

METHOD AND SYSTEM FOR DETECTION OF LOOP BETWEEN CONTROL PLANE AND DATA PLANE BASED ON SOFTWARE DEFINED NETWORK (SDN)

Field of the Invention

This disclosure relates to a method and system for detecting loop between control plane and data plane in a Software Defined Network (SDN) system. Particularly, this disclosure relates to a method and system for monitoring messages from the SDN controller and detecting loop based on the monitored messages.

Summary of the Prior Art

Software-Defined Networking (SDN) is a new network system. Compared with traditional network system, SDN physically separates the control layer and data layer of a network. The control layer is responsible for computing routes for packets, while the data layer is responsible for forwarding packets. In a traditional network, distributed algorithms are adopted to compute routes, and a router or switch comprises both control layer and data layer. This makes network configuration and management very complex. In contrast, SDN separates these two layers physically. Particularly, the control layer is implemented in a centralized server named SDN controller while the data layer remains at switches (the term router is no longer used in SDN). SDN makes network configuration easy. A network administrator only needs to configure SDN controller to manage the network.

Figure 1 shows the architecture of SDN 100. Essentially, SDN has three layers, namely, data layer 130, control layer 120 and application layer 110. The data layer 130 includes devices for forwarding packets, i.e. switches. The data layer 130 is responsible for data processing, forwarding, and status collection based on flow tables. The control layer 120 provides interfaces to control data layer and is responsible for processing abstraction of resources on a data plane, supports collection and maintenance of network topologies and status information, and invokes, based on applications, different resources on the data plane so as to make decisions for the data plane. The application layer 110 includes different services and applications and can manage and control forwarding and processing policies corresponding to the network. The interface between control layer and data layer is called southbound interface 150, while the interface between control layer and application layer is called northbound interface 140.

Through the southbound interface 150 of SDN, control layer 120 controls the forwarding in data layer 130, and data layer 130 reports its status to control layer 120. There are several southbound interface 150 protocols such as OpenFlow, ForCES, PCE-P, etc. Among them, OpenFlow is the most mature and most used. The OpenFlow protocol in the present disclosure refers to version 1.3 or later.

At present, in the network field, no related technologies are provided to detect forwarding loop inside switches. Hence, those skilled in the art are striving to provide a method and system of detecting loop in a SDN system.

Summary of the Invention

The above and other problems are solved and an advance in the state of art is made by systems and methods provided by embodiments in accordance with the disclosure. A first advantage of embodiments of systems and methods in accordance with the disclosure is that forwarding loop arising due to interaction between the control plane and the data plane inside a switch and across switches can be detected in real time. A third advantage of embodiments of systems and methods in accordance with the disclosure is that the systems and methods impose little impact on the network. This is because the systems and methods in accordance with the disclosure are dependent on only lightweight operations, for example, intercepting flow-mod and packet-out messages.

A first aspect of the disclosure describes a method for detecting a loop between a control plane and a data plane in a Software Defined Network (SDN) having a SDN controller being connectable to a plurality of network elements in the data plane. The method comprises: monitoring and intercepting a flow-mod message and a packet-out message from the SDN controller to the network element; updating a forwarding graph according to flow-mod message in response to intercepting the flow-mod message; analysing the information in the packet-out message against the forwarding graph associated to the network element to determine whether a loop exists in response to intercepting the packet-out message; and raising an action ticket in response to determining a loop exists.

According to an embodiment of the first aspect of the disclosure, the step of updating the forwarding graph comprises: retrieving a forwarding graph associated to the network element indicated in the flow-mod message; updating the forwarding graph according to the information in the flow-mod message; and storing the updated forwarding graph.

According to an embodiment of the first aspect of the disclosure, the step of updating the forwarding graph further comprises: reading flow entry command in the flow-mod

message; and revising the forwarding table by adding, modifying, deleting nodes and connection between the nodes in the forwarding graph based on the flow entry command, a node being a representation of a forwarding table of a network element or the SDN controller and wherein the flow entry command comprises adding, modifying and deleting commands.

According to an embodiment of the first aspect of the disclosure, the step of updating the forwarding graph further comprises: analysing the updated forwarding graph to detect whether a loop exist in the data plane raising an action ticket in response to determining a loop exist; and releasing the flow-mod message in response to determining a loop does not exist.

According to an embodiment of the first aspect of the disclosure, the step of analysing the updated forwarding graph to detect whether a loop exist in the data plane comprises: traversing nodes from a first node in the updated forwarding graph along a unidirectional edge of the flow in the updated forwarding graph; and determining a loop exist if the data packet is able to traverse back to the first node, wherein the first node is a representation of a first forwarding table of the network element.

According to an embodiment of the first aspect of the disclosure, the step of analysing the information in the packet-out message comprises: determining whether an action in the packet-out message is an indication to output to a first forwarding table of the network element; retrieving the forwarding graph associated to the network element indicated in the packet-out message in response to determining the action in the packet-out message is to output to the first forwarding table; traversing nodes from a node representing the first forwarding table of the network element along a unidirectional edge of the flow in the forwarding graph; and determining a loop exist if the data packet is able to traverse back to the node representing the SDN controller.

According to an embodiment of the first aspect of the disclosure, the step of analysing the information in the packet-out message comprises: determining whether an action in the packet-out message is an indication to output to a first forwarding table of the network element; retrieving the forwarding graph associated to the network element indicated in the packet-out message in response to determining the action in the packet-out message is to output to the first forwarding table; connecting a node representing the SDN controller to a first node representing the first forwarding table of the network element; traversing nodes from the node representing the SDN controller along a unidirectional edge of the flow in the forwarding graph; and determining a loop exist if the data packet is able to traverse back to the first node.

A second aspect of the disclosure describes a detection system for detecting a loop between a control plane and a data plane in a Software Defined Network (SDN) having a SDN controller being connectable to a plurality of network elements in the data plane. The detection system comprises a processor, a memory, and instructions stored on the memory and executable by the processor to: monitor and intercept a flow-mod message and a packet-out message from the SDN controller to the network element; update a forwarding graph according to flow-mod message in response to intercepting the flow-mod message; analyse the information in the packet-out message against the forwarding graph associated to the network element to determine whether a loop exists in response to intercepting the packet-out message; and raise an action ticket in response to determining a loop exists.

According to an embodiment of the second aspect of the disclosure, the instruction to update the forwarding graph comprises instructions to: retrieve a forwarding graph associated to the network element indicated in the flow-mod message; update the forwarding graph according to the information in the flow-mod message; and store the updated forwarding graph.

According to an embodiment of the second aspect of the disclosure, the instruction to update the forwarding graph further comprises instructions to: read flow entry command in the flow-mod message; and revise the forwarding graph by adding, modifying, or deleting nodes and connection between the nodes in the forwarding graph based on the flow entry command, the nodes being a representation of the forwarding tables of the network element or the SDN controller, and wherein the flow entry command comprises adding, modifying and deleting commands.

According to an embodiment of the second aspect of the disclosure, the instruction to update the forwarding graph further comprises instructions to: analyse the updated forwarding graph to detect whether a loop exist in the data plane; raise an action ticket in response to determining a loop exist; and release the flow-mod message in response to determining a loop does not exist.

According to an embodiment of the second aspect of the disclosure, the instruction to analyse the updated forwarding graph to detect whether a loop exist in the data plane further comprises instructions to: traverse nodes from a first node in the updated forwarding graph along a unidirectional edge of the flow in the updated forwarding graph; and determine a loop exist if the data packet is able to traverse back to the first node, wherein the first node is a representation of a first forwarding table of the network element.

According to an embodiment of the second aspect of the disclosure, the instruction to analyse the information in the packet-out message comprises instructions to: determine whether an action in the packet-out message is an indication to output to the first forwarding table of a network element; retrieve the forwarding graph associated to the network element indicated in the packet-out message in response to determining the action in the packet-out message is to output to a first forwarding table; traverse nodes from a node representing the first forwarding table of the network element along a unidirectional edge of the flow in the forwarding graph; and determine a loop exist if the data packet is able to traverse back to the node representing the SDN controller.

According to an embodiment of the second aspect of the disclosure, the instruction to analyse the information in the packet-out message comprises instructions to: determine whether an action in the packet-out message is an indication to output to the first forwarding table of a network element; retrieve the forwarding graph associated to the network element indicated in the packet-out message in response to determining the action in the packet-out message is to output to a first forwarding table; connecting a node representing the SDN controller to a first node representing the first forwarding table of the network element; traverse nodes from the node representing the SDN controller along a unidirectional edge of the flow in the forwarding graph; and determine a loop exist if the data packet is able to traverse back to the first node

A third aspect of the disclosure describes a detection system for detecting a loop between a control plane and a data plane in a Software Defined Network (SDN) having a SDN controller being connectable to a plurality of network elements in the data plane. The detection system comprises a detection module and a forwarding graph module. The detection module is configured to: monitor and intercept a flow-mod message and a packet-out message from the SDN controller to the network element; forward information in the flow-mod message to the forwarding graph module in response to intercepting the flow-mod message; analyse the information in the packet-out message against a forwarding graph associated to the network element to determine whether a loop exists in response to intercepting the packet-out message; and raise an action ticket in response to determining a loop exists. The forwarding graph module is configured to update a forwarding graph according to the information from the detection module.

According to an embodiment of the third aspect of the disclosure, the forwarding graph module is configured to: receive the information from the detection module; retrieve a forwarding graph associated to the network element indicated in the information; update the forwarding graph according to the information; and store the updated forwarding graph.

According to an embodiment of the third aspect of the disclosure, the forwarding graph module is configured to: create a new forwarding graph for the network element indicated in the information if a forwarding graph not available.

According to an embodiment of the third aspect of the disclosure, the step to update the forwarding graph further comprises the forwarding graph module to: read flow entry command in the flow-mod message; and revise the forwarding graph by adding, modifying, deleting nodes and connection between the nodes in the forwarding graph based on the flow entry command, the nodes being a representation of the forwarding tables of the network element or the SDN controller, wherein the flow entry command comprises adding, modifying and deleting commands.

According to an embodiment of the third aspect of the disclosure, the step to update the forwarding graph further comprises the forwarding graph module to: analyse the updated forwarding graph to detect whether a loop exist in the data plane; raise an action ticket in response to determining a loop exist; and release the flow-mod message in response to determining a loop does not exist.

According to an embodiment of the third aspect of the disclosure, the step to analyse the updated forwarding graph to detect whether a loop exist in the data plane comprises the detection module to: traverse nodes from a first node in the updated forwarding graph along a unidirectional edge of the flow in the updated forwarding graph; and determine a loop exist if the data packet is able to traverse back to the first node, wherein the first node is a representation of a first forwarding table of the network element.

According to an embodiment of the third aspect of the disclosure, the step to analyse the information in the packet-out message comprises the detection module to: determine whether an action in the packet-out message is an indication to output to the first forwarding table of a network element; retrieve the forwarding graph associated to the network element indicated in the packet-out message in response to determining the action in the packet-out message is to output to a first forwarding table; traverse nodes from a node representing the first forwarding table of the network element along a unidirectional edge of the flow in the forwarding graph; and determine a loop exist if the data packet is able to traverse back to the node representing the SDN controller.

According to an embodiment of the third aspect of the disclosure, the step to analyse the information in the packet-out message comprises the detection module to: determine whether an action in the packet-out message is an indication to output to the first forwarding table of a network element; retrieve the forwarding graph associated to the network element

indicated in the packet-out message in response to determining the action in the packet-out message is to output to a first forwarding table; traverse nodes from a node representing the first forwarding table of the network element along a unidirectional edge of the flow in the forwarding graph; and determine a loop exist if the data packet is able to traverse back to a node representing the SDN controller node.

Brief Description of the Drawings

The above advantages and features in accordance with this invention are described in the following detailed description and are shown in the following drawings:

Figure 1 illustrating the architecture of the Software-Defined Networking (SDN);

Figure 2 illustrating an example of a process based on the OpenFlow protocol;

Figure 3 illustrating a system for detecting a loop between the control plane and the data plane in a Software Defined Network in accordance with an embodiment of this disclosure;

Figure 4 illustrating an exemplary processing system in a server computer representing the processing systems in the SDN controller that execute instructions to perform the processes in accordance with embodiments of this disclosure;

Figure 5 illustrating a timing diagram for detecting a loop between the control plane and data plane in a SDN system in accordance with an embodiment of this disclosure;

Figure 6 illustrating a detection process performed by the SDN controller to detect a loop between the control plane and data plane in a SDN system in accordance with an embodiment of this disclosure;

Figure 7 illustrating an embodiment of a system architecture for detecting a loop between the control plane and data plane in a SDN system in accordance with an embodiment of this disclosure;

Figure 8 illustrating an example of a forwarding graph;

Figure 9 illustrating a simple network topology between the SDN controller and two switches in accordance with an embodiment of this disclosure;

Figure 10 illustrating the forwarding graph being updated in accordance with an embodiment of this disclosure;

Figure 11A illustrating a loop involving four nodes;

Figure 11B illustrating a loop involving two nodes; and

Figure 12 illustrating a timing diagram of detecting a loop between the control plane and data plane in a SDN system via the detection system shown in figure 7.

Detailed Description

This disclosure relates to a method and system for detecting loop between control plane and data plane in a Software Defined Network (SDN) system. Particularly, this disclosure relates to a method and system for monitoring messages from the SDN controller and detecting loop based on the monitored messages.

The present disclosure provides a method and system for detecting a loop arising between the control plane and the data plane in real time in an SDN scenario and for resolving the problem. The present disclosure supports loop detection in a single switch and across switches, and packet-out messages sent by controllers are also detected. In addition, the present disclosure also provides a solution for resolving loop in real time.

The present disclosure involves two processes, namely, loop detection on a switch and loop resolution. We shall first discuss the OpenFlow protocol that is relevant to the method and system according to the present disclosure.

According to the OpenFlow protocol, the control layer and the data layer use packet-in, packet-out, and flow-mod messages for interaction and switches use flow tables and group tables to process data packets. When a data packet arrives at an ingress switch, the switch processes the data packet according to a corresponding entry in a flow table. If the data packet is a new flow and the switch does not have a regular corresponding entry, a table-miss is triggered. A flow table having a special entry named table-miss entry defines how to process the packets unmatched by regular entries. If the action of the table-miss entry is output:controller, the switch sends a packet-in message to a controller to report the new flow. Based on the information in the packet-in message such as source and destination addresses of the flow, the controller queries a forwarding path of the flow, sends a packet-out message to a current switch, to inform the current switch of a port (which could be a normal output port or a special port OFPP_TABLE for sending the data packet to the first forwarding table) to which the data packet is forwarded, and adds the table entries to all switches on the path through flow-mod messages so that the ingress switch and all the switches on the path can process subsequent data packets by using the table entries without triggering table-miss any more. It is useful to note that a packet-in message is not triggered for table-miss only, and may also be triggered due to output:controller actions in the regular

flow table entries or group table entries or due to another manner specified in the OpenFlow protocol. It is useful to note that non-ingress switches may have table entries with output:controller actions as well.

When required to execute multiple instructions for one flow, a switch may use flow tables of multiple levels, namely, a pipeline. Figure 2 illustrates a pipeline processing procedure. The data packet enters from the ingress port and is processed according to the flow tables in turn until the last flow table no longer points to another flow table. To enable the data packet to be processed by all flow tables on the switch, the OpenFlow specification introduces a table reserved port, used for a packet-out message so that the switch can process, from the first flow table in pipeline, the data packet that triggers a packet-in message.

Figure 3 illustrates a system 300 for detecting a loop between the control plane 301 and the data plane 302 in a Software Defined Network having a SDN controller 305 being connectable to network elements in the data plane 302.

The network elements 310 are typically switches. The SDN controller 310 may be a computing system such as a server computer or a virtual machine running on a server computer.

Figure 4 illustrates an example of a processing system 400 in a server computer representing the processing systems in the SDN controller 310 that execute instructions to perform the processes described below in accordance with embodiments of this disclosure. One skilled in the art will recognize that the instructions may be stored and/or performed as hardware, firmware, or software without departing from this invention. Further, one skilled in the art will recognize that the exact configuration of each processing system may be different and the exact configuration of the processing system executing processes in accordance with this invention may vary and processing system 400 shown in figure 4 is provided by way of example only.

Processing system 400 includes a processor 410, a display 420, a keypad 430, a memory 440, network device 450 and an I/O device 460.

The display 420, keypad 430, memory 440, network device 45 and I/O device 460 and any number of other peripheral devices connect to processor 410 to exchange data with processor 410 for use in applications being executed by processor 410.

The display 420 receives display data from processor 410 and display images on a screen for a user to see. The display 440 may be a liquid crystal display (LCD) or organic light-emitting diode (OLED) display. The keypad 420 receives user input and transmits the

input to processor 410. In some embodiments, the display 440 may be a touch sensitive surface that functions as a keypad to receive user input.

The memory 440 is a device that transmits and receives data to and from processor 410 for storing data to a memory. The data being stored include instructions for performing the processes as will be described below and the flow tables and group tables. The memory may comprise non-volatile memory such as a Read Only Memory (ROM) or volatile memory, such as Random Access Memory (RAM). Non-volatile memory stores instructions and data needed to operate various sub-systems of processing system 400. Volatile memory stores the instructions and data needed by processor 410 to perform the processes for providing a system in accordance with this invention. One skilled in the art will recognize that any non-transitory memory may be used and the exact type used is left as a design choice to those skilled in the art.

The network device 450 connects processor 410 to the SDN system for transmission of data to and from other network elements within the SDN system.

The processor 410 is a processor, microprocessor, or any combination of processors and microprocessors that execute instructions to perform the processes in accordance with the present disclosure. The processor has the capability to execute various application programs that are stored in the memory 440. These application programs can receive inputs from the user via the keypad 430.

Figure 5 illustrates a timing diagram of a process 500 for detecting a loop between the control plane 301 and data plane 302 a SDN system. Process 500 begins with step 515 where the SDN controller analyses the flow tables of the switches in the network it controls and creates a forwarding graph for the network. The forwarding graph associated to the network is stored in the memory.

A short brief regarding a forwarding graph will now be discussed. A forwarding graph is a directed graph that includes nodes and directed edges between nodes. There is one filter on each edge, and the filter is an intersection of matching domains of the two nodes. One node indicates one rule (i.e. table entry) on the switch and a directed edge indicates a next hop dependency between rules. As shown in figure 8, if the data packet matching domain belongs to the filter, node A 810 forwards data packets to node B 820 for processing after processing the data packets. Therefore, the sequence for processing data packets by flow entries, that is, a data packet forwarding path, can be indicated by creating a forwarding graph of all flow entries. A forwarding graph is a dependant graph of rules, which has been used in SDN to detect the violation of routing policy. Differently from existing forwarding

graphs, the forwarding graph of this disclosure also view the SDN controller as a node, and the nodes with output:controller action all have a direct edge to the controller node.

Figure 9 illustrates a simple network topology between the SDN controller 305 and two switches 810 and 820. The SDN controller 305 is a node the forwarding graph. The matching domain of the controller node is wildcard. As each edge in the forwarding graph can indicate the data processing logic, there is a directed edge 910 between the controller node and each node that can trigger a packet-in message (i.e. each node with output:controller action).

Matching domains of flow entries are expressed in binary. For simplicity, the domain length in all the examples is 8 bits long. In practice, the domain length is hundreds of bits. When matching domains of two flow entries have intersection and a physical link exists between switches at which the two flow entries are located, a dependency exists between nodes of the two flow entries, that is, a unidirectional edge exists. According to all flow entries, port IDs of the switch are uniquely expressed in ascending order. In the forwarding graph, flow entries are expressed as nodes and each node corresponds to a unique port ID. According to dependencies between nodes, directed edges are added in the forwarding graph. As shown in figure 9, node A outputs from port 2 to node B as node A has a direct filter being 10101xx1 to node B. In short, the nodes are representation of the flow tables of the network element and SDN controller. Forwarding graph is widely known and specific details are therefore omitted for brevity.

In step 520, the switch sends a packet-in message to the SDN controller. In step 525, in response to the packet-in message, the SDN controller generates and transmits a flow-mod message. In particular, as mentioned above, the SDN controller would, based on the information in the packet-in message such as source and destination addresses of the flow, queries a forwarding path of the flow, sends a packet-out message to a current switch, to inform the current switch of a port (either a normal output port or a special port OFPP_table) to which the data packet is forwarded, and adds the table entries to all switches on the path through flow-mod messages so that the ingress switch and all the switches on the path can process subsequent data packets by using the table entries without triggering table-miss any more. Prior to sending the flow-mod message to the network element 310, the SDN controller updates the forwarding graph associated to the switches on the path to process the data packets according to the information in the flow-mod message. In particular, the SDN controller reads the command in the flow entry of the flow-mod message which comprises add, modify and delete commands. The SDN controller revises the forwarding graph by amending, modifying or deleting the nodes and connections in the forwarding

graph according to the relevant commands in the flow entry. After updating the forwarding graph, the flow-mod message is released to the network element 310.

A packet-in message is a way for the switch to send a captured packet to the controller. Relevant to this disclosure, a packet-in is transmitted to the SDN controller when a table:miss is triggered by the switch due to inability to process a data packet. One skilled in the art will recognise that other reasons may trigger the switch to transmit a packet-in message to the SDN controller such as ttl error without departing from the disclosure.

The SDN controller sends a flow-mod to modify the state of a switch. Essentially, the flow-mod message contains updated flow entries in order for the switch to process the data packet in response to receiving a table:miss in a packet-in message.

Figure 10 illustrates updating of the forwarding graph in response to intercepting a flow-mod message. Assuming a data packet whose matching domain is 1010xx10 arrives at switch 1 1010 which does not have a flow entry for processing the data packet. Switch 1 would send a packet-in message to the SDN controller 305 to trigger table:miss. In such circumstances, a dotted line 1011 indicates the switch does not have flow entries for processing the data packet. In response, the SDN controller generates and transmits a flow-mod message containing updated flow entries in order for switch 1 to process the data packet. In this instance, the updated flow entries update the switches to process the data packet on the following path: switch 1 -> switch 2 -> switch 3. Therefore, switch 1 1010, switch 2 1020, and switch 3 1030 all deliver flow entries. In the forwarding graph, three nodes representing the three switches 1010, 1020 and 1030 connected with solid lines and dependencies between the nodes are added. When the next data packet arrives, the flow entry processing path is the part of solid lines. It is noted that each node in the forwarding graph is a representation of a forwarding table which contains information on where data packets are forwarded upon receiving the data packets. Further, one switch may have more than one forwarding table. Hence, the illustration as shown in figure 10 is based on the assumption that each switch has only one forwarding table.

Essentially, when flow table information is updated, forwarding entries are updated synchronously and a controller node is added to the forwarding graph. Accordingly, there is a directed edge from each node that can trigger a packet-in message to the controller node. When receiving a flow-mod message, the SDN controller adds or modifies nodes and dependencies between the nodes in the forwarding graph after the message passes inspection.

In step 530, the SDN controller 305 generates and transmits a packet-out message to the switch to inject the data packets. Prior to sending the packet-out message to the network element 310, the SDN controller detects whether a loop exists in the forwarding graph. If loop exists, the SDN controller raised an action ticket before releasing the packet-out message to the network element 310.

The SDN controller is able to inject data packets into the data plane of a particular switch via a packet-out message. The packet-in message can either carry a raw packet to inject into the switch, or indicate a local buffer on the switch containing a raw packet to release. Upon intercepting a packet-out message from the controller node, the detection system looks up the forwarding graph for the forwarding path of the data packet associated with the packet-out message, and checks if the data packet is able to traverse back to the controller node again. If so, loop is detected. Figure 11A illustrates a loop where the data packet is able to traverse back to the controller node with switch 3 1030 having a direct edge back to the controller node. Particularly, in this instance, a raw packet (10100010) is injected into switch 1 1010. The forwarding path would entail the raw packet to be traversed on the following path: controller 305 → switch 1 1010 → switch 2 1020 → switch 3 1030 → controller 305. A switch may have one or more forwarding tables. However, this illustration is based on the assumption that each switch has only one forwarding table. Hence, a loop between the control plane and data plane is detected. If loop is detected, the SDN controller reports the loop and resolves the loop in ways such as discarding the packet-out message and limiting the rate of the packet-in message. Figure 11A illustrates a loop between the control plane and the data plane involving the controller in the control plane and three switches in the data plane. In the simplest form, a loop between the control plane and the data plane can also involve the controller in the control plane and only one switch in the data plane as illustrated in figure 11B.

When monitoring the packet-out message, the detection system detects whether a loop between control plane and data plane exist.

Figure 6 illustrates a detection process 600 performed by the SDN controller to detect a loop between the control plane 301 and data plane 302 in a SDN system.

Process 600 begins with step 605 by monitoring and intercepting messages between the SDN controller and the switches. If the message is a flow-mod message, process 600 proceeds to step 610. If the message is a packet-out message, process 600 proceeds to step 615. The monitoring and intercepting method could be implemented by having the intermediate layer working as a proxy or hooking the TCP/IP stack in the controller.

In step 610, process 600 retrieves the forwarding graph associated to the switch and updates the forwarding graph according to the information in the flow-mod message. Thereafter, the updated forwarding graph is stored on the memory and the flow-mod message is released. Process 600 updates the forwarding graph in the following manner. Particularly, process 600 reads the command in the flow entry of the flow-mod message which comprises add, modify and delete commands. Process 600 then revises the forwarding graph by amending, modifying or deleting the nodes and connections in the forwarding graph according to the relevant commands in the flow entry. Differently from current practice, the forwarding graph is updated including the controller node. Details of updating a forwarding graph has been described above with reference to figure 10. In another embodiment, after updating forwarding graph in step 610, the detection process 600 may proceed to step 615 to analyse the updated forwarding graph to detect whether a loop exist in the data plane, i.e. data packets traverse within switch or switches in a loop. Particularly, process 600 traverses nodes starting from the node representing the first forwarding table along the unidirectional edge of the flow in the forwarding graph and checks whether the data packet is able to traverse back to the same node representing the first forwarding table to determine whether a loop exist. If a loop is detected, process 600 raises an action ticket. If a loop is not detected, process 600 releases the flow-mod message and repeats from step 605.

In step 615, process 600 analyses the information in the packet-out message and retrieves the forwarding graph associated to the switch indicated in the packet-out message to determine whether a loop exists. Particularly, process 600 checks whether the action in the packet-out message is output:table where the output:table action is an indication of data packet being caused to be outputted to the first forwarding table of a switch. If the action is output:table, process 600 verifies whether the packet-out message may cause the data packet to travel back to the controller. In particular, process 600 retrieves the forwarding graph associated to the network element indicated in the packet-out message and connects the node representing the controller node to the node representing the first forwarding table in the forwarding graph, the first forwarding table being the node processing the data packet in the destination switch indicated in the packet-out message, and the destination switch being the destination of the packet-out message. Process 600 then traverses nodes starting from the node representing the SDN controller along the unidirectional edge of the flow in the forwarding graph and checks whether the data packet is able to traverse back to the node representing the SDN controller again. If so, loop is detected and process 600 proceeds to step 620 and raises an action ticket to resolve the loop and releases the packet-out message. If a loop does not exist, process 600 releases the packet-out message and

repeats from step 605 to monitor and intercept messages. Details of analysing and determining whether a loop exists have been described above with reference to figures 11A and 11B.

In another embodiment, process 600 is not required to connect the node representing the SDN controller to the first node representing the first forwarding table in the forwarding graph. Instead, process 600 can detect whether a loop exist by traversing nodes starting from the node representing the first forwarding table along the unidirectional edge of the flow in the forwarding graph and checks whether the data packet is able to traverse to the node representing the SDN controller to determine whether a loop exist. Essentially, instead of having to first connect the controller node to the first node in the forwarding graph, process 600 can detect whether a loop exist by traversing from the network element and check whether the data packets traverse to the SDN controller node.

In step 620, an action ticket is raised to alert that a loop exists. Some possible ways of resolving the loop are discarding the packet-out message and limiting the rate of the packet-in message. The exact implementation of resolving the loops are left to the skilled in the art.

The detection process 600 may be provided as an intermediate layer between the control plane and the data plane, monitoring for certain messages being transmitted out from the SDN controller. The intermediate layer may be a processing system that is connected to the SDN controller. Alternatively, the detection process 600 may be provided by a virtual machine, running on the resources of the SDN controller, to monitor for certain messages being transmitted out from the SDN controller.

Figure 7 illustrates an embodiment of a system architecture 700 in accordance with the present disclosure. An intermediate layer 710 is established between the control plane 301 and the data plane 302. In the intermediate layer 710, a detection system 720 is provided. The detection system 720 may be provided by a processing system that is connected to the SDN controller or a virtual machine which is running on the resources of the SDN controller. Alternatively, the detection system 720 may be provided by adding an application to the SDN controller.

The detection system 720 includes a detection module 721 and a forwarding graph module 722. The detection module 721 monitors for packet-out and flow-mod messages between the SDN controller 305 and network elements 310. One method of monitoring and intercepting messages between the SDN controller and network elements 310 is having the detection module 721 being implemented as a proxy or hooking the TCP/IP stack in the SDN

controller to monitoring and intercept the messages. By monitoring packet-out and flow-mod messages sent by the SDN controller 305 to the network elements 310, the detection module 721 updates stored flow table information in real time in the forwarding graph module 722. According to dependencies between stored flow tables, a forwarding graph is created by the forwarding graph module 722. One skilled in the art will recognise that the detection system 720 as described includes the two modules to illustrate that the processes executable by the detection system 720 can be divided out using two modules. In fact, the processes executable by the detection system 720 may not require separate modules. Furthermore, more than 2 modules may be implemented without departing from the present disclosure.

Figure 12 illustrates a timing diagram of a process 1200 of detecting a loop between the control plane 301 and data plane 302 in a SDN system via a detection system 720 having a detection module 721 and a forwarding graph module 722. Process 1200 begins with step 1230 where the SDN controller 305 sends a flow-mod message to the network element 310. In step 1235, the detection module 721 intercepts the flow-mod message and forwards the relevant information in the flow-mod message to the forwarding graph module 722. In particular, the commands in the flow entry of the flow-mod message are forwarded to the forwarding graph module 722. In step 1240, the forwarding graph module 722 retrieves the forwarding graph associated to the network element and updates the forwarding graph according the information in the flow-mod message. Particularly, the forwarding graph module revises the forwarding graph by amending, modifying or deleting nodes and connections in the forwarding graph according to the relevant commands received from the detection module 721. If a forwarding graph has not been created for the network element involved, a new forwarding graph is created and stored on the memory. After the forwarding graph has been updated or created by the forwarding graph module 722, the flow-mod message is released by the detection module 721 in step 1245.

In another embodiment, after updating forwarding graph, the forwarding graph module 722 forwards the updated forwarding graph to the detection module 721. In response to receiving the updated forwarding graph, the detection module 721 analyses the updated forwarding graph to detect whether a loop exist in the data plane, i.e. data packets traverse within switch or switches in a loop. Particularly, the detection module 721 traverses nodes starting from the node representing the first forwarding table along the unidirectional edge of the flow in the forwarding graph and checks whether the data packet is able to traverse back to the same node to determine whether a loop exist. If a loop is detected, the

detection module 721 raises an action ticket. If a loop is not detected, the detection module 721 releases the flow-mod message.

In step 1250, the SDN controller sends a packet-out message to the network element 310. In step 1255, the detection module 721 intercepts the packet-out message and checks whether the action is output:table, wherein the output:table action is an indication of data packet being caused to be outputted to the first forwarding table of a network element. If the action is output:table, the detection module 721 retrieves the forwarding graph associated to the network element.

In step 1250, the detection module 721 analyses the information in the packet-out message to determine whether a loop exists. Particularly, the detection module 721 connects the node representing the SDN controller to the first node in the forwarding graph, the first node in the forwarding graph being the first forwarding table. The detection module 721 then traverses nodes starting from the node representing the SDN controller along the unidirectional edge of the flow in the forwarding graph and checks whether the data packet is able to traverse back to the node representing the SDN controller again. If so, loop is detected and the detection module 721 raises an action ticket to resolve the loop and releases the packet-out message in step 1265 and process 1200 ends. In one embodiment, the packet-out message is released only after the loop is resolved. If a loop does not exist, the detection module 721 releases the packet-out message in step 1265 and process 1200 ends.

In another embodiment, the detection module 721 is not required to connect the node representing the SDN controller to the first node in the forwarding graph. Instead, the detection module 721 traverses nodes starting from the first node in the forwarding graph along the unidirectional edge of the flow in the forwarding graph and checks whether the data packet is able to traverse to the node representing the SDN controller node to determine whether a loop exist. Essentially, instead of having to first connect the controller node to the first node in the forwarding graph, the detection module 721 can detect whether a loop exist by traversing from the first node and check whether the data packets traverse to the controller node.

The above is a description of embodiments of a method and system for detecting loop between the data plane and the control plane of a SDN system. It is foreseeable that those skilled in the art can and will design alternative method and system based on this disclosure that infringe upon this invention as set forth in the following claims.

CLAIMS:

1. A method for detecting a loop between a control plane and a data plane in a Software Defined Network (SDN) having a SDN controller being connectable to a plurality of network elements in the data plane comprising:

monitoring and intercepting a flow-mod message and a packet-out message from the SDN controller to the network element;

updating a forwarding graph according to flow-mod message in response to intercepting the flow-mod message;

analysing the information in the packet-out message against the forwarding graph associated to the network element to determine whether a loop exists in response to intercepting the packet-out message; and

raising an action ticket in response to determining a loop exists.

2. The method according to claim 1 wherein the step of updating the forwarding graph comprises:

retrieving a forwarding graph associated to the network element indicated in the flow-mod message;

updating the forwarding graph according to the information in the flow-mod message; and

storing the updated forwarding graph.

3. The method according to claim 2 wherein the step of updating the forwarding graph further comprises:

reading flow entry command in the flow-mod message; and

revising the forwarding graph by amending, modifying or deleting nodes and connection between the nodes in the forwarding graph based on the flow entry command, the nodes being a representation of the forwarding tables of the network elements or the SDN controller, wherein the flow entry command comprises adding, modifying and deleting commands.

4. The method according to claim 3 wherein the step of updating the forwarding graph further comprises:

analysing the updated forwarding graph to detect whether a loop exist in the data plane;

raising an action ticket in response to determining a loop exist; and

releasing the flow-mod message in response to determining a loop does not exist.

5. The method according to claim 4 wherein the step of analysing the updated forwarding graph to detect whether a loop exist in the data plane comprises:

traversing nodes from a first node in the updated forwarding graph along a unidirectional edge of the flow in the updated forwarding graph; and

determining a loop exist if the data packet is able to traverse back to the first node, wherein the first node is a representation of a first forwarding table of the network element.

6. The method according to any one of claims 1-5 wherein the step of analysing the information in the packet-out message comprises:

determining whether an action in the packet-out message is an indication to output to a first forwarding table of the network element;

retrieving the forwarding graph associated to the network element indicated in the packet-out message in response to determining the action in the packet-out message is to output to the first forwarding table;

traversing nodes from a node representing the first forwarding table of the network element along a unidirectional edge of the flow in the forwarding graph; and

determining a loop exist if the data packet is able to traverse back to a node representing the SDN controller.

7. The method according to any one of claims 1-5 wherein the step of analysing the information in the packet-out message comprises:

determining whether an action in the packet-out message is an indication to output to a first forwarding table of the network element;

retrieving the forwarding graph associated to the network element indicated in the packet-out message in response to determining the action in the packet-out message is to output to the first forwarding table;

connecting a node representing the SDN controller to a first node representing the first forwarding table of the network element;

traversing nodes from the node representing the SDN controller along a unidirectional edge of the flow in the forwarding graph; and

determining a loop exist if the data packet is able to traverse back to the first node.

8. A detection system for detecting a loop between a control plane and a data plane in a Software Defined Network (SDN) having a SDN controller being connectable to a plurality of network elements in the data plane comprising:

a processor, a memory, and instructions stored on the memory and executable by the processor to:

monitor and intercept a flow-mod message and a packet-out message from the SDN controller to the network element;

update a forwarding graph according to flow-mod message in response to intercepting the flow-mod message;

analyse the information in the packet-out message against the forwarding graph associated to the network element to determine whether a loop exists in response to intercepting the packet-out message; and

raise an action ticket in response to determining a loop exists.

9. The detection system according to claim 8 wherein the instruction to update the forwarding graph comprises instructions to:

retrieve a forwarding graph associated to the network element indicated in the flow-mod message;

update the forwarding graph according to the information in the flow-mod message; and

store the updated forwarding graph.

10. The detection system according to claim 9 wherein the instruction to update the forwarding graph further comprises instructions to:

read flow entry command in the flow-mod message; and

revise the forwarding graph by adding, modifying or deleting nodes and connection between the nodes in the forwarding graph based on the flow entry command, the nodes being a representation of the forwarding tables of the network element or the SDN controller, wherein the flow entry command comprises adding, modifying and deleting commands.

11. The detection system according to claim 10 wherein the instruction to update the forwarding graph further comprises instructions to:

analyse the updated forwarding graph to detect whether a loop exist in the data plane;

raise an action ticket in response to determining a loop exist; and

release the flow-mod message in response to determining a loop does not exist.

12. The detection system according to claim 11 wherein the instruction to analyse the updated forwarding graph to detect whether a loop exist in the data plane comprises instructions to:

traverse nodes from a first node in the updated forwarding graph along a unidirectional edge of the flow in the updated forwarding graph; and

determine a loop exist if the data packet is able to traverse back to the first node , wherein the first node is a representation of a first forwarding table of the network element.

13. The detection system according to any one of claims 8-12 wherein the instruction to analyse the information in the packet-out message comprises instructions to:

determine whether an action in the packet-out message is an indication to output to a first forwarding table of the network element;

retrieve the forwarding graph associated to the network element indicated in the packet-out message in response to determining the action in the packet-out message is to

output to a first forwarding table;

traverse nodes from a node representing the first forwarding table of the network element along a unidirectional edge of the flow in the forwarding graph; and

determine a loop exist if the data packet is able to traverse back to the a node representing the SDN controller.

14. The detection system according to any one of claims 8-12 wherein the instruction to analyse the information in the packet-out message comprises instructions to:

determine whether an action in the packet-out message is an indication to output to a first forwarding table of the network element;

retrieve the forwarding graph associated to the network element indicated in the packet-out message in response to determining the action in the packet-out message is to output to a first forwarding table;

connect a node representing the SDN controller to a first node representing the first forwarding table of the network element;

traverse nodes from the node representing the SDN controller along a unidirectional edge of the flow in the forwarding graph; and

determine a loop exist if the data packet is able to traverse back to the first node.

15. A detection system for detecting a loop between a control plane and a data plane in a Software Defined Network (SDN) having a SDN controller being connectable to a plurality of network elements in the data plane comprising:

a detection module configured to:

monitor and intercept a flow-mod message and a packet-out message from the SDN controller to the network element;

forward information in the flow-mod message to the forwarding graph module in response to intercepting the flow-mod message;

analyse the information in the packet-out message against a forwarding graph associated to the network element to determine whether a loop exists in response to intercepting the packet-out message;

raise an action ticket in response to determining a loop exists; and

a forwarding graph module configured to:

update a forwarding graph according to the information from the detection module.

16. The detection system according to claim 15 wherein the forwarding graph module is configured to:

receive the information from the detection module;

retrieve a forwarding graph associated to the network element indicated in the information;

update the forwarding graph according to the information; and

store the updated forwarding graph.

17. The detection system according to claim 16 wherein the forwarding graph module is configured to:

create a new forwarding graph for the network element indicated in the information if a forwarding graph not available.

18. The detection system according to claim 16 wherein the step to update the forwarding graph further comprises the forwarding graph module to:

read flow entry command in the flow-mod message;

revising the forwarding graph by adding, modifying or deleting nodes and connection between the nodes in the forwarding graph based on the flow entries, the nodes being a representation of the forwarding tables of the network elements or the SDN controller, wherein the flow entry command comprises adding, modifying and deleting commands.

19. The detection system according to claim 18 wherein the step to update the forwarding graph further comprises the forwarding graph module to:

analyse the updated forwarding graph to detect whether a loop exist in the data plane;

raise an action ticket in response to determining a loop exist; and

release the flow-mod message in response to determining a loop does not exist.

20. The detection system according to claim 19 wherein the step to analyse the updated forwarding graph to detect whether a loop exist in the data plane comprises the detection module to:

traverse nodes from a first node in the updated forwarding graph along a unidirectional edge of the flow in the forwarding graph; and

determine a loop exist if the data packet is able to traverse back to the first node, wherein the first node is a representation of a first forwarding table of the network element.