On the Performance Benefits of Heterogeneous Virtual Network Function Execution Frameworks

Haruna Umar Adoga
School of Computing Science
University of Glasgow
Glasgow, UK
h.adoga.1@research.gla.ac.uk

Yehia Elkhatib
School of Computing Science
University of Glasgow
Glasgow, UK
yehia.elkhatib@glasgow.ac.uk

Dimitrios P. Pezaros
School of Computing Science
University of Glasgow
Glasgow, UK
dimitrios.pezaros@glasgow.ac.uk

***Abstract*—As the adoption of softwarized network functions (NFs) keeps growing, we evaluate the performance benefits of SDN-aware data-plane implementations when compared to diverse acceleration and process-based NFV frameworks. Typical network functions have been implemented using four alternative frameworks scenarios, an SDN-aware software switch (data-plane), a virtual machine (VM), a Data-Plane Development Kit (DPDK) NF, and a containerized NF. Results from our experiments show that the data-plane NF implementation yields much higher bandwidth and packets per second (pps) rates. The bandwidth obtained is 14% more than the user-space scenario while retaining CPU utilization. The DPDK NFs in our evaluation can process packets at a much higher rate for 64B packets, on a single CPU core, which is 7 times higher than the containerized NF implementations, also tied to a single core. Our results also show the performance gains from deploying virtual network functions on heterogeneous frameworks.**

***Index Terms*—SDN, Data-plane, Network Function Virtualization, Network Softwarization, Containerized Network Functions, DPDK, Virtual Network Functions, Virtual Machines, Performance Comparison, NFV frameworks.**

I. Introduction

The flexibility offered by network function virtualization (NFV) and software-defined networking SDN allows service providers to deploy Virtual Network Functions (VNFs) in a flexible and scalable manner, with the simple goal of meeting the packet processing requirements of applications. The important components that must be present for NFV to be implemented are: (i) the NFV Infrastructure (NFVI), which consists of a virtualization layer for hosting VNFs; (ii) the abstracted network function, which runs on the NFVI; (iii) the NFV Management and Orchestration framework (NFV-MANO) for the management and orchestration of virtualized resources and the VNF lifecycle [1].

The data-plane of the network serves as a suitable alternative for implementing VNFs. Hence network operators can leverage the data-plane as well as process-based NFV frameworks for deploying softwarized network functions. Doing so comes with the benefit of cost reduction on procuring high-end servers for NFV deployment, and reduction also on the overheads for compute-bound network functions (Table I). Typically, VNF deployment is carried out using VMs [2], containers [3], Unikernels [4], Click-based processing elements, and Data-Plane Development Kit (DPDK) which uses zero-copy and kernel bypass to speed up packet processing [5].

At the same time, service providers need to understand how to leverage the network data-plane for the deployment of network functions. In this paper, we carry out a practical performance evaluation on a real testbed to compare the implementation of representative network functions as part of an SDN-aware data-plane against having such functionality abstracted and implemented as a VNF running on a commodity server – using heterogeneous VNF execution frameworks. One of the problems is to find out where to best implement these functions, which should allow for faster service deployments and for specific applications to be deployed using the suitable processing pipeline in the network.

We make the following contributions:

- We present a classification of virtual network functions (Table I) based on the type of operations they perform on packets. After motivating the need for data-plane NF deployments, we designed a testbed that is suitable for getting comparative results (for both scenarios, i.e., software switch and separate virtualized functions on commodity servers).
- A significant difference in the performance of heterogeneous virtualized packet processing frameworks was identified. We achieved this by carefully evaluating packets of various sizes at the data-plane, compared to having equivalent network functions on commodity servers.
- Our discussion in §II and §IV sheds light on what network functions are suitable for both scenarios, based on our evaluation. We analyzed and presented our findings, with packet processing acceleration frameworks, using Intel DPDK and lightweight containers, which shows a significant improvement in the performance of the network functions evaluated.

Next, we present the design of the testbed used for the first performance evaluation, which also contains a classification of network functions and their complexity. The evaluation of both scenarios is presented in Section III. A high-performance setup and the results obtained using this approach are presented in Section III-C, with DPDK and Docker containers. In Section

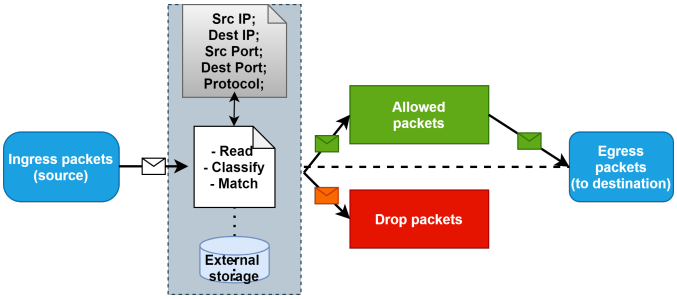

Figure 1. An overview of the operation of I/O-bound NFs, e.g., ACL/Firewall, Header Classifier, and caching. Their tasks involve IP header lookup and data store access.

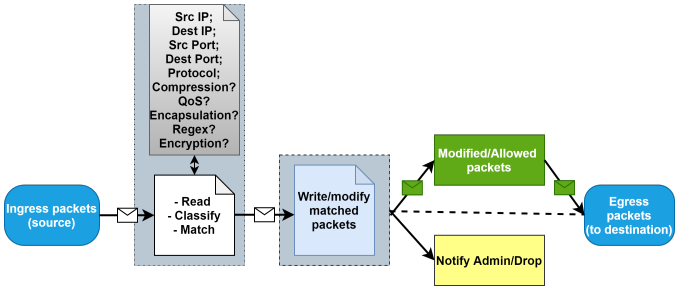

Figure 2. An overview of the operation of compute-bound NFs, e.g., NAT, IDS and QoS optimization. Their tasks involve more complex types of processing performed on packets.

IV, we shed some light on what network functions are ideal for deployment using the environments considered, including some design implications. We present the related work in Section V, and conclude the work in Section VI.

II. Design - VNF Classification and Complexity

In Table I, we capture some of the most popular NFs that can be abstracted and implemented in software, and their classification as discussed below. We classify NFs based on how they process packets and the typical operations they carry out along the processing pipeline. We use Figure 1 to illustrate how packets traverse representative NFs that perform similar operations on received packets, such as an ACL/ Firewall. These NFs perform IP header lookups to read fields such as source and destination addresses and port numbers, and the protocol in use. In the case of an ACL/Firewall, packets are checked against predefined match-action rules to either discard or forward them.

Figure 2 illustrates the typical mode of operation of the second group of representative NFs (Table I), i.e., NFs that perform operations beyond basic packet header lookup against predefined match-action rules. Packets traversing these NFs go through more fine-grained inspection of payload data, depending on the type of NF. A typical operation might involve performing regex matches, compression parameters for Quality of Service optimizations or encryption parameters.

We arrived at the conclusion of Table I after carefully considering the behavior of virtual network functions and frameworks that were implemented in notable works such as [6]–[9], where representative virtual network functions such as IDS/DPIs and L3 routers exhibit compute-bound characteristics, while others such as ACL/firewalls are I/O-bound due to the nature of operations they carry out on packets traversing them [10], [11].

Answering the question about what network functions are suitable for data-plane deployments is important, because the SDN-aware data-plane has some limitations alongside benefits in terms of its capabilities which we explore in the remainder of the paper.

A. Testbed design

We describe the testbed we designed to evaluate the performance of heterogeneous VNF execution frameworks. We started our design first by identifying the NFs that can be equally deployed at the data-plane level of an SDN-aware software switch and as a separate user-space VNF entity in the network. Looking at Table I, with their typical mode of operation and complexity in mind, NFs that are I/O-bound can be deployed using both approaches for a fair evaluation. Also, for any significant performance degradation/improvement to be observed, at least two different VNF implementations can produce comparable results.

Two typical scenarios are a Longest Prefix Match (LPM) router and a stateless firewall that, on the one hand, are configured using predefined match-action rules on a virtual switch, and on the other hand having the same functionalities running as separate entities (i.e. VNFs) in the network. This aspect of our testbed represents the required abstracted NF component of the ETSI NFV specification [1].

B. Virtual switches

SDN-aware virtual switches such as the Open vSwitch (OvS) have a data-plane component that handles traffic forwarding. In terms of NFV implementations, the network data-plane can be represented using a production-grade virtual switch [12]. Virtual switches can be configured to emulate typical middlebox functionalities such as firewall/access control, Network Address Translation (NAT), routing, and switching. When functioning as a router, it matches the longest source and destination IP prefix and forwards packets, using specified ports. Firewall functionality matches IP addresses and specific TCP/UDP port numbers, and denies or permits packets (based on match-action flow rules).

C. Data-plane scenario design

For performance reasons, hosting all VNFs (middleboxes) in a single server can lead to resource contention [13]. This informed our decision to isolate the NFs from the packet generator (Figure 3) by making use of multiple (two) physical servers in our first testbed. We investigated both NFs using two servers which are connected back-to-back to also eliminate overheads that are caused by switches. NIC speeds were set to 1000Mbps for consistency throughout our evaluations.

The testbed is depicted in Figure 3. We configured an OvS virtual switch on a physical server running Linux kernel 5.4.0-59-generic, on Ubuntu 20.04 (DUT in Figure 3). The CPU

Table I
VIRTUAL NETWORK FUNCTIONS OPERATIONS AND CLASSIFICATION

VNF	Description	Compute-bound	I/O-bound
DPI – IDS/IDP	Traffic logging and inspection, stateful/stateless security	✓	✗
Virtual Private Network	User traffic encryption and security	✓	✗
Network Address Translation (NAT)	Private to public IP translation and vice versa	✓	✗
WAN Optimizer	ISP traffic optimization for QoS	✓	✗
Router	Packet routing based on L3 details	✓	✗
L2 Switch	Packet switching based on MAC addresses	✓	✗
Load Balancer	Traffic load balancing, based on application level policies	✗	✓
ACL/Firewall	Device or application level access control	✗	✓
Caching	Performance improvements for better QoS/QoE	✗	✓
IPv4/IPv6 proxy	IPv4 to IPv6 connections proxy	✗	✓
Header Classifier	Classification, based on IP header fields	✗	✓

is an Intel® Core™ i7-8700 @ 3.20GHz, with 6 CPU cores. Virtual interfaces are set to 1GB speeds for the software switch scenario. For the firewall NF configuration, up to 15 different match-action firewall rules were pre-installed on the virtual switch to process HTTP (port 80) traffic that is destined for the web-server or the FTP server (ports 20 and 21) at any given time in our scenario. Paravirtualised (virtio-net) NICs were used on the VMs for better performance. Packets destined for the HTTP and FTP servers were filtered to allow or drop, based on protocol type, data link and transport layer protocols. We reused the same setup for the router NF tests this time by creating match-action rules to route packets coming from the packet generator and destined for the FTP and HTTP service based on layer 3 information using LPM lookup.

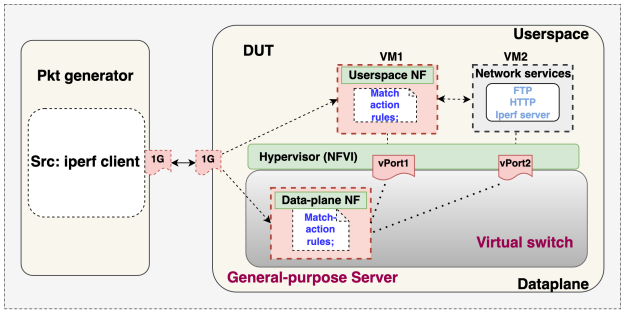

Figure 3. Ingress packets destined for the FTP and HTTP services on VM2, are made to traverse the data-plane or user-space network function at any given time. User-space function runs on VM1 while data-plane rules are inserted on the virtual switch.

D. VNF scenario design

For a comparable scenario to be achieved, the commodity server (DUT) hosting the network functions setup as our NFVI is the same server used for the data-plane scenario (§II-C). VMs running on Kernel-based Virtual Machine (KVM) with minimal versions of the Linux kernel 3.10.0 were used, with virtual network interfaces set to 1GB speeds. We configured a kernel-based firewall function on VM1, an implementation of equivalent functionality to the data-plane firewall rules in terms of the operations carried out on received packets, i.e., matching packets based on source and destination ports or IP addresses. A simple webserver rendering a webpage on port 80 and an FTP server listening on ports 20 and 21 (VM2) is also configured to receive requests. We configured the router NF, a simple Linux kernel-based router to route packets between two separate networks, with the packet generator and DUT server connected back to back (Figure 3).

III. ASSESSING VIRTUAL NETWORK FUNCTIONS PERFORMANCE

We now present the results from the first testbed in our work, described in the previous section. We measure representative performance parameters such as throughput, packet rate and CPU utilization for each scenario. The presented bandwidth measurements in Figure 4 are averaged over 10 runs each using `iperf3` (varying the `len` option to accommodate different packet sizes).

A. Data-plane performance

The supported bandwidth achieved on the data-plane firewall scenario was observed to be only 226Mbps for 64B packets. Meanwhile, the largest packets (1500B) were transferred at a bandwidth of 957Mbps, which is about 96% of the line rate. The data-plane router NF can process 185Mbps at small packet size (64B), and 905Mbps for large ones (1500B). To put the results into perspective, Table II and Figure 4 summarize the bandwidth for the data-plane and user-space implementations of firewall and router NFs, with various packet sizes. The data-plane implementation outperforms the user-space scenario, yielding about 14% more bandwidth for 1500B packets (Figure 4). Both NFs (router and firewall) produced a reasonably high bandwidth when compared with the user-space VNF implementations. The drop in performance, which was observed with small packet sizes is due to the increase in packet-per-second processing rate, which is seen as a drawback in most software-based middleboxes.

B. VNF performance

Like the data-plane scenario, the performance while sending various packet sizes to the HTTP and FTP services with packets traversing the user-space network function was measured and plotted in Figure 4. With the firewall functionality in place and the client machine (packet generator) attempting to gain access to the server hosting the services (VM2), the bandwidth

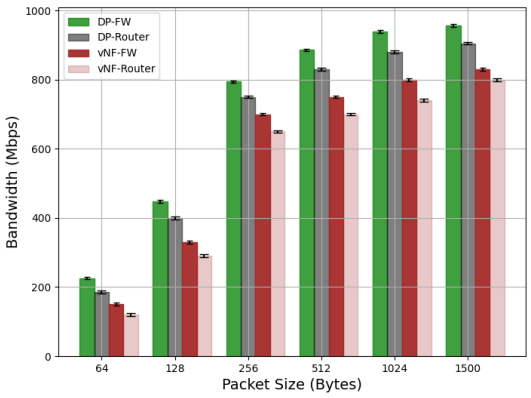

Figure 4. Mean bandwidth rates for data-plane and user-space deployments of network functions. The data-plane functions (both firewall and router), consistently outperforms user-space functions in this scenario.

Table II
MEAN BANDWIDTH RATES (IN MBPS) FOR DATA-PLANE AND VNF DEPLOYMENTS.

Packet size (B)	DP-FW	VNF-FW	DP-Router	VNF-Router
64	226	150	185	120
128	448	330	400	290
256	794	700	750	650
512	886	750	830	700
1024	939	800	880	740
1500	957	830	905	800

peaked at 150Mbps (for 64B packets), which is a significant decrease from 226Mbps observed in the data-plane scenario. Similar tests were carried out for the router NF, producing a bandwidth of 120Mbps for 64Bytes packets. MTU-sized packets produced 800Mbps, which is about about 11% below the bandwidth reached in the corresponding data-plane router scenario i.e, 905Mbps.

For MTU-sized packets, bandwidth of 830Mbps (for the firewall NF scenario) was achieved, which is 13% less than the data-plane firewall scenario. For this first set of experiments, we observed that the data-plane scenarios outperform our user-space VNF implementations, in terms of maximum bandwidth and CPU utilization, this also holds true for the packets per second rates (Figure 4).

CPU utilization of the commodity server hosting the testbed is depicted in Figure 5, which gives the processing cost of having the NFs deployed in both scenarios. By implication, in terms of CPU utilization cost, the data-plane implementation of our firewall requires less resources.

The CPU utilization for VNF and data-plane scenarios were observed for the router and firewall NFs used in our evaluations. CPU spikes were much higher in the firewall VNF scenario (up to 120%, which is exactly 10% of the overall CPU cores on the server. Higher CPU utilization was achieved for smaller packet sizes, which decreases as packet size increases (Figure 5).

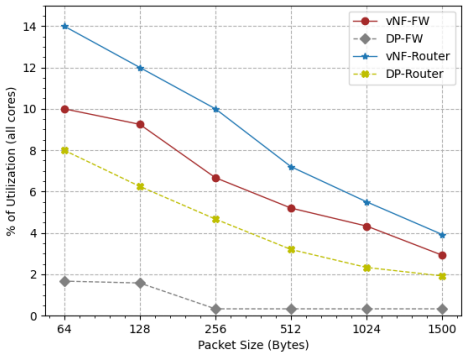

Figure 5. CPU utilization of the DUT – data-plane firewall and router (grey and yellow lines) maintain much lower CPU utilization, when compared with user-space functions (blue and red lines).

C. DPDK And Containerised Network Functions

In this section, we present our findings using an implementation that handles line-rate packet generation and processing. This helps in finding out the behaviour of the same representative network functions on a high performance vNF execution framework and lightweight containers. We carried out this part of the experiment using the Intel Data-Plane Development Kit (DPDK)[1] framework and compared the packet processing rates with containerized NFs scenario which involves the deployment of the same network function on Docker containers.

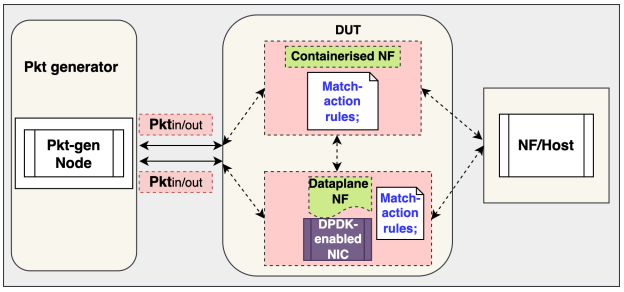

Figure 6. DPDK and Containerised NFs testbed - packets are generated and made to traverse the containerised NF or Dataplane NF at any given time. Match-action rules are inserted, based on type of functionality (a router or firewall function).

This part of our evaluation leverages the OpenNetVM NFV framework, which is ideal for building high-speed vNFs that can handle line-rate packet processing using zero-copy and the DPDK poll-mode driver, rather than interrupts [5]. The OpenNetVM manager represents the NFV MANO component, as it is responsible for NF life-cycle management. We made use of a commodity servers (depicted in Figure 6), one as a packet generator with `pktgen` packet generator, the other as the Device Under Test (DUT), with network functions built on the OpenNetVM framework. The CPU on the DUT is an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, with 6 cores (without hyper-threading), for improved performance per core.

[1]https://www.dpdk.org/

We made use of pktgen-dpdk[2], which generates packets at line-rate, and observe the packet processing rate per second, as packets are processed by the firewall NF. Note that our ability to scale the number of NFs in this part of our evaluation, depends on the number of available CPU cores, as each NF is assigned a single CPU core to run. As with the DPDK NFs, we also restricted our containerized NFs to run on single CPU cores for a fair comparison.

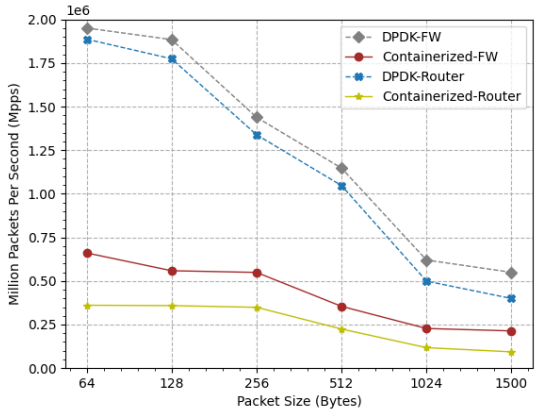

Figure 7. Million Packets Per Second – DPDK and Containerized NF. The DPDK firewall NF processed about 2Mpps. All NFs are tied to separate CPU cores, for performance isolation and measurement consistency.

We run the NFs as depicted in Figure 6, for the firewall NF scenario, we initialised the NF by creating an initial batch of 16,000 packets from the packet generator, which increases while the NF runs, and the packets are sent back to the RX queue of the generating NF (for measurements). This scenario was first carried out for the firewall NF, and using the router LPM NF. The results are as shown in Figure 7, where the packets per second rates obtained are depicted for the minimum-sized (64B) and MTU-sized (1500B) packets.

IV. Discussion and Recommendations

For us to fully appreciate how deployment choices by network operators can affect NFV performance, we carried out our evaluation with multiple physical commodity servers. Also, containerized and DPDK NF scenarios benefited from our design choice of running NFs on separate CPU cores.

Another design consideration worth mentioning is the behaviour of NFs that are built using kernel-bypass frameworks such as the Intel DPDK library, the number of available CPU cores is especially relevant when service providers decide to adopt this approach. This is even more so as NFs are often tied to available cores, and depending on the scenario, some cores need to be made available for the NF manager or controller.

In our first testbed (Figure 3), although this is not a claim that is generalizable for all dataplane implementations out there, the data-plane deployment consistently outperforms the user-space VNF when it comes to achieved bandwidth in the environment we have evaluated in our work. This is particularly noticeable with larger packet sizes, and more so in the case of the firewall VNF tests. This consistency also holds true for CPU utilization. By implication, service providers can leverage the data-plane of the network, especially in situations where compute-intensive NFs are deployed. The overall performance of network functions evaluated increased significantly using DPDK, even for 64B packets. Service providers can also leverage this for deploying compute-intensive vNFs such as a stateful IDS, for faster packet processing.

The mode of operation of some network functions in Table I such as WAN optimisers involves carrying out tasks such as traffic compression/decompression, caching, floating-point operations and sometimes content duplication [14]. Such operations are not readily supported by current data-plane technology hence service providers can leverage other heterogeneous deployment options.

One of the practical scenarios that would benefit from the results of our work is a network service provider that provides a set of network functions comprising a gateway firewall and an Intrusion Detection System (IDS). This chain of network functions which are currently being implemented at the user-space of the network can be split in a hybrid manner between the data-plane and user-space to improve performance and reduce processing and service deployment costs.

A network function such as an IDS which is stateful, requires the use of security modules, and is computationally intensive [9]. This will benefit from the available resources (memory, CPU and I/O) at the user-space of commodity servers while having the firewall functionality or other lightweight functions deployed at the network data-plane. Also, in some existing and emerging Edge network use cases such as e-healthcare, self-driving cars and mixed reality (MR), depending on application profiles and requirements, a hybrid deployment scheme can be employed. We can achieve better performance by delegating simple, time-critical operations to the data-plane and leaving only more complex stateful and resource-hungry processing for user-space vNFs; which should also help eliminate processing redundancy.

V. Related Work

The effect of having concurrent NFs run on a commodity server was characterized by Pitaev *et al.* [15], using OvS-DPDK, SR-IOV, and FD.IO VPP. They were able to show that having multiple VNFs deployed on a single host produces some performance bottlenecks, especially as the number of VNFs are increased. We evaluated our NFs implementation using two physical servers connected back-to-back, and report on some of the benefits of doing so. We carried out our experiments in the context of SDN, where match-action rules are inserted to the data-plane of virtual switches.

Rasoul *et al.* [16] evaluated the performance of virtualization technologies (unikernels, VMs and containers) for NFV deployments, with a focus on edge networks. They deployed two services (Redis and Apache) to examine the behaviour of these virtualization environments. They recommended the

[2]https://git.dpdk.org/apps/pktgen-dpdk

use of unikernels for applications with high context switching between userspace and kernel-space. VMs and containers demonstrated stable performance behaviour when compared to unikernels, with VMs having additional hypervisor overheads. We take a different approach by considering not only the effect of NF deployment on VMs and containers but using heterogeneous execution frameworks with commonly used network functions (a router and a firewall).

SDN-VNF performance was analysed by Gedia *et al.* [17], in the context of VM and container implementations. They considered parameters such as CPU and memory utilization, service provisioning time and throughput. In their work, an ONOS SDN controller was deployed using containers and VMs, they attempted to answer a question that would help network operators in choosing the ideal platform for hosting VNF/SDN services. Their results show much better performance using the containerised NF approach. This informed our decision to further explore what is obtainable using a high performance packet processing approach such as DPDK and consider the implementation of NFs that incorporates the typical behaviour of middleboxes found in service provider environments rather than controllers.

VI. Conclusions

We have evaluated the performance of different VNF implementation alternatives and motivate the deployment of network functions as part of the SDN-aware data-plane of the network for better service delivery, especially where latency-sensitive and bandwidth-intensive network functions are to be implemented. Two unique testbeds were carefully designed, to implement commonly used network functions (a firewall and a router) using the data-plane component of a production-grade virtual switch, and network functions deployed on separate virtualized processing elements. We also considered a high-performance scenario, by using DPDK and Docker containers for deploying the same network functions, which shows a significant increase in the performance of the network functions considered.

There are different ways in which this work could be built upon. Firstly, the dimensions of the evaluation can be extended by evaluating the data-plane scenario on a programmable hardware switch, which provides the option of offloading packet processing to SmartNICs, and also provides more in-network processing options, as these technologies are gradually gaining traction in service provider networks. Secondly, we envisage hybrid deployment scenarios where service providers deploy NFs using both data-plane and user-space processing elements, depending on application profiles, NF complexity and resource availability in the network infrastructure. In such circumstances, the outcomes of our work can be used to build a knowledge base to be used by a deployment decision support system.

Acknowledgments

This work was supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/N033957/1, the PETRAS National Centre of Excellence for IoT Systems Cybersecurity, which has been funded by the UK EPSRC under grant number EP/S035362/1, and the Petroleum Technology Development Fund (PTDF) Nigeria, grant 1563/19.

References

[1] F. Asquini, A. Bujari, D. Munaretto, C. E. Palazzi, and D. Ronzani, "An ETSI NFV implementation for automatic deployment and configuration of a virtualized mobile core network," in *International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2021, pp. 357–362.

[2] K. N. Qureshi, E. Ahmad, M. Anwar, K. Z. Ghafoor, and G. Jeon, "Network functions virtualization for mobile core and heterogeneous cellular networks," *Wireless Personal Communications*, vol. 122, no. 3, pp. 2543–2559, 2022.

[3] Z. Huang, N. Samaan, and A. Karmouch, "A novel resource reliability-aware infrastructure manager for containerized network functions," in *International Conference on Communications (ICC)*. IEEE, 2021.

[4] X. Wang, J. Du, and H. Liu, "Performance and isolation analysis of runc, gvisor and kata containers runtimes," *Cluster Computing*, pp. 1–17, 2022.

[5] W. Zhang, G. Liu, W. Zhang, N. Shah, P. Lopreiato, G. Todeschi, K. Ramakrishnan, and T. Wood, "OpenNetVM: A platform for high performance network service chains," in *Workshop on Hot topics in Middleboxes and Network Function Virtualization*, 2016, pp. 26–31.

[6] S. G. Kulkarni, G. Liu, K. Ramakrishnan, M. Arumaithurai, T. Wood, and X. Fu, "Reinforce: Achieving efficient failure resiliency for network function virtualization based services," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2018, pp. 41–53.

[7] D. Perino, M. Varvello, L. Linguaglossa, R. Laufer, and R. Boislaigue, "Caesar: A content router for high-speed forwarding on content names," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2014, pp. 137–148.

[8] H. U. Adoga and D. P. Pezaros, "Network function virtualization and service function chaining frameworks: A comprehensive review of requirements, objectives, implementations, and open research challenges," *Future Internet*, vol. 14, no. 2, p. 59, 2022.

[9] X. Chen, H. Liu, D. Zhang, Z. Meng, Q. Huang, H. Zhou, C. Wu, X. Liu, and Q. Yang, "Automatic performance-optimal offloading of network functions on programmable switches," *IEEE Transactions on Cloud Computing*, 2022.

[10] G. P. Katsikas, T. Barbette, D. Kostic, R. Steinert, and G. Q. Maguire Jr, "Metron: NFV service chains at the true speed of the underlying hardware," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2018, pp. 171–186.

[11] A. Bremler-Barr, Y. Harchol, and D. Hay, "OpenBox: a software-defined framework for developing, deploying, and managing network functions," in *ACM SIGCOMM Conference*, 2016, pp. 511–524.

[12] W. Tu, Y.-H. Wei, G. Antichi, and B. Pfaff, "Revisiting the open vSwitch dataplane ten years later," in *ACM SIGCOMM Conference*, 2021, pp. 245–257.

[13] D. Saxena and A. K. Singh, "OFP-TM: an online VM failure prediction and tolerance model towards high availability of cloud computing environments," *The Journal of Supercomputing*, pp. 1–22, 2022.

[14] Y. He, X. Zhang, Z. Xia, Y. Liu, K. Sood, and S. Yu, "Joint optimization of service chain graph design and mapping in nfv-enabled networks," *Computer Networks*, vol. 202, p. 108626, 2022.

[15] N. Pitaev, M. Falkner, A. Leivadeas, and I. Lambadaris, "Characterizing the performance of concurrent virtualized network functions with OVS-DPDK, FD.IO VPP and SR-IOV," in *ACM/SPEC International Conference on Performance Engineering (ICPE)*, 2018, pp. 285–292.

[16] R. Behravesh, E. Coronado, and R. Riggio, "Performance evaluation on virtualization technologies for nfv deployment in 5g networks," in *Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 24–29.

[17] D. Gedia and L. Perigo, "Performance evaluation of SDN-VNF in virtual machine and container," in *Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2018.