An OpenFlow-based Dynamic Service Chaining Approach for Hybrid Network Functions Virtualization

ChiSheng Su^{a,*}, Fu-Min Chang^b, Shang-Juh Kao^a

^aNational Chung Hsing University, No.250, Guoguang Rd., South Dist., Taichung City 402, Taiwan

^bChaoyang University of Technology, Taichung, 413, Taiwan

*Corresponding Author: cssu@csmail.nchu.edu.tw

Abstract

Network Functions Virtualization (NFV) technology enables network services (performed virtually). One or more virtualized network functions (VNFs) together with the traditional middle boxes can be established a complex enterprise network. To provide an efficient NFV environment, we are facing the challenges of routing path decision, service ordering, and the hybrid communications between switches and middle boxes. In this paper, we propose a framework for building an OpenFlow-based dynamic services chaining and present a routing path for a hybrid NFV environment.

In order to control legacy switches in hybrid NFV environment, we use simple network management protocol (SNMP) to manage Forwarding Information Base (FIB) on legacy switches. The reserved fake MAC-address is being substituted for a destination MAC-address, and then the packets are forwarded to the policy designating port. To solve the problem of middle boxes being unable to modify the packet's fields, we adopt MAC-address based forwarding method and Multiprotocol Label Switching (MPLS) protocol into the system to handle routing paths. The proposed architecture and simulation environment are reported. The performance evaluation of the system under different cases is also presented and discussed.

Keywords: NFV, OpenFlow, MPLS, SNMP, dynamic service chaining.

1. Introduction

Software Defined Networking (SDN) architecture is a promising approach for programmable networks due to its novel features of decoupling the control plane and data plane

of a network, such that the switches can be absorbed in packet forwarding. It is logically centralized software-based controllers. Several approaches were proposed to archive the network virtualization, OpenFlow⁽¹⁾⁻⁽³⁾ has been approved as a way for researchers to build their experiments. It standardizes the communication between switches and controllers and provide a series version of OpenFlow specifications⁽³⁾⁻⁽⁷⁾. The OpenFlow controller manages one or more OpenFlow switches through a secure channel to setup flow tables.

Network Functions Virtualization (NFV) technology⁽⁸⁾⁻⁽⁹⁾ enables the network services that once require dedicated hardware virtually by the software and virtualization technology. NFV framework consists of three components, virtualized network functions (VNFs), network function virtualization infrastructure (NFVI), and network functions virtualization management and orchestration architectural framework (NFV-MANO Architectural Framework). VNFs are virtualized software implementations of network functions that can be deployed on a general-purpose computer. For example, NAT, firewalls, IDS/IPS, DNS, and proxy can be implemented as x86 programs running on virtual machines. NFVI is the environment for VNFs. NFV-MANO Architectural Framework is the collection of network functions, data repositories and reference points and exchange information for the purposes of managing and orchestrating NFVI and VNFs.

Actually, NFV does not need to deploy on SDN environment, it can be deployed standalone in the existing legacy network environment. It means that one or more virtualized network functions (VNFs) together with the traditional middle boxes can be promptly established for a complex enterprise network. Under this situation, one of challenges is how to provide the routing path decision⁽¹⁰⁾⁻⁽¹¹⁾, service ordering, and the communications between switches

and middle boxes. In this paper, we propose a framework for building an OpenFlow-based dynamic services chaining mechanism and present a routing path management for a hybrid NFV environment.

In a pure OpenFlow environment, the policy-based routing (PBR) decision system⁽¹²⁾ is easier to reach the goal because it is logically centralized control with uniform protocols. We provide more elastic and flexible policy control by managing the ingress and egress rules separately using controller. On the other hand, in hybrid NFV environment, we define a policy unit in the policy-based routing (PBR) decision system by classifying the network packets according to their IP address, MAC address, and UDP/TCP ports. Each policy, a pair of IP address and TCP port, assigns a specific routing path, which forms services chains in a mapping table. In order to control and manage legacy switches, SNMP4SDN⁽²⁴⁾, a simple network management protocol (SNMP) tool, is used to modify Forwarding Information Base (FIB). A reserved fake MAC-address replace the original destination MAC-address, such that the policy manager manages legacy switches according to the fake MAC-address mapping table. However, middle boxes are not allowing changing packet's fields, the general solution is putting the middle box behind OpenFlow enabled switch. In this paper, we assume that we cannot modify the physical topology. Based on the assumption, we adopt Multiprotocol Label Switching (MPLS) protocol into the system to handle routing paths. As a control group, tunneling technology is also involved in the experience.

The rest of this paper proceeds as follows. In section 2, we describe the problem and solution of pure OpenFlow architecture and hybrid architecture, respectively. After that, we present the performance evaluation in section 3. Finally, we draw a conclusion.

2. NFV Architecture

In the environment of network functions virtualization, the operations of both simple and hybrid OpenFlow architectures will be exemplified in this section. The following discussions are based on the system with OpenFlow version 1.0 to 1.3, OpenDaylight Hydrogen controller⁽¹³⁾, and SNMP4SDN module being applied.

2.1 Simple OpenFlow architecture

With simple OpenFlow architecture, the system is equipped with OpenFlow-enabled switches and all

Fig. 1. Simple OpenFlow Architecture

communications among the controller, middle boxes, and switches are following the OpenFlow protocol. Fig. 1 shows a simple OpenFlow architecture for our presentation example. In the architecture, a controller connects five OpenFlow switches. Three middle boxes, a firewall, a NAT server, and a proxy server, are connected directly to three switches separately, while two hosts are connected to another two switches. The host01 has a link with switch01 and is able to connect to Internet directly using a public IP address, and the host02 is assumed to be in a private LAN with requiring

Table 1. Service order from host to Internet.

Host	Routing Station Order
Host01	Host01 → Firewall → Proxy → Internet
Host02	Host02 → Firewall → Proxy → NAT → Internet

Table 2. Service order from Internet to host.

Host	Routing Station Order
Host01	Internet → Firewall → Host01
Host02	Internet → Firewall → NAT → Host02

NAT service for Internet access.

Assume that the service ordering from hosts to the Internet and from the Internet to hosts are listed as Table 1 and Table 2 respectively. Table 1 shows the routing order of each host before it reaches the Internet, and Table 2 lists the orders for the requesting services from the Internet. All packets from and to the host02 are passed through the NAT server. The controller plays the role of controlling all OpenFlow switches and acts as a policy decision manager. During the communications, the information of IP address, MAC addresses, and TCP/UDP ports are monitored by the controller for the purpose of packet forwarding.

As indicated in Table 1, when the Host01 has a request for http service, it must first pass the Firewall and visit the Proxy server. Similarly, the Host02 needs to visit the Firewall,

the Proxy server, and the NAT server in sequence. By equaling the weight among the switches, the controller then can easily apply the Dijkstra's algorithm⁽¹⁴⁾ to determine the path of shortest hop count. Accordingly, we got two ordered lists which represents two routing paths: {Host01, Switch01, Switch03, Firewall, Switch03, Switch05, Internet} and {Host02, Switch02, Switch03, Firewall, Switch03, Switch05, NAT, Switch05, Internet} for http's requests from Host01 and Host02 respectively. Due to all switches are OpenFlow-enabled, all packets are forwarded without headers being modified. In simple OpenFlow architecture, we only need to set up flow rules, matching fields, and forwarding ports on each OpenFlow switch. Conversely, similar procedures are performed for a request from the Internet. That is, when a request is issued from the Internet, as shown in Table 2, the shortest path of hop count is determined through inspecting the packet's headers by the controller.

2.2 Hybrid OpenFlow architecture

The Hybrid OpenFlow architecture consists of a group of OpenFlow switches and L2/L3 legacy switches, in addition to the controller and all necessary middle boxes. Several dynamic routing approaches are possible⁽¹⁸⁻²³⁾ for the hybrid architecture. Multiple Protocol Label Switching⁽²²⁾ (MPLS) is one of the approaches. MPLS could be efficient, but it is not supported in OpenFlow 1.0⁽⁴⁾. VLAN tag⁽²⁶⁾ use is another approach, which is available in OpenFlow 1.1.0⁽⁵⁾ with a tag limitation of 4094. When the network is already partitioned as pre-defined VLAN, any additional routing policy will become difficult. Hence, using VLAN tag approach may lack of dynamicity. In this paper, we propose a method by applying substitution over the packet headers. The header substitution could be easy to implement on common legacy switches. There could be two types of substitutions: either with the virtualized network functions (VNFs) or with the middle boxes.

Fig. 2 is type-1 architecture related to substituting packet headers method with VNFs. The basic ideal is modify the destination MAC address, one of FIB object⁽¹⁶⁾, as a label for routing but it must be recovered before entering any VNFs and target. For this reason, we suggest put an OpenFlow

Fig. 3. Hybrid OpenFlow Architecture (type-1)

Fig. 2. Hybrid OpenFlow Architecture (type-2)

switch in front of the VNFs. It is easy to set up a software implemented OpenFlow switch, Open vSwitch⁽¹⁷⁾ and CPqD⁽¹⁸⁾ ofsoftswitch13 are both ease of use. The type-1 architecture can be solved by using the OpenFlow embedded NVF appliance.

Fig. 3 is type-2 architecture represented to substituting packet headers method with legacy middle boxes. Due to several reasons, enterprise may not change the switch on the top of middle boxes. In this case, our approach is using MPLS via `mplsInterfaceInPackets` and `mplsInterfaceInDiscards` for dynamic routing. Similar to type-1, the additional labels must pop out before entering middle boxes. We give a summary of type-1 and type-2 hybrid OpenFlow architecture here. The critical points of hybrid OpenFlow architecture are the ingress and egress switches of hosts, VNFs, middle boxes, and entire network. There are several established techniques for the intra connections between legacy switches but limit at particular devices and inelastic of deployment.

The hybrid OpenFlow environment is very hard to generate rules. We develop a policy rule generator on the controller to prevent policy collision and loop. Fig. 4 shows that the policy generator working flow. It accept a pair of nodes, source node and destination node at first, next it will calculate and generate an ordered node list for representing a path with ordered nodes. The generator checks each node's type and generates rules for node, and then deploy the rules to system. With the policy rule generate, we can manage the

Fig. 4. Policy Rule Generator Flow Chart

system efficiently and safely

3. Feasibility and Performance Evaluation

Fig. 5. OpenFlow simulation environment

In previous section, we describe how dynamic routing deployment works by modifying packet headers and MPLS via management information base (MIB) objects. A path consists of multiple switches, in order to manage the routing paths dynamically, we adopt SNMP to control legacy switch in hybrid SDN environment. The policy decision system determines the forwarding rule on each switch, and then deploys flow rule on OpenFlow switches and forwarding table or MPLS label rule on legacy switches, respectively. There are also other approaches reaching this goal, the most common method is tunneling. Tunneling technologies provide good isolation of network. It can provide service for those unsupported or managed network devices between server and client. The main limitation of tunneling technologies is lack of performance. The remainder of this section, we give a report of those methods' performance measure and take it as a reference for deploying NFV environment.

In the following experiments, we simplify the processing of packets pass in and out of VNFs. In order to simulate the behavior of packets pass in and out of switches, we set a series of flow rules for controlling routing path. We use the method of modifying packet headers as an example as Fig. 5 shows. In the test 1, we use *ping* command to test network latency. In the test 2, the request generator triggers a *scp* command to copy a 500M file from Host02 to Host01 to get elapsed time between Host01 and Host02. When the packet goes through each switches, the OpenFlow switches will match, modify, and forward according to the flow rules, and record the elapsed time as a performance measure. The experiments are all in the same machine with FX(tm)-6100 Six-Core Processor and RAM 8G. We use namespace as hosts, Open vSwitch as OpenFlow switch, and OpenDaylight Hydrogen as controller. The performance evaluation results in Table 3 and 4 show that the L2 normal routing behavior is fastest and very closed to other approaches. Our approach achieves the dynamic routing policy feature certainly, even

though they are a little slower than L2 normal routing behavior.

4. Conclusions

Table 3. Ping Latency

Table 4. Scp elapsed time

In this paper, we present an architecture for OpenFlow-based dynamic service chaining by collaborating legacy switches in a hybrid NFV environment. This architecture includes two main mechanisms: OpenFlow-based NFV forwarding rule mechanism and controller-based policy decision mechanism. By integrating the SNMP4SDN module into the OpenFlow controller, we are able to efficiently manage the legacy switches in hybrid NFV environment. Particularly, we adopt forwarding information base (FIB) for switch management. We use the management objects, such as *ipForwardTable*, *mplsInterfaceInPackets* and *mplsInterfaceInDiscards*, to determine the dynamic forwarding rules in routing path table accordingly.

The simple performance test results prove that the system is effective and feasible. According to the experimental data, each approaches have similar performance and close to L2 switch normal routing performance. Our approach is to make some simple, general mechanisms, and combines them to achieve the dynamic routing policy feature in both simple and hybrid OpenFlow architectures.

References

- (1) McKeown, Nick, et al. : “OpenFlow: enabling innovation in campus networks”, ACM SIGCOMM Computer Communication Review 38.2, pp. 69-74, 2008
- (2) Matias, Jon, et al. : “Implementing layer 2 network virtualization using OpenFlow: Challenges and solutions”, Software Defined Networking (EWSDN), European Workshop, IEEE, pp. 30-35, 2012
- (3) “Open Networking Foundation”, <https://www.opennetworking.org/>, [Online; accessed 04-Dec-2015], 2015
- (4) “OpenFlow Switch Specification, Version 1.0.0 (Wire Protocol 0x01)”, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf>, [Online; accessed 04-Dec-2015], 2009
- (5) “OpenFlow Switch Specification, Version 1.1.0 Implemented (Wire Protocol 0x02)”, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf>, [Online; accessed 04-Dec-2015], 2011
- (6) “OpenFlow Switch Specification, Version 1.2 (Wire Protocol 0x03)”, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.2.pdf>, [Online; accessed 04-Dec-2015], 2011
- (7) “OpenFlow Switch Specification, Version 1.3.3 (Wire Protocol 0x04)”, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.3.pdf>, [Online; accessed 04-Dec-2015], 2013
- (8) Batalle, Josep, et al. : “On the implementation of NFV over an OpenFlow infrastructure: Routing Function Virtualization”, Future Networks and Services (SDN4FNS), 2013 IEEE SDN for. IEEE, pp. 1-6, 2013
- (9) Masutani, Hitoshi, et al. : “Requirements and design of flexible NFV network infrastructure node leveraging SDN/OpenFlow”, Optical Network Design and Modeling, 2014 International Conference on IEEE, pp. 258-263, 2014
- (10) Szabo, Robert, et al. : “Elastic network functions: opportunities and challenges”, Network, IEEE 29.3, pp. 15-21, 2015
- (11) Haleplidis, Evangelos, et al. : “Towards a network abstraction model for SDN”, Journal of Network and Systems Management 23.2, pp. 309-327, 2014
- (12) Braun, Hans-Werner : “Models of policy based routing”, 1989
- (13) “OpenDaylight Platform”, [Online; accessed 04-Dec-2015], 2015
- (14) Skiena S : “Dijkstra's Algorithm”, Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley, pp. 225-227, 1990
- (15) Kempf, James, et al. : “OpenFlow MPLS and the open source label switched router”, Proceedings of the 23rd International Teletraffic Congress. International Teletraffic Congress, pp. 8-1, 2011
- (16) McCloghrie, Keith, and Marshall T. Rose : “Management Information Base for network management of TCP/IP-based internets”, No. RFC 1156, 1990
- (17) “Open vSwitch Features”, <http://www.openvswitch.org/>, [Online; accessed 04-Dec-2015], 2015
- (18) Y. Rekhter, B. Davie, Rosen Eric, G. Swallow, D. Farinacci, D. Katz : “Tag switching architecture overview”, Proceedings of the IEEE , vol.85, no.12, pp. 1973-1983, Dec 1997
- (19) “RouteFlow project”, <http://cpqd.github.io/RouteFlow/>, [Online; accessed 04-Dec-2015], 2015
- (20) Lara, Adrian, Anisha Kolasani, and Byrav Ramamurthy : “Network innovation using openflow: A survey”, Communications Surveys & Tutorials, IEEE, pp. 493-512, 2014
- (21) Feng, Tao, Jun Bi, and Hongyu Hu : “OpenRouter: OpenFlow extension and implementation based on a commercial router”, Network Protocols (ICNP), 2011 19th IEEE International Conference, IEEE, pp. 141-142 2011
- (22) E. Rosen, A. Viswanathan, R. Callon : “Multiprotocol Label Switching Architecture”, No. RFC 3031, Jan. 2001
- (23) El Ferkouss, Omar, et al : “On the flexibility of MPLS applications over an OpenFlow-enabled network”, Global Telecommunications Conference (GLOBECOM 2011), IEEE, pp. 1-6, 2011
- (24) Matias, Jon, et al. : “An OpenFlow based network virtualization framework for the cloud”, Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference, pp. 672-678, 2011

- (25) “SNMP4SDN: Architecture and Design”,
<https://wiki.opendaylight.org/view/SNMP4SDN:Main>,
[Online; accessed 04-Dec-2015], 2015
- (26) T. Otsuka, M. Koibuchi, Akiya Jouraku, H. Amano :
“VLAN-based minimal paths in PC cluster with
Ethernet on mesh and torus”, ICPP 2005. International
Conference, pp. 567-576, 14-17 June 2005