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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 January 2017 (26.01.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/014769 A1

(51) International Patent Classification:
H04L 12/24 (2006.01) *H04L 12/801* (2013.01)
H04L 12/911 (2013.01)

(21) International Application Number:
PCT/US2015/041546

(22) International Filing Date:
22 July 2015 (22.07.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **LEE, Jeongkeun**; 1501 Page Mill Road, Palo Alto, California 94304-1100 (US). **TURNER, Yoshio**; 4394A 24th Street, San Francisco, California 94114 (US). **PRAKASH, Chaithan M.**; 450 Oak Grove Drive, Apt. 304, Santa Clara, California 95054 (US).

(74) Agents: **FERGUSON, Christopher Ward** 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).

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))*

WO 2017/014769 A1

(54) Title: PROVIDING A COMPOSITE NETWORK POLICY

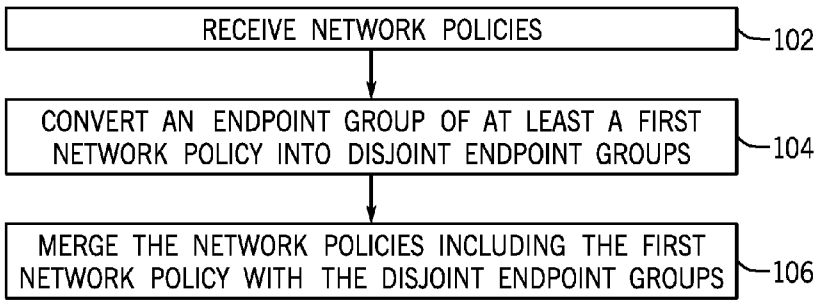

FIG. 1

(57) Abstract: In some examples, each of network policies specifies at least one characteristic of communications allowed between endpoint groups. An endpoint group of a first network policy is converted into disjoint endpoint groups. The network policies are merged to provide a composite network policy.

PROVIDING A COMPOSITE NETWORK POLICY

Background

[0001] A network can be used to communicate data between various endpoints. The network can include interconnecting devices (*e.g.* routers, switches, etc.) for forwarding data along respective paths between endpoints. In addition, various service functions can be implemented with service function boxes deployed in the network, where the service functions can be applied on data packets communicated along paths in the network.

Brief Description Of The Drawings

[0002] Some implementations of the present disclosure are described with respect to the following figures.

[0003] Fig. 1 is a flow diagram of an example process according to some implementations.

[0004] Figs. 2A-2D are graphs representing corresponding different example network policies that can be provided to govern communications in a network, according to some examples.

[0005] Fig. 3 is a flow diagram of an example normalization process according to some implementations.

[0006] Fig. 4 is a schematic diagram of a hierarchical arrangement of labels associated with endpoint groups, according to some implementations.

[0007] Figs. 5A and 5B are graphs showing an example of normalizing an input policy graph according to some implementations.

[0008] Fig. 6 is a graph showing an example edge formed from combining edges of input policy graphs, according to some implementations.

[0009] Figs. 7A and 7B are graphs illustrating an example of combining input policy graphs to form a composite policy graph, according to some implementations.

[0010] Figs. 8 and 9 are block diagrams example systems according to some implementations.

Detailed Description

[0011] Network policies can be provided for governing communications of data in a network. As used here, the term "network" can refer to an arrangement of devices and paths that allow for communications between endpoints. Examples of endpoints include a server, a virtual machine, a client device, a subnet, an end user, and so forth. In some cases, in examples where there are multiple networks that are coupled to each other, a network can also be considered an endpoint. More generally, an endpoint can be a smallest unit of abstraction for which a network policy is applied.

[0012] A network policy can specify at least one characteristic of communications allowed between endpoint groups (EPGs), where each endpoint group (EPG) includes one or multiple endpoints. Examples of characteristics that can be specified by a network policy include port numbers to use for communications between respective EPGs, one or multiple service functions to apply to data that is communicated between respective EPGs, and/or other characteristics. A port number can refer to a Transmission Control Protocol (TCP) port number. Stated differently, a network policy can specify a behavior of a portion of a network in processing data (*e.g.* data packets). The processing of data packets can include forwarding data packets, modifying data packets (such as by changing values of headers of the data packets, dropping the data packets, etc.), applying service functions, and/or other types of processing.

[0013] Examples of service functions, which can be implemented by service function boxes, include load balancing to balance data communication load across multiple devices, protection services (such as firewall protection, intrusion detection, network authorization or authentication, etc.), network address translation (to

translate an address of a data packet between a first address and a second address), and/or other service functions. A service function box can refer to a hardware device or a program (machine-readable or machine-executable instructions) configured to perform a respective service function.

[0014] In some example implementations, a service function box can represent an abstract function that takes a packet as input and returns a set of zero or more packets. In such implementations, a network programming language can be used to describe the function, behaviors, and properties of a service function box. In various implementations, a Pyretic network programming language can be used. Pyretic can use real IP/MAC addresses to implement network programs. Pyretic can be extended to write programs/policies regarding logical EPG parameters (*e.g.* 'web.ip' to indicate IP addresses of a Web EPG). Examples of functions that can be provided by Pyretic programs include a drop function (to drop packets), a forward function (to forward a packet), and so forth.

[0015] Different network policies can be provided by respective different policy writers. Examples of policy writers can include network administrators, service providers, network operators, application developers, tenants of a cloud infrastructure, and so forth. A cloud infrastructure can refer to an arrangement of resources (including processing resources, storage resources, and/or other resources) that are available over a network to devices of tenants (which are users that are able to selectively access the cloud resources). Network policies can also be provided by automated entities, such as control programs, applications, network services, and so forth. Thus, a policy writer can refer to any entity (a human, a machine, or a program) that is able to provide a network policy.

[0016] In some examples, network policies can be provided by multiple different policy writers in the context of Software Defined Networking (SDN). SDN can refer to a technique for implementing computer networking environments using software (or more generally, machine-readable or machine-executable instructions) to control the configuration and allocation of networking resources in the network. In such a network, the hardware resources (*e.g.* routers, switches, server, etc.) or virtual

network and compute resources (*e.g.* virtual layer 2/layer 3 (L2/L3) networks, virtual machines) can be programmed to allocate networking and computing resources according to the network policies of various policy writers.

[0017] Network policies can be expressed using any of various different programming languages. In some examples, as discussed in the present disclosure, network policies can be represented using graphs.

[0018] As the number of independent network policies provided by policy writers increase, the management of communications in a network can become more complex, due to possible conflicts between the network policies. Given a collection of network policies from respective policy writers, a composite network policy can be produced by merging the network policies. Such merging of the network policies is also referred to as a composition of the network policies. Communications in a network can then be controlled using one or multiple controllers (e.g. one or multiple SDN controllers) according to the composite network policy.

[0019] Merging network policies can involve combining the network policies while accounting for any conflicts between the network policies. To properly merge multiple network policies into a composite network policy (a process referred to as network policy composition), an understanding of the intents of respective policy writers in formulating respective network policies is first determined. Manually merging network policies (particularly a large number of network policies) can be time and labor intensive, and may result in incorrect composition of the network policies or production of a composite network policy that is inconsistent with an intent of a policy writer.

[0020] In accordance with some implementations of the present disclosure, techniques or mechanisms are provided to allow for automated composition of network policies to form a composite network policy. As shown in Fig. 1, a process (which can be performed by a network policy composer as discussed further below) receives (at 102) network policies, where each network policy of the network policies specifies at least one characteristic of communications allowed between EPGs, each

EPG of the EPGs including at least one endpoint. The process converts (at 104) an EPG of at least a first network policy of the network policies into disjoint EPGs. Disjoint EPGs are EPGs with mutually exclusive memberships—in other words, a given endpoint cannot be assigned to multiple disjoint EPGs such that the disjoint EPGs do not share any endpoints. As discussed further below, the converting of an EPG of a network policy into disjoint EPGs is part of a normalization process.

[0021] The process then merges (at 106) the network policies including the first network policy with the disjoint EPGs. The merging of the network policies provides a composite network policy. Note that each other network policy that is merged with the first network policy can also include disjoint EPGs formed by converting an EPG of the other network policy.

[0022] In some implementations, a network policy can be represented as a policy graph. A policy graph (or more simply "graph") can include vertices that represent respective EPGs, and an edge between the vertices represent allowed communications between the EPGs (or more specifically, communications between endpoints of the EPGs). An EPG can refer to a group of arbitrary addressable endpoints or a group of endpoints that can perform a common logical role or share a common property (also referred to as a "label"). An EPG includes endpoints that satisfy a membership predicate specified for the EPG. A membership predicate can be provided as a label (any endpoint with a given label is a member of a given EPG). In general, a membership predicate can be provided as a Boolean expression over labels—for example, if a Boolean expression containing at least one given label of an endpoint evaluates to true, then the endpoint is a member of a respective EPG. In further examples, a Boolean expression can include multiple labels that are subject to a Boolean operator, such as AND, OR, and so forth. An example of a Boolean expression that can be associated with a given EPG is LABEL1 AND LABEL2—if an endpoint has both LABEL1 and LABEL2, then the endpoint is a member of the given EPG.

[0023] Endpoints are addressable using Internet Protocol (IP) addresses, Media Access Control (MAC) addresses, virtual local area network (VLAN) identifiers, and/or other types of addresses.

[0024] Endpoint properties (labels) can be assigned and changed dynamically at runtime, to cause respective endpoints to change membership between different EPGs. In response to an endpoint changing membership from a first EPG to a second EPG, the network policy that can be applied on communications of the endpoint can change from a first network policy (associated with the first EPG) to a second network policy (associated with the second EPG). As a result, changing an endpoint property can cause different network policies to be dynamically assigned to an endpoint as the endpoint property changes over time.

[0025] Figs. 2A-2D illustrate examples of policy graphs (or more simply "graphs") that are used to represent respective example network policies. Fig. 2A is a graph representing a first example network policy provided by an administrator for departments of an enterprise. The graph of Fig. 2A includes an IT vertex that represents an IT department (first EPG) and an ENGG vertex that represents an engineering department (second EPG). An edge between the IT vertex and the ENGG vertex indicates that traffic is allowed from any endpoint of the IT department to any endpoint of the engineering department using specified protocol port numbers (22, 23, or 5900 in the example of Fig. 2A).

[0026] Fig. 2B is a graph representing a second example network policy provided by a web application administrator. The graph of Fig. 2B includes a Departments vertex (representing a first EPG including departments of an enterprise), a Web vertex (representing a second EPG including one or multiple Web applications), and a DB vertex (representing a third EPG including one or multiple databases). An edge between the Departments vertex and the Web vertex in the graph of Fig. 2B specifies that traffic is allowed from any department to access a Web application using port 80 in the example, and also specifies that the traffic is to be load balanced using a load balancer (LB) service function box. An edge between the Web vertex and the DB vertex specifies that traffic is allowed from a Web application to a

database tier, using port 3306 in the example. The graph of Fig. 2B also shows an edge from the DB vertex to itself, which allows a database within the database tier to communicate with another database using port 7000 in the example.

[0027] Fig. 2C is a graph representing a third example network policy provided by an SDN application for domain name system (DNS)-based security protection. The graph of Fig. 2C includes a first graph model 202 having an NML vertex (representing an EPG including endpoints having a "normal" security status) connected over an edge having a deep packet inspection (DPI) service function box to a DNS vertex (an EPG including one or multiple DNS servers). The first graph model 202 specifies that traffic from the NML EPG to the DNS EPG is allowed if the traffic uses port 53, and further specifies that DPI is to be applied on the traffic.

[0028] The graph of Fig. 2C further includes a second graph model 204 having a QN vertex (representing an EPG including endpoints that have a "quarantined" status) connected over an edge to an RMD vertex (representing an EPG that includes one or multiple security remediation servers). The "*" indication on the edge in the second graph model 204 indicates that the traffic from the QN EPG to the RMD EPG is allowed for any port number. The network policy represented by the graph of Fig. 2C specifies that DNS traffic from network endpoints with the "normal" security status is be inspected by a DPI service function box when DNS lookups of DNS server(s) are performed. The network policy represented by the graph of Fig. 2C also specifies that network endpoints that have the "quarantined" status can only send their traffic (of any type) to a security remediation server in the RMD EPG.

[0029] Fig. 2D is a graph representing a fourth example network policy provided by a data center administrator. The graph of Fig. 2D includes a first graph model 206 and a second graph model 208. The first graph model 206 specifies that traffic coming into a data center (represented by the DC vertex) from the Internet (represented by the Internet vertex) can use any port number (indicated by the "*") and is to pass through a firewall (FW) service function box (that provides firewall protection) and a byte counter (BC) service function box (that counts a number of bytes of data). In addition, the first graph model 206 includes an edge, including a

byte counter (BC) service function box, from the DC vertex to itself, which specifies that traffic within the data center also traverses the BC service function box.

[0030] The second graph model 208 allows communication of traffic (on port 9099 in the example) between endpoints in the data center.

[0031] Although example policy graphs representing respective example network policies are depicted in Figs. 2A-2D, it is noted that there can be other network policies represented by other policy graphs.

[0032] Each of the example network policies shown in Figs. 2A-2D specify access control whitelisting (ACL), which grants specific entities access rights to other entities if a specified condition is satisfied. An edge of each policy graph in Figs. 2A-2D can thus be referred to as an access control whitelisting edge, which provides an access control whitelisting rule. In addition, Figs. 2B-2D represent network policies that specify service function chaining, in which one or multiple service functions are included in an edge to apply to data.

[0033] As noted further above, endpoints can be assigned labels dynamically at runtime, causing the endpoints to move from one EPG to another EPG. For example, a server that was assigned the label NML ("normal" status) can subsequently be relabeled QN ("quarantined" status) when a network monitor detects the server issuing a DNS query for a known malicious Internet domain.

[0034] Thus, a policy graph (such as any of those depicted in Figs. 2A-2D) can represent a set of one or multiple network policies that are applied dynamically to each endpoint according to the endpoint's status changes over time. Moreover, note that the composition of network policies represented by graphs into a composite network policy is performed only in response to changes in network policies, such as when a network policy is added, modified, or removed. The composition of network policies does not have to be performed in response to a change in membership of an endpoint from one EPG to another EPG. Instead, a runtime system only has to perform a relatively lightweight operation of looking up and applying the respective

network policies for each endpoint depending on the endpoint's current EPG membership.

[0035] Each of the graphs shown in Figs. 2A-2D includes a directed edge that specifies allowed communication from any endpoint in a source EPG to any endpoint in a destination EPG. Each edge can be associated with a classifier, which matches packet header fields of a data packet to determine the respective network policy (*e.g.* an access control whitelisting rule) is to be applied. For example, in Fig. 2A, the classifier associated with the edge between the IT vertex and the ENGG vertex determines if values of the packet header fields of a packet indicate that a source of the packet is an endpoint in the IT department, a destination of the packet is an endpoint in the engineering department, and a destination port number of 22, 23, or 5900 is used. Stated differently, the classifier compares the values of the packet header fields (*e.g.* source address field, destination address field, destination port number field) of the packet to corresponding values (*e.g.* source address value, destination address value, destination port number value) of the respective network policy to determine if a match condition of the edge is satisfied. If the match condition of the edge is satisfied as determined by the classifier, then communication of the packet from the IT department endpoint to the engineering department endpoint is allowed.

[0036] Although Figs. 2A-2D depict a single edge between respective pairs of EPG vertices, it is noted that there can be multiple directed edges from a first EPG vertex to a second EPG vertex, where each edge is associated with a respective different classifier.

[0037] In accordance with some implementations, prior to merging (composition) of network policies to form a composite network policy, a normalization process is first performed. Fig. 3 is a flow diagram of an example normalization process. The normalization process converts (at 302) EPGs of an input policy graph (representing a respective network policy) into disjoint EPGs. Task 302 corresponds to task 104 in Fig. 1. The converting of input EPGs of an input policy graph into disjoint EPGs produces a normalized policy graph. A reason for separating input EPGs of input

network policies (prior to performing composition of the input network policies) into equivalent sets of disjoint EPGs is that the input EPGs can have overlapping EPG membership (specified as arbitrary Boolean expressions over the label space), such that an endpoint may be part of multiple EPGs. The normalization process removes overlapping EPG memberships.

[0038] The normalization process can further replicate and/or merge (at 304) a composition constraint of an input policy graph in a normalized input graph. The merging of composition constraints forms a merged composition constraint. In addition, the normalization process can further replicate and/or merge (at 306) an edge of an input policy graph in the normalized graph, subject to replicated and/or merged composition constraint(s).

[0039] Composition constraints specified in network policies can represent intents of policy writers with respect to communications allowed by the corresponding network policies. A number of different composition constraints can be specified, and these composition constraints can be used in identifying and resolving conflicts between network policies when performing network policy composition. The composition constraints specified by network policies can govern what policy changes are allowed when the network policies are composed. In some implementations, composition constraints can be specified for any pair of EPGs in a network policy.

[0040] In some examples of the present disclosure, the composition constraints can be represented using different types of edges in policy graphs that represent the corresponding network policies.

[0041] The following lists examples of some composition constraints:

- A composition constraint that specifies that communications between respective EPGs must be allowed.
- A composition constraint specifying that communications between respective EPGs can be allowed.

- A composition constraint specifying that communications between respective EPGs are to be blocked.

- A composition constraint included in a first network policy and specifying at least one service function to be conditionally applied to communications between respective EPGs, if and only if another network policy specifies that the communications between the respective EPGs are allowed.

[0042] Table 1 below provides example specific composition constraints from a source EPG to a destination EPG.

Table 1

	Classifier		Service Function Box	
Match	Add	Remove	Drop	Modify
Port 80	Y	N	N	DSCP = 16, 18, 20
Port 88	N			

[0043] The table indicates that port 80 traffic cannot be disallowed through composition with other graphs (and example of a composition constraint that specifies that communications between respective EPGs must be allowed), and function boxes that are added cannot drop packets but are allowed to modify the DSCP packet field to a set of specific values (16, 18, 20). The table also shows that port 88 cannot be allowed through composition, but otherwise all other traffic can be allowed, with no restriction on function boxes.

[0044] In some implementations, converting input EPGs into equivalent disjoint EPGs can involve first producing locally disjoint EPGs from the input EPGs, followed by producing globally disjoint EPGs from the locally disjoint EPGs. Locally disjoint

EPGs are EPGs that are disjoint within a given input policy graph that represents a respective network policy. Globally disjoint EPGs are disjoint across multiple input policy graphs. Converting input EPGs into locally disjoint EPGs can employ a label hierarchy (which includes a hierarchical arrangement of labels). Producing globally disjoint EPGs uses label mapping (discussed further below).

[0045] An example label hierarchy is shown in Fig. 4, which depicts two sets of labels arranged in respective hierarchical trees (or more generally hierarchical structures) 400 and 410. The set of labels depicted in the tree 400 include labels relating to locations of endpoints. The labels at the leaves of the tree 400, e.g. ZONE_A, ZONE_B, ZONE_C, COMPUTER_ROOM, and STORAGE_ROOM, are basic elements that represent specific locations. A non-leaf label is a composite label that represents a logical disjunction (Boolean OR) of the descendant leaf labels, e.g. CAMPUS = ZONE_A OR ZONE_B OR ZONE_C.

[0046] The set of labels of the tree 410 include labels relating to status of endpoints. The leaves of the tree 410 are NML (representing a normal status of an endpoint) and QN (representing a quarantined status of an endpoint).

[0047] A label hierarchy can be used to determine labels that are mutually exclusive; mutually exclusive labels cannot simultaneously be assigned to a given endpoint. For example, the ZONE_A label, ZONE_B label, and ZONE_C label of Fig. 4A cannot all be assigned to the given endpoint, since the given endpoint cannot be at the three different locations at the same time. Similarly, the NML label and the QN label of Fig. 4B cannot both be assigned to the given endpoint, since the given endpoint cannot have both a normal status and quarantined status at the same time.

[0048] Specifically, in any single tree of the label hierarchy, any set of labels that do not have an ancestor relationship are mutually exclusive, e.g. {ZONE_A, ZONE_B, ZONE_C, COMPUTER_ROOM, STORAGE_ROOM}, {NML, QN}.

[0049] An input EPG in an input policy graph is split into locally disjoint EPGs by algebraically rewriting the input EPG's membership predicate expression into an

equivalent positive disjunctive normal form, where each term in the resulting expression describes a locally disjoint EPG. More specifically, each composite label (e.g. CAMPUS or DC in the tree 400, DNSP in the tree 410) in the expression is replaced with its leaf label equivalents. For example, CAMPUS in the expression can be replaced with ZONE_A OR ZONE_B OR ZONE_C. As a more specific example, assume that the membership predicate expression for the input EPG is CAMPUS & NML (in other words, an endpoint whose location is on campus and which has a normal status is a member of the input EPG). This expression is rewritten into the equivalent positive disjunctive normal form as follows: ZONE_A & NML OR ZONE_B & NML OR ZONE_C & NML.

[0050] In addition, any negated label is replaced with the disjunction of all the other sibling leaf labels in the hierarchy (since they are mutually exclusive). For example, an expression ~ZONE_A & NML (where ~ represents a negation) can be replaced with (ZONE_B OR ZONE_C) & NML.

[0051] Once the foregoing replacement of composite labels and negated labels are performed to produce a resultant expression that uses leaf labels (but not composite labels) in positive form, the resultant expression is converted into disjunctive normal form, i.e. ORing a collection of terms where each term is the AND of leaf labels. In the example above where ~ZONE_A & NML is replaced with (ZONE_B OR ZONE_C) & NML, the conversion to the disjunctive normal form produces the following output expression: ZONE_B & NML OR ZONE_C & NML, where ZONE_B & NML is a first conjunctive term, and ZONE_C & NML is a second conjunctive term.

[0052] Any conjunctive term in the output expression that has mutually exclusive labels results in an empty set, and so such a conjunctive term can be deleted from the output expression. Each remaining conjunctive term in the output expression defines an EPG that is disjoint within the same input policy graph.

[0053] To obtain globally disjoint EPGs, each locally disjoint EPG may further be divided. For each conjunctive term, the normalization process checks a label

mapping to identify all other potentially related labels. The related labels can be determined using a label mapping.

[0054] The label mapping is a symmetric relation F: L → L, where L is the set of all leaf labels. If labels (x, y) ∈ F, then a single endpoint can be assigned both labels x and y. An example label mapping is provided as follows:

{IT:ZONE_A, ENGG: ZONE_B, …}

[0055] According to the foregoing example label mapping, an IT endpoint can also be a ZONE_A endpoint (i.e. the endpoint can be assigned both the IT label and the ZONE_A label). Also, according to the foregoing example label mapping, an ENGG endpoint can also be a ZONE_B endpoint.

[0056] According to the foregoing example label mapping, IT maps to ZONE_A, and ENGG maps to ZONE_B.

[0057] Assume that a conjunctive term of an output expression (formed after splitting input EPGs to locally disjoint EPGs) is label1 & label2. The normalization process checks the label mapping that starts from each of these labels (label1 and label2), and adds related labels to the conjunctive term. As an example, the label mapping can be as follows:

{label1: label3, label2: label4, label2: label5}.

[0058] In the foregoing example, label1 maps to label3 (i.e. label3 is related to label1 in the label mapping), and label2 maps to both label4 and label5 (i.e. label4 and label5 are related to label2 in the label mapping). Based on the foregoing, label1 & label2 can be rewritten as (label1 & label3) & (label2 & label4 & label5).

[0059] In some cases, identified related labels may be mutually exclusive, e.g., NML and QN in the tree 410 of Fig. 4. For example, if the related labels label4 and label5 are mutually exclusive, then two terms can be formed from the original conjunctive term: 1) label1 & label2 & label3 & label4, and 2) label1 & label2 & label3

& label5. Splitting of the terms can continue until no term has mutually exclusive labels. This final set of terms corresponds to globally disjoint EPGs.

[0060] Once the normalized EPGs are generated, task 304 in Fig. 3 replicates and/or merges composition constraints from the input policy graphs. An example of composition constraint replication is described in connection with the example of Figs. 5A and 5B. Fig. 5A shows an input policy graph that includes EPG S and EPG D, with an edge associated with constraint C1 from EPG S to EPG D.

[0061] Assuming that EPG S has been split into globally disjoint EPGs S1 and S2, as shown in Fig. 5B, then the composition constraint C1 is replicated for the edge between EPG S1 and EPG D, and the edge between EPG S2 and EPG D.

[0062] As a further example, suppose that EPG S in the input policy graph is translated to normalized EPGs S1, S2, ..., Sm, and EPG D is normalized to EPGs D1, D2, ..., Dn, then composition constraint C1 is replicated in the normalized policy graph for ordered EPG pairs (Si, Dj), $\forall i$=1…m, $\forall j$=1…n.

[0063] In some cases, composition constraints may be merged to form a merged composition constraint. If the input policy graph has EPGs (G, H), and these EPGs overlap with EPGs (S, D) (in other words, EPG G and/or H share endpoints with EPG S and/or D), then a constraint C1 specified for the pair of EPGs (G, H) may be merged with a constraint C2 for the pair of EPGs (S, D). Merging composition constraints entails adopting a union of restrictive invariants of the overlapping constraints.

[0064] As an example, assume constraint C1 is as follows: MATCH(dstport=80) -> FUNCTION BOX MODIFY:Yes; and constraint C2 is as follows: MATCH(protocol=1) -> FUNCTION BOX MODIFY:DSCP=12.

[0065] Constraint C1 specifies that if a packet has a destination port value of 80, the packet is subjected to modification performed by the FUNCTION BOX MODIFY. Constraint C2 specifies that if a packet has a protocol value of 1, the FUNCTION BOX MODIFY is applied to modify the DSCP field of a packet to the value 12.

[0066] The merging of constraints C1 and C2 above is performed by performing a union of restrictive invariants to produce: MATCH(dstport=80 and protocol=1) -> FUNCTION BOX MODIFY:DSCP=12. Note that FUNCTION BOX MODIFY:DSCP=12 is more restrictive than FUNCTION BOX MODIFY:Yes.

[0067] Conflicts may be detected if an invariant from one composition constraint, e.g., never allow port 80 traffic, is opposed by an invariant from another composition constraint, e.g., never deny port 80 traffic. Such conflicts are error conditions that can be flagged to operators.

[0068] Task 306 of Fig. 3 replicates and/or merges edges of the input policy graph. In the example of Fig. 5A-5B, the edge between EPG S and EPG D of the input policy graph depicted in Fig. 5A is replicated in the normalized policy graph depicted in Fig. 5B due to the splitting of EPG S into EPG S1 and EPG S2. More generally, if EPG S has an edge to EPG D in the input policy graph, then the edge is replicated to the normalized graph's EPGs (Si, Dj), $\forall i$ = 1…m, $\forall j$ = 1...n.

[0069] Also, if the input graph has edges between EPGs (G, H), and these EPGs overlap with EPGs (S, D), any edges for the pair of EPGs (G, H) are merged with edges for the pair of EPGs (S, D) in the normalized EPG pairs that constitute the overlap. This merging is governed by the (merged) composition constraints. As an example, assume input policy graphs include those depicted in Figs. 2B and 2D. The input policy graph of Fig. 2B has an edge DB->DB, and the input policy graph of Fig. 2D has an edge DC->DC and an edge DC->DC that goes through a the BC service function box. In some examples, these edges can be merged as shown in Fig. 6, where DD = DB & DC.

[0070] Once normalization is performed, merging of input network policies can be performed (task 106 in Fig. 1). The merging of input network policies is a composition process that involves a union of normalized policy graphs corresponding to the input network policies.

[0071] In normalized form, EPGs in two different policy graphs are either disjoint or equal. Partial overlap is not possible since the normalization process has converted EPGs into globally disjoint EPGs. This property enables multiple normalized graphs to be composed together using a simple union operation. The composition forms a composite policy graph that includes a union of all the EPGs of the normalized policy graphs. The union operation also copies and merges composition constraints and directed edges from the individual normalized policy graphs to the composite policy graph. Composition constraint merging is performed in similar fashion as described for the normalization process.

[0072] As edges from the individual graphs are added to the composite policy graph, the edges are checked against the composition constraints for the source and destination EPGs. This check determines whether an edge's classifier satisfies the composition constraints or the edges are to be narrowed to be in compliance with the composition constraints. If a new edge satisfies the composition constraints with a non-null surviving classifier, then the new edge can be added or merged with existing edges from source to destination EPGs.

[0073] As noted above, one composition constraint (referred to as a "must composition constraint") specifies that communications between the source EPG (S) and the destination EPG (D) must be allowed. Another composition constraint (referred to as a "can composition constraint") specifies that communications between the source EPG and the destination EPG can be allowed. Yet another composition constraint (referred to as a "block composition constraint") specifies that communications between the source EPG and the destination EPG are to be blocked. A further composition constraint (referred to as a "conditional composition constraint") specifies at least one service function to be conditionally applied to communications between the source EPG and the destination EPG, if and only if another network policy specifies that the communications between the source EPG and the destination EPG are allowed

[0074] In some implementations, a must composition constraint or a can composition constraint overrides a conditional composition constraint, while a block

composition constraint overrides a can composition constraint. The must composition constraint or can composition constraint of a first network policy overriding the conditional composition constraint of a second network policy can refer to allowing the communications between the source EPG and the destination EPG, subject to application of the service function chain (including one or multiple service function boxes) of the conditional composition constraint of the second network policy. The block composition constraint overriding the can composition constraint can refer to blocking communications between the source EPG and the destination EPG according to a first network policy, even though a second network policy allows the communications between the source EPG and the destination EPG.

[0075] A conflict between a must composition constraint in a first network policy and a block composition constraint in a second network policy can be resolved based on ranks assigned to the first and second network policies or ranks assigned to the policy writers of the first and second network policies. For example, if the first network policy is ranked higher than the second network policy, the must composition constraint of the first network policy overrides the block composition constraint of the second network policy, such that communications between the source EPG and the destination EPG are allowed pursuant to the first network policy, even though the second network policy specifies that such communications are to be blocked. In the foregoing example, the second network policy is considered to be a dropped network policy, since the second network policy has been disregarded. A dropped network policy can be reported to a target entity, such as a policy writer or some other entity.

[0076] In other cases, if the ranks of the first and second network policies are the same, then the conflict between the first and second network policies remains unresolved. In such case, the unresolved conflict can be reported to a target entity, such as a policy writer or other entity for resolution, revision, and possible re-submission.

[0077] When adding a new edge (an edge of an individual normalized policy graph) to a composite policy graph, the composition process first determines if a

classifier flow space of the edge intersects with flow spaces of existing classifiers in the composite policy graph. As noted above, each edge can be associated with a classifier, which matches packet header fields of a data packet to determine the respective network policy (*e.g.* an access control whitelisting rule) is to be applied. A classifier flow space of a given classifier can refer to a portion of the packets that are matched by the given classifier. If the classifier flow space associated with the new edge does not intersect (i.e. does not overlap) with the classifier flow space of existing edges in the composite policy graph, then the new edge and its service function boxes can be added directly to the composite policy graph, subject to service function box composition constraints (or more simply "service chain constraints").

[0078] A first type service chain constraint can set restrictions on the behavior of service function boxes that are added to a resultant service function chain produced from combining service function chains of input the policy graphs. For example, a first type service chain constraint can set a restriction on packet header field modifications and packet drop operations that respective service function boxes can perform on packets. Composition analysis performed can check whether adding a specific service function box to a given service chain would violate first type service chain constraints given by input policy graphs that are being composed together.

[0079] Second type service chain constraints can specify restrictions on a change characteristic of a given service function box that is already present on the edge from the source EPG to the destination EPG. A change characteristic of a service function box indicates whether or not the service function box can be changed (*e.g.* dropped or modified) in a certain way.

[0080] Other types of service chain constraints can be specified.

[0081] If the classifier flow space of the new edge intersects with the classifier flow space of at least one existing edge in the composite policy graph, then the composition process merges the new edge with the at least one existing edge. Assuming that the classifier flow space of the new edge intersects with classifier flow

spaces of multiple existing edges in the composite policy graph, the composition process breaks down the intersecting space into respective subspaces, where each of the subspaces correspond to the overlap of the classifier flow space of the new edge with the classifier flow space of a respective existing classifier. By identifying the subspaces, the composition process can merge, for each subspace, the new edge with the respective existing edge.

[0082] For each pair of edges to be merged, if either edge has a service chain (of one or multiple service function boxes) for the intersecting subspace, then service function composition can be performed that combines service function boxes from the service chains of the edges. When combining service function boxes, the composition process provides a proper ordering of the service function boxes in the composed chain. In some examples, the proper order of the service function boxes can be determined by detecting dependencies between the service function boxes based on analysis of the boxes' packet processing functions. Detected dependencies are used to determine valid orderings

[0083] In further examples, the following invariants and rules can also be considered when combining service function boxes.

1. In some further implementations of the present disclosure, an atomic sub-chain can be specified on the edge from the source EPG to the destination EPG. An atomic sub-chain includes at least two service function boxes, and does not allow for the insertion of another service function in the atomic sub-chain. The service function boxes of the atomic sub-chain can share a common second type constraint(s); in other words, the second type constraint(s) is (are) associated with the atomic sub-chain at the granularity of the atomic sub-chain, rather than individually with the service function boxes in the atomic sub-chain.

2. Atomic sub-chains that include service function boxes violating composition constraints can be dropped as policy graphs are composed together.

3. Null input space of any function box in a composed chain is avoided. A service function box can be composed of a set of match-actions. If any service function box has a null input space that the box is matching against, this means that the service function box is not functioning and becomes useless.

4. Generation of null output space from a composed chain is also avoided. For example, in a chain B1>>B2 (where B1 and B2 are service function boxes), if B2 is dropping all the flow packets outputted by B1, the chain will create a null output space, and no packet will pass through the chain.

[0084] Fig. 7A depicts two example policy graphs P1 and P2 (representing respective network policies) that are to be combined by a policy composer according to some implementations. The policy graph P1 has a graph model 702 specifying that endpoints in a marketing EPG are allowed to access a customer relationship management (CRM) EPG (including one or multiple CRM servers). The edge between the marketing vertex and the CRM vertex specifies that port 7000 is to be used, and that a load balancing (LB) service function box is to be applied on the traffic between the marketing EPG and the CRM EPG.

[0085] The policy graph P1 also includes another graph model 704 including an edge associated with a block composition constraint between a non-marketing EPG and the CRM EPG. The block composition constraint specifies that traffic of endpoints in the non-marketing EPG (endpoints that are not in the marketing EPG) to the CRM EPG is blocked.

[0086] The policy graph P2 specifies that endpoints of an employees EPG can access endpoints of a servers EPG using ports 80, 334, and 7000, and that the traffic passes through a firewall (FW) service function. Note that endpoints of the marketing EPG are a subset of the employees EPG, and the endpoints of the CRM EPG are a subset of the servers EPG. Note also that the port range (port 7000) of the policy graph P1 is a subset of the port range (ports 80, 334, 7000) of the policy

graph P2. As a result, the EPGs and port range of the policy graph P1 are completely encompassed by the EPGs and the port range in the policy graph P2

[0087] Since the EPGs and port range of the policy graph P1 are completely encompassed by the EPGs and the port range in the policy graph P2, one may naively compose the access control whitelisting rules of the policy graphs P1 and P2 by prioritizing P1 over P2, but this would incorrectly allow traffic of non-marketing EPG endpoints to reach endpoints of the CRM EPG. In addition, it can be assumed that the intended order of the service function chain is FW followed by LB, so that the graph composition would have to consider this intended order.

[0088] By using the graph model 704 in the policy graph P1, the intent of the policy writer of the policy graph P1 that traffic of endpoints of non-marketing employees to CRM servers are to be blocked can be captured and considered by the policy composer. Note that the access control whitelisting rules of the policy graphs P1 and P2 conflict since P1 blocks non-marketing employees' traffic to CRM servers, while P2 allows the traffic from all employees (including non-marketing employees) to all servers (including CRM servers). By including the composition constraint represented by the graph model 704 in the policy graph P1, the conflict can be resolved by overriding P2's policy to allow non-marketing employees to access CRM servers with the composition constraint in the policy graph P1 that blocks traffic of non-marketing employees to the CRM servers.

[0089] An example composite policy graph based on combining the policy graphs P1 and P2 is shown in Fig. 7B. In the composite policy graph of Fig. 7B, the {Employees – Marketing} vertex represents an EPG made up of non-marketing employees, and the {Servers – CRM} vertex represents an EPG made up of non-CRM servers. Also, in the composite policy graph of Fig. 7B, the order of the FW-LB chain between the marketing EPG and the CRM EPG complies with the intended order of the FW and LB service functions.

[0090] In combining the service function chain (including FW) of the policy graph P2 with the service function chain (including LB) of the policy graph P1, to provide

FW-LB chain between the marketing EPG and the CRM EPG of the composite policy graph of Fig. 7B, the policy composer can determine the proper order of the service function boxes by detecting dependencies between the service function boxes based on analysis of the boxes' packet processing functions. Detected dependencies are used to determine valid orderings.

[0091] Also, in forming the service function chain in the composite policy graph produced by the policy composer, the policy composer also considers any service chain constraints as discussed above, wherein each service chain constraint can set restrictions on the behavior of service function boxes that are added in the composite policy graph.

[0092] Fig. 8 is a block diagram of a system 800 according to some implementations. The system 800 can include a computer or an arrangement of multiple computers. The system 800 includes a processor (or multiple processors) 802, which can execute machine-readable instructions. A processor can include a microprocessor, a microcontroller, a physical processor module or subsystem, a programmable integrated circuit, a programmable gate array, or another physical control or computing device.

[0093] The processor(s) 802 can execute machine-readable instructions of a policy composer 804, which when executed perform various tasks as discussed above. The policy composer 804 includes graph receiving instructions 806 to receive input policy graphs representing respective network policies. The policy composer 804 also includes normalization instructions 808 to perform a normalization process as discussed above, which can normalize a given endpoint group represented by a first policy graph into disjoint EPGs that have mutually exclusive memberships, as to form a normalized policy graph. The policy composer 804 also includes composition instructions 810 to perform a composition process as discussed above, which combines the normalized policy graph with another policy graph (or other policy graphs) to form a composite policy graph representing a composite network policy.

[0094] Fig. 9 is a block diagram of a system 900 according to some implementations. The system 900 includes a non-transitory machine-readable or computer-readable storage medium (or storage media) 902. The storage medium (or storage media) 902 can store machine-readable instructions that are executable on one or multiple processors. The machine-readable instructions include normalization instructions 904 to normalize an EPG of a network policy, to produce disjoint EPGs based on the normalized EPG, the disjoint EPGs having mutually exclusive memberships of endpoints. The machine-readable instructions further include merge instructions 906 to merge the network policy including the disjoint EPGs with another network policy, the merging of the network policies providing a composite network policy.

[0095] The storage medium (or storage media) 902 can include one or multiple different forms of memory including semiconductor memory devices such as dynamic or static random access memories (DRAMs or SRAMs), erasable and programmable read-only memories (EPROMs), electrically erasable and programmable read-only memories (EEPROMs) and flash memories; magnetic disks such as fixed, floppy and removable disks; other magnetic media including tape; optical media such as compact disks (CDs) or digital video disks (DVDs); or other types of storage devices. Note that the instructions discussed above can be provided on one computer-readable or machine-readable storage medium, or alternatively, can be provided on multiple computer-readable or machine-readable storage media distributed in a large system having possibly plural nodes. Such computer-readable or machine-readable storage medium or media is (are) considered to be part of an article (or article of manufacture). An article or article of manufacture can refer to any manufactured single component or multiple components. The storage medium or media can be located either in the machine running the machine-readable instructions, or located at a remote site from which machine-readable instructions can be downloaded over a network for execution.

[0096] In the foregoing description, numerous details are set forth to provide an understanding of the subject disclosed herein. However, implementations may be

practiced without some of these details. Other implementations may include modifications and variations from the details discussed above. It is intended that the appended claims cover such modifications and variations.

What is claimed is:

1. A method comprising:

receiving, by a system comprising a processor, network policies, each network policy of the network policies specifying at least one characteristic of communications allowed between endpoint groups;

converting, by the system, an endpoint group of a first network policy of the network policies into first disjoint endpoint groups; and

merging, by the system, the network policies including the first network policy with the first disjoint endpoint groups, the merging of the network policies providing a composite network policy.

2. The method of claim 1, further comprising converting, by the system, an endpoint group of a second network policy of the network policies into second disjoint endpoint groups,

wherein the merging comprises merging the network policies including the first network policy with the first disjoint endpoint groups and the second network policy with the second disjoint endpoint groups.

3. The method of claim 1, further comprising:

representing each network policy of the network policies as a graph, wherein the graph representing the first network policy comprises vertices each representing a respective endpoint group, and an edge,

wherein the converting causes modification of the graph representing the first network policy to replace a vertex representing the converted endpoint group with vertices representing the first disjoint endpoint groups.

4. The method of claim 3, wherein the converting further comprises replicating the edge to form edges to the first disjoint endpoint groups.

5. The method of claim 3, wherein the converting further comprises merging edges in the graph.

6. The method of claim 1, wherein the converting of the endpoint group of the first network policy into the first disjoint endpoint groups is according to a hierarchical relationship of labels associated with endpoint groups.

7. The method of claim 1, wherein the converting of the endpoint group of the first network policy into the first disjoint endpoint groups is according to a mapping of labels in a plurality of different hierarchical structures, each hierarchical structure of the hierarchical structures specifying hierarchical relationships among labels associated with endpoint groups.

8. The method of claim 1, further comprising dividing, by the system, a given endpoint group of the first disjoint endpoint groups into further disjoint endpoint groups,

wherein the network policies merged comprise the first network policy with the further disjoint endpoint groups.

9. The method of claim 1, wherein the converting further comprises:

merging composition constraints for a first collection of endpoint groups with composition constraints for a second collection of endpoint groups, the merging of the composition constraints forming merged composition constraints,

wherein the merging of the network policies is according to the merged composition constraints.

10. The method of claim 1, wherein the converting further comprises:

replicating composition constraints responsive to the converting of the endpoint group of the first network policy into the first disjoint endpoint groups,

wherein the merging of the network policies is according to the replicated composition constraints.

11. A system comprising:

at least one processor to:

receive graphs representing respective network policies, each network policy of the network policies specifying at least one characteristic of communications allowed between endpoint groups, wherein a first graph of the graphs includes vertices representing respective endpoint groups and an edge between the endpoint groups;

normalize a given endpoint group represented by the first graph into endpoint groups that have mutually exclusive memberships; and

combine the normalized graph with another graph of the graphs to form a composite graph representing a composite network policy.

12. The system of claim 11, wherein the combining is according to composition constraints included in the network policies, the composition constraints comprising a first composition constraint specifying that communications between respective endpoint groups are to be blocked.

13. The system of claim 12, wherein the composition constraints comprise a second composition constraint specifying that communications between respective endpoint groups must be allowed.

14. The system of claim 11, wherein the combining of the normalized graphs with the another graph comprises a union of the normalized graph and the another graph, the composite graph including a union of endpoint groups of the normalized graph and the another graph.

15. An article comprising at least one non-transitory machine-readable storage medium storing instructions that upon execution cause a system to:

normalize a first endpoint group of a network policy that specifies at least one characteristic of communications allowed between endpoint groups, the first endpoint group including a plurality of endpoints, and wherein the normalizing produces disjoint endpoint groups based on the first endpoint group, the disjoint endpoint groups having mutually exclusive memberships of endpoints; and

merge the network policy comprising the normalized endpoint group with another network policy, the merging of the network policies providing a composite network policy.

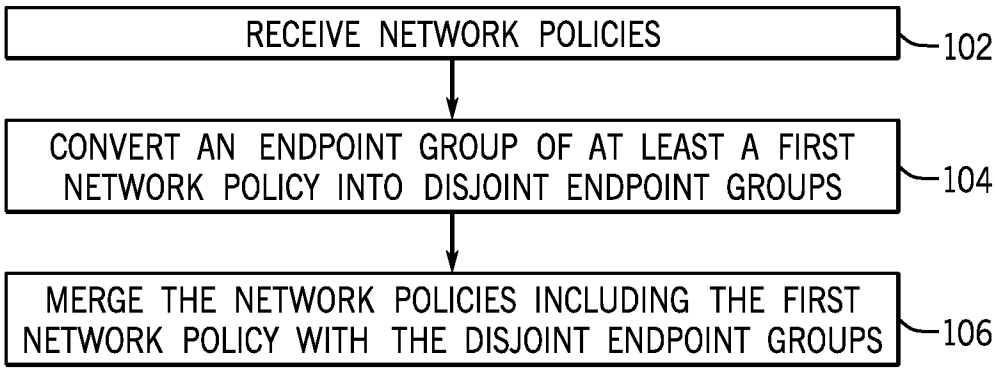
RECEIVE NETWORK POLICIES
102
CONVERT AN ENDPOINT GROUP OF AT LEAST A FIRST NETWORK POLICY INTO DISJOINT ENDPOINT GROUPS
104
MERGE THE NETWORK POLICIES INCLUDING THE FIRST NETWORK POLICY WITH THE DISJOINT ENDPOINT GROUPS
106

FIG. 1

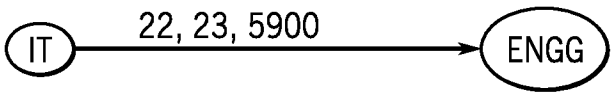

FIG. 2A

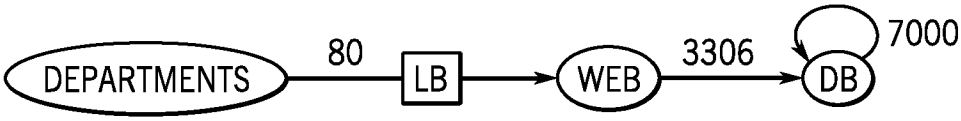

FIG. 2B

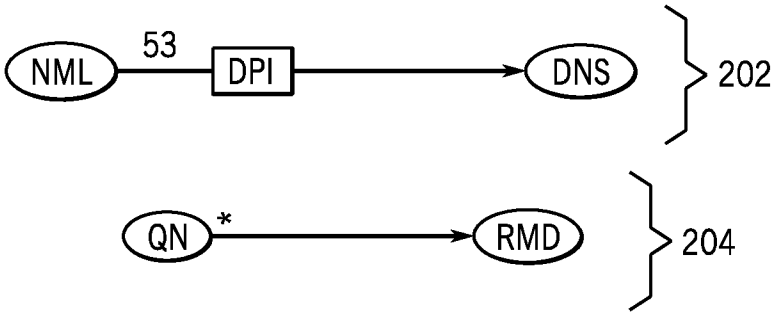

FIG. 2C

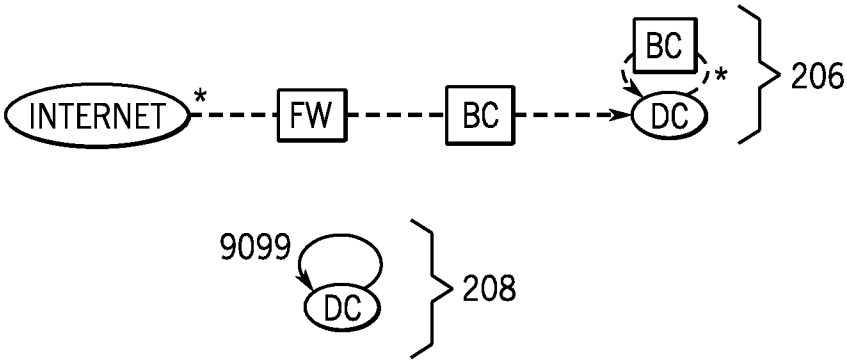

FIG. 2D

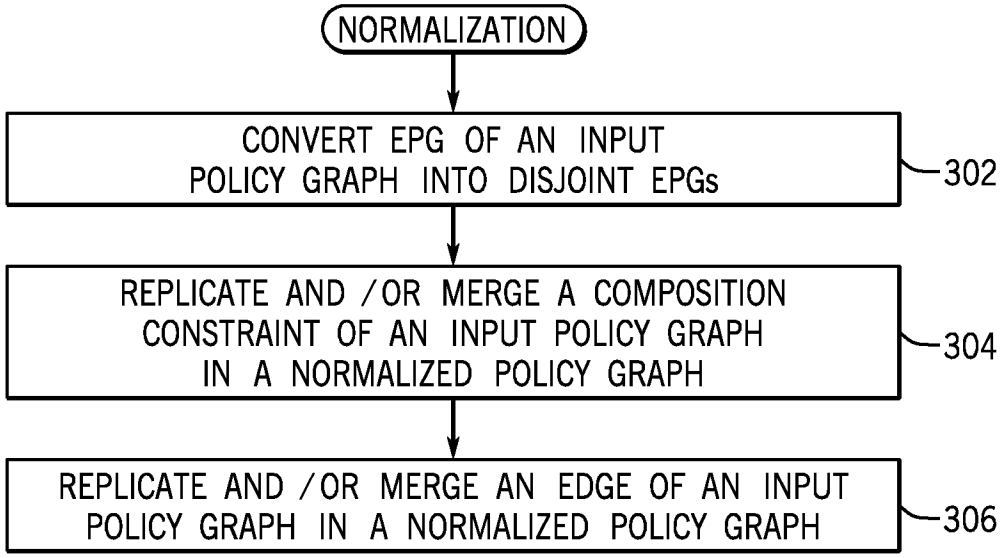
NORMALIZATION
CONVERT EPG OF AN INPUT POLICY GRAPH INTO DISJOINT EPGs
302
REPLICATE AND / OR MERGE A COMPOSITION CONSTRAINT OF AN INPUT POLICY GRAPH IN A NORMALIZED POLICY GRAPH
304
REPLICATE AND / OR MERGE AN EDGE OF AN INPUT POLICY GRAPH IN A NORMALIZED POLICY GRAPH
306

FIG. 3

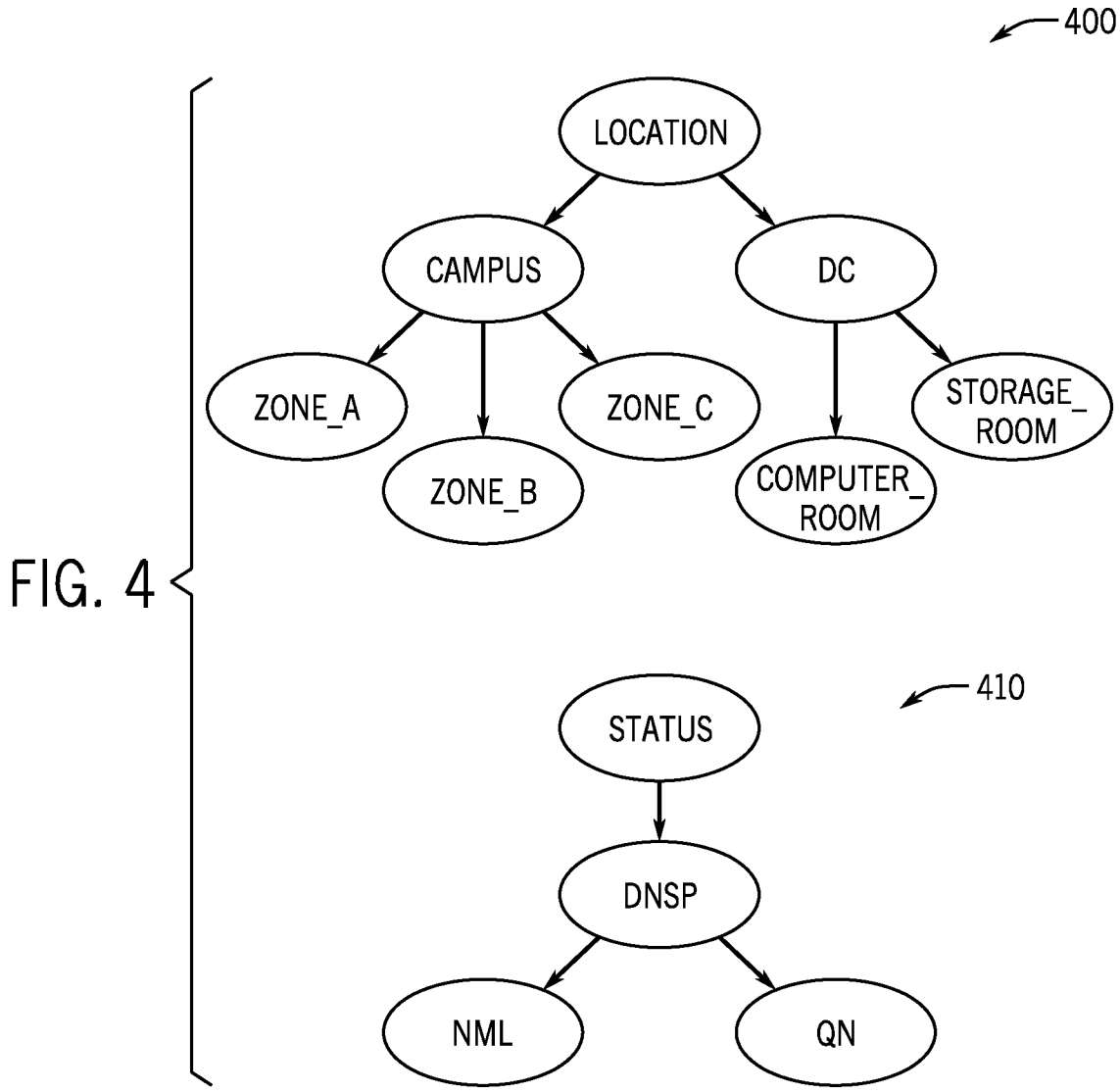
400
LOCATION
CAMPUS
DC
ZONE_A
ZONE_B
ZONE_C
COMPUTER_ ROOM
STORAGE_ ROOM
410
STATUS
DNSP
NML
QN

FIG. 4

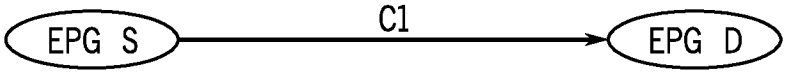

FIG. 5A

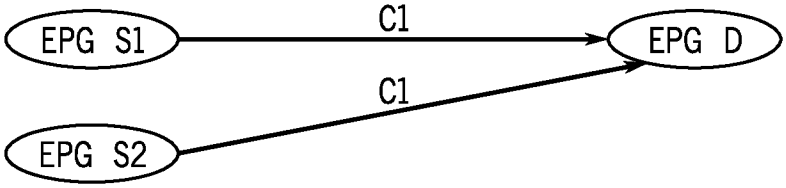

FIG. 5B

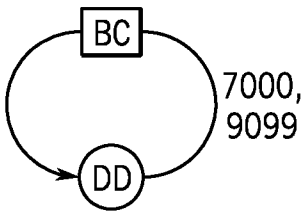

FIG. 6

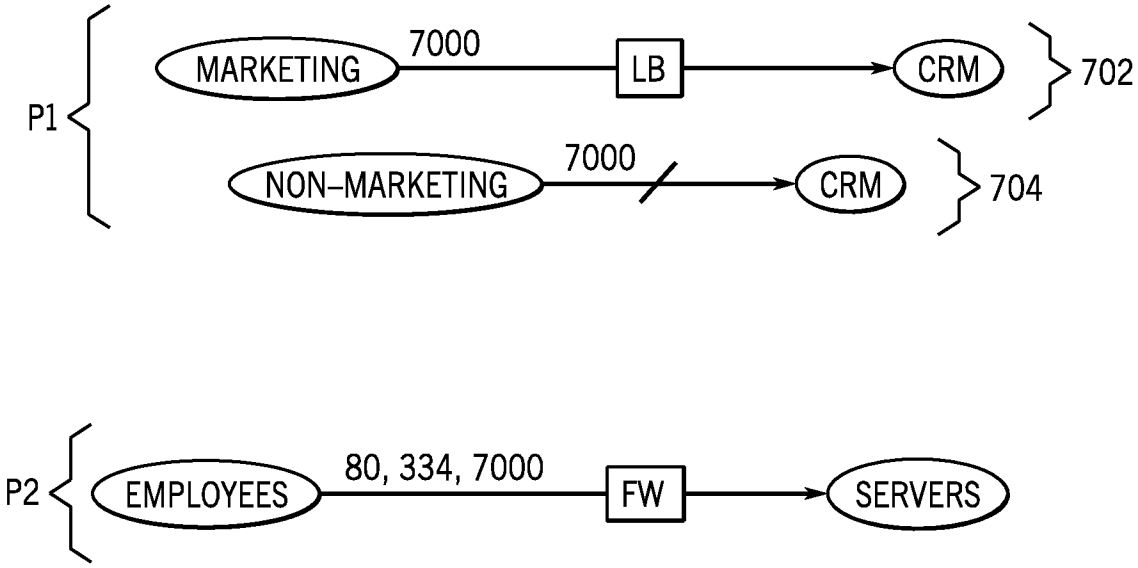
P1
MARKETING
7000
LB
CRM
702
NON–MARKETING
7000
CRM
704
P2
EMPLOYEES
80, 334, 7000
FW
SERVERS

FIG. 7A

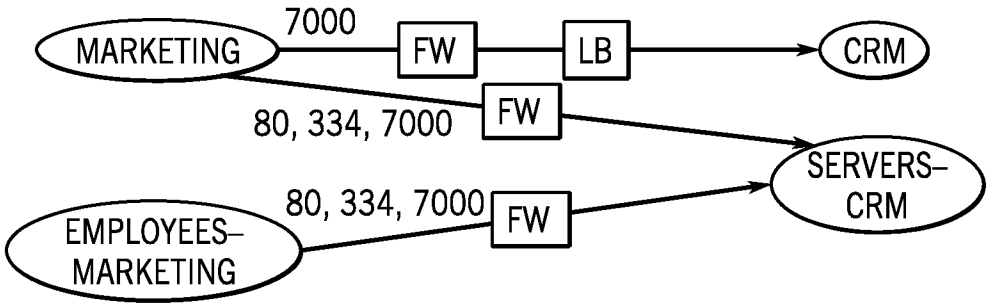
MARKETING
7000
FW
LB
CRM
FW
80, 334, 7000
SERVERS–
CRM
EMPLOYEES–
MARKETING
80, 334, 7000
FW

FIG. 7B

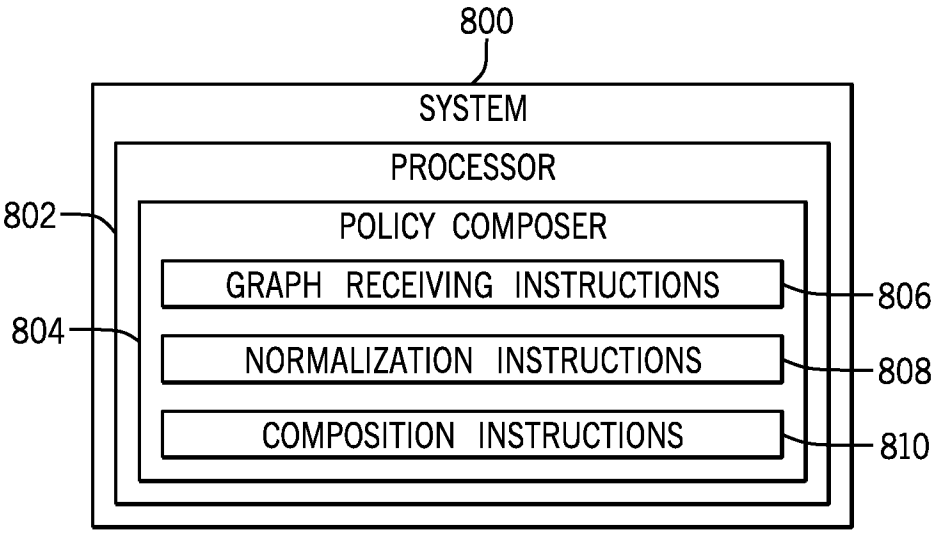
800
SYSTEM
PROCESSOR
802
POLICY COMPOSER
GRAPH RECEIVING INSTRUCTIONS
806
804
NORMALIZATION INSTRUCTIONS
808
COMPOSITION INSTRUCTIONS
810

FIG. 8

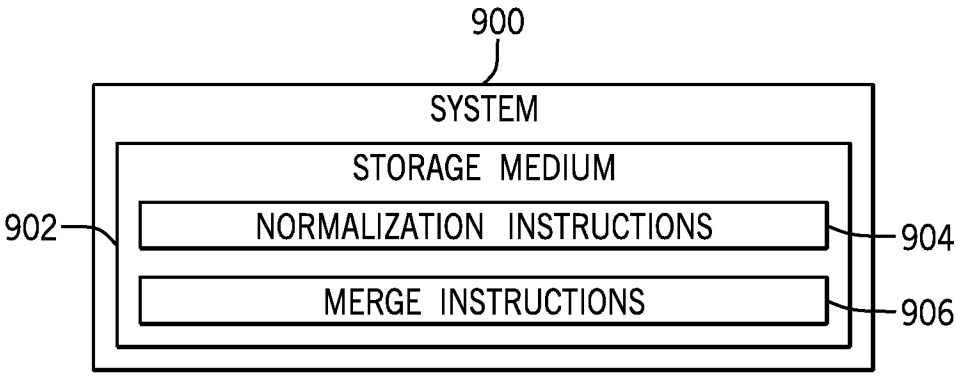
900
SYSTEM
STORAGE MEDIUM
902
NORMALIZATION INSTRUCTIONS
904
MERGE INSTRUCTIONS
906

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/041546

A. CLASSIFICATION OF SUBJECT MATTER

H04L 12/24(2006.01)I, H04L 12/911(2013.01)I, 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 12/24; G06Q 10/10; G06F 17/00; H04L 12/851; H04L 29/06; H04L 12/911; H04L 12/801

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: network, policy, merge, graph

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2009-0249431 A1 (RAYMOND W. MCCOLLUM et al.) 01 October 2009 See paragraphs [0017], [0112]-[0130], claims 1-10 and figures 7, 11, 12.	1-15
Y	US 2015-0063102 A1 (CISCO TECHNOLOGY, INC.) 05 March 2015 See paragraphs [0019], [0024], [0026], [0028], [0064] and figure 4.	1-15
A	US 2014-0150053 A1 (JUNIPER NETWORKS, INC.) 29 May 2014 See paragraphs [0042]-[0047], claims 28-36 and figure 5.	1-15
A	US 2015-0135265 A1 (MYDIGITALSHIELD, INC.) 14 May 2015 See abstract, paragraphs [0077], [0078] and figure 7.	1-15
A	US 2014-0136676 A1 (CALIFORNIA INSTITUTE OF TECHNOLOGY) 15 May 2014 See abstract, claims 1-18 and figures 12, 13, 15.	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
19 August 2016 (19.08.2016)	**08 September 2016 (08.09.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
KIM, Seong Woo
Telephone No. +82-42-481-3348

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2015/041546

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009-0249431 A1	01/10/2009	US 8166516 B2	24/04/2012
US 2015-0063102 A1	05/03/2015	US 2016-0036707 A1 US 9203765 B2	04/02/2016 01/12/2015
US 2014-0150053 A1	29/05/2014	US 2013-0133027 A1 US 8369224 B1 US 8644167 B2	23/05/2013 05/02/2013 04/02/2014
US 2015-0135265 A1	14/05/2015	None	
US 2014-0136676 A1	15/05/2014	None	