each network element independent from the routing and switching functions running on that network element. As the above examples illustrate, such applications can modify traffic, settings, or other aspects of the network element itself, but can also generate events processed by other network elements. Doing so allows the collection of networking elements (switches, routers, etc.) to perform a variety of distributed functions from within the network itself in response to custom events, referred to as software defined networking. Further, because custom events are published across the network, multiple network elements can communicate and respond to the same event. Thus, unlike currently available event management systems, custom events (and responding applications) can be used to create and coordinate software defined networking within a network infrastructure.

[0020] In the following, reference is made to embodiments of the invention. However, the invention is not limited to any specifically described embodiment. Instead, any combination of the following features and elements, whether related to different embodiments or not, is contemplated to implement and practice the invention. Furthermore, although embodiments of the invention may achieve advantages over other possible solutions and/or over the prior art, whether or not a particular advantage is achieved by a given embodiment is not limiting of the invention. Thus, the following aspects, features, embodiments and advantages are merely illustrative and are not considered elements or limitations of the appended claims except where explicitly recited in a claim(s). Likewise, reference to “the invention” shall not be construed as a generalization of any inventive subject matter disclosed herein and shall not be considered to be an element or limitation of the appended claims except where explicitly recited in a claim(s).

[0021] Aspects of the present invention may be embodied as a system, method or computer program product. Accordingly, aspects of the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, *etc.*) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a “circuit,” “module” or “system.” Furthermore, aspects of the present invention may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon.

[0022] Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples a computer readable storage medium include: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the current context, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus or device.

[0023] The flowchart and block diagrams in the Figures illustrate the architecture, functionality and operation of possible implementations of systems, methods and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. Each block of the block diagrams and/or flowchart illustrations, and combinations of blocks in the block diagrams and/or flowchart illustrations can be implemented by special-purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

[0024] Embodiments of the invention may be provided to end users through a cloud computing infrastructure. Cloud computing generally refers to the provision of scalable computing resources as a service over a network. More formally, cloud computing may be defined as a computing capability that provides an abstraction between the computing resource and its underlying technical architecture (*e.g.*, servers, storage, networks), enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Thus, cloud computing allows a user to access virtual computing resources (*e.g.*, storage, data, applications, and even complete virtualized computing systems) in “the cloud,” without regard for the underlying physical systems (or locations of those systems) used to provide the computing resources. A user can access any of the resources that reside in the cloud at any time, and from anywhere across the Internet.

[0025] As an example, in context of the present invention, a cloud hosting company (hosting a data center with a hosts connected to a network infrastructure) may monitor and charge for certain types of bandwidth. Each tenant of the provider may share the underlying network infrastructure and the provider may use Multi-Protocol Label Switching (MPLS) tags to separate different virtual network flows. In such a case, the event manager could identify tenant flows based on the MPLS tags. The application running on the network elements could then calculate aggregate bandwidth used by each tenant and report it to a logging application running outside of the networking infrastructure.

[0026] Figure 1 illustrates an example computer networking infrastructure 100 configured to support software defined networking, according to one embodiment. The networking infrastructure 100 is included to be a representative view of a variety of networking environments. For example, a collection of server systems and network hardware inside a typical corporate data center (represented here as dashed line 102). As shown, multiple hosts $105_{1\text{-}N}$ and $110_{1\text{-}N}$ are connected to switches 115_1 and 115_2, respectively. Each host 105, 115 generally corresponds to a computing device. For example, hosts 105 and 110 could be a set of blade servers disposed within a chassis. In such a case, each host 105, 110 could be a server blade running an operating system and applications directly. Hosts 105, 110 could also be one of multiple virtual machine instances executing on a server blade.

[0027] Illustratively, the computing systems of data center 102 are each attached to a network switch. Switch 115_1 (for hosts 105) and network switch 115_2 (for hosts 110). Continuing with the example of a blade server (or other converged infrastructure), switch 115 could be a “top-of-rack switch” connecting the computing devices within the blade server to an aggregation switch 120.

Switch 115_{1-2} provides a network device with interface ports, a processor, a memory, and forwarding logic used to process network traffic.

[0028] Like switch 115_1 the aggregation switch 120 provides a network device with a memory, processor, and set of network interfaces. In this example, aggregation switch 120 is generally configured to forward data traffic from hosts (via switches 115_{1-2}) towards a destination (e.g., from one of hosts 105, 110 to another or from one of hosts 105, 110 towards network 150). Aggregation switch 120 is connected to an edge router 130. In turn, edge router 130 connects the systems (switches hosts, etc.) of data center 102 to an external network 150. Like switches 115 and aggregation switch 120, edge router 130 provides a network device configured to process network data flows from other switches, other hosts and other networks. Accordingly, edge router 130 can include a memory, processor, and network interfaces to connect other system within the networking infrastructure 100 to the edge router 130.

[0029] In one embodiment, switches 115, 120, or router 130, may include an execution environment (container) used to execute custom applications which respond to events published by the network elements (switches, routers) in the networking infrastructure 100 or published by another network element of network infrastructure 100. Additionally, the switches 115, 120 or router 130 may include an event matching application configured to monitor for, and publish, the occurrence of custom events to the network infrastructure 100. The events, execution container, and event matching component, are each discussed in further detail below.

[0030] Note, the computer networking infrastructure 100 is simplified as needed to present an infrastructure which supports programmability for arbitrary events in a software defined networking environment. One of ordinary in the art will recognize that networking infrastructure 100 is representative of a broad variety of network hardware, e.g., switches, routers, bridges, etc., and the computing hosts servers, blades, appliances, virtual machine instances, etc.) as well as converged infrastructure elements (e.g., a blade server chassis with integrated networking elements).

[0031] Figure 2 further illustrates a network element 200 configured to support programmability for arbitrary events in a software defined networking environment, according to one embodiment. Again, network element 200 generally corresponds to a routing, switching, bridge, firewall, appliance, or other network hardware used to process, forward, route, or otherwise provide and manage a computer network.

[0032] Illustratively, network element includes a CPU/ASIC (application specific integrated circuit) 205, forwarding/routing data 210, an application container 215, event monitor 235, and custom event definitions 230. The CPU/ASIC 205 provides a processor generally configured to execute instructions to process network packets, frames, flows, etc., received and forwarded over ports/interfaces 240. In general, connected hosts (or other connected devices) forward network frames/packets to network element 200. In turn, the CPU 205 executes forwarding logic to determine how to process any network traffic. Typically, e.g., to make a forwarding decision based on forwarding / routing data 210. Once a decision is made, a network packet/frame received on one interface 240 is forwarded out on another interface 240.

[0033] The custom events 230 correspond to any user specified event approaite, as needed for a particular case. For example, events may be related to OIR (Online Insertion and Removal), CLI (Command Line Interface), Syslog, XML-RPC, IP SLAs,. NetFlow, application specific events, configuration changes, interface counters, redundancy framework, SNMP notification (i.e. when the device receives a trap), resource usage, Timers, operating system process, counters, diagnostics, environmental (e.g. temperature), routing, object tracking, etc. Note, some events may be provided by the operating system of a network element. For example, a number of these events may be generated by an embedded event manager (EEM) on a network element running a version of Cisco's IOS. In addition to the event itself, a custom event may include data related to an event. For example, an application specific event could be one in which bulk/filtered metric data is sent on a periodic basis containing summary information of other events that have occurred within the related time period. Similarly, a custom event 230 could be composed as a compound of multiple events such, as the interface statistics changing over a certain time period correlated with related interface configuration changes. Further, the custom events 230 could be defined based on an evaluation of network flows, including events corresponding to switch functionality, events triggered using deep packet inspection, events related to quality of service (QoS) or service level agreements (SLA), events triggered by entries in a system log, events triggered by interface states, or the application of a forwarding rule.

[0034] However defined, in one embodiment, the event monitor 235 is configured to detect the occurrence of a custom event 230. For example, the event monitor 235 may monitor network activity (e.g., frames received and forwarded over ports/interfaces 240), monitor system state of the network element, counters, or timers, etc., to determine when a custom event 235 has occurred. When the event monitor 235 detects the occurrence of a custom event 235, the event monitor 235 publishes an event received by any network element registered to receive that event. For example, in one embodiment, the network element is broadcast over the control plane of the network infrastructure, allowing each network element to receive the event and raise it to any registered applications. Further, the event monitor 235 may publish the event to other connected devices (e.g., other switches, routers, servers, etc.). In one embodiment, the event may be specified using a two integer identification mechanism, an application identifier and a type. These values are arbitrarily chosen and agreed upon by both the event publisher and subscriber.

[0035] Additionally, software defined networking (SDN) applications 225 executing on the network element 200 can register to receive an event, shown in Figure 2 as registered events 225. For example, an SDN application 225 may register a callback function with the event monitor 235. In such a case, when the event monitor 235 detects an event registered by SDN application 225, the event monitor 235 invokes the callback function of the SDN application 225. In one embodiment, the SDN application 215 executes on the network element within an application container 215. The application container 215 provides an execution environment separate from the routing and forwarding functions of the network element 200. Note, however while the container separates the SDN application 215 from the other routing/forwarding functions of the network element 200, the SDN application 225 can generally access, modify, configure, data in packets, frames, network flows, as well as configure ports/interfaces 240, update, change, modify forwarding/routing data 240, etc. Further, the SDN application 225 can generate custom events 230, communicate with other network elements, applications, or services, as needed to perform a software defined networking function within a common network infrastructure.

[0036] Figure 3 illustrates a method 300 for a network device to respond to arbitrary events within a software defined networking environment, according to one embodiment. As shown, the method 300 begins at step 305, where an SDN application is launched on one or more network elements. As noted, the SDN application may be configured to register to receive notifications of custom network events (step 310). For example, the SDN application may register a callback function with an event monitor. In such a case, when the event monitor detects a registered event, it can invoke the callback function (step 315). Alternatively, the SDN application may monitor for events published to a network control plane by that network element or published to the control plane by another network element. Once the SDN application is launched, and registered for any custom events, the SDN application can perform a variety of functions from within the network itself, in response to the occurrence of the registered events.

[0037] Figure 4 illustrates a method 400 for a network element to publish arbitrary events within a software defined networking environment, according to one embodiment. As shown, the method 400 begins at step 405, where an event monitor executing on a network element (e.g., switch, router, etc.) monitors for the occurrence of a custom event (step 410). At step 415, once one of the custom events occurs, the event monitor may identify any data associated with the event and publish the event to any registered SDN applications (step 420) or to a control plane for a network infrastructure generally. For example, assume a custom event is used to monitor the link state of interfaces on the network element. Such an event could evaluate system log entries using a regular expression to identify a change in the link state from “UP” to “DOWN.” In such a case, the application could identify the particular interface referenced in a system log entry. Further, once the SDN application is notified of the interface state, it can take actions to reroute network flows, repair the link state, or any other programmatic action from within the network itself. Further still, the same custom event can be shared across SDN applications on a group of network elements in a network infrastructure – resulting in multiple elements (and any connected hosts) responding to link state changes (or other custom events) in a consistent and coordinated manner.

[0038] As another example, a custom event could identify traffic flow of a particular protocol (e.g., SMTP traffic) using deep packet inspection. An event could pass the contents of such traffic to an SDN application for analysis. Similarly, events could be related to MPLS tags and used to monitor data use for traffic flows for multi-tenant data center. As still another example, a custom event could be triggered by a service running on the network element. The network element could have a service configured to monitor response times (e.g., ICMP responses) from certain hosts and generate custom events when the latency falls below (or exceeds) a given threshold. When an event is published (indicating the threshold has been crossed), the latency data can be passed to an application running in a container on the network element – or to an application running on other network elements.

[0039] The latter example is further shown in Figure 5, which illustrates multiple network elements configured to respond to user-defined events, within a software defined networking environment, according to one embodiment. As shown, network elements 500, 525, 530, and 535 provide a set of switches, routers within a common network infrastructure. As such, the network elements 500, 525, 530, and 535 may be connected over a set of interfaces. For example, network element 500 may be an aggregation switch for traffic from switches 525, 530, and 535 (and ultimately, from hosts connected to switches 525, 530, and 535). Assume that network element 450 includes an IPSLA (IP layer service level agreement) component 520 used to monitor the latency of a given host 550. For example, a service provider may have guaranteed a maximum latency for hosts 550. In such a case, the IPSLA component 520 may periodically send ICMP echo requests to network element 525 (or to hosts 550). Further, the IPSLA component 520 may publish IPSLA events, write to a system log, etc. At the same time, the embedded event manager (EEM) applet 515 registers to receive IPSLA events. If the latency passes the threshold specified for an SLA requirement, an event is raised to the EEM applet. In turn, the EEM applet (effectively acting as an event monitor) publishes a custom event with IPSLA event data to network application 510. In response, the network application 510 can execute arbitrary code from within container 505 to, e.g., publish events to SDN applications running on other network elements, reconfigure interfaces on network element 500 or network element 525, notify an administrator, or perform other actions to comply with the provisions of an SLA. Importantly, in this example, the events are published, along with any relevant data, for a SDN application. Further, the custom event can propagate, e.g., as one event shared across the control plane of a network infrastructure or when an SDN application registered to receive the event on one network element publishes an event raised to an SDN application on another SDN application.

[0040] Figure 6 illustrates an example computing system configured with network management application used to configure a software defined networking environment, according to one embodiment. As shown, the computing system 600 includes, without limitation, a central processing unit (CPU) 605, a network interface 615, a network interface 615, a memory 620, and storage 630, each connected to a bus 617. The computing system 600 may also include an I/O device interface 610 connecting I/O devices 512 (*e.g.*, keyboard, display and mouse devices) to the computing system 600. Further, in context of this disclosure, the computing elements shown in computing system 600 is included to be generally representative of both physical computing system (e.g., a system in a data center) as well as virtual computing instances executing within a computing cloud.

[0041] The CPU 605 retrieves and executes programming instructions stored in the memory 620 as well as stores and retrieves application data residing in the memory 630. The interconnect 617 is used to transmit programming instructions and application data between the CPU 605, I/O devices interface 610, storage 630, network interface 615, and memory 620. Note, CPU 605 is included to be representative of a single CPU, multiple CPUs, a single CPU having multiple processing cores, and the like. And the memory 620 is generally included to be representative of a random access memory. The storage 630 may be a disk drive storage device. Although shown as a single unit, the storage 630 may be a combination of fixed and/or removable storage devices, such as fixed disc drives, solid state storage, or optical storage, network attached storage (NAS), or a storage area-network (SAN).

[0042] Illustratively, the memory 620 includes an SDN development tool 622, SDN application 624, and registered events 624. And the storage 630 includes SDN configuration data 632 and network element data 634. The SDN development tool 622 provides one or more applications configured to allow developers to compose SDN applications, e.g., SDN application 624. For example, the SDN development tool 622 can include a variety of code editors, compilers, code generators, etc. SDN development tool 622 may also allow a user to configure SDN applications on network elements, define custom events, configure an event detector with the custom events, register SDN applications to receive events, etc. SDN configuration data 632 specifies what SDN applications are configured on what network elements, what custom events they should receive, and the custom event definitions. Similarly network element data 634 can store any information related to a software defined network, e.g., the current configuration of SDN applications within a network infrastructure. Further, as noted the SDN application 624 can execute on a host system connected to a network element (e.g., a server blade attached to a switch in a data center). Once running, the SDN application 624 can register to receive events (shown as registered events 626) published from within the network infrastructure as well as publish events raised to the applications within an SDN network.

[0043] As described, embodiments presented herein provide techniques for using arbitrary criteria to define events occurring within a common network infrastructure, as well and techniques for detecting and responding to the occurrence of such custom events. That is, embodiments described above provide techniques that support programmability for arbitrary events in a software defined networking environment. Doing so allows the collection of networking elements (switches, routers, etc.) to perform a variety of distributed functions from within the network itself, referred to as software defined networking. Further, because custom events are published across the network, multiple network elements can communicate and respond to the same event. Thus, unlike currently available event management systems, custom events (and responding applications) can be used to create and coordinate software defined networking within a network infrastructure.

[0044] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof, and the scope thereof is determined by the claims that follow.

WHAT IS CLAIMED IS:

1. A networking device, comprising:
a processor;
a traffic management component; and
a memory storing a container, independent from the traffic management component, hosting an application, which, when executed on the processor, performs an operation for responding to a network event, the operation comprising:
receiving a notification that an instance of the network event has occurred, wherein the notification indicates an event type and includes metadata associated with the instance of the network event, and
in response, executing one or more functions of the application.

2. The networking device of claim 1, wherein the notification is received from an event monitor on the networking device.

3. The networking device of claim 2, wherein the application registers a callback function with the event monitor, and wherein the callback function is invoked with the notification.

4. The networking device of claim 1, wherein the notification is received from a second networking device over a network control plane of the common networking infrastructure.

5. The network device of claim 1, wherein at least one of the executed functions generates an event published to application on a second network device over a network control plane of the common networking infrastructure.

6. The networking device of claim 1, wherein the event is generated in response to a service monitoring component executing on the network device determining that a monitored service metric as passed a threshold value.

7. The networking device of claim 1, wherein the event is generated in response to network traffic managed by the traffic management component being processed on the network device.

8. The networking device of claim 1, wherein the networking device is one of a switch and a router.

9. A method for processing network events on a network device in a common network infrastructure, the method comprising:

launching, in a container independent from a traffic management component of the network device, an application;

registering the application to receive notifications of at least a first network event;

receiving, by the application, a notification that an instance of the first network event has occurred, wherein the notification indicates an event type and includes metadata associated with the instance of the network event; and

in response, executing one or more functions of the application.

10. The method of claim 9, wherein the notification is received from an event monitor on the networking device.

11. The method of claim 10, further comprising, registering a callback function with the event monitor, wherein the callback function is invoked with the notification.

12. The method of claim 9, wherein the notification is received from a second networking device over a network control plane of the common networking infrastructure.

13. The method of claim 9, wherein at least one of the executed functions generates an event published to application on a second network device over a network control plane of the common networking infrastructure.

14. The method of claim 9, wherein the event is generated in response to a service monitoring component executing on the network device determining that a monitored service metric as passed a threshold value.

15. The method of claim 9, wherein the event is generated in response to network traffic managed by the traffic management component being processed on the network device.

16. The method of claim 9, wherein the networking device is one of a switch and a router.

17. A computer-readable storage medium storing instructions, which, when executed on a processor, performs an operation for processing network events on a network device in a common network infrastructure, the operation comprising:

launching, in a container independent from a traffic management component of the network device, an application;

registering the application to receive notifications of at least a first network event;

receiving, by the application, a notification that an instance of the first network event has occurred, wherein the notification indicates an event type and includes metadata associated with the instance of the network event; and

in response, executing one or more functions of the application.

18. The computer-readable storage medium of claim 17, wherein the notification is received from an event monitor on the networking device.

19. The computer-readable storage medium of claim 18, wherein the operation further comprises, registering a callback function with the event monitor, wherein the callback function is invoked with the notification.

20. The computer-readable storage medium of claim 17, wherein the notification is received from a second networking device over a network control plane of the common networking infrastructure.

21. The computer-readable storage medium of claim 17, wherein at least one of the executed functions generates an event published to application on a second network device over a network control plane of the common networking infrastructure.

22. The computer-readable storage medium of claim 17, wherein the event is generated in response to a service monitoring component executing on the network device determining that a monitored service metric as passed a threshold value.

23. The computer-readable storage medium of claim 17, wherein the event is generated in response to network traffic managed by the traffic management component being processed on the network device.

24. The computer-readable storage medium of claim 17, wherein the networking device is one of a switch and a router.

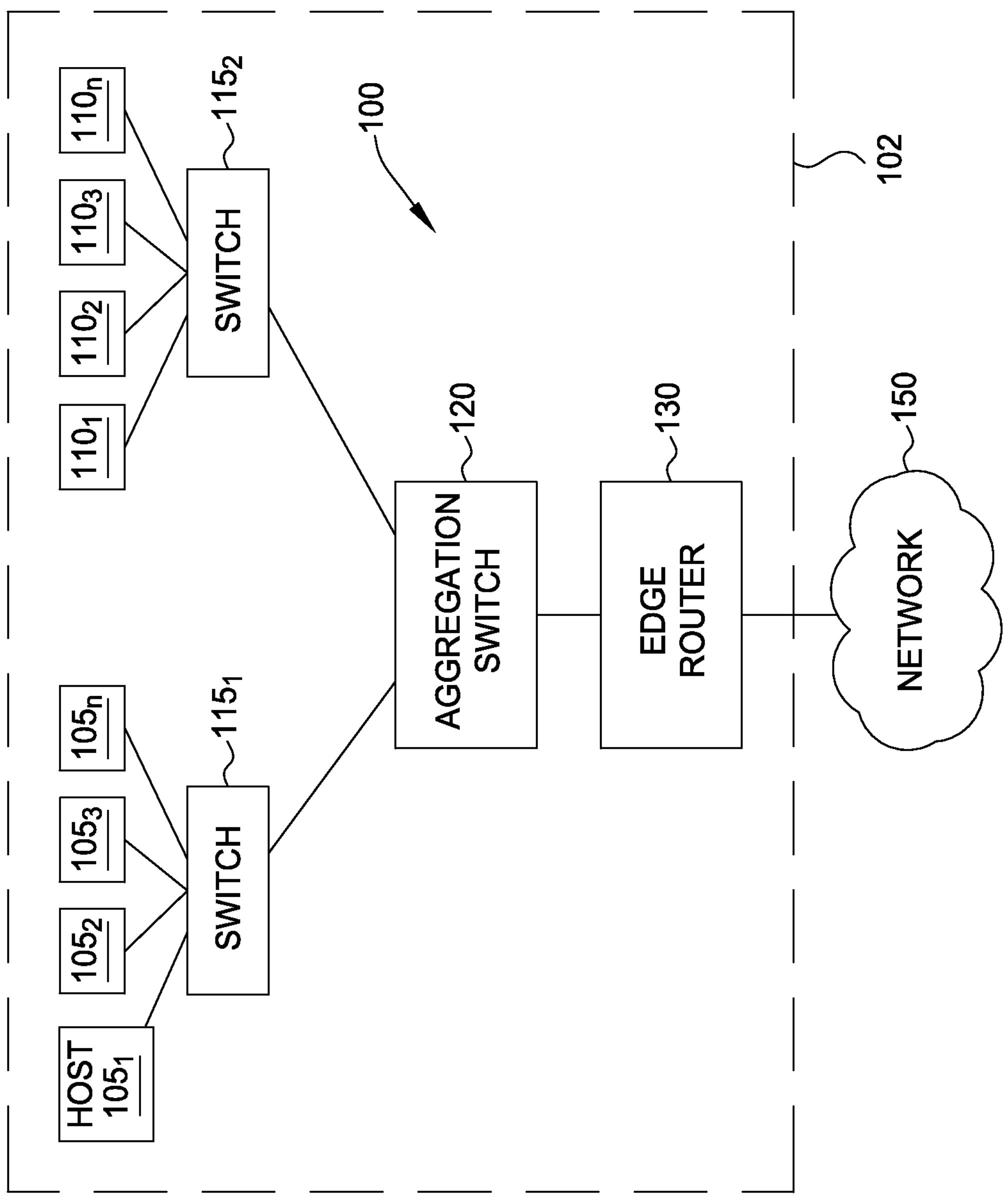
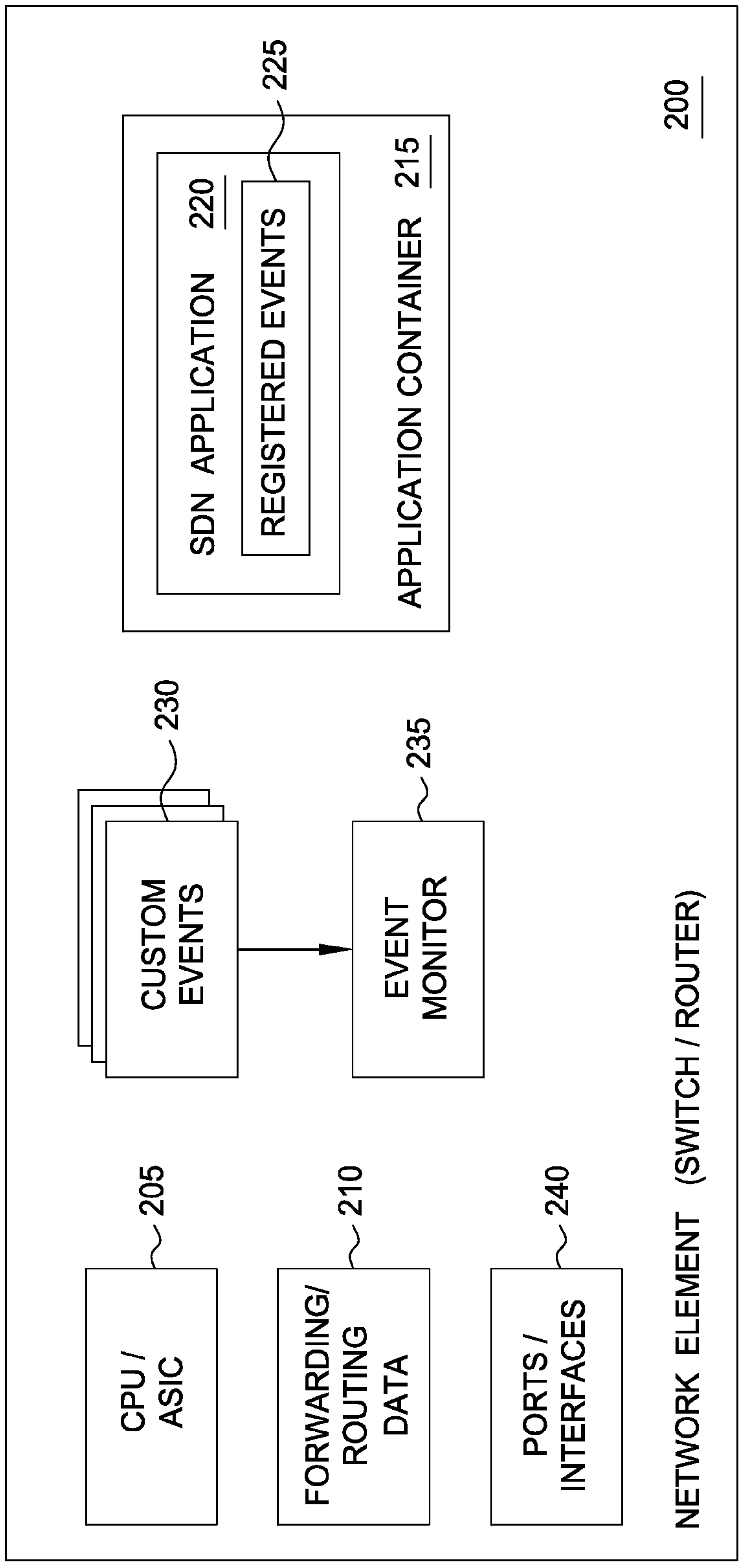

HOST 105_1
105_2
105_3
105_n
SWITCH
115_1
110_1
110_2
110_3
110_n
SWITCH
115_2
100
AGGREGATION SWITCH
120
EDGE ROUTER
130
102
NETWORK
150

FIG. 1



NETWORK ELEMENT (SWITCH / ROUTER)
200
CPU / ASIC
205
FORWARDING/ ROUTING DATA
210
PORTS / INTERFACES
240
CUSTOM EVENTS
230
EVENT MONITOR
235
APPLICATION CONTAINER
215
SDN APPLICATION
220
REGISTERED EVENTS
225

FIG. 2

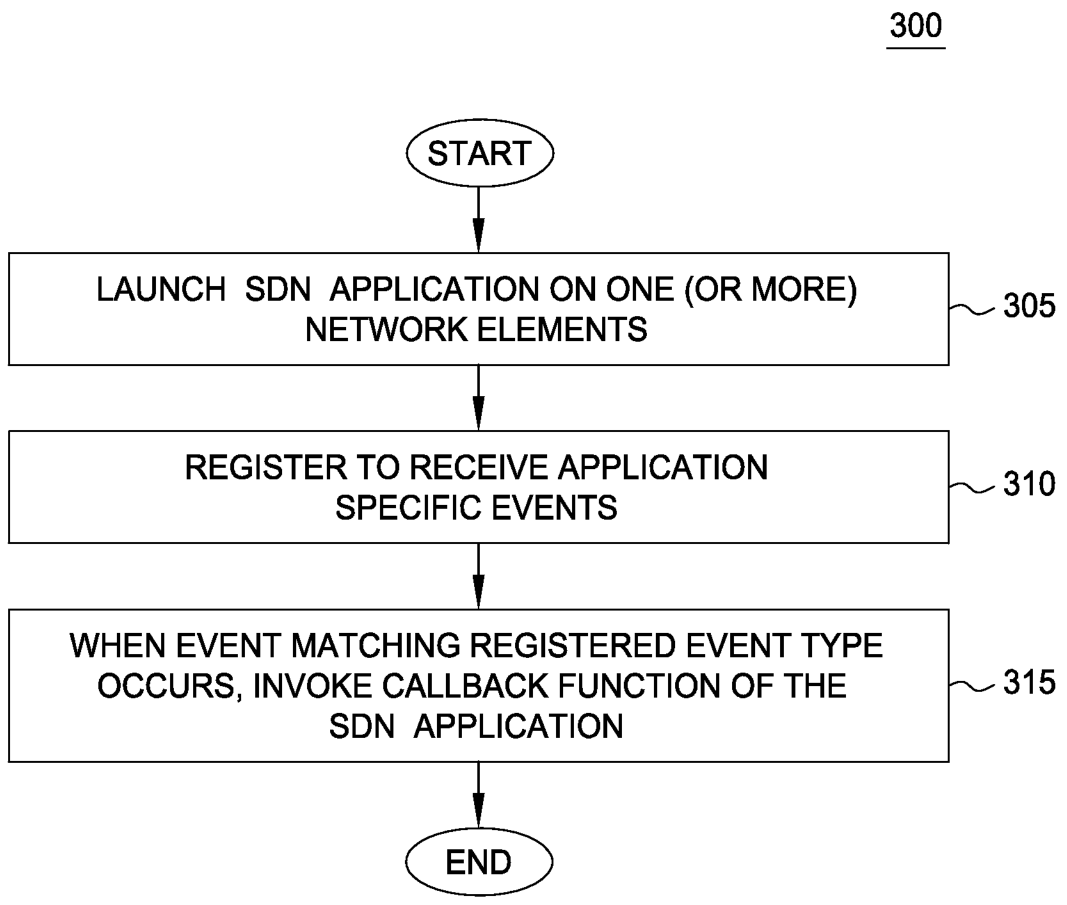

300
START
LAUNCH SDN APPLICATION ON ONE (OR MORE) NETWORK ELEMENTS
305
REGISTER TO RECEIVE APPLICATION SPECIFIC EVENTS
310
WHEN EVENT MATCHING REGISTERED EVENT TYPE OCCURS, INVOKE CALLBACK FUNCTION OF THE SDN APPLICATION
315
END

FIG. 3

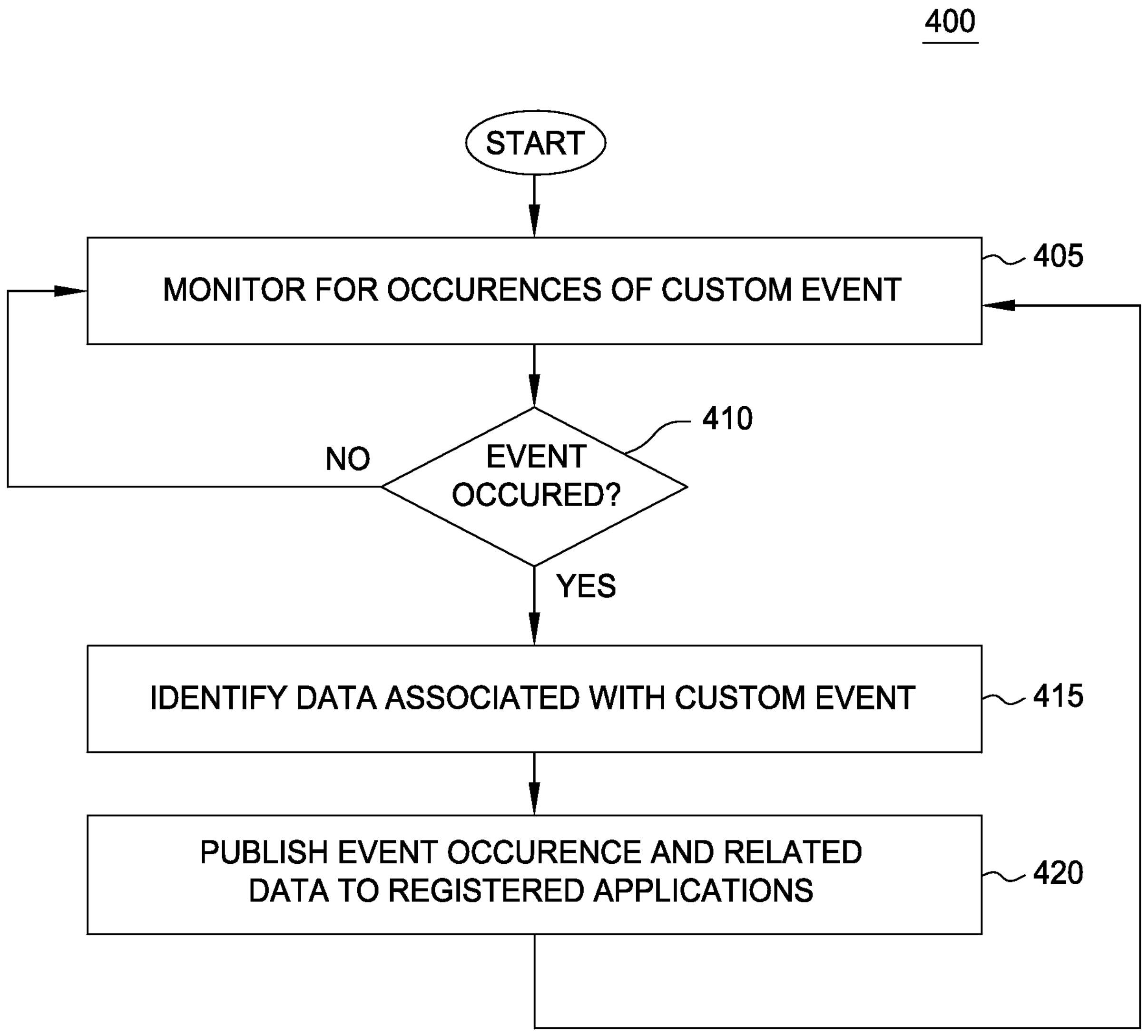

400
START
MONITOR FOR OCCURENCES OF CUSTOM EVENT
405
EVENT OCCURED?
410
NO
YES
IDENTIFY DATA ASSOCIATED WITH CUSTOM EVENT
415
PUBLISH EVENT OCCURENCE AND RELATED DATA TO REGISTERED APPLICATIONS
420

FIG. 4

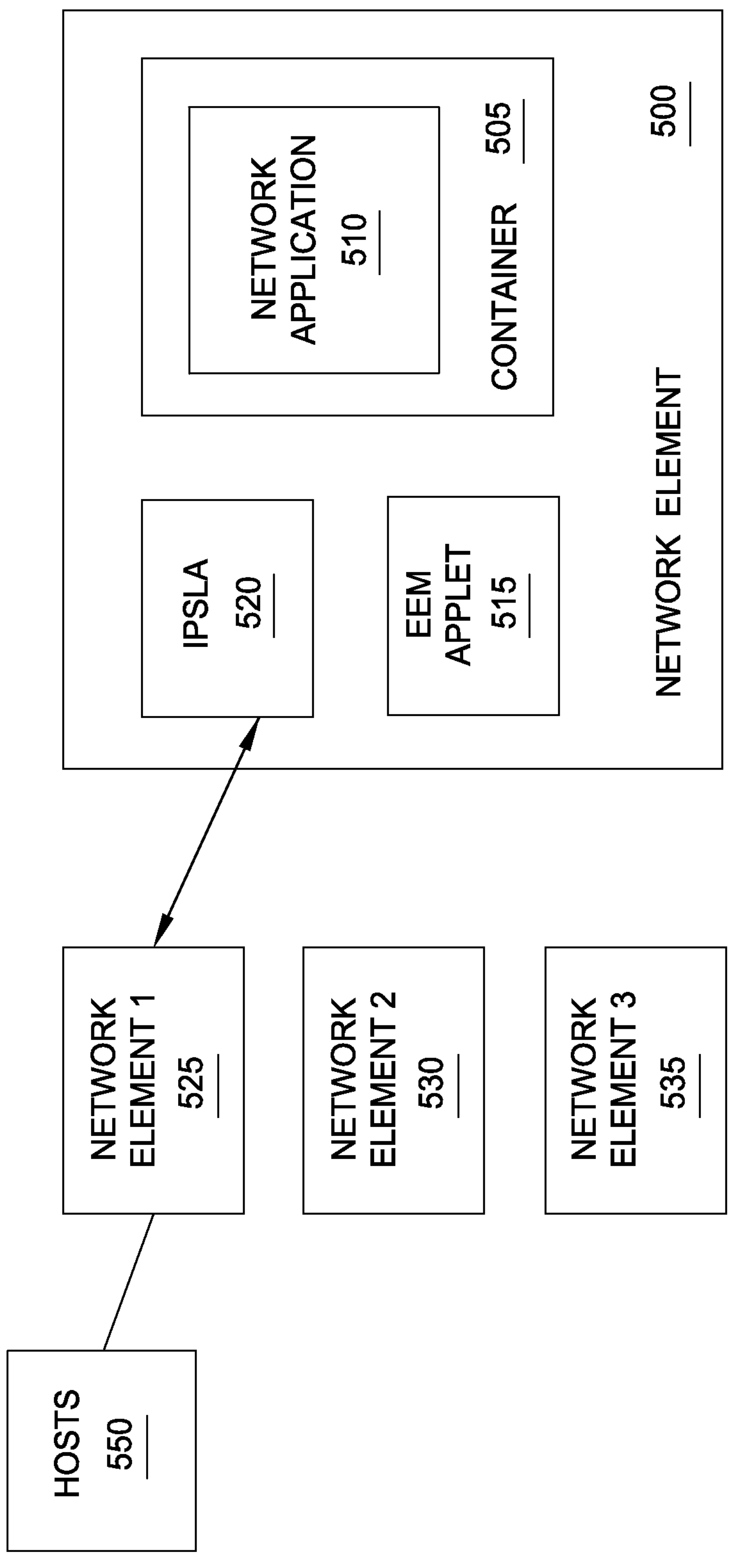

NETWORK APPLICATION 510
CONTAINER 505
500
IPSLA 520
EEM APPLET 515
NETWORK ELEMENT
NETWORK ELEMENT 1 525
NETWORK ELEMENT 2 530
NETWORK ELEMENT 3 535
HOSTS 550

FIG. 5

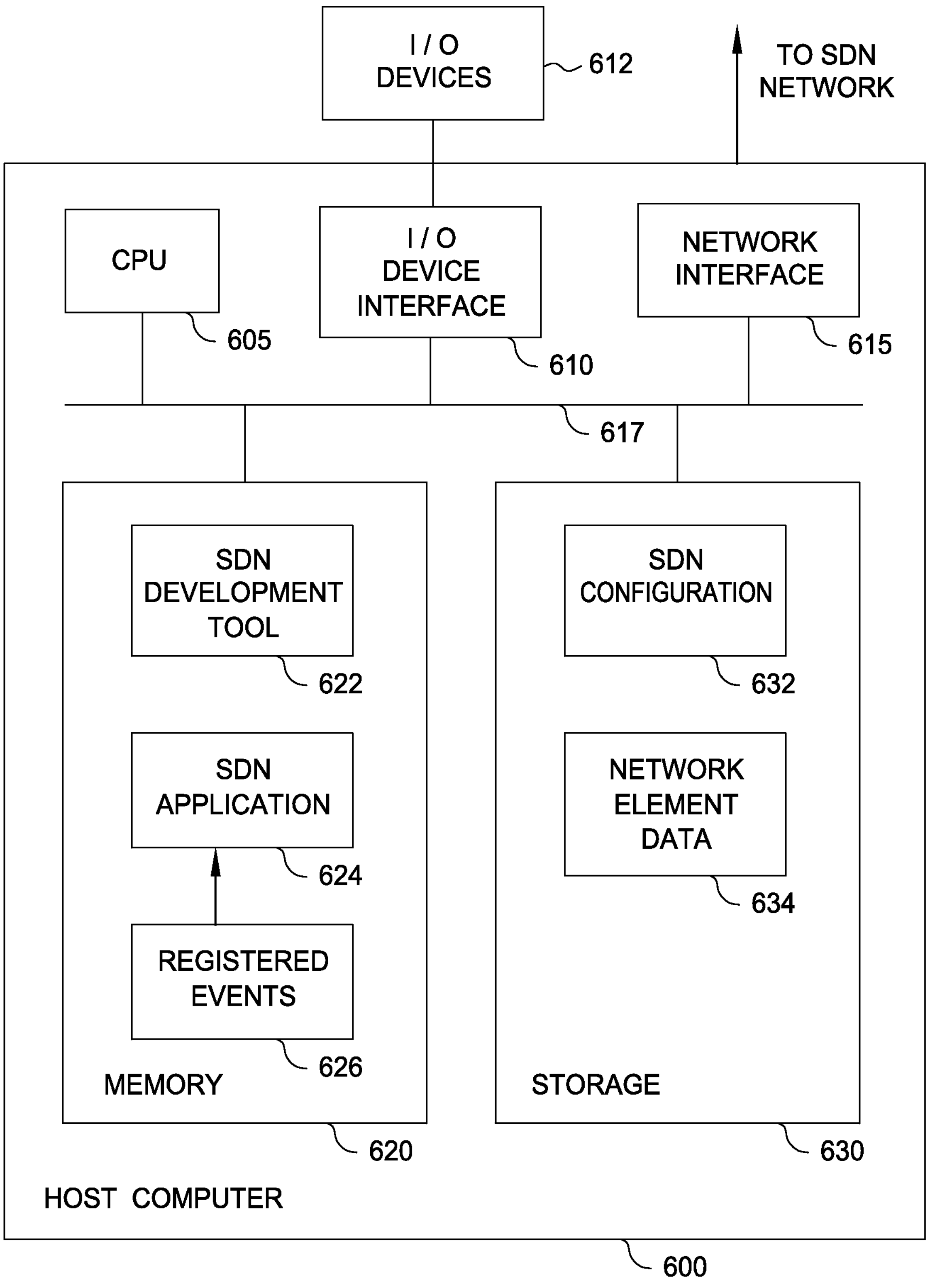

I / O DEVICES
612
TO SDN NETWORK
CPU
605
I / O DEVICE INTERFACE
610
NETWORK INTERFACE
615
617
SDN DEVELOPMENT TOOL
622
SDN APPLICATION
624
REGISTERED EVENTS
626
MEMORY
620
SDN CONFIGURATION
632
NETWORK ELEMENT DATA
634
STORAGE
630
HOST COMPUTER
600

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/022055

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04W4/00 H04W4/20
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04W

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

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

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2005/273787 A1 (KOVACHKA-DIMITROVA MONIKA M [BG] ET AL) 8 December 2005 (2005-12-08) abstract claims 1,7,9 figure 7 paragraph [0063] - paragraph [0068] -----	1-24
A	US 8 180 883 B1 (CLEMM LUDWIG ALEXANDER [US] ET AL) 15 May 2012 (2012-05-15) abstract claim 1 column 10, line 50 - column 11, line 4 figure 8 column 17, line 1 - line 14 -----	1-24
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

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

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

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search 27 May 2014	Date of mailing of the international search report 06/06/2014
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Authorized officer Bub, Armin

1

International application No
PCT/US2014/022055

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 7 656 818 B1 (BAROUDI MIHYAR [US] ET AL) 2 February 2010 (2010-02-02) abstract claim 1 figure 2 column 2, line 44 - column 4, line 28 figure 3 column 4, line 29 - column 5, line 17 -----	1-24
A	US 8 352 769 B1 (GHOSE TIRTHANKAR [US] ET AL) 8 January 2013 (2013-01-08) abstract column 13, line 52 - line 67 claim 1 -----	1-24
A	US 8 010 668 B1 (ROTHSTEIN JESSE A [US] ET AL) 30 August 2011 (2011-08-30) abstract claim 1 figure 4 figure 2 column 7, line 52 - column 8, line 21 -----	1-24

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2014/022055

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2005273787 A1	08-12-2005	NONE	
US 8180883 B1	15-05-2012	NONE	
US 7656818 B1	02-02-2010	NONE	
US 8352769 B1	08-01-2013	US 8352769 B1	08-01-2013
		US 2013158729 A1	20-06-2013
US 8010668 B1	30-08-2011	US 8010668 B1	30-08-2011
		US 8024483 B1	20-09-2011
		US 8326984 B1	04-12-2012
		US 8516113 B1	20-08-2013