# Performance Characterization and Profiling of Chained CPU-bound Virtual Network Functions

Sebastian Troia, Marco Savi, Giulia Nava, Ligia Maria Moreira Zorello, Thomas Schneider and Guido Maier

***Abstract*—The increased demand for high-quality Internet connectivity resulting from the growing number of connected devices and advanced services has put significant strain on telecommunication networks. In response, cutting-edge technologies such as Network Function Virtualization (NFV) and Software Defined Networking (SDN) have been introduced to transform network infrastructure. These innovative solutions offer dynamic, efficient, and easily manageable networks that surpass traditional approaches. To fully realize the benefits of NFV and maintain the performance level of specialized equipment, it is critical to assess the behavior of Virtual Network Functions (VNFs) and the impact of virtualization overhead. This paper delves into understanding how various factors such as resource allocation, consumption, and traffic load impact the performance of VNFs. We aim to provide a detailed analysis of these factors and develop analytical functions to accurately describe their impact. By testing VNFs on different testbeds, we identify the key parameters and trends, and develop models to generalize VNF behavior. Our results highlight the negative impact of resource saturation on performance and identify the CPU as the main bottleneck. We also propose a VNF profiling procedure as a solution to model the observed trends and test more complex VNFs deployment scenarios to evaluate the impact of interconnection, co-location, and NFV infrastructure on performance.**

***Index Terms*—Network Function Virtualization, Software Defined Networking, Virtual Network Function, Service Function Chain, Profiling, Monitoring, Virtual Router, Virtual Firewall**

## I. Introduction

Network Function Virtualization (NFV) is a game-changing technology that empowers digital transformation. It allows service providers to replace traditional, proprietary specialized hardware systems such as routers, firewalls, and Customer Premises Equipments (CPEs) with virtualized functions that are easy to upgrade and maintain. This means that new services can be quickly and efficiently rolled out, resulting in increased profits for providers.

NFV revolutionizes the way service providers manage their networks by transferring intelligence and workloads into software. This allows providers to easily and efficiently scale their Information Technology (IT) and network resources to meet changing traffic and service usage demands. With its immense commercial value and crucial role in the evolution of telecommunications networks, NFV has become a vital part of operator's development strategies. The global NFV market has seen tremendous growth in the past years, and is expected to continue to soar with a projected Compounded Average Growth Rate (CAGR) of 28% from 2022-2026 [1]. This growth is further propelled by the advent of 5G technology and the rise in services and applications with rigorous and varied performance requirements such as cloud computing and Internet of Things (IoT). Instead of relying on dedicated hardware, traditional network functions are now performed by software running on commercial-off-the-shelf servers.

The introduction of new services and technologies requires a major upgrade to the existing network architecture to meet their performance needs. Virtual Network Functions (VNFs), or virtualized network services that run on open computing platforms, offer a flexible solution. These VNFs can be deployed and configured in various ways, giving operators the ability to allocate and scale resources on demand, unlike traditional isolated physical appliances. However, the performance of VNFs can be unpredictable and can depend on various factors such as resource configuration, traffic load and underlying hardware characteristics. The virtualization overhead can also lead to bottlenecks, which can negatively impact Quality of Service (QoS) and degrade service performance. The operator needs to be aware of these potential performance issues and work to mitigate them to ensure a smooth and reliable service experience. Proper resource planning, such as scaling, requires a clear understanding of the relationship between resources and VNF performance. This allows for more accurate predictions of how many VNF instances to deploy or how many resources to allocate, instead of relying on time-consuming and inefficient trial and error methods.

This paper delves into the impact of resource allocation, resource consumption, and traffic load on VNF performance, using various metrics. The goal is to provide a Testing Tool (TT) able to perform a detailed analysis of how different factors affect VNF performance, and to develop analytical functions that can accurately describe this impact. The proposed TT offers the opportunity to test heterogeneous VNFs under different input workload features and traffic load, while identifying the VNFs resource consumption and the maximum QoS level that can be guaranteed. It also helps to analyze the VNF behaviour starting from the observation of the collected samples. By exploiting the obtained results, the trends of each tested VNF are described as a function of the resource allocation and consumption. Then, VNF profiling is proposed as a solution to model the observed trends. It provides a model to accurately describe how the specific VNF reacts under a

Sebastian Troia and Guido Maier are with the Department of Electronics, Information and Bioengineering (DEIB), Politecnico Di Milano, Italy.

Ligia Maria Moreira Zorello and Giulia Nava were with the Department of Electronics, Information and Bioengineering (DEIB), Politecnico Di Milano, Italy.

Marco Savi is with the Department of Informatics, Systems and Communication (DISCo), University of Milano-Bicocca, Italy.

Thomas Schneider is with ADVA Optical Networking, Munich, Germany.

Corresponding author: Sebastian Troia, e-mail: sebastian.troia@polimi.it

certain amount of workload and in a specific configuration. To validate our model, we perform different tests by exploiting multiple VNFs on both commercial testbeds provided by ADVA company[1] and experimental testbeds at the BONSAI lab[2] of Politecnico di Milano. The experiments include two scenarios: stand-alone VNFs and Service Function Chains (SFCs). The first scenario examines the impact of individual carrier-grade VNFs, while the second allows for the evaluation of the impact of multiple chained VNFs on the same server.

The remainder of this paper is organized as follows: In Section II, we provide an in-depth review of the existing research in the field of virtualized network functions (VNFs) and service function chains (SFCs) testing. This section is divided into three sub-sections, covering VNF performance characterization, VNF and SFC profiling, and testing tools for performance characterization and profiling. In Section III, we present the proposed testing tool architecture, including its different components and their functionalities. In Section IV, we describe the two testbeds used for conducting the experiments, including their configurations and limitations. Section V provides a detailed overview of the VNFs and SFCs used in this work and the challenges associated with testing them. In Section VI, we present the results of the experimental evaluation of our testing tool, including the test scenarios and performance metrics used in the evaluation. Finally, Section VII concludes the paper.

## II. Related Work

In this section we focus on the related work. At first, we report existing research works dealing with *performance characterization* of VNFs. Then, we focus on VNFs' and SFCs' *profiling*. Finally, we discuss the most relevant *testing tools* that have been developed for both performance characterization and profiling.

### A. VNF performance characterization

Ref. [2] focuses on the existing dependency between introduced processing delay as a function of processing virtual resources assigned to a VNF. Even though the introduced delay strongly depends on specific VNF implementations and underlying hardware, they assume a common trend where the higher the resources assigned, the lower the delay. However, they consider the existence of a saturation point, where delay is not further reduced by assigning processing resources above a threshold. Ref. [3] start from similar assumptions, as in [2], but focusing on exploiting such information to optimize VNFs placement, with the goal of meeting QoS service thresholds. Our research supports these findings and confirms the assumptions made by the authors in [2] and [3].

An important aspect is the impact of the VNF type on resource consumption and performance. To investigate this aspect, different VNFs have been studied in the literature. Ref. [4] tests the performance of a virtual router, deployed on a Kernel-based Virtual Machine (KVM) hypervisor environment. The authors identify the main throughput bottlenecks on the underlying infrastructure, which can have a negative influence on latency. Ref. [5] focuses on the performance of a virtual firewall. The authors investigate the service performance degradation that virtualization introduces compared to the traditional use of dedicated hardware. Ref. [6] evaluates the performance of a Firewall, an Intrusion Detection System (IDS), and a Network Address Translator (NAT) when deployed as Amazon EC2 Cloud instances. The analyzed metrics are resource utilization (in terms of CPU, memory and disk occupation), and packet loss percentage. The results obtained show that CPU is most likely to become a bottleneck causing performance degradation when it reaches the saturation level. VNFs that experience this behaviour are called *CPU-bound* VNFs. Furthermore, input packet rate is shown to be the determining factor in the VNFs performance as it is the most important parameter impacting on CPU consumption. Our work focuses on the performance characterization of a virtual router and of a virtual firewall (both CPU-bound VNFs) but, with respect to the previous works, it adds a more extensive analysis with respect to metrics such as throughput, delay and packet loss.

### B. VNF and SFC profiling

*1) VNF Profiling:* VNF profiling is proposed as an effective strategy to enhance the awareness of network operators when deploying VNFs in a highly dynamic and complex environment as an NFV infrastructure. By profiling VNFs' performance and resource consumption, it is possible to know in advance (i.e., before VNF deployment) the maximum amount of traffic that can be forwarded to a VNF instance while ensuring that a desired QoS service threshold is guaranteed.

Ref. [7] profiles the performance of four VNFs: virtual router, virtual firewall, Open vSwitch (OVS)[3]. and a Squid Cache server[4]. These functions are tested under a varying workload and different resource configurations. To collect data for profiling, each Device Under Test (DUT) is subjected to an increasing input workload. Starting from the gathered samples, the profiling models are trained using different techniques, such as Linear Regression, k-Nearest Neighbours, Interpolation, Artificial Neural Networks (ANN) and Curve Fit, with the latter being the one performing best.

Similar approaches using Machine Learning (ML) techniques are adopted in Refs. [8] [9]. These works focus on the need of estimating in an automated way the VNFs' CPU consumption as a function of the input traffic. Ref. [8] focuses on the characterization of the CPU usage (i.e., VNF's CPU profile) of OVS and Snort[5]. The authors exploit ANNs to create the VNF profile. Ref. [9] uses different methods

[1]Weblink (last access 14/04/2023): https://www.adva.com/

[2]Weblink (last access 14/04/2023): https://www.bonsai.deib.polimi.it/

[3]Weblink (last access 14/04/2023): https://www.openvswitch.org/. Open vSwitch is a high-quality, multi-layer virtual switch that is available under the open-source Apache 2.0 license.

[4]Weblink (last access 14/04/2023): http://www.squid-cache.org/. Squid acts as a caching proxy for the Web and can handle HTTP, HTTPS, FTP and other protocols. By caching frequently requested web pages, it decreases bandwidth usage and enhances response speed.

[5]Weblink (last access 14/04/2023): https://www.snort.org/. Snort is an Open Source Intrusion Prevention System (IPS). This IPS operates through the utilization of a series of rules, which are designed to identify malicious network activity.

to achieve the exact same goal, namely Linear Regression, Support Vector Regression (SVR), Decision Trees, Ensemble Learning and Neural Networks. Two different VNFs are tested and profiled: a Squid Cache and a Nginx Proxy.

Ref. [10] focuses on profiling the maximum throughput achievable under different CPU resource configurations for some software application: a Nginx with Apache Web Server, a Video Encoder with a Database and Snort. For the Web Server, Video Encoder and Database, the performance metric evaluated are the number of requests per second, the number of frames per second and the response time, respectively. The results show that the impact of the available CPU resources on throughput depends not only on the considered application, but also on the different possible configuration of each application.

Our work builds upon previous research by expanding the scope of performance metrics to be evaluated in relation to the input rate. We focus on three key metrics: packet loss, end-to-end delay and throughput. Previous studies have only examined a subset of these metrics, but by looking at all of them together, we can create more comprehensive profiles of VNFs.

*2) SFC profiling:* The previously mentioned VNF profiling approaches assume that VNFs perform the same way whether they are standalone components or interconnected to form an SFC. However, research studies in Refs. [11] [12] [13] [14] [15] have shown that the performance of chained VNFs can differ from that of standalone VNFs. It is crucial to take this into account when profiling both individual VNFs and the SFC as a whole.

Ref. [11] presents an SFC profiling system that includes the components necessary for performing the desired profiling. The system is validated using a linear chain including the Nginx TCP load balancer, the TCP relay Socat, and the Squid Proxy. Similarly, Ref. [12] tests a video streaming SFC in standalone and chaining scenarios, using Support Vector Regression and Polynomial Regression techniques to predict the minimum required vCPUs for a video encoder and a Squid cache. The results obtained with standalone VNFs are then compared to those obtained in the chaining scenario, showing a deviation in the expected CPU consumption of the cache component.

Ref. [13] presents a profiling model for an SFC composed of three functional blocks: a load balancer, a file server, and the client. The goal is to create a model that predicts the number of server instances required to meet a specific workload and QoS target. Various methods are used for profiling, including ANNs, Lasso, Random Forest, Multivariate Linear Interpolation, and Multi Lasso.

Ref. [14] presents an alternative approach, i.e. a two-step deployment workflow. At first, it uses VNF standalone profiles to make an initial estimation of resource allocation and SFC performance, and then updates them in a second phase using online data collected from the SFC. Four VNFs are tested: a load balancer, a proxy server, a firewall, and a virtual switch. The models are trained using ANNs, Lasso, and Multi Lasso.

Ref. [15] introduces an automated analysis system that can detect any abnormal behavior of the VNFs. The authors conduct tests with various combinations of VNFs to see how the addition of a new component can affect overall service performance. Although the authors do not focus on SFC profiling, the results demonstrate a significant deviation from expected behavior as more VNFs are added, highlighting the need for appropriate SFC profiling when multiple VNFs are chained together.

Building on previous research, we propose a testing tool that allows for the characterization and profiling of chains of VNFs with various topologies, while also highlighting differences with standalone deployment scenarios.

### *C. Testing tools for performance characterization and profiling*

A number of studies in literature have proposed testing tools for the characterization and profiling of VNFs and SFCs. Having a well-designed testing tool is crucial as it enables the characterization and profiling of a wide range of VNFs and SFCs in an automated or semi-automated manner, something that is not possible with the results-oriented and highly specific studies reported in previous sections.

Ref. [16] proposes a framework for VNF testing, with the goal of retrieving the VNF configuration that leads to the best performance, for instance in terms of Virtual Machine (VM) size, while Ref. [17] proposes a framework for VNF performance profile construction similar to the one proposed by our work. While such paper focuses only on VNF performance characterization, in this work we propose a VNF/SFC testing tool that also provides valuable profiling functionalities. Instead of solely focusing on VNF performance characterization like the aforementioned study, we take a holistic approach by considering a variety of performance metrics such as end-to-end delay, throughput, and packet loss. Our proposed tool provides the user with the capability to optimize different Virtual Network Functions (VNFs) in a more comprehensive manner, thereby resulting in improved overall performance. Moreover, the main difference with our proposal is that our testing tool is testbed-agnostic, and that we have adopted it to characterize and profile two different VNFs (i.e., a virtual router and a virtual firewall) on two different testbeds.

Ref. [18] proposes a more comprehensive framework that embeds a set of reusable VNF testing procedures and simplifies the operations to be performed for VNF performance and profile characterization. The framework's functionalities are demonstrated in Ref. [19]. Ref. [20] presents a new tool for VNF scalability (scale up/out) and energy efficiency benchmarking. This is very important for ensuring a correct dimensioning of the VNFs to be deployed. Although relevant, we do not consider this aspect; we instead focus on testing the performance of VNFs that have already been dimensioned. Ref. [21] presents a tool that enables the correlation of VNF performance with node resource utilization. However, the authors acknowledge that the tool has only been tested on a small testbed and its evaluation did not encompass real-world VNFs. Ref. [22] proposes a ML-based system to automate the detection of defects and bugs in VNFs by identifying performance degradation. In our work we do not perform VNF testing from a functional perspective, while we deeply focus on non-functional tests.

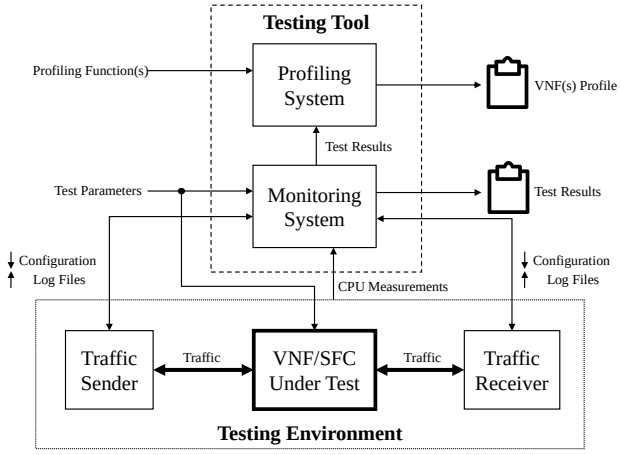


Fig. 1. Pictorial view of the Testing Tool architecture.

Ref. [11] is instead among the first works proposing a tool for SFC profiling. Ref. [23] proposes a framework similar to that of Ref. [11], but it is focused on performance characterization more than on SFC profiling. An important added value of Ref. [23] is that it characterizes SFC performance considering different hardware topologies. However, one limitation is that it only considers a linear SFC, limitation that we overcome in this work.

### D. Paper Contribution

Our work builds upon the existing literature on VNF performance characterization and profiling, but it extends it in several ways. We focus on the performance characterization of a virtual router and a virtual firewall, which are both CPU-bound VNFs, but we conduct a more extensive analysis with respect to metrics such as throughput, delay, and packet loss. This is different from Refs. [4]–[6] which mainly investigate the impact of VNF types on resource consumption and performance, but do not provide a comprehensive analysis of the VNF performance. Then, we propose a novel approach to estimate the VNFs' CPU consumption. Our approach is able to accurately state the CPU usage of both virtual router and virtual firewall under different workloads, while minimizing the need for extensive profiling. This is in contrast to Refs. [7]–[9], which mainly focus on profiling techniques to estimate the VNF performance. Overall, our work provides a more comprehensive and accurate analysis of the performance of CPU-bound VNFs, and proposes a novel approach to estimate their CPU consumption. Additionally, we highlight the importance of resource allocation strategies to improve VNF performance in NFV environments.

## III. Testing Tool architecture

In this section we describe the proposed Testing Tool (TT) for CPU-bound VNF/SFC performance monitoring and profiling. The TT is designed to be deployed on an NFV Testing Environment (TE). Figure 1 depicts a high-level view of the TT architecture and its interaction with the TE. The TT collects *log files* from the TE's nodes that are in charge of generating (*Traffic Sender*) and receiving (*Traffic Receiver*) the network traffic processed by the VNF or SFC under test, which is treated as a black box. Additionally, it receives measurements related to consumed CPU as recorded and stored by the TE.

As shown in Figure 1, the TT includes two sub-components: the *Monitoring System* and the *Profiling System*. The Monitoring System is in charge of processing data collected from the TE and of providing the user with the processed *test results*. These processed data are also provided as input to the Profiling System, together with some pre-defined *profiling functions* as selected by the user. These inputs are used for VNF profiling, and the obtained *VNF profile(s)* are output to the user. The user can also specify some *test parameters* (e.g. test duration, amount of virtual resources to be dedicated to the VNF under test, etc.) that drive the most appropriate configuration of all the nodes/components involved in the test execution (i.e., Traffic Sender, Traffic Receiver, VNF/SFC Under Test, Monitoring System), so that specific behaviours can be tested. The TT was designed and developed as an external component with respect to the TE, to which it seamlessly interfaces with the TE to gather crucial data and measurements. In this paper we adopt our tool to test VNFs/SFCs deployed in two different TEs, whose details will be provided in Section IV. More details on Monitoring and Profiling Systems are provided in the next subsections.

### A. Monitoring System

The Monitoring System provides an automatic and easily configurable tool for retrieving performance samples of the VNFs and SFCs under test. It can be used to test any CPU-bound VNF prior its deployment, characterizing its behaviour and defining the maximum workload that can be processed without affecting the running service. The tool, developed in Python programming language, tests the *VNF/SFC performance* by analyzing three metrics as a function of the VNF/SFC *input rate*. The three considered metrics are: *packet loss* (i.e., the amount of packets that are discarded), *throughput* (i.e., the amount of processed data per time unit), and *end-to-end delay* (i.e., the time needed by a generated packet to reach the destination while traversing the VNF/SFC). A fourth considered parameter is the VNF's *CPU consumption* on the TE. The latter is collected as a function of the input rate and, although not directly linked to VNF/SFC performance, it is fundamental to evaluate CPU-bound VNFs.

Going more into detail, three functionalities are guaranteed by the Monitoring System:

- *Test parameters gathering:* The user can specify the set of parameters necessary to perform the desired tests. Among the most relevant parameters, we can mention the duration of each test, the packets' size, the transport protocol to use, the number of concurrent flows to be generated and the range of input traffic rates to test. As parameter, it is also possible to specify that a *bursty traffic* scenario has to be tested. In this case, the user must also specify the *duty cycle* (i.e., ON and OFF periods).
- *Traffic generation and measurement:* The Monitoring System initiates the traffic generation process. To this

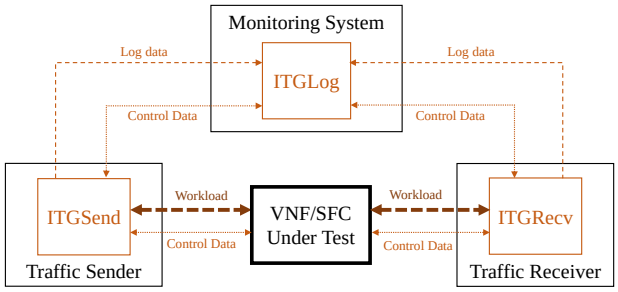


Fig. 2. Adoption of D-ITG modules in the TT [27].

aim, it exploits two open-source tools that are (partially) deployed in the TE: Distributed Internet Traffic Generator (D-ITG) [24] and the well-known UNIX System Activity Reporter (SAR). D-ITG requires an agent to be installed on the nodes of the TE acting as Traffic Sender and Traffic Receiver. SAR must be deployed only on the Traffic Sender and it is used to measure the actual traffic generated by the D-ITG sender agent (i.e., the VNF/SFC input rate). All the generated data is stored in log files. Based on the evaluation from the authors in [25] [26], we preferred to use D-ITG [24] as traffic generator as it guarantees higher performance compared to other software traffic generators[6]. Moreover, D-ITG makes it possible to easily measure the One-Way Delay (OWD) and Round-Trip Time (RTT) of each packet transmitted, and it is capable of generating packets according to patterns following a variety of probability distributions (such as uniform, exponential and normal) and using specific transport protocols (e.g. UDP, TCP, ICMP). To avoid any error due to incorrect synchronization among the sender and receiver nodes and ease the tool implementation and deployment, we decided to set up a two-way communication channel and to estimate the OWD as half of the RTT. How D-ITG is used as part of the TT is illustrated in Fig. 2. Specifically, the following D-ITG's modules are adopted:

- *ITGSend*: it is responsible for traffic generation. It can operate in *single-flow* or *multi-flow* mode. The TT uses the multi-flow mode, so that multiple concurrent flows are generated to better test the VNF/SFC behaviour. ITGSend agent is embedded in the Traffic Sender.
- *ITGRecv*: it is in charge of receiving the packets as generated by ITGSend. It is embedded in the Traffic Receiver.
- *ITGLog*: it is responsible for storing all the log information on the traffic and flows, including measurements related to packet loss, throughput and end-to-end delay. It is embedded in the Monitoring System.

Additionally, measurements related to the VNFs CPU consumption are sampled and collected by the TE. To do so, Testing-Environment-specific tools are used, as we will show in Section IV.

[6]We would like to remind the reader that evaluating and comparing different traffic generators is beyond the scope of this work.

**Algorithm 1** Monitoring System workflow

**Require:** $test\ traffic\ parameters$
**Ensure:** $VNF\ performance\ characterization,\ saturation\ point$
$t \leftarrow\ duration\ of\ a\ single\ test$
$n_{flows} \leftarrow\ number\ of\ flows$
$p \leftarrow\ protocol$
$p_{size} \leftarrow\ packets\ size$
$S_{allsamples} \leftarrow\ empty\ set\ of\ samples$
**if** $traffic$ is constant **then**
  $R \leftarrow\ set\ of\ input\ rates\ to\ test$
  $S_{statistics} \leftarrow\ empty\ set\ of\ average\ samples$
  **for** $r$ in R **do**
    $generate\ traffic\ with\ rate\ r\ on\ each\ flow\ for\ t\ seconds$
    $use\ Paramiko\ to\ retrieve\ receiver\ log$
    $compute\ input\ rate,\ throughput,\ end-to-end\ delay\ and\ packet\ loss$
    $i \leftarrow\ input\ rate\ samples$
    $o \leftarrow\ throughput\ samples$
    $p \leftarrow\ packet\ loss\ samples$
    $d \leftarrow\ delay\ samples$
    $i_{mean},\ o_{mean},\ p_{mean},\ d_{mean} \leftarrow compute\ mean\ of\ metrics$
    $S_{allsamples}.append(\ i,\ o,\ p,\ d)$
    $S_{statistics}.append(\ i_{mean},\ o_{mean},\ p_{mean},\ d_{mean})$
  $generate\ plots\ from\ S_{statistics}$
  $start\ the\ profiling\ procedure$
**else if** $traffic$ is bursty **then**
  $T_{pattern} \leftarrow\ traffic\ pattern$
  $generate\ traffic\ according\ to\ T_{pattern}$
  $use\ Paramiko\ to\ retrieve\ receiver\ log$
  $compute\ input\ rate,\ throughput,\ end-to-end\ delay\ and\ packet\ loss$
  $i \leftarrow\ input\ rate\ samples$
  $o \leftarrow\ throughput\ samples$
  $p \leftarrow\ packet\ loss\ samples$
  $d \leftarrow\ delay\ samples$
  $S_{allsamples}.append(\ i,\ o,\ p,\ d)$
  $generate\ plots\ from\ S_{allsamples}$
**return** samples datasets

- *Data gathering and elaboration:* The Monitoring System collects the Traffic Sender and Traffic Receiver log files as well as CPU consumption measurement from the TE, elaborates them and saves the results in Comma-Separated Values (CSV) format files. These files can be then plotted (if needed) by the user and are used as input by the Profiling System.

*1) Explanatory workflow:* In this section we report an explanatory workflow on how the Monitoring System can be used. A pseudo-code is shown in Algorithm 1.

The user must first specify the *input parameters*, i.e., duration of a single test, number of concurrent flows, protocol to use, packet size and traffic pattern. As specified, two types of tests can be performed: with *constant input traffic* and with *bursty traffic*.

In the case of constant input traffic, the *minimum* and *maximum* input rates to test are defined in terms of packets per second, together with a *step* parameter. The tool will then test sequentially all the $R$ traffic rates between the minimum and maximum according to the specified step, by also taking as input the other specified input parameters. After each test the

results, as collected and elaborated by the data retrieved from the TE (e.g. D-ITG logs), are saved in CSV files. Two files are obtained, generated respectively starting from the $S_{statistics}$ and $S_{allsamples}$ sets. $S_{statistics}$ elaborates average values over the test duration for each tested input rate: the corresponding average delay, packet loss, throughput, together with CPU consumption, are recorded and stored. The Monitoring System also allows the user to plot the graphs of the average values of the considered metrics and CPU consumption as a function of the input rate. $S_{allsamples}$ collects instead all the fine-grained measurements for each metric and each input rate, where a sample is recorded every second.

In the case of bursty input traffic, a single input rate is specified together with the *duty cycle*. The results obtained for each second of the test are saved in $S_{allsamples}$. From the file, the graphs of the performance metrics are plotted. No average statistics are saved in $S_{statistics}$ for this specific test.

The measurements dataset $S_{statistics}$ can then be used by the Profiling System to profile the VNFs Under Test.

*2) Detailed explanation of Algorithm 1:* The Monitoring System necessitates the user to establish the input parameters concerning the traffic pattern to be generated. These parameters include the duration of a single test (line 1), the number of concurrent flows (line 2), the protocol to be utilized (line 3), the packet size (line 4), and the traffic pattern (lines 6 and 22). It is noteworthy that the Monitoring system facilitates two distinct types of tests - those that employ a constant input traffic pattern and those that adopt a bursty traffic pattern.

1) For a constant input traffic (line 6), the algorithm defines the minimum and maximum input rates to be tested along with a step parameter (line 7). Subsequently, the algorithm tests all the $R$ traffic rates between the minimum and maximum rates in a sequential manner, incrementing the traffic rate by the specified step as indicated in line 9. After each test, the algorithm retrieves the results from the receiver log through Paramiko[7], as shown in line 11, and saves them in CSV files. In line 18, the algorithm generates a file called S_allsamples that records fine-grained measurements for each metric and input rate, with a sample being recorded every second. Another set of samples, S_statistics, is obtained in line 19, which collects the metric values (average delay, packet loss, throughput) as the average over the test duration for each tested input rate. Additionally, the Monitoring System provides the functionality to plot graphs of the average metric values and CPU consumption as a function of the input rate, as indicated in line 20. Furthermore, the experiment continues in line 21 by launching the Profiling System on the measurements dataset S_statistics to profile the VNF being tested.
2) In the case of bursty input traffic (line 22), a traffic pattern needs to be chosen as indicated in line 23. The results obtained for each second of the test are saved in S_allsamples, which records the input rate (line 27), throughput samples (line 28), packet loss samples (line 29), and delay samples (line 30). The performance metric graphs are plotted from this file in line 32. It should be noted that no average statistics are computed for this specific test as the focus is on the traffic pattern.

[7]Weblink: https://www.paramiko.org/ (last access: 14/04/2023)

### B. Profiling System

The Profiling System is in charge of analyzing the data collected by the Monitoring System and to profile a VNF. This makes it possible to understand the expected behavior of a VNF, when deployed on a physical infrastructure, prior its deployment, so that the optimal resource allocation (e.g. in terms of vCPUs) is performed, based on the expected input traffic.

The ultimate goal of the Profiling System is thus to elaborate a *VNF profile*, starting from some pre-defined input *profiling functions*. Starting from the measurements obtained by the Monitoring System, the best fitting profiling function is chosen for each performance metric and the VNF profile is constructed. Each VNF profile is specific for (i) any VNF implementation and (ii) underlying hardware specification and configuration.

The VNF profiling strategy that we propose is suitable for *CPU-bound VNFs*, which are very common and their salient properties have been already well investigated in literature [3] [6] [7]. However, a similar approach could be adopted for VNFs whose performance is mostly affected by scarcity of other resources (e.g. memory-bound VNFs). Our proposed profiling strategy takes inspiration from [7]. Similarly, it identifies two different working regions for CPU-bound VNFs:

- *No CPU saturation*: In this region the CPU consumption is below its maximum. The input traffic intensity and CPU consumption are highly correlated: an increase on the input traffic reflects to an increase on the CPU consumption. The processing capability is enough to guarantee almost constant delay and throughput, with no (or negligible) packet losses.
- *CPU saturation*: In this region the CPU consumption is at its maximum. The input traffic intensity and CPU consumption are poorly correlated, while an increase in the input traffic reflects to higher delay and packet losses, and to lower throughput: this means that the input workload is highly (positively or negatively) correlated to these performance metrics.

As we will show in Section VI, this behaviour is confirmed for the VNFs tested in this paper. We define the input rate leading to the border point between the two regions as *CPU saturation break-point* [7]. The Profiling System models the VNF behaviour (in terms of throughput, delay and packet loss as a function of the input workload) with a *monotonic function*. The expected output is a set of functions $f_m(r)$ that profile the VNF, where $m$ indicates the metric $m \in \{throughput, delay, packet\ loss\}$ and $r$ the input traffic.

The method used to obtain $f_m(r)$ is Curve Fit, as it has been proven to provide the best accuracy with respect to other methods such as Linear Regression or Interpolation [7]. It consists in fitting a pre-defined set of functions $f(r)$ to the samples collected for each metric $m$ and as a function of $r$, to find the most accurate (i.e., the one that minimizes the

**Algorithm 2** Profiling System workflow

**Require:** $S_{statistics}$
**Ensure:** $f_m(r)$ with $m \in \{throughput, delay, packet\ loss\}$
1: $S_{nonsat} \leftarrow \emptyset$
2: $S_{sat} \leftarrow \emptyset$
3: Identify $r_{CPU_{saturation\_breakpoint}}$ from $CPU_{mean}(r)$
4: **for** $s \in S_{statistics}$ **do**
5: **if** $r < r_{CPU_{saturation\_breakpoint}}$ **then**
6: $S_{nonsat}.append(s)$
7: **else**
8: $S_{sat}.append(s)$
9: Use Curve Fit method on $S_{nonsat}$ and $S_{sat}$
10: **for** $m \in \{throughput, delay, packet\ loss\}$ **do**
11: $f_{m,nonsat}(r) \leftarrow$ Most accurate function on $S_{nonsat}$ for *no CPU saturation* region for metrics $m$
12: $f_{m,sat}(r) \leftarrow$ Most accurate function on $S_{sat}$ *CPU saturation* region for metrics $m$
13: Save fitted coefficients $C_m$ for $f_{m,nonsat}(r)$ and $f_{m,sat}(r)$
14: Define $f_m(r) = \begin{cases} f_{m,nonsat}(r) & No\ CPU\ saturation\ region \\ f_{m,sat}(r) & CPU\ saturation\ region \end{cases}$
15: Plot fit functions $f_m(r)$ with $m \in \{throughput, delay, packet\ loss\}$
16: **return** $f_m(r)$ and $C_m$ for $m \in \{throughput, delay, packet\ loss\}$

standard error). Curve Fit is performed independently for the *no CPU saturation* and the *CPU saturation* regions, and the possible profiling functions are taken from [28]. By adopting this approach, $f_m(r)$ are piece-wise functions defined in the following way:

$$f_m(r) = \begin{cases} f_{m,nonsat}(r) & No\ CPU\ saturation\ region \\ f_{m,sat}(r) & CPU\ saturation\ region \end{cases} \quad (1)$$

$f_{m,nonsat}(r)$ and $f_{m,sat}(r)$ are the profiling functions, taken from the set of input functions $f(r)$, that best fit to the collected samples, in any of the two regions, according to the Curve Fit method.

*1) Explanatory workflow:* Algorithm 2 reports an explanatory workflow of the profiling procedure carried out by the Profiling System. As input, the $S_{statistics}$ set, as obtained by the Monitoring System, is given. The Profiling System splits $S_{statistics}$ in two different subsets, $S_{nonsat}$ and $S_{sat}$, according to the CPU working zone they belong to. Then, the Curve Fit method is applied on the two subsets. The best function, for each specific performance metric and working region, is retrieved. The Curve Fit procedure also returns the *fitted coefficients* for each chosen function, saved in CSV files together with the name of the chosen function. The functions can also be plotted by the Profiling System, and all together specify the VNF profile.

*2) Detailed explanation of Algorithm 2:* Given the dataset S_statistics, obtained by the Monitoring System during the constant traffic tests, the Profiling System employs an algorithmic process to identify the CPU saturation breakpoint (line 3) and subsequently segregates each sample of the dataset into two distinct subsets, namely S_nonsat (line 1) and S_sat (line 2), based on the CPU working zone they occupy. This process is represented in the pseudo algorithm outlined in lines 4 to 8. Subsequently, the Curve Fit method is invoked to analyze the two aforementioned subsets (line 9), enabling the retrieval of the optimal function for a given performance metric and working region (line 11 and 12). The Curve Fit process, besides ascertaining the most suitable functions, also provides the relevant fitted coefficients for each of the selected functions (line 13). Furthermore, the Profiling System is equipped with the capability of plotting these functions, which in their entirety define the VNF profile (line 15 and 16).

## IV. Testing Environments

Two different TEs have been set up and used together with the TT to perform the experiments reported and described in Section VI. The two TEs are built upon two testbeds that have been designed and are currently used for different purposes:

- *ADVA TE:* It is built on top of a carrier-grade testbed hosted at ADVA premises that is used for testing of production-grade software.
- *BONSAI TE:* It is built on top of a testbed realized for research and academic purposes, hosted in the premises of BONSAI Lab of Politecnico di Milano.

In the following, we describe in detail the TEs and how they have been set up.

### A. ADVA Testing Environment

The ADVA TE is based on a commercial NFV solution, called *Ensemble* [29], deployed on the testbed in ADVA's premises.

*1) Hardware components:* The ADVA TE is illustrated in Fig. 3 and includes the following *hardware* components:

- Three Lanner 5210 servers acting as *computing nodes*, equipped with Intel Atom C3958 CPUs with 16 Cores, 32 GB RAM and 512 GB SSD.
- A Cisco Nexus *network switch*, used to connect the computing nodes and steer network traffic between VNFs deployed on different computing nodes.
- An HP Proliant server, hosting the ADVA proprietary software for ETSI-compliant NFV Management and Orchestration (MANO).

*2) Software components:* Concerning *software*, it is entirely part of the ADVA's *Ensemble* suite [29]. An important component of the TE is the *Connector* [30]. It is deployed on each computing node and acts as a virtual network Operating System. It provides not only cloud-native computing technology, but also a virtual infrastructure manager based on Linux, KVM and OpenStack [31] that implements all the functions related to the virtualisation layer of the NFV Infrastructure (NFVI), which includes the three computing nodes and the network switch implemented in this work.

The adopted NFV Management and Orchestration software component (*ADVA MANO*) is the heart of the TE[8]. It is in charge of controlling the NFV Infrastructure (NFVI) and manages the lifecycle of the VNFs and the deployment of the SFCs. The sub-components of ADVA MANO are the

[8]The reader can find more information on ADVA MANO in the following white paper (last access 14/04/2023): https://www.adva.com/en/resources/resources-gated-page/reports/eantc-adva-smartwan-using-6wind-vsr-with-intel

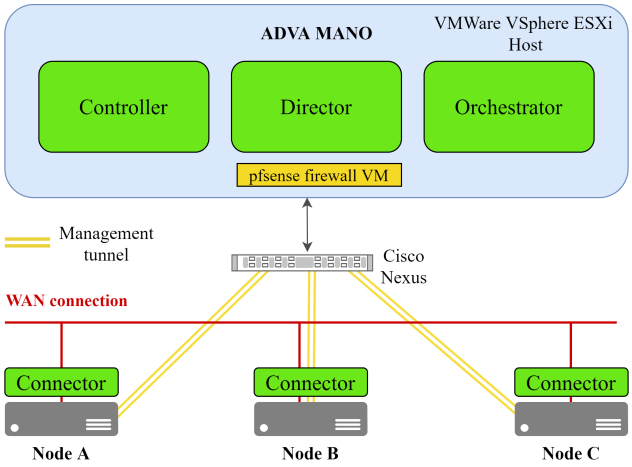


Fig. 3. ADVA testbed architectural components.

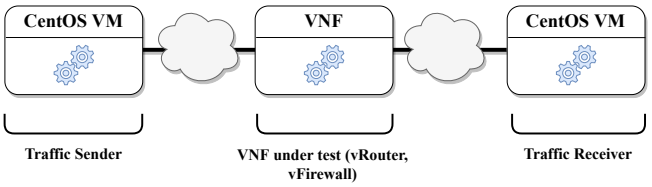


Fig. 4. Stand-alone VNF testing scenario.

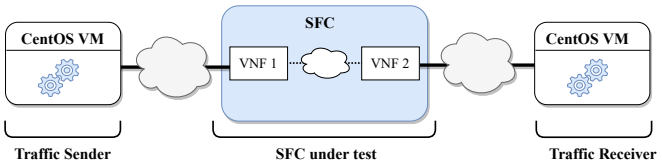


Fig. 5. SFC testing scenario.

*Controller*, the *Virtualization Director* [32] and the *Orchestrator* [33]. The three sub-components interact by means of open and standard Application Programming Interfaces (APIs). Specifically, the TE requires the functionalities of the Virtualization Director and of the Orchestrator, while it does not require those offered by the Controller.

The *Virtualisation Director* offers a single-cloud or multi-cloud automated NFV service, end-to-end service visibility, and performance management. It deals with the faults and events generated by the Connector, offering troubleshooting tools and management support.

The *Orchestrator* is an ETSI MANO-compliant solution [34]. It provides both an NFV Orchestrator (NFVO) and a generic VNF Manager (VNFM) and offers a single point of entry for end-to-end network services and VNFs lifecycle management, with features for the VNF on-boarding, network service design and deployment. The front-end of the Ensemble Orchestrator provides a web user interface, guaranteeing a consistent exposure layer to the Operational Support System (OSS).

### B. BONSAI Testing Environment

The BONSAI TE is built upon a testbed composed of 12 computing nodes, interconnected between each other. The TE is used by students and researchers for academic and research purposes. It relies on a resource virtualisation environment based on OpenStack [31] (Stein version), which handles all the computing nodes and provides functionalities related to instance creation, interconnection and resource allocation. By following NFV MANO specifications, the BONSAI TE uses OpenStack as a Virtual Infrastructure Manager (VIM) to assign the required resources to the virtualized network services (i.e., VNFs and SFCs).

For the purposes of this work, a single computing node is used to launch VNF instances and to host the Traffic Sender and the Traffic Receiver. It is equipped with an Intel Xeon E5-2620 v4 processor with 8 cores, and Ubuntu 18.04 LTS is used as an Operating System.

### C. Testing scenarios

For both TEs, if a single VNF has to be tested, three virtual machines are instantated: one VM hosts the VNF Under Test and two VMs, equipped with CentOS, act as Traffic Sender and Traffic Receiver, respectively. We call this setup *stand-alone VNF* testing scenario. If a SFC has instead to be tested, and thus more than one VNF must be deployed, one VM per chained VNF is instantiated. In this case, we referred to it as *SFC* testing scenario. The two different scenarios are illustrated in Fig. 4 and Fig. 5, and will be both investigated in Section VI.

### D. Differences between Testing Environments

As already pointed out, the two TEs are different in scope. While the ADVA TE has been specifically set up to test production-grade software artifacts, the BONSAI TE has been created by exploiting open source software to allow students and researchers carry on their academic projects.

Concerning their configuration, one of the main difference is related to *hyper-threading*. The ADVA TE configuration has hyper-threading disabled by default, ensuring a one-to-one mapping between virtual CPUs and physical CPU cores. This approach is taken to prevent any computational resource competition among virtual CPUs that would result from assigning them to the same physical core. On the other hand, the BONSAI TE configuration has hyper-threading enabled, with virtual machine instances typically sharing computing resources. This type of resource competition can be expected to result in decreased VNF performance due to performance inference, as previously documented in related work [35], [36] and experimentally demonstrated in [37] and confirmed in Section VI.

In general, such a performance interference makes us claim that disabling hyper-threading should be always done whenever possible, i.e., when enough computational resources are available in the virtualization system. This is the reason why we decided to disable hyper-threading in the ADVA TE (where more computational resources are available), but we did not disable it in the BONSAI TE, which is a less-capable TE. In

TABLE I
VIRTUAL ROUTER FLAVORS.

| # Flavor | Resource requirements |
|---|---|
| 1 | 1 vCPU, 2 GB RAM, 10 GB Disk |
| 2 | 2 vCPU, 2 GB RAM, 10 GB Disk |
| 3 | 3 vCPU, 2 GB RAM, 10 GB Disk |

TABLE II
VIRTUAL FIREWALL FLAVOR.

| # Flavor | Resource requirements |
|---|---|
| 1 | 2 vCPU, 6.5 GB RAM, 60 GB Disk |

addition, another aspect should be considered. Hyper-threading has a much worse impact on data-plane-intensive VNFs such as those considered in this paper (i.e., virtual router and virtual FW) than on control plane functions (e.g. Dynamic Host Configuration Protocol, DHCP). So, also the type of VNFs that have to be executed (i.e., data-plane-intensive or not) should be taken into consideration by the network operator when deciding whether hyper-threading should be disabled or not in the virtualization system.

## V. VNFs AND SFCs UNDER TEST

This Section describes the carrier-grade VNFs that have been tested using the TT, with the goal to extrapolate common trends occurring for CPU-bound VNFs. Specifically, the VNFs Under Test are two carrier-grade functions, namely a *virtual router* and a *virtual firewall*, which have also been replicated and chained together following different SFC topologies.

### A. Virtual Network Functions

*1) Virtual router:* The first considered VNF is the Mikrotik proprietary virtual router named *Mikrotik Cloud Hosted Router* [38]. It is feasible to designate different *flavors* for this VNF. As specified in Table I, three distinct flavors have been defined. These flavors differ in the number of assigned virtual CPUs, providing the ability to assess the performance of the virtual router under varying computational constraints.

*2) Virtual firewall:* The second VNF considered is a proprietary virtual firewall by *Palo Alto Networks* [39], named VM-100 and belonging to the *VM-series*. The VM-series has been developed as a set of solutions to guarantee an efficient and advanced threat prevention in a virtual environment, reaching the same performance as next-generation firewall hardware appliances. With respect to the virtual router, the adopted business model is different. Only one flavor can be used for the VM-100, as shown in the Table II. For more virtual resources, a more powerful virtual firewall has to be provided (e.g. VM-200).

### B. Service Functions Chains

Two types of SFCs are considered. The first one, illustrated in Fig. 6, is a simple linear chain (*SFC linear* in brief) concatenating a virtual router and a virtual firewall. This is the typical SFC that can be adopted to access a De-Militarized

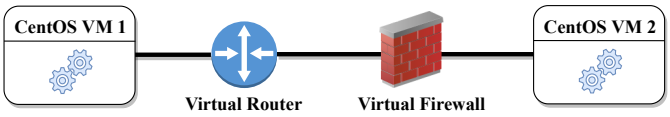


Fig. 6. Linear Service Function Chain (*SFC linear*).

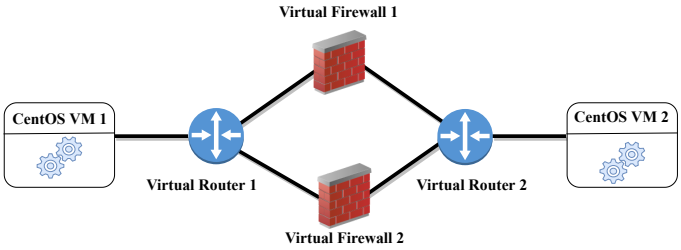


Fig. 7. Service Function Chain with split (*SFC split*)

Zone (DMZ). All the traffic sequentially traverses the two VNFs in a given order.

The second one is illustrated in Fig. 7. It is characterized by two branches, starting and ending in two different virtual routers, and each branch includes a virtual firewall. Traffic load is balanced by the first virtual router between the two branches, and merged towards the receiver by the second virtual router. A SFC of this type can be adopted when a single virtual firewall does not have enough computational resources to process all the incoming traffic. In brief, we will call it *SFC split*. The resource requirements for the VNFs of each of the two SFCs are listed in Table III. Note that *SFC split* is defined by taking as an inspiration the results obtained in [40] [41], where multiple VNFs can be executed in parallel. The only difference with respect to those works is that we consider the particular case where the VNFs that are executed in parallel are of the same type (i.e., virtual firewall), but all the considerations made later in this paper can easily be generalized to adhere to the SFCs considered in [40] [41].

## VI. EXPERIMENTAL EVALUATION

In this Section we report the results obtained by running the proposed TT (Section III) on the two considered TEs (Section IV) with respect to the VNFs and SFCs introduced in Section V. We first report the results obtained by running the Monitoring System sub-component, then we focus on VNF profiling as performed by the Profiling System. The goal is to characterize the considered VNFs and SFCs and to extrapolate some common trends that occur for (chained) CPU-bound VNFs.

### A. Monitoring System - VNF/SFC Performance Characterization

We used the Monitoring System to test the VNFs and SFCs Under Test by generating UDP traffic. We conducted a series of experiments on a single VNF configuration. To capture a comprehensive range of possible input rates, we

TABLE III
SFCs FLAVORS.

| SFC | Resource requirements (per VNF) | |
|---|---|---|
| Linear | 1 Virtual router | 1 vCPU, 2 GB RAM, 10 GB Disk |
| | 1 Virtual firewall | 2 vCPU, 6.5 GB RAM, 60 GB Disk |
| Split | 2 Virtual router | 1 vCPU, 500 MB RAM, 10 GB Disk |
| | 2 Virtual firewall | 2 vCPU, 6.5 GB RAM, 60 GB Disk |

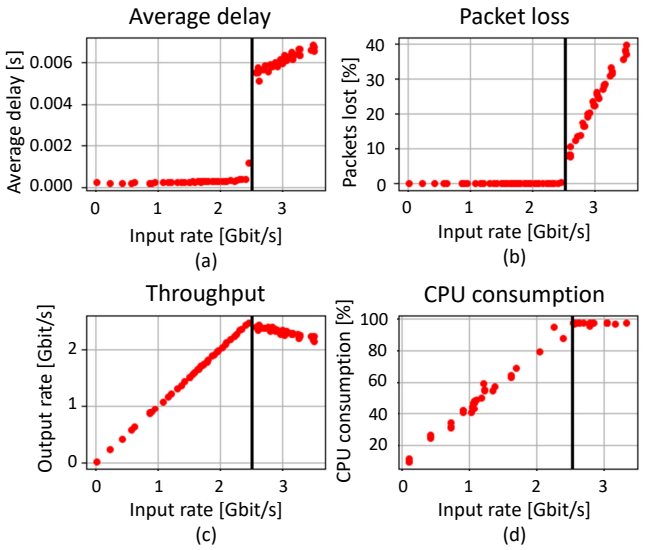


Fig. 8. Virtual router performance on ADVA TE (packet size: 1460 bytes).

specified a minimum and maximum range (1000, 80000 pps[9]), while also imposing a pre-determined step size between each measurement point (1000 pps). In order to obtain accurate and precise measurement data, each input rate was sustained for a constant time period of 30 seconds. During this interval, we recorded samples of critical performance metrics such as throughput, packet loss, average delay, and CPU consumption. The measurement outcomes were derived by computing the mean values of these performance metrics over the 30-second observation period. In the following, we report the obtained results.

*1) Virtual router:*

*a) ADVA TE:* The results for the virtual router are reported in Fig. 8. Flavor 1 (see Table I) is used and the packet size for UDP-generated traffic is set to 1460 bytes. The analysis of the collected measurements confirms the existence of a different behavior in the two regions (*no CPU saturation* and *CPU saturation*). In the *no CPU saturation* region the VNF does not shows any relevant packet loss, the throughput increases linearly with the input rate and the end-to-end delay is almost constant and below 1ms. Moreover, also the CPU consumption increases almost linearly with the input rate. In the *CPU saturation* region, the CPU is always consumed at 100% of its capacity and VNF performance degrades. The average delay shows a sudden steep increase (often higher than 6ms), and the VNF starts dropping a relevant amount of packets. Such a high packet loss is also related to throughput

[9]pps: packet per second

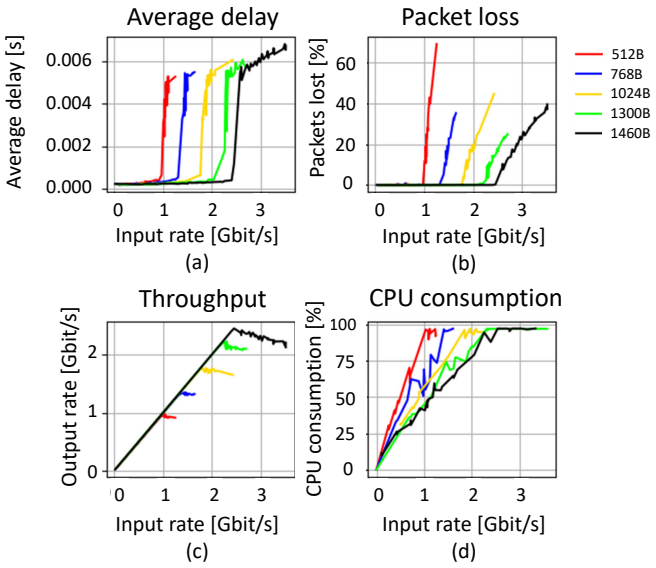


Fig. 9. Virtual router performance on ADVA TE considering different packet sizes.

reduction, which is more evident as the input rate increases. The black vertical line depicted in each sub-figure, at an input rate of around 2.5 Gbit/s, shows the CPU saturation break-point. This break-point gives an indication about the maximum input rate that can be handled by the virtual router without any relevant performance degradation.

**Packet size variation.** We used the TT to evaluate how a packet size variation impacts on virtual router performance. Results are shown in Fig. 9. Samples have been collected with packet sizes of 512 bytes, 768 bytes, 1024 bytes, 1300 bytes and 1460 bytes. All the other parameters have been set as in the previous experiment. It can be seen that the CPU saturation break-point is at lower input rates for smaller packets. This happens as a consequence of the increase in the number of packets to be processed at the same data rate and it is not surprising: the more packets need to be processed, the more CPU needs to be consumed. This important result shows that the average expected packet size is another crucial parameter to be considered while choosing the best configuration for a VNF.

**Flavor variation.** We performed tests by varying the flavor type (see Table I) of the VM where the virtual router is executed. In this way it is possible to evaluate the impact of the amount of assigned virtual resources on VNF performance. The results are shown in Fig. 10, and confirm the strict dependency between amount of assigned virtual resources and VNF performance, showing the advantages in assigning more virtual resources. Specifically, being the virtual router a CPU-bound VNF, what really matters is the number of assigned vCPUs for each flavor (i.e., 1, 2 and 3 respectively). We performed tests by varying virtual memory and virtual disk but, as expected, performance is not significantly affected and thus results are not reported.

By observing the differences between the *1 vCPU* (flavor 1)

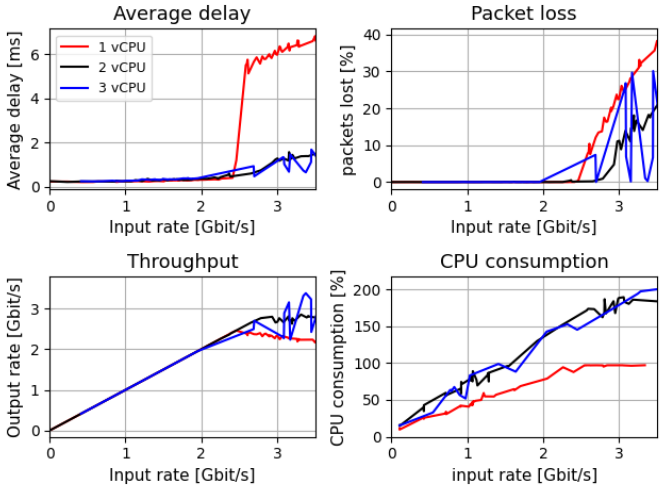


Fig. 10. Virtual router performance on ADVA TE considering different VNF flavors (packet size: 1460 bytes).

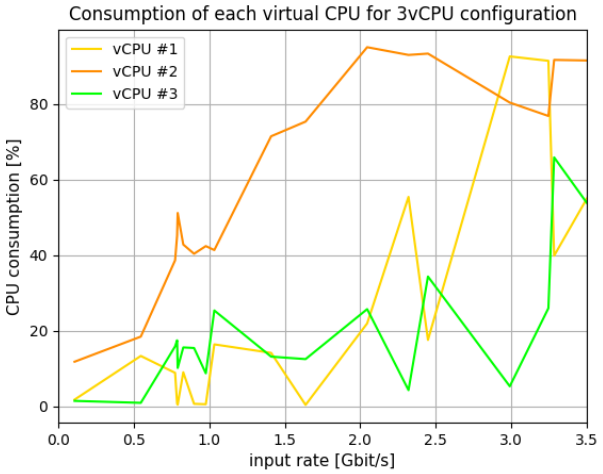


Fig. 11. Breakdown of CPU consumption for the *3 vCPU* case and virtual router.

and *2 vCPU* (flavor 2) cases, it can be noticed that an increase on the number of virtual CPUs reflects to a VNF performance improvement, and that the VNF is capable of guaranteeing no losses for a higher input rates. The mostly-affected metric is the end-to-end delay. At low input rates, it is almost the same for *1 vCPU* and *2 vCPU*, but for higher input rates it is consistently lower in the *2 vCPU* case. Also packet loss and throughput experience a lower degradation when *2 vCPU* are allocated.

However, some unexpected behavior occurs for the *3 vCPU* case, as the virtual router becomes unstable in the *CPU saturation* region. What can be seen is that the CPU consumption never overcomes 200%, meaning that the equivalent of two full vCPUs is utilized at most. This is due to the how virtual router is implemented, which is ineffective in balancing the load among the vCPUs. This is confirmed by looking at Fig. 11, which plots the average CPU consumption of each vCPU in the *3 vCPU* case, and discloses how the occupation is well below 100% for vCPU #2 and vCPU #3, while it is always close to 100% for vCPU #1.

In the *no CPU saturation* region some interesting behavior is seen concerning end-to-end delay, as shown in Fig. 12: it experiences a slight increase in *2 vCPU* and *3 vCPU* with respect to *1 vCPU*. This happens because of the *upscaling costs* that incur when a VNF has to balance the load across multiple CPU cores in multi-core implementations, as investigated in a previous paper [35]. In line with [35], Fig. 12 confirms an almost linear dependence between the number of adopted CPU cores and the average delay.

**Bursty traffic.** We performed various experiments with different duty cycles and input rates for the ON period. The results are consistent with what shown in the previous subsections. In OFF periods, the CPU consumption is negligible, while in ON periods the behavior depends on whether the input rate belongs to the *CPU saturation* or *no CPU saturation* region.

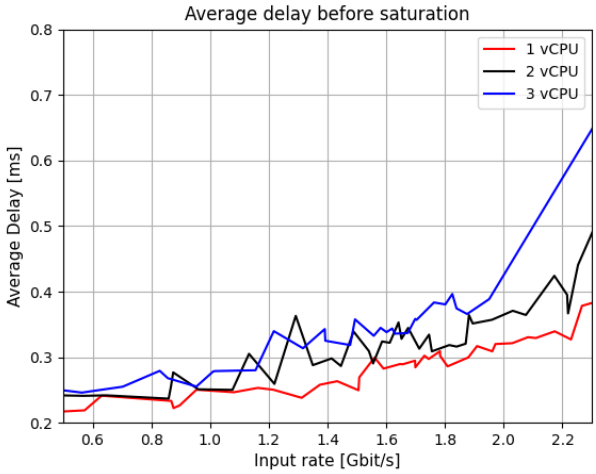


Fig. 12. Average delay in the *no CPU saturation* region for virtual router.

*b) BONSAI TE:* All the previously described experiments have been performed on the ADVA TE. We tested the virtual router also on the BONSAI TE, choosing as input parameters the same as for the tests reported in Fig. 8. In the BONSAI TE the vCPUs assigned to each VM (see Fig. 4) are physically shared by exploiting *hyper-threading*, which causes resource contention. This is needed because of the limited computational resources available in the computing node. As already mentioned, hyper-threading is instead not adopted in the ADVA TE due to the greater processing availability, which makes it possible a one-to-one mapping between vCPUs and physical cores (see Section IV-D for a deeper discussion).

Results are shown in Fig. 13. We can easily identify the CPU saturation break-point, which is at around 0.87 Gbit/s and around three times lower than that experienced on the ADVA TE. Additionally, worse performance is generally experienced. This is not surprising, being the BONSAI TE based on general-purpose open source software and exploiting hyper-

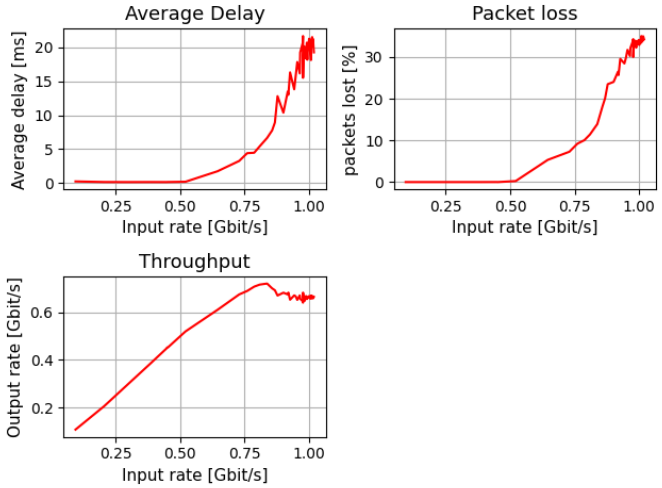


Fig. 13. Virtual router performance on BONSAI TE (Packet size: 1460 bytes).

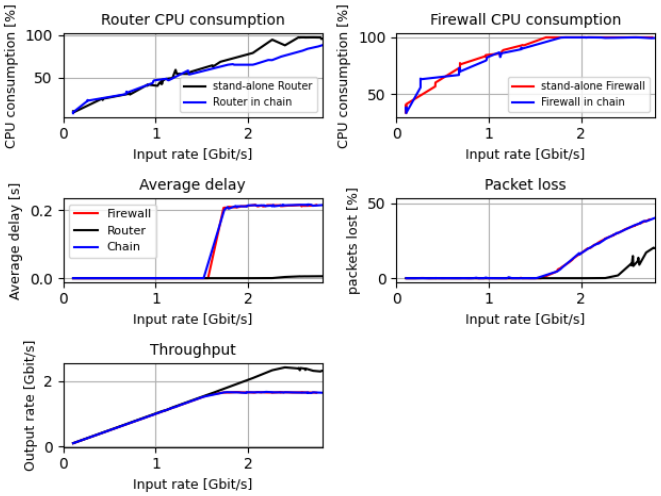


Fig. 15. *SFC linear* performance on ADVA TE (packet size: 1460 bytes).

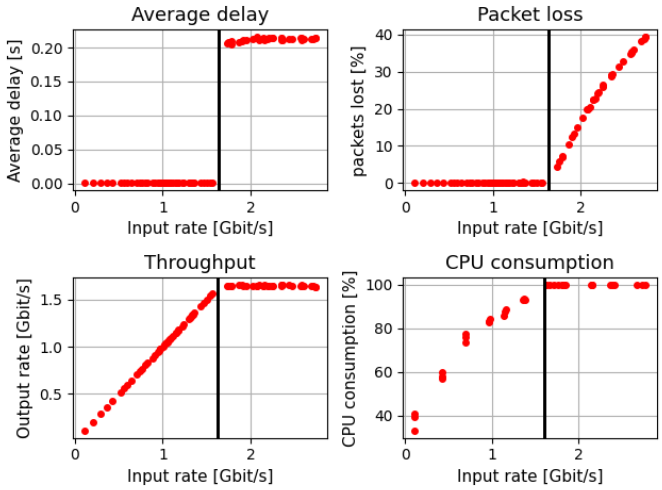


Fig. 14. Virtual firewall performance on ADVA TE (packet size: 1460 bytes).

threading. This confirms the need of accurately assessing the VNF performance on the specific NFV infrastructure where it will be deployed so that no service degradation, given the expected input traffic, is ensured. A similar performance degradation has been experienced also for virtual firewall and chained VNFs. However, we do not include additional results on the BONSAI TE due to space limitations. All the following tests have been performed on the ADVA TE.

### B. Virtual firewall

We tested the virtual firewall by generating UDP traffic (with packet size set to 1460 bytes) and test duration of 30 seconds. Fig. 14 shows the obtained results. Even though the functionalities offered by a virtual firewall are very different by those offered by a virtual router, a similar trend is experienced for each performance metric and for CPU consumption. However, we should note that both throughput and delay have different slopes compared to virtual router. The average delay shows a discontinuity around the CPU saturation break-point, leading to much higher latency in the *CPU saturation* region (around 200ms) with respect to *no CPU saturation* (around 1ms). Instead, throughput stabilizes around a value (close to 1.7 Gbit/s), which appears to be the maximum achievable. Another difference is the much lower input rate at which the CPU saturation break-point occurs, i.e., 1.5 Gbit/s instead of 2.5 Gbit/s. Clearly, a virtual firewall is more complex than a virtual router, as more operations need to be performed on each packet. This reflects on a higher CPU consumption at a same input rate.

The performance difference between the virtual router and the virtual firewall is an important indicator on how performing a detailed VNF characterization is important to avoid service disruptions due to incorrect VNF dimensioning given an expected average input rate.

### C. Service Function Chains

In this subsection we perform SFC performance characterization using the Monitoring System for the two SFCs specified in Section V.

*a) SFC linear (Fig. 6):* Fig. 15 reports the SFC performance when a *SFC linear* is deployed. Concerning CPU consumption, a comparison between each of the two VNFs deployed in the *stand-alone VNF* testing scenario (see Fig. 4) and in the *SFC* testing scenario (see Fig. 5) is reported. Concerning the performance metrics, the figure reports the value in the *stand-alone VNF* testing scenario, for each of the two VNFs, and for the *SFC* testing scenario.

The CPU consumption of the two VNFs follows almost exactly the same profile as in the *stand-alone VNF* testing case. This is due to one-to-one mapping between vCPUs and physical CPUs in the ADVA TE, which guarantees no resource contention. For what concerns end-to-end delay, throughput and packet loss, the SFC performance is very similar to the performance of virtual firewall. As the virtual firewall has its CPU saturation break-point at lower input rates than the

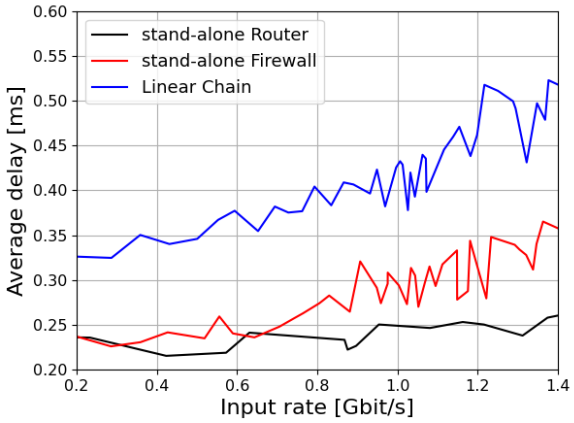


Fig. 16. Average delay in the *no CPU saturation* region for *SFC linear*, compared to the *stand-alone VNF* testing scenarios.

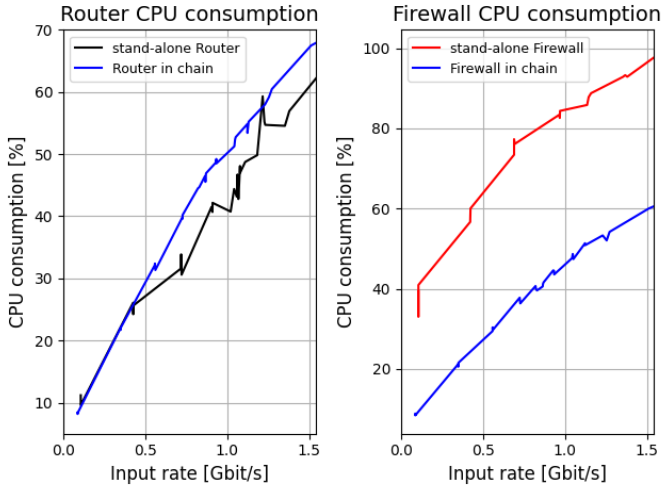


Fig. 17. Virtual router and virtual firewall CPU consumption for *Split SFC*.

virtual router, the saturation break-point of the linear chain is naturally determined by the virtual firewall. As a consequence, the packet loss and throughput of the chain follow exactly the same trend experienced by the virtual firewall. The average end-to-end delay shows instead an additive behavior, as better shown in Fig. 16, which illustrates the average delay of *SFC linear* in the *no CPU saturation* zone, compared to that obtained in the *stand-alone VNF* scenario.

It is than clear that, in the case of deployment of linear chains of CPU-bound VNFs, throughput and packet loss are dominated by the CPU-hungrier VNF, while the delay is the sum of the delays introduced by any chained VNF, plus any transmission and propagation delay (negligible in our tests).

*b) SFC split (Fig. 7):* Since the virtual firewall is the dominating VNF in *SFC linear*, a possible strategy to improve SFC performance could be deploying two parallel instances of such a VNF, as done for *SFC split*. However, by scaling out the virtual firewall an additional level of complexity is introduced, due to the need for traffic load balancing at the ingress virtual router and recombination at the egress virtual router. To avoid performance degradation, the virtual routers must be carefully dimensioned based on the expected amount of traffic to be processed.

Fig. 17 shows the CPU consumption as measured for the ingress virtual router and one of the Firewall in the *SFC split* and compared to the *VNF stand-alone* case, with focus on the *no CPU saturation* region. Very similar results are obtained for the other virtual firewall and virtual router.

As the input traffic is split equally among the two virtual firewall, their CPU consumption is reduced (slightly less than halved). Conversely, the virtual router CPU consumption slightly increases due to the additional load balancing operations that have to be performed. As we focused our tests on the *no CPU saturation* region, no packet loss is experienced and the throughput coincides with the input rate, as already extensively showed in the previous subsections. Also in this case, the average delay shows an additive behaviour, and it is slightly higher than in the case of *SFC linear*. In fact, one more VNF needs to be traversed by any packet. We do not report graphs for the performance metrics due to space constraints.

### D. Profiling System - VNF Profiling

In this subsection we test the capabilities of the Profiling System to profile VNFs in terms of delay, packet loss and throughput as a function of the input rate. The goal is to define analytical models that can be used to predict the behavior of the VNF given an expected amount of input traffic. We focus our evaluation on the virtual router and virtual firewall VNFs whose performance has been already characterized in the previous subsection.

*1) Virtual router:*

*a) Packet loss:* The Profiling System models the packet loss by choosing the following profiling functions:

- In the *no CPU saturation* region, an exponential function in the form $f_{packet\_loss,nonsat}(r) = ae^{b(r-c)}$ is chosen, where $a$, $b$ and $c$ are the scaling factors and $e$ the Euler's number. $r$ is the input rate.
- In the *CPU saturation* region, it is modelled by the function $f_{packet\_loss,sat}(r) = 100 \cdot (1 - \frac{d}{r-g})$ where $d$ and $g$ are the scaling factors. Being the packet loss expressed as a percentage, the maximum value it can reach is 100%.

Hence, the analytical model used to profile packet loss of the virtual router is the following:

$$f_{packet\_loss,VR}(r) = \begin{cases} ae^{b(r-c)} & No\ CPU\ saturation \\ 100 \cdot (1 - \frac{d}{r-g}) & CPU\ saturation \end{cases} \quad (2)$$

The left-hand side of Fig. 18 reports the profile as obtained using the Curve Fit method, together with the samples using by the fitting method, considering the *flavor #1* on the ADVA TE. The relative error between estimated and real value, for the collected samples, is always below 1.5% for any of the two regions. This confirms that the analytical model well profiles the packet loss of the VNF. The right-hand side of Fig. 18 shows the profile obtained from the samples collected on the

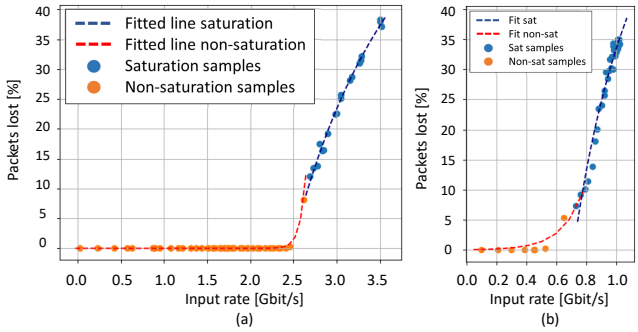


Fig. 18. Packet loss profile for the Virual Router on the ADVA TE (left) and BONSAI TE (right) (flavor #1, packet size: 1460 bytes).

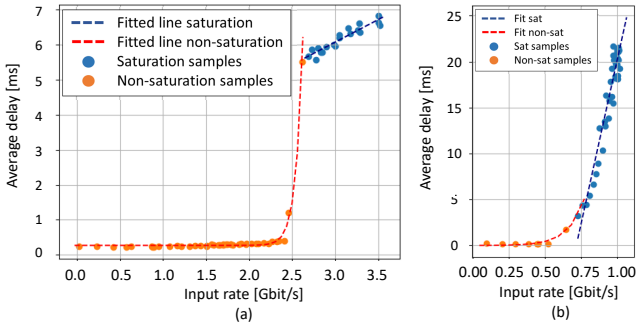


Fig. 19. Delay profile for the Virual Router on the ADVA TE (left) and BONSAI TE (right) (flavor #1, packet size: 1460 bytes).

TABLE IV
PACKET LOSS PROFILING FUNCTIONS' COEFFICIENTS.

| Testing Environment | Coefficients *No CPU saturation* | | | Coefficients *CPU saturation* | |
|---|---|---|---|---|---|
| | a | b | c | d | g |
| ADVA | 8.503e-17 | 14.95 | 0 | 1.722 | 0.754 |
| BONSAI | 0.042 | 6.999 | 0 | 0.569 | 0.143 |

TABLE V
AVERAGE DELAY PROFILING FUNCTIONS' COEFFICIENTS.

| Testing Environment | Coefficients *No CPU saturation* | | | Coefficients *CPU saturation* | |
|---|---|---|---|---|---|
| | a | b | c | d | g |
| ADVA | 12.697 | 2.484 | 0.272 | 1.286 | 2.214 |
| BONSAI | 8.999 | 0.595 | 7.15e-26 | 70.971 | -51.028 |

BONSAI TE. Even though the relative error is higher for some points (around 4%), the two functions described above are still the ones selected by the Profiling System. The values of the coefficients of the selected functions, for both TEs, are reported in Table IV.

*b) Average delay:* The following functions are selected by the Profiling System to model the average delay:

- In the *no CPU saturation* region an exponential function is chosen, as the one adopted to model packet loss in the previous paragraph.
- In the *CPU saturation* region, the linear function $f_{sat,delay}(r) = dr + g$ is adopted, where $d$ and $g$ are the gradient and the intercept respectively.

Hence, the analytical model used to profile the average delay of the virtual router is the following:

$$f_{delay,VR}(r) = \begin{cases} a + e^{b(r-c)} & No\ CPU\ saturation \\ dr + g & CPU\ saturation \end{cases} \quad (3)$$

The left-hand side of Fig. 19 reports the profile of the average delay on the ADVA TE. The relative error is always below 5% except around the CPU saturation break-point where it more pronounced (around 25%). Defining the profile of any performance metric around the break-point is challenging due to the very different behavior in the two regions. However, none of the other available profiling functions fitted better than the two considered. On the right-hand side the profile obtained from the samples collected on the BONSAI TE is shown. In this case, it is important to see that the relative error is a bit higher in the *CPU saturation* region (around 10%), and that the behavior is super-linear considering all the samples as a whole. In this case, modelling the profile using an exponential function on the the whole set of collected sample would have led to a better fit, but this is a corner case not experienced for any other metric and any other set up, so it cannot be generalized. The values of the coefficients of the selected functions are reported in Table V.

*c) Throughput:* The throughput is modelled by the Profiling System with a linear function for both the *no CPU saturation* and *CPU saturation* regions. The analytical model used to profile the throughput of the VNF thus is:

$$f_{throughput,VR}(r) = \begin{cases} ar + b & No\ CPU\ saturation \\ cr + d & CPU\ saturation \end{cases} \quad (4)$$

The profile obtained by applying the Curve Fit method to the throughput samples of the virtual router on the ADVA TE are illustrated in the right-hand side of Fig. 20. Larger errors in prediction are observed at higher input rates, when the VNF enters the CPU saturation region. The right-hand side of Fig. 20 shows the profile as obtained from the samples collected on the BONSAI TE. The profile well approximates the collected samples, with relative errors that are just slightly worse than those experienced on the ADVA TE. The functions' coefficient values are reported in Table VI.

*d) Flavor variation:* Having some models describing how the VNF reacts under an increasing traffic load is beneficial to decide the best *flavor* for the VM hosting a VNF, given the expected amount of input traffic. For the virtual router, the behaviour observed with *flavor #1* (1 vCPU) and *flavor #2* (2 vCPU) can be compared starting from the computed profiling models, showing how more assigned resources translate to better VNF performance. This is shown in Fig. 21. In the past subsections we did not show in detailing how packet loss, average delay and throughput are profiled for flavor #2. However, it can be seen that different profiling functions are chosen for packet loss by the Profiling System with respect to flavor #1.

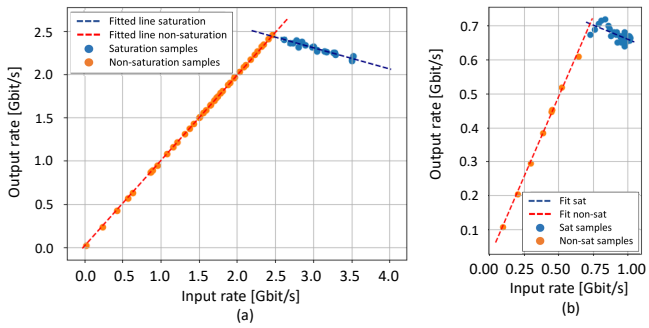


Fig. 20. Throughput profile for the Virual Router on the ADVA TE (left) and BONSAI TE (right) (flavor #1, packet size: 1460 bytes).

TABLE VI
THROUGHPUT PROFILING FUNCTIONS' COEFFICIENTS.

| Testing Environment | Coefficients *No CPU saturation* | | Coefficients *CPU saturation* | |
|---|---|---|---|---|
| | a | b | c | d |
| ADVA | 0.988 | 0.014 | -0.245 | 3.046 |
| BONSAI | 0.943 | 0.020 | -0.167 | 0.828 |

*2) Virtual firewall:* The Profiling System was also used to profile the virtual firewall from the samples collected on the ADVA TE. The results are illustrated in Fig. 22. It is important to note that the profiling functions selected by the Profiling System are among the ones selected for the virtual router. Even though more investigations are needed to generalize our finding, it seems that CPU-bound VNFs' performance profiles can be modelled using a very limited set of profiling functions.

## VII. CONCLUSION

VNFs have the potential to revolutionize the way we manage and optimize networks, but their performance can vary greatly depending on a number of factors such as resource configuration and traffic load. In addition, the underlying hardware, virtualization layer, and number of VNFs co-located can greatly impact the overall QoS experienced. As a result, deploying chains of VNFs, known as Service Chains, can sometimes lead to performance degradation due to their virtualized nature. To fully harness the power of VNFs, it's crucial that we understand and address these performance challenges. This paper delves into understanding the impact of resource allocation, consumption, and traffic load on the performance of VNFs through various performance metrics. Thanks to the use of real-world network testbeds, we were able to provide a detailed analysis of how different factors affect VNF performance, and to develop analytical functions that can accurately describe this impact. We found that the CPU was the main bottleneck at the VNF layer, and that the VNFs performance degradation was highly dependent on traffic features. The findings show that by adding more vCPUs to the VNF, an improvement is registered both in the saturation break-point position and also in the average delay experienced. However, some costs must be faced due to the load balancing procedure and the introduction of instability in the observed VNF behavior. We believe that the findings of this work can

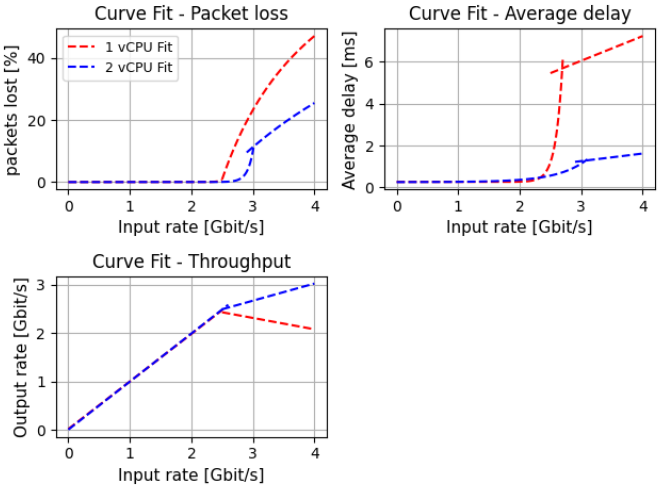


Fig. 21. Comparison between the profiles obtained for flavor #1 and flavor #2 (virtual router).

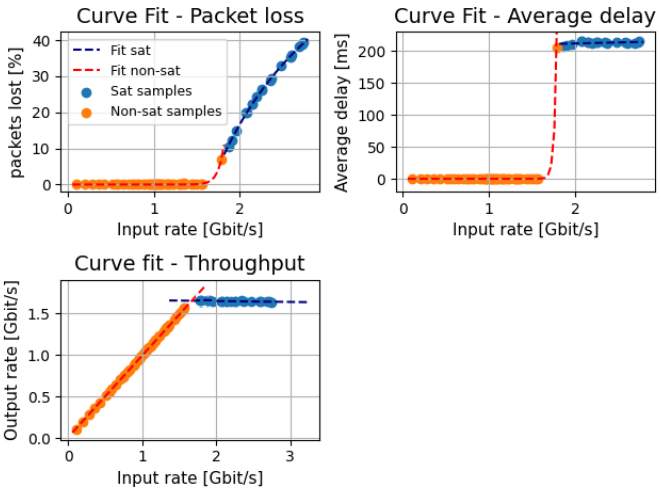


Fig. 22. Performance metrics profile for the Virual Firewall on the ADVA TE (flavor #1, packet size: 1460 bytes).

be used as a support during the deployment of SFCs in various network contexts. As a next step, we plan to analyze how the performance of the VNFs is affected by compute-intensive tasks across all computing cores. This analysis will factor in variables such as cache memory usage, hardware temperature, and memory sharing criteria that are dependent on specific hardware and software configurations. By doing so, we will provide an accurate evaluation of the impact of compute-intensive tasks on VNF performance.

## REFERENCES

[1] I. Group, "Network Function Virtualization (NFV) Market: Global Industry Trends, Share, Size, Growth, Opportunity and Forecast 2022-2027. [Online] https://www.giiresearch.com/report/imarc1120187-network-function-virtualization-nfv-market-global.html," IMARC, Tech. Rep., 2022.

[2] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware vnf placement and chaining based on a flexible resource allocation approach," in *2017 13th International Conference on Network and Service Management (CNSM)*, 2017, pp. 1–7.

[3] G. Garg, V. Reddy, A. Antony Franklin, and B. R. Tamma, "Davis: A delay-aware vnf selection algorithm for service function chaining," in *2019 11th International Conference on Communication Systems Networks (COMSNETS)*, 2019, pp. 436–439.

[4] M. Falkner, A. Leivadeas, I. Lambadaris, and G. Kesidis, "Performance analysis of virtualized network functions on virtualized systems architectures," in *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, 2016.

[5] W. Konikiewicz and M. Markowski, "Analysis of performance and efficiency of hardware and software firewalls," *Journal of Applied Computer Science Methods*, vol. 9, 2017.

[6] N. Ghrada, M. F. Zhani, and Y. Elkhatib, "Price and performance of cloud-hosted virtual network functions: Analysis and future challenges," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 482–487.

[7] S. Van Rossem, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, "Profile-based resource allocation for virtualized network functions," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1374–1388, 2019.

[8] A. Mestres, E. Alarcón, and A. Cabellos, "A machine learning-based approach for virtual network function modeling," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2018.

[9] S. Schneider, N. P. Satheeschandran, M. Peuster, and H. Karl, "Machine learning for dynamic resource allocation in network function virtualization," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020.

[10] M. Peuster and H. Karl, "Understand your chains: Towards performance profile-based network service management," in *2016 Fifth European Workshop on Software-Defined Networks (EWSDN)*, 2016, pp. 7–12.

[11] ——, "Profile your chains, not functions: Automated network service profiling in devops environments," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017, pp. 1–6.

[12] S. Dräxler, M. Peuster, M. Illian, and H. Karl, "Generating resource and performance models for service function chains: The video streaming case," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 318–322.

[13] S. Van Rossem, T. Soenen, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, "Adaptive amp; learning-aware orchestration of content delivery services," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 77–84.

[14] S. Van Rossem, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, "Vnf performance modelling: From stand-alone to chained topologies," *Computer Networks*, vol. 181, p. 107428, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620311178

[15] J. Nam, J. Seo, and S. Shin, "Probius: Automated approach for vnf and service chain analysis in software-defined nfv," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3185467.3185495

[16] L. Cao, P. Sharma, S. Fahmy, and V. Saxena, "Nfv-vital: A framework for characterizing the performance of virtual network functions," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, 2015, pp. 93–99.

[17] R. V. Rosa, C. E. Rothenberg, and R. Szabo, "Vbaas: Vnf benchmark-as-a-service," in *2015 Fourth European Workshop on Software Defined Networks*, 2015, pp. 79–84.

[18] R. V. Rosa, C. Bertoldo, and C. E. Rothenberg, "Take your vnf to the gym: A testing framework for automated nfv performance benchmarking," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 110–117, 2017.

[19] R. V. Rosa and C. E. Rothenberg, "Automated vnf testing with gym: A benchmarking use case," in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, 2018, pp. 1–2.

[20] N. Schmitt, J. von Kistowski, and S. Kounev, "Towards a scalability and energy efficiency benchmark for vnf," in *Performance Evaluation and Benchmarking for the Analytics Era*, R. Nambiar and M. Poess, Eds. Cham: Springer International Publishing, 2018, pp. 41–54.

[21] Y. Sharma, M. G. Khan, J. Taheri, and A. Kassler, "Performance benchmarking of virtualized network functions to correlate key performance metrics with system activity," in *2020 11th International Conference on Network of the Future (NoF)*, 2020, pp. 73–81.

[22] G. Piao, P. K. Nicholson, and D. Lugones, "Env2vec: Accelerating vnf testing with deep learning," in *Proceedings of the Fifteenth European Conference on Computer Systems*, ser. EuroSys '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3342195.3387525

[23] I. J. Sanz, D. M. F. Mattos, and O. C. M. B. Duarte, "Sfcperf: An automatic performance evaluation framework for service function chaining," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–9.

[24] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[25] D. Emma, A. Pescape, and G. Ventre, "Analysis and experimentation of an open distributed platform for synthetic traffic generation," in *Proceedings. 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004. FTDCS 2004.*, 2004, pp. 277–283.

[26] A. Botta, A. Dainotti, and A. Pescapé, "Do you trust your software-based traffic generator?" *IEEE Communications Magazine*, vol. 48, no. 9, pp. 158–165, 2010.

[27] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, "D-itg distributed internet traffic generator," in *First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings.*, 2004, pp. 316–317.

[28] V. Sit, M. Poulin-Costello, and W. Bergerud, *Catalogue of curves for curve fitting*. Citeseer, 1994.

[29] "Ensemble Suite, [online] https://www.adva.com/en/resources/downloads/data-sheets/ensemble-cloud-suite."

[30] "Ensemble Connector, [online] https://www.adva.com/en/products/network-virtualization/ensemble-connector."

[31] "OpenStack, [online] https://www.openstack.org/."

[32] "Ensemble Director, [online] https://www.adva.com/en/products/network-virtualization/ensemble-virtualization-director."

[33] "Ensemble Orchestrator, [online] https://www.adva.com/en/products/network-virtualization/ensemble-orchestrator."

[34] L. Mamushiane, A. A. Lysko, T. Mukute, J. Mwangama, and Z. D. Toit, "Overview of 9 open-source resource orchestrating etsi mano compliant implementations: A brief survey," in *2019 IEEE 2nd Wireless Africa Conference (WAC)*, 2019.

[35] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, 2015, pp. 191–197.

[36] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware vnf placement for service-customized 5g network slices," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 2449–2457.

[37] C. Zeng, F. Liu, S. Chen, W. Jiang, and M. Li, "Demystifying the performance interference of co-located virtual network functions," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018.

[38] "Mikrotik CHR, https://wiki.mikrotik.com/wiki/Manual:CHR."

[39] "Palo Alto Networks Firewall VM Series, https://docs.paloaltonetworks.com /vm-series.html."

[40] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu, "Nfp: Enabling network function parallelism in nfv," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, p. 43–56.

[41] Y. Zhang, B. Anwer, V. Gopalakrishnan, B. Han, J. Reich, A. Shaikh, and Z.-L. Zhang, "Parabox: Exploiting parallelism for virtual network functions in service chaining," in *Proceedings of the Symposium on SDN Research*, 2017, p. 143–149.