논문 19-44-10-22 The Journal of Korean Institute of Communications and Information Sciences '19-10 Vol.44 No.10
https://doi.org/10.7840/kics.2019.44.10.1949

SFC 테스트를 위한 에이전트 기반 메커니즘

응웬쭝당찐◆, 유 명 식°

An Agent-Based Mechanism for Testing SFC

Trinh Nguyen◆, Myungsik Yoo°

요 약

네트워크 기능 가상화 (NFV)는 사용자에게 네트워크 서비스를 제공하는 새로운 방법이다. 전용 하드웨어와 달리 NFV 네트워크의 네트워크 기능은 일반 서버에서 실행되며 서비스 기능 체인 (SFC)은 가상 네트워크 기능 (VNF)을 함께 묶어 네트워크 서비스를 실행한다. SFC의 신뢰성과 품질을 보장하기 위해서는 실제 시스템에 배치하기 전에 성능 평가가 필요하다. 기존 방식들에서는 SFC 상호 운용성에 대한 내용이 다루어지지 않았다. 따라서 이 논문에서 Agent 기반 SFC 상호 운용성 테스트 방식을 제안한다.

Key Words : NFV, SFC, Testing, Agent-based, Interoperability

ABSTRACT

Network function virtualization (NFV) is a new way to provide network services to customers. Different from dedicated hardware, the network functions of an NFV network are executed on general servers, and service function chaining (SFC) chains virtual network functions (VNFs) to work together to form a network service. To ensure the reliability and dependability of the SFCs, there is a need for performance evaluation and functionality correctness of SFCs before deploying it in production. The conventional approaches do not focus on the SFC interoperability tests. Hence, this article proposes an agent-based mechanism for testing SFC interoperability.

Ⅰ. Introduction

NFV opens a new era of network service provision by leveraging virtualization technologies. ETSI[1] introduced the NFV architectural framework so that instead of using fixed hardware resources to deploy network services, software is used to implement the functionalities that will be installed in commodity hardwares. That way, services will be easy for operators to deploy, adjust, or even adding more complex features. In addition, by releasing the need to buy dedicated and expensive hardwares, network operators now only need to use normal servers and install one or more VNFs on top of them which significantly reduces the cost. Moreover, more advanced SFCs can be formed by chaining different kinds of VNFs because of the flexibility nature of software. This helps service providers to

※ This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2018-2017-0-01633) supervised by the IITP(Institute for Information & communications Technology Promotion).

◆ First Author : Trinh Nguyen, Department of ICMC Convergence Technology, Soongsil University, Seoul, Republic of Korea, dangtrinhnt@soongsil.ac.kr, 학생회원

° Corresponding Author : Myungsik Yoo, School of Electronic Engineering, Soongsil University, Seoul, Republic of Korea, myoo@ssu.ac.kr, 종신회원

논문번호 : 201907-137-D-RN, Received July 24, 2019; Revised August 8, 2019; Accepted August 8, 2019

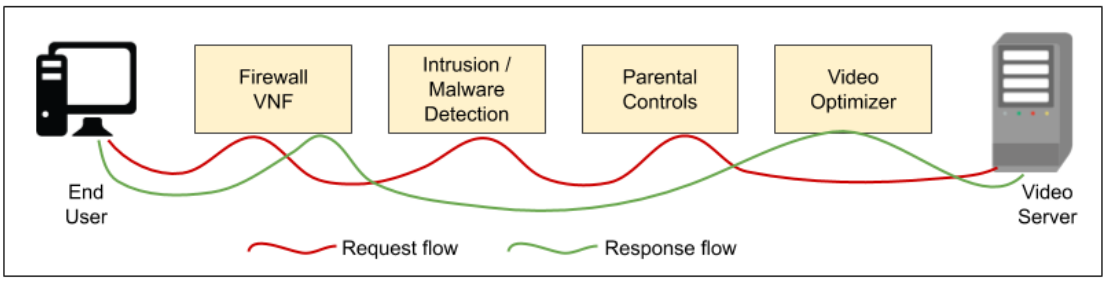

Fig. 1. a sample SFC

create more value-added services that aligns with their requirements. Figure 1 illustrates a sample SFC that contains four VNFS which are the firewall, instrusion/malware detection, parental controls, and video optimizer.

Besides allowing more advanced SFCs to be formed, NFV also adds complexity to the network by different virtualization layers. That makes testing more difficult because unlike traditional testing methods, we have to deal with layers of software and delays which may not provide accurate information for measurement. Comparing to hardware-based SFCs testing which uses real-time and advanced sensors to provide needed metrics for testing, virtualized SFCs requires more effort and new methods to test. Some effort has been made to provide testing solution in NFV but does not focus on SFCs which is the interoperability aspect of a NFV system. In section II we describe some background knowledge and existing work.

The rest of this paper is organized as follows. Section II introduces some background and related work. Section III describes the architecture and building blocks of the proposed monitoring system for container-based OpenStack. Section IV shows the experimental results to demonstrate the feasibility and effectiveness of the designed solution. Finally, section V concludes the paper and discuss future works.

Ⅱ. Background

When network is moving to NFV architecture, the approach of conventional network functions coming from only one vendor transforms to a collection of virtualized middle-boxes built by different vendors running on a virtualized system, with different complex service deployment model. Therefore, the old testing methods and mechanism should also be updated to support more cases and settings. Among different types of testing methodologies as defined by [2] and [3], there are two major concepts which are conformance testing and interoperability testing.

2.1 Conformance testing

Conformance testing is the testing procedure executed at the interfaces that normally not visible to the users. The purpose of conformance testing is to check whether a feature has been implemented correctly. Moreover, the tests are produced by a specialized testing system that has the ability to collect all input and output information. Conformance testing only works with one System Under Test (SUT), and the System Under Test can be a Function Under Test (FUT). Figure 2 represents the core idea of conformance testing.

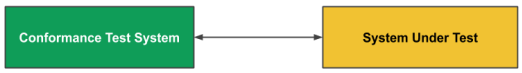

Fig. 2. Conformance testing method

2.2 Interoperability testing

Unlike conformance testing which focuses on the feature, interoperability testing concentrates on checking the service that the System Under Test provides users. The System Under Test can comprise many different Functions Under Test developed by different vendors. Interoperability

testing shows that the end-to-end feature of at least two functions working as expected. Tests in this method are produced through the Application Programming Interfaces (APIs). Because interoperability testing requires network developers to look for many components along the implement process, there are not much effort putting into it. Hence, there is a need to design an interoperability testing system for the NFV system to prevent failures. Figure 3 illustrates the main concept of interoperability testing.

As described in [4] testing in NFV should be a combination of conformance and interoperability testing since they are complimentary techniques that each has benefits and limitations. Recently, there are several work has been proposed to tackle the testing in NFV but mostly focuses on conformance testing while this article concentrates on testing the SFCs which employs the interoperability methodology. The GYM framework in [5] that offers a comprehensive architecture for the testing the VNFs can only test the functionalities of the VNFs under test. Other drawbacks of GYM is that it is designed as a separate system, not integrated with NFV and does not cover SFC. The 5TANGO project [6] provides an approach for NFV testing but only targets the 5G networks and does not support testing the orchestration and SFCs. SFCPerf [7] provides repeatability for experiencing different network

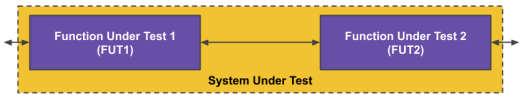

Fig. 3. Interoperability testing method

functions and virtualization infrastructure using both active and passive data collection for analysis. Even though SFCPerf offers such benefits, it has some disadvantages such as the management module features overlap with the NFV MANO under test, active data collection service is only deployed at the end points, and the test cases cannot be designed flexibly. In addition to those framework, the work in [8] [9] [10] [11] [12] are open source collaboration projects that target mainly on performance and functionality testing of the NFV implementation such as OPNFV, OpenStack, etc.

Studying the above work, an agent-based mechanism for SFC testing is proposed. Detail architecture of the proposed system will be discussed in Section III.

Ⅲ. The Agent-Based Mechanism for Testing SFC

3.1 Scope of the proposed mechanism

Tackling issues of the existing work, the proposed system is trying to provide the ability to test the interoperability, orchestration of SFCs rather than

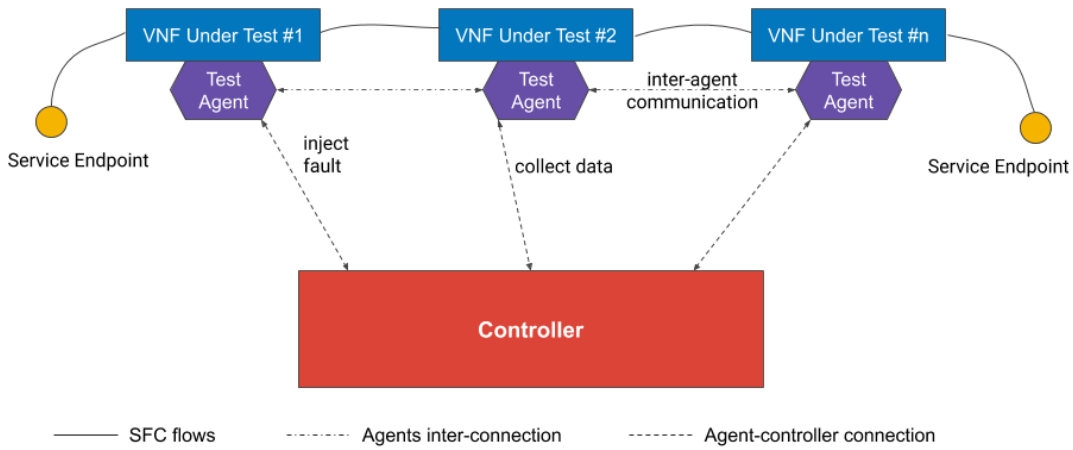

Fig. 4. Detail architecture of the agent-based mechanism for testing SFC

the infrastructure or performance. The scope of the proposed mechanism are SFC correctness which is to check if the designed SFC is working correctly and provides user the desired network services. Procedure includes decoupling the VNFs that SFC contains and extracting information of the running package flows.

3.2 Detail architecture

The proposed mechanism comprises two main components, the test agent and the controller. A test agent is installed in each VNF Function Under Test (VNF FUT). Agents will collect metrics from the VNF under test and send back to the Controller. Agents can execute test commands inside the VNF under test following by instructions of the Controller.

Agents communicate with each other to test the VNFs within a SFC (throughput, port lock, package loss, etc.). Different type of agents can be supported in a way that a corresponding agent driver in the Controller will be made. One important testing technique is to leverage the Controller-Agents connection. In this technique, the Controller tells the Test Agent to inject faults (e.g., loads, packages that go to forbidden path, etc.) into the VNF FUT. Faults were designed to test the performance and functionality of the VNF and the SFC under test. Faults are varied by test cases and defined in the test profile.

The Controller is the center of the proposed mechanism where all the testing flows are coordinated. The controller consists of different elements which are explained below:

* Test agent drivers: to collect data or inject faults depends on the test defined in the profile. They can analyze the metrics from collected data.

They also support different VNF vendors (e.g., KVM, VMWare VMs, containers, etc.)

* Controller Engine: controller coordinates the test procedures or test cases by using Agent Drivers and shows the results to the users.

3.3 Workflow

By deploying the agents to each VNF FUT, the proposed system can acquire the information of the packages flowing inside the running SFC to check for the correctness. The workflow is as follows.

Firstly, the SFC under test need to be pre-deployed and running so that IP addresses of the VNF FUT are known. The Test Agents can be installed when the VNF FUT are instantiated or after they are up and running. The Test Agents also need to be installed in the start and end points of the SFC. After the Test Agents are deployed and the Controller recognizes them, the user tells the Controller to execute the test. The Controller asks each Test Agents to send information of the host VNF for it to analyze via the REST APIs provided by the Test Agents. Test Agents harvests the packages flowing through the host VNFs and return the needed information for the Controller. Finally, the Controller analyzes all of the gathered information and shows the results the user.

Ⅳ. Experiment

An implementation is produced to evaluate the proposed mechanism in term of feasibility and extensibility. OpenStack Tacker [13] was used as the target testing NFV MANO while all other cloud infrastructure services are instantiated using Devstack [14]. One single computer was used to deploy the test scenario as illustrated in Figure 5. The test scenario is a single SFC that comprises of two VNFs, one is the firewall, and the other one is the Content Filtering VNF. The end point of the network service is a HTTP Server. In this experiment, the SFC does not allow traffic going through port 8080 (VNF1) and filters access to the "/restricted" path (VNF2) of the HTTP Server. So our test results will be showing if the packages going through or not and if the access to the "/restricted" URI is blocked.

Table 1 is the system configuration used for the experiment.

In this article, our focus is designing the mechanism for network operators (i.e., the users) to be able to gather information for the running VNFs of a SFC so that it can do the analysis and provides

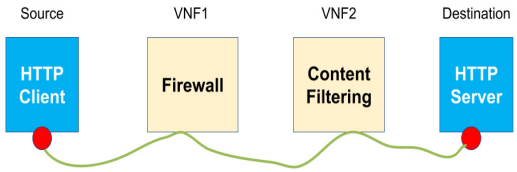

Fig. 5. The test scenario

useful information about the SFC. That means the Test Agent will have to create a special interface for the Controller to get the information it needs. Therefore, we decided to implement the REST APIs on each Test Agent. The Controller then will only need to call these APIs to communicate with the Agents. Figure 6 displays an example of the API implementation using Python programming language. In the example, the Agent exposes the tcpdump [15] data it gets from the host VNF to the Controller.

Figure 7 and 8 illustrate the analysis results in command lines after the Controller received information from the Test Agents where 172.24.4.250 is the IP address of the HTTP Server.

In figure 7, the Controller first asked the client to send request the "/restricted" path of the server. It then told VNF1 (i.e., the firewall) to check if the request packages are through, VNF2 (i.e., the

Table 1. Specifications of the Experiment

Item	Configuration
CPU	Intel (R) Xeon (R) CPU E3-1240 V3 @ 3.40GHz, 8 cores
Memory	16GB DDR3, 1333 MT/s
Operating System	Ubuntu 18.04 LTS
Software	OpenStack Tacker, Devstack, OpenWrt, CirrOS, Python 2.7

```
...
@app.route('/tcpdump/<int:timeout>/<int:port>')
def tcpdump(timeout, port):
    filename = _tcpdump(timeout, port)
    return filename
...
```

Fig. 6. An example of the API

```
================================================
Results of test criteria content_filtered is:
================================================

+Src client: Request to path restricted of server 172.24.4.250 sent.

+VNF1 result: Has packet

+VNF2 result: Access denied

+Server result: No packet

================================================
```

Fig. 7. Results of testing if the "/restricted" URI is blocked.

content filtering) to check if it filter the restricted path. Finally, the Controller checked if the server has some packet. In this scenario, if the VNF1 has packet, VNF2 denies access to the provided path, and the Server also has some packets, the content filtering feature of the SFC is working properly.

The results showing in figure 8 validates if the port 8080 is blocked by the VNF1. So that, if the VNF1 has packet, VNF2 and the HTTP Server have no packet going through, the SFC is working correctly.

```
================================================
Results of test criteria port_block is:
================================================

+Src client: Request to port 8080 of server 172.24.4.250 sent.

+VNF1 result: Has packet

+VNF2 result: No packet

+Server result: No packet

================================================
```

Fig. 8. Results of testing if port 8080 of the HTTP Server is blocked.

V. Conclusion and Future Work

Even though testing in NFV is a mature topic, state of the art works only focus on conformance testing which is only cover performance and functionality of the VNFs or the underlying infrastructure. Therefore left the interoperability testing especially the SFCs unnoticed. This article by reviewing some of the related work in NFV testing methodologies proposes an agent-based testing mechanism for SFCs. Future work will focus on making the agent's capability to harvest more information of the host VNFs that belong to the SFC under test. Moreover, the Controller will be updated as well to produce advanced analysis to

provide more insight for network operators.

References

[1] ETSI, *Network Function Virtualisation (NFV); Architectural Framework* [Online]. Available: https://www.etsi.org/deliver/etsi_eg/202100_202199/202107/01.01.01_60/eg_202107v010101p.pdf

[2] International Telecommunication Union, *X.290 : OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications.* [Online]. Available: https://www.itu.int/rec/T-REC-X.290/en

[3] ETSI, *Methods for Testing and Specification (MTS); Planning for validation and testing in the standards-making process.* [Online]. Available: https://www.etsi.org/deliver/etsi_eg/202100_202199/202107/01.01.01_60/eg_202107v010101p.pdf

[4] ETSI, *Network Functions Virtualisation (NFV); Testing Methodology; Report on NFV Interoperability Testing Methodology*. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/002/01.01.01_60/gs_nfv-tst002v010101p.pdf

[5] R. V. Rosa, C. Bertoldo, and C. E. Rothenberg, "Take your VNF to the gym: A testing framework for automated NFV performance benchmarking," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 110-117, Sep. 2017.

[6] P. Twamley, et al., "5GTANGO: An approach for testing NFV deployments," *2018 IEEE EuCNC*, Ljubljana, Slovenia, Slovenia, Jun. 2018.

[7] I. J. Sanz, Diogo M. F. Mattos, and Otto Carlos M. B. Duarte, "SFCPerf: An automatic performance evaluation framework for service function chaining," *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symp. IEEE*, Taipei, Taiwan, Apr. 2018.

[8] *OPNFV Yardstick*, [ONLINE], https://wiki.opnfv.org/display/yardstick/Yardstick

[9] *OPNFV QTIP*, [ONLINE], https://wiki.opnfv.org/display/qtip/Platform+Performance+Benchmarking

[10] *OPNFV Bottlenecks*, [ONLINE], Available: https://wiki.opnfv.org/display/bottlenecks/Bottlenecks

[11] *OpenStack RefStack*, [ONLINE], Available: https://refstack.openstack.org/

[12] *OpenStack Tempest*, [ONLINE], Available: https://docs.openstack.org/tempest/latest/

[13] *OpenStack Tacker*, [ONLINE], Available: https://docs.openstack.org/tacker/latest/

[14] *OpenStack Devstack*, [ONLINE], Available: https://docs.openstack.org/devstack/latest/

[15] *tcpdump*, [ONLINE], Available: https://www.tcpdump.org/manpages/tcpdump.1.html

응웬쫑당찐 (Trinh Nguyen)

Trinh Nguyen received his B.Eng. degree in Computer Networking from University of Information Technology, VNU-HCM, Ho Chi Minh City, Vietnam, in 2012. He has been pursuing the Master's degree in ICT at Soongsil University since Fall 2017. His research interests include Software-Defined Networking, Network Function Virtualization and Cloud Computing.

[ORCID:0000-0001-6885-5935]

유 명 식 (Myungsik Yoo)

Myungsik Yoo received his B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, Republic of Korea, in 1989 and 1991, and his Ph.D. in electrical engineering from State University of New York at Buffalo, New York, USA in 2000. He was a senior research engineer at Nokia Research Center, Burlington, Massachusetts. He is currently a professor in the school of electronic engineering, Soongsil University, Seoul, Republic of Korea. His research interests include visible light communications, sensor networks, Internet protocols, control, and management issues.

[ORCID:0000-0002-5578-6931]