Abstract:

Title: DATAFLOW CONSISTENCY VERIFICATION

(57) Abstract: In some examples, a network controller includes a flow repository and a consistency verification engine. The flow repository may store a dataflow configuration previously specified by an application. The consistency verification engine may access the dataflow configuration from the flow repository, retrieve flow table entries from a flow table of a network device, identify an inconsistency between the dataflow configuration previously specified by the application and the flow table entries of the network device, and respond to the identified inconsistency.
— 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))
DATAFLOW CONSISTENCY VERIFICATION

BACKGROUND

[0001] High speed communication networks form part of the backbone of what has become indispensable worldwide data connectivity. Within such communication networks, various network devices including endpoint devices and switching devices direct network traffic from source ports to destination ports, helping to eventually guide a data packet from a source to a destination. Improvements in the reliability and efficiency of these communication networks will increase the effectiveness of communicating data.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Certain examples are described in the following detailed description and in reference to the drawings.

[0003] Figure 1 shows an example of a system that supports dataflow consistency verification.

[0004] Figure 2 shows an example of an architecture that includes a consistency verification engine to support dataflow consistency verification.

[0005] Figure 3 shows an example of a dataflow consistency verification process that a consistency verification engine may perform.

[0006] Figure 4 shows an example of logic that a system or device may implement to support dataflow consistency verification.

[0007] Figure 5 shows an example of a device that supports dataflow consistency verification.
DETAILED DESCRIPTION

[0008] The disclosure herein describes systems, methods, devices, and logic that may support dataflow consistency verification. In particular, the features described herein may support verification that network traffic forwarding or processing requirements previously specified by applications are actually implemented and correctly effectuated by network devices in a communication network. The dataflow consistency verification features described herein may be applied to a software defined networking (SDN) architecture, and an SDN controller may implement dataflow consistency verification features to increase network reliability, verify that network devices are appropriately configured to process network traffic according to the specified requirements, and increase the efficiency of packet processing logic of the network devices.

[0009] Figure 1 shows an example of a system 100 that supports dataflow consistency verification. The system 100 may take the form of a computing system, including a single or multiple computing devices such as application servers, network devices, network controllers, compute nodes, desktop or laptop computers, smart phones or other mobile devices, tablet devices, embedded controllers, and more. The system 100 may be part of a communication network, and may thus take the form of an endpoint device, a controller device that manages switching functions in the communication network, network devices such as a switch, router, hub, gateway, an edge device, or any other device that transmits, receives, or otherwise controls the routing of data through the communication network.

[0010] The system 100 includes a network controller 102, which may be any physical or logical entity that manages network functions of a communication network. The system 100 may implement the network controller 102 (and components thereof) in various ways, for example as hardware and programming. The programming for the network controller 102 may take the form of processor-executable instructions stored on a non-transitory machine-readable storage medium and the hardware for the network controller 102 may include a processing resource to execute those instructions. A processing resource may
take the form of single processor or multi-processor systems, and in some examples, the system 100 implements multiple engines using the same computing system features or hardware components (e.g., a common processing resource).

[0011] In some examples, a communication network managed by the network controller 102 is implemented and controlled through a software defined networking (SDN) architecture. The network controller 102 in these examples may implement an SDN controller. As an SDN controller, the network controller 102 may be physically separate from the network devices managed by the network controller 102. In that regard, the SDN architecture may decouple the system that makes decisions about where network traffic is sent (which may be referred to as a control plane) from the underlying systems that forward the network traffic to the selected destination (which may be referred to as a data plane). The SDN controller may control how network traffic is processed or forwarded based on traffic requirements or configurations specified by endpoint systems executing applications that communicate the network traffic (which may be referred to as an application plane). In some examples, the network controller 102 and managed network devices may, for example, support the OpenFlow protocol and exchange control plane communications according to the OpenFlow protocol. However, any other SDN protocol may be supported by the network controller 102 and managed network devices.

[0012] The network controller 102 may control how data traverses across the data plane. The network controller 120 may do so by determining and assigning data routes in the communication network for traffic flows and configuring managed network devices to route a particular dataflow according to a determined data route. The network controller 102 may thus assign network switching functions to managed network devices in the communication network, for example through flow table entries that specify matching criteria and an accompanying action to perform for network traffic that satisfies the matching criteria.

[0013] The network controller 102 may propagate flow table entries to various network devices in a managed communication network based on dataflow
configurations specified by an application. A dataflow configuration may refer to any configuration, characteristic, accompanying action, or requirement for network traffic communicated across a communication network. Thus, example dataflow configurations may include a required level of quality of service (QoS), network paths for communicated traffic to traverse through, packet processing or security actions to perform on network traffic, flow dependencies or links to other dataflows, or any other rule, requirement, action, constraints or any other configuration to apply to network traffic. Dataflow configurations may thus specify network resources or requirements of an application that the network controller 102 provides through control and management of network devices in the data plane.

[0014] In some examples, a dataflow configuration specified by an application applies to a particular dataflow or a particular set of dataflows. A dataflow may refer to a stream of data (e.g., sequence of packets) communicated between a particular source and a particular destination. Thus, the application may specify dataflow configurations for dataflows communicated to various, different destinations (e.g., other applications, end hosts, devices, etc.) and do so on a per-dataflow basis.

[0015] As described in greater detail herein, the network controller 102 may verify that dataflow configurations specified by applications are actually implemented as flow table entries in network devices of a communication network. To do so, the network controller 102 shown in Figure 1 includes a flow repository 108. The flow repository 108 may store dataflow configurations from any number of applications of the application plane, and the network controller 102 (or system 100 implementing the network controller 102) may implement the flow repository 108 as any type of combination of data structures, such as a table, list, or in various other forms. In Figure 1, the flow repository 108 is part of the network controller 102. In other examples, the flow repository 108 may be remote or logically separate from the network controller 102.

[0016] The network controller 102 shown in Figure 1 also includes a consistency verification engine 110. The consistency verification engine 110 may be a
component of the network controller 102, and may thus take the form of hardware and programming or any other physical implementation form of the network controller 102. The consistency verification engine 110 may include various engine components through which the consistency verification engine 110 may verify that dataflow configurations specified by applications are properly (e.g., correctly, accurately, or consistently) implemented in network devices of a communication network. In the example shown in Figure 1, the consistency verification engine 110 includes an engine component to access the dataflow configuration from the flow repository 108; an engine component to retrieve flow table entries from a flow table of a network device; an engine component to identify an inconsistency between the dataflow configuration previously specified by the application and the flow table entries of the network device; and an engine component to respond to the identified inconsistency.

[0017] An inconsistency between the dataflow configuration and flow table entries may occur when flow table entries of a network device cause the network device to process network traffic of a particular dataflow differently than as specified by a dataflow configuration applicable to the particular dataflow. Thus, the dataflow consistency verification features described herein may support increased effectiveness, reliability, or efficiency in network traffic communication. Some example features relating to dataflow consistency verification are described in greater detail next.

[0018] Figure 2 shows an example of an architecture 200 that includes the consistency verification engine 110 to support dataflow consistency verification. The example architecture 200 shown in Figure 2 includes an application layer with three applications, labeled as the applications 201, 202, and 203. The applications may be applications of any type or function, and in an SDN architecture, the applications 201, 202, and 203 may be referred to as SDN applications. The architecture 200 also includes the network controller 102, which in the example in Figure 2 includes the flow repository 108 and the consistency verification engine 110.
The architecture 200 shown in Figure 2 also includes the network devices 211, 212, and 213, which the network controller 102 may manage, controlling how network traffic is routed across the network devices 211, 212, and 213. The network devices 211, 212, and 213 may be switching devices or any other device that processes or routes network traffic across a communication network. The network controller 102 may configure the network devices 211, 212, and 213 through control plane communications, e.g., according to the OpenFlow protocol. Thus, network traffic (e.g., to and from the applications) may be routed by the network devices 211, 212, and 213 across a data plane.

The network controller 102 may receive a dataflow configuration 220 from an application, such as the application 203 in Figure 2. As noted above, the dataflow configuration 220 may specify any number of requirements, constraints, characteristics, or configurations for network traffic such as a particular dataflow or set of dataflows. In particular, the dataflow configuration 220 specified by the application 203 may apply to a particular dataflow (or set of dataflows) received or transmitted by the application 203. The network controller 102 may store entries in the flow repository 108 for the dataflow configurations specified by multiple applications, such as the applications 201, 202, and 203.

In some examples, an application itself inserts an entry into the flow repository 108 for a specified dataflow configuration. In that regard, the network controller 102 may provide an interface (e.g., an application program interface (API)) by which SDN applications may create, read, update, or delete entries in the flow repository 108 for dataflow configurations specified by the applications. As an SDN application may determine the specific requirements and characteristics of network traffic communicated by the SDN application, the consistency verification engine 110 may aggregate the dataflow configurations specified by the applications in the flow repository 108, but may rely upon SDN applications themselves to specify, control, and update the particular dataflow configurations required by the SDN applications.

The dataflow configuration 220 may include various data fields to indicate the particular configuration elements specified by the dataflow
configuration 220. Example data fields may include target network devices to effectuate (and thus, implement) the dataflow configuration 220 on, matching criteria for network traffic (e.g., that specifies a particular dataflow that the dataflow configuration 220 applies to), an action to perform for network traffic satisfying the matching criteria, and various other elements. In some implementations, the OpenFlow protocol provides a set of possible data fields that an application can populate in communicating the dataflow configuration 220, such as source or destination data fields (e.g., specified as IP addresses), ethertype fields, packet priority, or other matching criteria, packet forwarding or processing actions to perform, applicable time periods of the dataflow configuration 220, and more. The network controller 102 may implement the dataflow configuration 220 by configuring the network devices 211, 212, and 213 (as applicable) to process and forward network traffic according to the dataflow configuration 220.

[0023] The consistency verification engine 110 may verify that a network device configured to implement the dataflow configuration 220 actually does so. Put another way, the consistency verification engine 110 may verify that the flow table entries of a network device configured to implement the dataflow configuration 220 are actually present in flow tables of the network device. As such, the consistency verification engine 110 may perform a dataflow/consistency verification after the dataflow configuration 220 is specified by the application 203 and after the initial implementation in the networking devices by the network controller 102.

[0024] To perform a dataflow consistency verification for the application 203, the consistency verification engine 110 may access the dataflow configurations 220 previously specified by the application 203, e.g., from the flow repository 108. As the application 203 may control the dataflow/configurations 220 requested for network traffic communicated to or from the application 203, in some examples the consistency verification engine 110 queries the application 203 itself for the dataflow configurations 220 to receive an up-to-date or most recent version of the dataflow configurations 220. As another example, the consistency verification
engine 110 may request the application 203 to update the dataflow configurations 220 in the flow repository 108, which may allow the consistency verification engine 110 to retrieve an updated version of the dataflow configurations 220 for verification.

[0025] In performing the dataflow consistency verification, the consistency verification engine 110 may also retrieve the flow table entries of network devices in the communication network that apply the dataflow configurations 220 of the application 203. In Figure 2, the consistency verification engine 110 retrieves the flow table entries 230 from the network device 213. Then, the consistency verification engine 110 may identify whether an inconsistency exists between the dataflow configurations 220 accessed from the flow repository 108 (or application 203) and the flow table entries 230 retrieved from the network device 213. The consistency verification engine 110 may identify an inconsistency through a determination that the flow table entries 230 of the network device do not include an entry corresponding to at least one of the dataflow configurations 220 previously specified by the application 203.

[0026] In response to identifying an inconsistency, the consistency verification engine 110 may respond by logging the identified inconsistency, taking a corrective action to address the identified inconsistency, or both. As such, the consistency verification engine 110 may perform a dataflow consistency verification to ensure that the dataflow configurations 220 specified by the application 203 are implemented correctly by the network device 213 (for the specific dataflow configurations 220 implemented by the network device 213). The consistency verification engine 110 may likewise perform the dataflow consistency verification for other network devices in the communication network, such as for the specific dataflow configurations 220 implemented by the network devices 211 and 212 respectively. Thus, the consistency verification engine 110 may verify the consistency of any number of dataflow configurations previously specified by any number of applications of an application layer in an SDN architecture.
[0027] By performing the dataflow consistency verification subsequent to an initial implementation on managed network devices, the consistency verification engine 110 may account for implementation failures, lingering (and stale) flow table entries for crashed applications that inefficiently consume flow table resources, failure of table entry update or delete operations by the network controller 102, and various other scenarios by which an inconsistency may develop between the dataflow configurations specified by applications and the actual flow table entries of managed network devices. Dataflow consistency verifications by the consistency verification engine 110 may also protect against update failures for flow operations, e.g., failure to add a table entry corresponding to or updating a particular dataflow configuration in a flow table of a network device. Update failures may occur due to network failures, flow table overflow (e.g., when a flow table is full), channel starvation, and specification of an invalid dataflow for a particular network device. The consistency verification engine 110 may identify inconsistencies in the flow tables of network devices caused by any of these various scenarios and address the identified inconsistencies.

[0028] In some examples, the consistency verification engine 110 periodically performs the dataflow consistency verifications, thus verifying the network state with application requirements at various different times to account for update failures, application crashes, or other events that may cause an inconsistency between the dataflow configurations specified by SDN applications and flow table entries of the data layer network devices processing and routing the application network traffic. As another example, the consistency verification engine 110 may perform the dataflow consistency verifications at user-specified times, in response to certain network events (e.g., when a network device or application server reboots), or according to any other verification trigger criteria (e.g., when network bandwidth consumption exceeds a threshold amount in the aggregate or for a particular network device.

[0029] Figure 3 shows an example of a dataflow consistency verification process that the consistency verification engine 110 may perform. In particular,
Figure 3 illustrates an example dataflow consistency verification that the consistency verification engine 110 may perform for an application 201.

[0030] The consistency verification engine 110 may perform a dataflow consistency verification in response to receiving a consistency verification request 310 from the application 201. The consistency verification request 310 may take the form of any communication or message, and may indicate a request from the application 201 to subscribe to dataflow consistency verifications by the consistency verification engine 110. The consistency verification request 310 may request dataflow consistency verifications for the application 201 at a periodic rate, at on-demand times specified by the application 201, or both.

[0031] Consistency verification requests 310 may also be a mechanism by which the consistency verification engine 110 determines which applications to perform dataflow consistency verifications for. The consistency verification engine 110 may perform dataflow consistency verifications for the particular applications of an application layer from which a consistency verification request 310 has been received, but not for other applications of the application layer from which a consistency verification request 310 has not been received. As such, the consistency verification engine 110 may perform dataflow consistency verifications for some or all of the applications of an SDN application layer, depending on which applications provide a consistency verification request 310 to the consistency verification engine 110.

[0032] To perform the dataflow consistency verification for the application 201, the consistency verification engine 110 may retrieve both the dataflow configurations specified by the application 201 and the flow table entries of any network device that implements at least one of the dataflow configurations specified by the application 201. Thus, the consistency verification engine 110 may send a flow repository query 320 to the flow repository 108 (e.g., as a table query for any dataflow configurations previously specified by the application 201). As a response to the flow repository query 320, the consistency verification engine 110 may receive the dataflow configurations 321 previously specified by the application 201.
[0033] The consistency verification engine 110 may also send a network device query 330 to the network device 211, for example as a message communicated according to the OpenFlow protocol. The network device 211 may respond with with the flow table entries 313 presently implemented by the network device 211, which may allow the consistency verification engine 110 to ascertain how the network device 211 is configured to process and forward traffic. That is, the flow table entries 313 may indicate the dataflows that the network device is matching for, as each of the flow table entries 313 may correspond to a particular dataflow (e.g., via matching criteria) and specify the corresponding action to perform for network traffic of the particular dataflow.

[0034] The consistency verification engine 110 may detect an inconsistency between the dataflow configurations 321 and the flow table entries 331 when at least one of the dataflows configured in the dataflow configurations 321 does not have a corresponding entry among the flow table entries 331 of the network device 211. Put another way, the dataflow configurations 321 may specify a particular action to perform, traffic requirement to adhere to, or network route to communicate through for network traffic of a particular dataflow. An inconsistency may exist when the network device 211 does not include a flow table entry for the particular dataflow, and thus does not process or route network traffic of the particular dataflow according to the specified action, requirement, or network route. As another example, the consistency verification engine 110 may identify an inconsistency when the action, requirement, or network route specified in a dataflow configuration of the application 201 differs from the action, requirement, or network route indicated in a flow table entry among the flow table entries 331 that corresponds to the particular dataflow (e.g., with matching criteria that identifies the particular dataflow). In such a case, the network device 211 may identify the network traffic of the particular dataflow, but process or forward the identified network traffic differently or inconsistency with the dataflow configuration specified by the application 201.

[0036] To compare the retrieved dataflow configurations 321 and the flow table entries 331, the consistency verification engine 110 may convert the dataflow
configurations 321 of the application 201 and the flow table entries of the network device 211 into a common object format. For example, the consistency verification engine 110 may convert the dataflow configurations 321 and the flow table entries 331 into the common object format by converting data fields of the dataflow configurations 321 to match a data format of the flow table entries 331 of the network device 211.

[0036] The OpenFlow protocol, for example, may provide a set of various data fields through which the application 201 may specify the dataflow configurations 321, and the consistency verification engine 110 may filter or parse the dataflow configurations 321 for the specific data fields that correspond to the data fields (or format) of the flow table entries 311 (e.g., source IP address, destination IP address, TCP addresses, communication protocols, ethertype fields, data fields corresponding to the tuple fields of a 5-tuple, or any other data field that identifies a dataflow). Then, the consistency verification engine 110 may compare the dataflow configurations 321 and flow table entries 331 converted into the common object format to identify an inconsistency (e.g., identify a missing entry among the flow table entries 311 or an entry with a corresponding action or other parameter that is inconsistent with a corresponding dataflow configuration).

[0037] To respond to an identified inconsistency, the consistency verification engine 110 may take a corrective action, such as by sending an inconsistency alert 340 to the application 201. The inconsistency alert 340 may be a message sent to the application 201, and may identify the network device 211 and dataflow configuration that is inconsistently (or not) applied by the network device 211. In response to receiving the inconsistency alert 340, the application 201 itself may initiate an update to the network controller 102, e.g., by sending a request to the network controller 102 to update the flow table entries of the network device 211 to include an entry corresponding to any of the dataflow configurations that the network device 211 does not implement or implements incorrectly. As another example of a corrective action, the consistency verification engine 110 itself may update the flow table entries of the network device 211, e.g., through any network device update mechanism supported by OpenFlow or another SDN protocol. The
consistency verification engine 110 may additionally or alternatively log the identified consistency, e.g., for subsequent analysis or use.

[0038] As described above, the consistency verification engine 110 of a network controller 102 may support dataflow consistency verifications to ensure that the requested network traffic requirements and configurations are appropriately implemented on managed network devices of a communication network. As such, the consistency verification engine 110 may increase the efficiency and reliability of the communication network.

[0039] Through the features described herein, the consistency verification engine 110 may support increased flexibility in the specification of dataflow characteristics and requirements by SDN applications. For example, the consistency verification engine 110 may support application restriction of the placement of dataflows, allowing the applications to selecting a set of target devices to apply a dataflow configuration to. An SDN application may request, through a specified dataflow configuration, that a particular dataflow or set of dataflows target a specific network device, a group of network devices, or all network devices managed by the network controller 102, for example. The consistency verification engine 110 may verify that such dataflow restrictions are actually effectuated by the selected target network devices.

[0040] As another example, the consistency verification engine 110, through dataflow consistency verifications, may ensure that dataflow dependencies are met. For instance, an application may specify that dataflows F1 and F2 be part of a same dataflow group, and request that dataflows F1 and F2 be installed, effectuated, implemented, or removed from the same target network devices. The dataflow consistency verifications performed by the consistency verification engine 110 may ensure that such flow dependencies are actually implemented on the applicable target network devices, e.g., by verifying that the flow table entries of the target network devices include entries for both dataflows F1 and F2 as opposed to just one or none of the dataflows F1 and F2.

[0041] While a few examples have been described above, the consistency verification engine 110 may similarly verify that particular dataflow configurations
specified by various applications of an application layer are properly effectuated by network devices of a communication network. The consistency verification engine 110 may thus provide and support dataflow consistency verifications in a communication network.

[0042] Figure 4 shows an example of logic 400 that a system or device may implement to support dataflow consistency verification. The system 100, for instance, may implement the logic 400 as hardware, executable instructions stored on a machine-readable medium, or as combinations thereof. In some examples, the network controller 102 implements the logic 400 via the consistency verification engine 110, through which the network controller 102 may perform the logic 400 as a method or process to verify whether dataflow configurations specified by an application are consistent with the flow table entries implemented in network devices of a communication network.

[0043] The logic 400 may be implemented or performed by the consistency verification engine 110, and in doing so the consistency verification engine 110 may maintain a flow repository 108 that stores dataflow configurations of multiple applications (402). The consistency verification engine 110 may also perform a dataflow consistency verification for a particular application among the multiple applications (404).

[0044] In performing the dataflow consistency verification, the consistency verification engine 110 may retrieve a dataflow configuration previously specified by the particular application from the flow repository 108 (406), retrieve flow table entries from a flow table of a network device (408), and identify whether an inconsistency is present between the dataflow configuration previously specified by the particular application and the flow table entries of the network device (410). To identify whether the inconsistency is present between the dataflow configuration and the flow table entries, the consistency verification engine 110 may determine that the flow table entries of the network device do not include an entry corresponding to the dataflow configuration previously specified by the application.
[0045] When the inconsistency is present, the consistency verification engine 110 may respond to the identified inconsistency (412), for example by logging the identified inconsistency, taking a corrective action to address the identified inconsistency, or both. In taking the corrective action b address the identified inconsistency, the consistency verification engine 110 may alert the identified application to update the flow table entries of the network device b include an entry that corresponds to the dataflow configuration. As another example, the consistency verification engine 110 may take corrective action by itself adding an entry corresponding to the dataflow configuration into a flow table of the network device.

[0046] In some examples, the consistency verification engine 110 may implement or perform the logic 400 further to receive a consistency verification request from the particular application, and, in response, periodically performing the dataflow consistency verification for the particular application. As another example, the consistency verification engine 110 may implement or execute the logic 400 further to determine not to perform a dataflow consistency verification for any of the multiple applications from which a consistency verification request has not been received by the network controller 102.

[0047] Figure 5 shows an example of a device 500 that supports dataflow consistency verification. The device 500 may include a processing resource 510, which may take the form of a single or multiple processors. The processor(s) may include a central processing unit (CPU), microprocessor, or any hardware device suitable for executing instructions stored on a machine-readable medium, such as the machine-readable medium 520 shown in Figure 5. The machine-readable medium 520 may take the form of any non-transitory electronic, magnetic, optical, or other physical storage device that stores executable instructions, such as the instructions 522, 524, 526, and 528 shown in Figure 5. As such, the machine-readable medium 520 may be, for example, Random Access Memory (RAM) such as dynamic RAM (DRAM), flash memory, memristor memory, spin-transfer torque memory, an Electrically-Erasable Programmable Read-Only Memory (EEPROM), a storage drive, an optical disk, and the like.
[0048] The device 500 may execute instructions stored on the machine-readable medium 520 through the processing resource 510. Executing the instructions may cause the device 500 to perform any of the features described herein, including according to any features of the network controller 102, including the consistency verification engine 110. For example, execution of the instructions 522, 524, 526, and 528 by the processing resource 510 may cause the device 500 to maintain a flow repository that stores dataflow configurations of multiple applications (instructions 522), the dataflow configurations previously specified by the multiple applications; receive a consistency verification request from a first application of the multiple applications, but not from a second application of the multiple applications (instructions 524); perform a dataflow consistency verification for the first application by retrieving a dataflow configuration previously specified by the first application, retrieving flow table entries from a flow table of a network device, identifying whether an inconsistency is present between the dataflow configuration previously specified by the first application and the flow table entries of the network device, and when the inconsistency is present, responding to the identified inconsistency (instructions 526); and determine not to perform a dataflow consistency verification for the second application (instructions 528).

[0049] In some examples, the instructions are executable by the processing resource 510 to identify the inconsistency through a determination that the flow table entries of the network device do not include an entry for the dataflow previously specified by the first application.

[0050] The systems, methods, devices, and logic described above, including the network controller 102 and the consistency verification engine 110, may be implemented in many different ways in many different combinations of hardware, logic, circuitry, and executable instructions stored on a machine-readable medium. For example, the network controller or the consistency verification engine 110 may include circuitry in a controller, a microprocessor, or an application specific integrated circuit (ASIC), or may be implemented with discrete logic or components, or a combination of other types of analog or digital circuitry,
combined on a single integrated circuit or distributed among multiple integrated circuits. A product, such as a computer program product, may include a storage medium and machine readable instructions stored on the medium, which when executed in an endpoint, computer system, or other device, cause the device to perform operations according to any of the description above, including according to any features of the network controller, including the consistency verification engine 110.

[0051] The processing capability of the systems, devices, and engines described herein, including network controller 102 and the consistency verification engine 110, may be distributed among multiple system components, such as among multiple processors and memories, optionally including multiple distributed processing systems. Parameters, databases, and other data structures, such as the flow repository 108, may be separately stored and managed, may be incorporated into a single memory or database, may be logically and physically organized in many different ways, and may implemented in many ways, including data structures such as linked lists, hash tables, or implicit storage mechanisms. Programs may be parts (e.g., subroutines) of a single program, separate programs, distributed across several memories and processors, or implemented in many different ways, such as in a library (e.g., a shared library).

[0052] While various examples have been described above, many more implementations are possible.
CLAIMS

1. A network controller comprising:
   a flow repository that stores a dataflow configuration previously specified
   by an application; and
   a consistency verification engine to:
       access the dataflow configuration from the flow repository;
       retrieve flow table entries from a flow table of a network device;
       identify an inconsistency between the dataflow configuration
       previously specified by the application and the flow table
       entries of the network device; and
       respond to the identified inconsistency.

2. The network controller of claim 1, wherein the consistency verification
   engine is to identify the inconsistency through a determination that the flow table
   entries of the network device do not include an entry corresponding to the
   dataflow configuration previously specified by the application.

3. The network controller of claim 1, wherein the consistency verification
   engine is to identify the inconsistency by:
       converting the dataflow configuration accessed from the flow repository
       and the flow table entries retrieved from the network device into a common
       object format; and
       comparing the dataflow configuration and flow table entries converted
       into the common object format to identify the inconsistency.

4. The network controller of claim 3, wherein the consistency verification
   engine is to convert the dataflow configuration and the flow table entries into the
   common object format by converting data fields of the dataflow configuration to
   match a data format of the flow table entries of the network device.
5. The network controller of claim 1, wherein the consistency verification engine is further to maintain the flow repository to include flow configurations of multiple applications.

6. The network controller of claim 1, wherein the consistency verification engine is to respond to the identified inconsistency by logging the identified inconsistency, taking a corrective action to address the identified inconsistency, or both.

7. The network controller of claim 6, wherein the consistency verification engine is to take the corrective action to address the identified inconsistency by alerting the application to update the table flow table entries of the network device to include an entry that corresponds to the dataflow configuration.

8. A method comprising:
   through a network controller:
   maintaining a flow repository that stores dataflow configurations of multiple applications, the dataflow configurations previously specified by the multiple applications;
   performing a dataflow consistency verification for a particular application among the multiple applications by:
      retrieving a dataflow configuration previously specified by the particular application from the flow repository;
      retrieving flow table entries from a flow table of a network device;
      identifying whether an inconsistency is present between the dataflow configuration previously specified by the particular application and the flow table entries of the network device; and
      when the inconsistency is present, responding to the identified inconsistency.
9. The method of claim 8, further comprising receiving a consistency verification request from the particular application; and, in response, periodically performing the dataflow consistency verification for the particular application.

10. The method of claim 8, further comprising determining not to perform a dataflow consistency verification for any of the multiple applications from which a consistency verification request has not been received by the network controller.

11. The method of claim 8, wherein identifying whether the inconsistency is present between the dataflow configuration and the flow table entries comprises determining that the flow table entries of the network device do not include an entry corresponding to the dataflow configuration previously specified by the application.

12. The method of claim 8, wherein responding to the identified inconsistency comprises logging the identified inconsistency, taking a corrective action to address the identified inconsistency, or both.

13. The method of claim 12, wherein taking the corrective action to address the identified inconsistency comprises alerting the particular application to update the flow table entries of the network device to include an entry that corresponds to the dataflow configuration.

14. A non-transitory machine-readable medium comprising instructions executable by a processing resource to:

   maintain a flow repository that stores dataflow configurations of multiple applications, the dataflow configurations previously specified by the multiple applications:
receive a consistency verification request from a first application of the multiple applications, but not from a second application of the multiple applications;

perform a dataflow consistency verification for the first application by:

- retrieving a particular dataflow configuration previously specified by the first application;
- retrieving flow table entries from a flow table of a network device;
- identifying whether an inconsistency is present between the particular dataflow configuration previously specified by the first application and the flow table entries of the network device; and
- when the inconsistency is present, responding to the identified inconsistency; and

determine not to perform a dataflow consistency verification for the second application.

15. The non-transitory machine-readable medium of claim 14, wherein the instructions are executable to identify the inconsistency through a determination that the flow table entries of the network device do not include an entry corresponding to the particular dataflow configuration previously specified by the application.
Figure 4

1. Maintain a flow repository that stores dataflow configurations of multiple applications.
2. Perform a dataflow consistency verification for a particular application among the multiple applications.
3. Retrieve a dataflow configuration previously specified by the particular application from the flow repository.
4. Retrieve flow table entries from a flow table of a switching device.
5. Identify whether an inconsistency is present between the dataflow configuration previously specified by the particular application and the flow table entries of the switching device.
6. If inconsistency is present, respond to the identified inconsistency. If not, end the process.

End
Machine-Readable Medium 520

Instructions to maintain a flow repository that stores dataflow configurations of multiple applications, the dataflow configurations previously specified by the multiple applications  

Instructions to receive a consistency verification request from a first application of the multiple applications, but not from a second application of the multiple applications  

Instructions to perform a dataflow consistency verification for the first application by retrieving a particular dataflow configuration previously specified by the first application; retrieving flow table entries from a flow table of a switching device; identifying whether an inconsistency is present between the particular dataflow configuration previously specified by the first application and the flow table entries of the switching device; and when the inconsistency is present, responding to the identified inconsistency  

Instructions to determine not to perform a dataflow consistency verification for the second application  

Figure 5
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

H04L 12/859(2013.01)i, H04L 12/24(2006.01)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 12/859; G06F 15/173; G06F 17/00; H04L 12/26; H04L 29/06; H04L 12/70; G01R 31/08; H04L 12/24

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: application, data, flow, configuration, network, device, table, entries, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>US 2014-0269690 AI (QUALCOMM INCORPORATED) 18 September 2014</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td>See paragraphs [0009][0010]; [0026]; [0041]; and figures 1.</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US 2014-0108632 AI (CISCO TECHNOLOGY, INC.) 17 April 2014</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td>See paragraphs [0011]; [0029]; and figures 1-2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See column 2, line 25 - column 3, line 51; and figures 1-2.</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US 8500513 B2 (JUHA PARK et al.) 30 September 2014</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td>See column 4, line 57 - column 5, line 54; and figures 3-5.</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US 8504687 B2 (EUGENIO MARIA MAFFIONE et al.) 6 August 2013</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td>See column 6, line 1 - column 7, line 6; and figures 1-2.</td>
<td></td>
</tr>
</tbody>
</table>

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

24 August 2016 (24.08.2016)

Date of mailing of the international search report

24 August 2016 (24.08.2016)

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
LEE, EUN KYU

Telephone No. +82-42-481-3580

Form PCT/ISA/210 (second sheet) (January 2015)
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>wO 2014-165235 Al</td>
<td>09/10/2014</td>
</tr>
<tr>
<td>US 2014-0108632 Al</td>
<td>17/04/2014</td>
<td>CN 104737504 A</td>
<td>24/06/2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 2907276 Al</td>
<td>19/08/2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>wO 2014-062365 Al</td>
<td>24/04/2014</td>
</tr>
<tr>
<td>US 8504687 B2</td>
<td>06/08/2013</td>
<td>AR 074229 Al</td>
<td>29/12/2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CA 2744831 Al</td>
<td>03/06/2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CN 102292933 A</td>
<td>21/12/2011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CN 102292933 B</td>
<td>07/05/2014</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 2362990 Al</td>
<td>07/09/2011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IL 213179 DO</td>
<td>31/07/2011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>wO 2010-060480 Al</td>
<td>03/06/2010</td>
</tr>
</tbody>
</table>