A framework for functional testing of VNFs

Askhat Nuriddinov, Wouter Tavernier, Didier Colle, Mario Pickavet
Ghent University – IMEC
{askhat.nuriddinov, wouter.tavernier, didier.colle, mario.pickavet}@ugent.be

***Abstract*—Network Function Virtualization (NFV) is a promising technology which can significantly boost innovations in the area of telecommunication networks. However, to realize the anticipated benefits of NFV, network operators need to solve several challenges which include performance and functional testing of Virtualized Network Functions (VNF). Furthermore, the development process of VNFs is very complex and error-prone, therefore the developers also need to do functional testing of their VNFs.**

To this end, we introduce a new open-source framework for functional testing of VNFs. It allows to write tests in Python and use its simplicity to write tests with minimal effort. The framework has integration with virtual infrastructure to make the test process seamless and less time consuming.

I. Introduction

Network Function Virtualization aims to reduce time-to-market and accelerate the development and deployment of Virtualized Network Functions (VNF). This architecture makes it possible to introduce DevOps in the development process. DevOps is a software engineering approach that aims at unifying software development (Dev) and software operation (Ops). It makes the development seamless and less error-prone and increases deployment frequency. The key aspect of DevOps is automated testing.

Currently the industry standard for testing network functions is TTCN-3 [5] developed by ETSI. It is a strongly typed testing language used in conformance testing of communicating systems. A downside of TTCN-3 is that it is a domain-specific language which means it has little popularity, less documentation and limited expressiveness compared to general-purpose languages.

In this document we introduce a new functional testing framework for VNFs. The framework allows to write functional tests in Python in just a few lines of Python code. It supports all stages of testing process including infrastructure setup, VNFs instantiation, configuration of network, execution of arbitrary commands on running instances and retrieving results. The test execution process is shown on figure 1. In the following paragraphs we will describe each section in detail. The framework is developed in the context of the 5GTango project [1] and is part of the 5GTango SDK [2].

II. Framework

The testing framework is written in Python language and is based on a popular Python testing framework `pytest`. This allows the developers to write tests in Python and use its simplicity and expressiveness to write tests very clear and elegant.

The framework has an integration layer which provides an abstract unified infrastructure interface to set up the testing environment. The integration is written as a separate module for each platform and can be easily extended to support any different platforms. The implementation inherits the abstract interface which makes tests written with the help of the framework almost platform-independent. The only difference will be in the target platform.

III. Use cases

Since the framework was developed as a part of the 5GTango project ([1]) it has support of its infrastructure managers OpenStack-based Service Platform [3] and Docker-based Emulator [4].

The 5GTango Emulator is a light-weight emulation platform, which is a part of the 5GTango SDK, and is supposed to be run on the developer's machine. It can be used for the development of VNFs without any specific target platform.

A. Test-driven development

Test-driven development is an approach that requires developers to write tests of software before actually writing the software itself. The developer can use some light-weight virtual infrastructure like the 5GTango Emulator and the testing framework to set up sufficient environment for test-driven development. The framework can generate rich feedback and significantly accelerate the development process.

B. Continuous integration

Continuous integration is a development practice that requires developers to integrate code into a shared repository several times a day. By integrating regularly, developers can detect errors quickly and avoid "integration hell". The framework can be used to check the integrity between different components of a complex VNF.

C. Continuous delivery

Continuous delivery is an engineering technique in which developers produce software in short cycles, ensuring that the software can be reliably released at any time. With the help of the framework developers can set continuous delivery of a VNF and make it available for deployment at any time.

D. Continuous deployment

Continuous deployment is an extension of continuous delivery when every change that is made in software is automatically deployed to production. The framework can prove reliability of a VNF and reduce time-to-market.

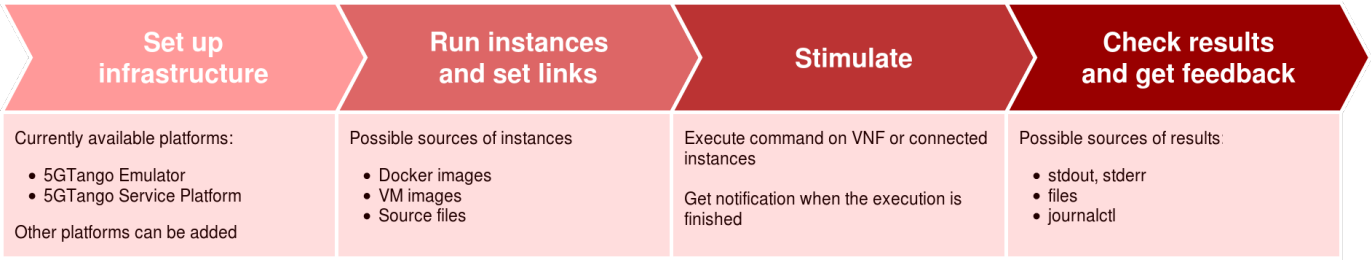

Fig. 1. Test execution process.

IV. Workflow

The framework aims to let developers write tests with minimal effort and define each stage of test in just a few lines of code. In the following paragraphs we describe the workflow of using the testing framework and give some details of each stage.

A. Setup

In order to execute tests the developer needs to run a virtual infrastructure. With the help of the framework this can be done by just initializing a class of appropriate infrastructure, e.g for the 5GTango Emulator it will look like `vim = Emulator()`. Then a VNF and extra instances can be launched and network links can be set up by appropriate methods of the infrastructure adapter class. There are several possible sources of instances: VM or Docker image and source files as well. With the last option an appropriate image will be automatically created. There is also a base image with the basic set of testing utilities which can be used as an extra instance to send traffic through the VNF and for creating new instances as well.

B. Stimulation

The test stimuli must be specified as arguments of a method of an instance which will send traffic or do some other required actions. If there is a need in some specific utilities to send traffic any arbitrary software can be installed in extra instances as it is described in the previous section.

C. Verification

When the stimulation starts the test execution become locked. After receiving the notification that the running process is finished the test is unlocked and the results can be checked. The results can be retrieved from `stdout` and `stderr` of the stimulation process, files and `journalctl` logs. There is also an option to retrieve only updates of file, i.e. rows written after the stimulation was started. In case of a failure of the test the developer will get feedback and see where the output differs from the expected.

V. Demonstration

The objective of the planned demonstration is to illustrate the usage of the framework. We will show the entire workflow of writing and executing functional tests of VNFs with the framework. For the demonstration we will use a chain of three instances:

1) Base image as an extra image to send traffic through VNFs with curl
2) Snort[1] — network intrusion detection system
3) Apache[2] HTTP server

The 5GTango Emulator will be used as a virtual infrastructure for this demonstration. We will show how the framework runs the environment, instantiate VNFs and the base image and sets network links. We will use Snort to pass some traffic and alert another. The test will check if Snort is properly configured and HTTP-server responds with correct headers.

We will also show how the Emulator and the framework can be used to set continuous integration. The entire process can be executed on a regular laptop.

VI. Conclusion

Our demonstrated testing framework greatly simplifies the process of testing VNFs and makes it less time-consuming. Hence, it significantly accelerates the development process and makes it less error-prone.

References

[1] 5GTANGO consortium, "5G Development and Validation Platform for global Industry-specific Network Services and Apps," 2018. [Online]. Available: https://5gtango.eu/

[2] S. V. Rossem, M. Peuster, L. Conceio, H. R. Kouchaksaraei, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, "A network service development kit supporting the end-to-end lifecycle of NFV-based telecom services," IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 12.

[3] S. Draxler, H. Karl, M. Peuster, H. R. Kouchaksaraei, M. Bredel, J. Lessmann, T. Soenen, W. Tavernier, S. Mendel-Brin, G. Xilouris, "SONATA: Service programming and orchestration for virtualized software networks," 2017 IEEE International Conference on Communications Workshops (ICC Workshops), May 2017, pp. 973978.

[4] M. Peuster, H. Karl, and Steven Van Rossem, "MeDICINE: Rapid Prototyping of Production-ready Network Services in multi-PoP Environments," IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN, 2016, pp. 148–153

[5] ETSI ES 201 873-1 V3.1.1 "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language", ETSI, 2005

[1] Snort: https://www.snort.org

[2] Apache HTTP: http://httpd.apache.org