Feasibility of Compound Chained Network Functions for Flexible Packet Processing

Wolfgang Hahn[1], Borislava Gajic[1], Florian Wohlfart[2], Daniel Raumer[2], Paul Emmerich[2], Sebastian Gallenmüller[2], Georg Carle[2]

[1] Nokia Bell Labs
St. Martin-Straße 76, 81541 Munich, Germany
{wolfgang.hahn; borislava.gajic}@nokia-bell-labs.com

[2] Technical University of Munich, Chair of Network Architectures and Services
Boltzmannstr. 3, 85748 Garching b. München, Germany
{wohlfart; raumer; emmericp; gallenmu; carle}@net.in.tum.de

Abstract - **The paper discusses and evaluates different implementation options of Service Function Chaining (SFC) in virtual computing environments. Classical SFC relies on dedicated virtual machines for each chained network function. The motivation is to increase the efficiency of the classical SFC approach with respect to latency and throughput. Therefore, compound SFC that are implemented in single virtual machines are proposed. The focus is on the evaluation of the latency of communication between network functions and on the impact of limited computing resources on the maximal executable workload. To evaluate the presented SFC concepts, the performance measurements in a hardware testbed using different network functions have been carried out. A prototype of compound SFC was implemented based on the networking toolkit Snabb [1]. Furthermore, a configurable synthetic network function was implemented and tested. The results show the sensitivity of different parameters with respect to the efficiency of SFC implementation options.**

Keywords— Service Function Chaining; Packet Processing; Virtualization; Latency; Resource Contention

I. INTRODUCTION AND MOTIVATION

Network Functions Virtualization (NFV) has been applied to telecommunication networks to reduce costs of operators and to increase the flexibility and agility in introducing new services. The concept of Service Function Chaining (SFC) allows to flexibly route data streams through network functions (such as NAT, TCP optimizers, firewalls, video optimizers), by exploiting the modularity of network functions. Applying NFV to SFC improves the flexibility in building network functions. New standards [2, 3] allow to increase chaining efficiency e.g. for managing traffic profiles by reprogramming the data path like with SDN technologies [4].

In the context of 5G, those technologies are envisaged as promising tools to increase the flexibility and programmability when implementing 5G RAN and core functions from modular micro services. The EU Horizon 2020 funded project 5G NORMA [5] aims to develop a network that is able to adapt its functions to multiple services in a context-aware way. However, applying SFC and NFV approaches can imply drawbacks especially in terms of latency [6].

Since the requirements and expectations in 5G networks for data throughput and latency are very high it needs to be investigated if the overhead introduced by virtualization and modularization is acceptable for the performance in 5G networks. Another question that rises in this context is if negative impacts of SFC and NFV on the performance can be reduced by applying novel approaches as proposed in this paper.

Service Functions (SFs) were former realized as physical "boxes". These physical devices shall now become Virtual Network Functions (VNFs) that rely on the data center to connect the different virtual machines (VMs). This implementation option is referred to as ***classical SFC design***. Fig.1 a) displays two SFs (also referred to as Network Function NF in this paper). One of the drawbacks of such approach is that it introduces additional latency due to communication between VNFs (even if accelerated). The novel approach that is evaluated in this paper combines a SFC inside one virtualized environment, this is termed **compound SFC**; see Fig.1 b).

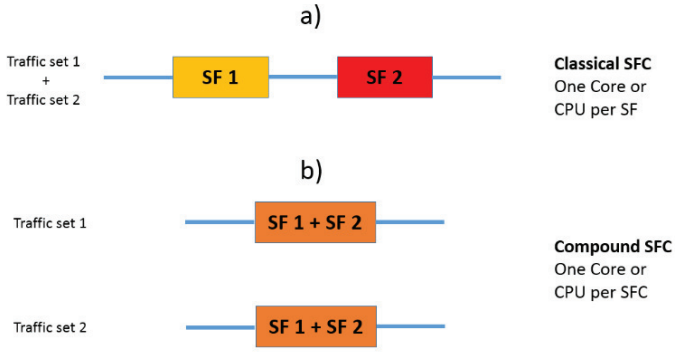

Fig. 1. Classical SFC a) and. compound SFC b) implementation option

Compound SFCs are implemented within a single VM. However, this concept is applicable to different virtualization technologies such as containers or cgroups. By this means, there is the potential of reducing the latency introduced due to communication between VNFs. On the drawback side,

compound SFC provides less isolation of individual SFs as isolation only exists for the compound element as a whole. However, this is less important for functions that are provided by one vendor. This is the case when network functions are decomposed into micro services. Development of micro services is more agile and improves the programmability of the data path in the network. Furthermore, requirements of the individual SFs need to be taken into account when composing the compound SFC. These requirements have to be translated into resource restrictions of the VMs.

Under the assumption that each box in Fig.1 (implemented as VM or container) consumes the same amount of computing resources, the assumption that connections of two boxes do not consume any resources and with perfect load distribution of the traffic flows to parallel SFCs both arrangements can handle the same throughput. The following sections investigate the differences and corresponding tradeoffs of the approaches. The investigation addresses two questions:

- Can the chaining overhead (expressed by latency when routing from one SF to another SF) be reduced by the compound SFC design and to what extend?
- Does the combined implementation of SF introduce limitations for the performance by different SF competing for same resources (like CPU cache)?

The first question aims at evaluating a potential reduction of the virtualization overhead of the classical SFC design by using the novel approach of compound SFC. The second question focusses on investigating the restrictions of the compound SFC approach in terms of resource contention by different SFs that are chained together.

The remainder of the paper is organized as follows. Section II gives an overview of the related work. Section III describes the implementation of the new concept of compound SFC and the chosen experimental test setup. Section IV investigates the chaining performance of both approaches especially with respect to latency and discusses the main findings. Section VI presents results of investigations related to resource limitations of the compound implementation. Finally, Section VI draws conclusions.

II. Related Work

The basic concept of the compound SFC aims at mitigating the communication latency between VMs. It was introduced in the previous work by Hahn et al. [7] and is further evaluated in this paper. In the previous work [7] the focus was on resource scaling of the different approaches for SFC. The overall number of computing resources were shown to be similar for both, the classical SFCs and the compound SFCs (see Fig.1). For adapting to varying traffic demands, a mechanism was introduced for reconfiguration of the compound SFC inside a VM (called “inner loop” control). As a result the alternative operation of increasing or decreasing the overall number of VMs for SFC by involving entities such as the Orchestrator (scaling in and out of SF (VMs), referred to as “outer loop”) needs to be done less frequently. An advantage of the inner loop management is that it can scale more dynamically. In this paper, a quantitative evaluation of compound SFC in terms of potential reduction of latency compared to classical SFC is provided.

NFV [8] and Software Defined Networking (SDN) [4] are key enablers for constantly delivering innovative network functions at lower costs. Whereas NFV enables deploying of NFs as a software on standardized hardware, SDN decouples the control plane and the data plane, thus allowing the programmability on the flow level. Often, NFV and SDN are seen as technologies to implement SFC. However, deploying the SFC as a composition of VMs running the individual SFs has a latency issue due to the communication between VMs [9, 6]. This has led to research activities to ensure fast packet processing in DC with general-purpose hardware [10, 11, 11] .

Recent software switches reduce the chaining overhead that is caused by connectivity between VMs compared to state of the art virtual switches like Open vSwitch or Hyper-V-Switch. Examples of accelerated switches are DPDK vSwitch [12] based on DPDK or VALE [13] based on netmap. Other work directly focused on accelerations for chaining of NFs [14, 15, 16] or used programmable NICs for this purpose [17]. Another option is to utilize the DirectPath I/O [18] acceleration technique or SR-IOV [19] for acceleration. Recent work [20] analyzed the benefits of using the Data Plane Development Kits (DPDKs) for running VNFs. Although performance overhead through virtualization can be reduced by these techniques, the latency introduced by the classical SFC deployment remains an issue [6].

III. Implementation of the Compound SFC and Evaluation Setup

In order to evaluate and compare the classical and the compound SFC approaches, both scenarios are implemented in a measurement lab setup. Both scenarios are tested with different chain lengths of up to 16 NFs. The resulting numbers are based on tests lasting 100 seconds each. All results are repeatable so confidence intervals in the graphs are not provided.

The classical SFC scenario is used as a reference for VM-based SFC with one VM per NF. The implementation relies on Kernel-based Virtual Machines (KVM) and interconnects VMs using Open vSwitch (OVS) in the hypervisor. Both OVS and KVM are considered as de-facto standard solutions. The NFs implement the standard OS socket interface for communication with neighboring elements in the SFC. Fig.2a illustrates the setup.

Evaluating the compound SFC approach is the main objective of this work. This scenario can be realized using modular packet processing frameworks such as Click [21], FastClick [22], or Snabb [1]. These frameworks can be used to establish a forwarding path across multiple modules using proprietary interfaces not compatible with the OS socket interface. This makes the packet processing modules independent from VMs or containers and allows the chaining of multiple NFs inside a virtual machine. The measurement setup is based on Snabb [1] which focuses on efficient interfaces and fast packet processing. Several NFs run within a single VM. These VMs are chained together via Snabb, as illustrated in Fig.2b. Like netmap Snabb

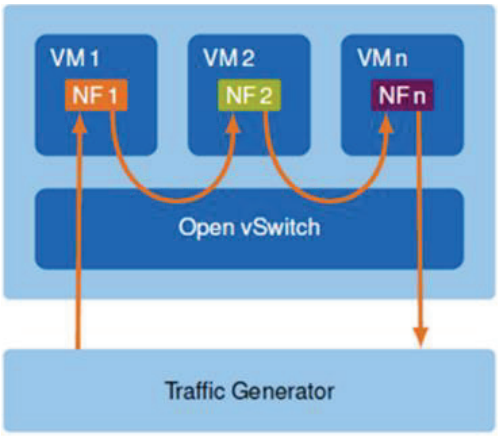

Fig. 2a. Test setup of classical SFC

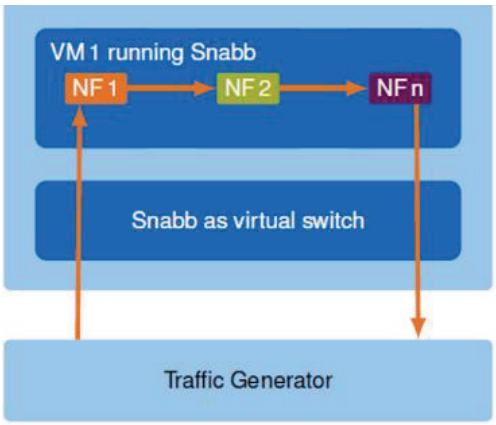

Fig. 2b. Test setup of compound SFC

circumvents performance limitations of classic packet reception and transmission. Details on these limitations are described in [10] [23].

The preferred metric for the required processing time is the number of CPU cycles consumed per packet (cycles/pkt). Compared to reporting the maximum throughput a given device is able to process (e.g. packets per second) this measure is more generic and allows comparability with other hardware configurations, e.g. the same CPUs at different frequencies. Nevertheless, the throughput metric can be easily derived as: throughput[B/s] = [CPU_freq[#cyc/s] / [#cyc]/packet[#pkt]] * packet_size[B].

The latency (measured in microseconds) introduced during packet processing is another important metric in the tests. Compared to the mere processing time it also includes time packets spend waiting in buffers. The latency metric is applicable to the measurement system as a whole. It depends on the processing time (measured in cycles/packet) and on factors, such as buffer size and batch size and strategies.

The machine hosting the device under test was equipped with the following hardware: Intel(R) Xeon(R) CPU E31230 (3.20GHz clock speed), 16 GB RAM, Intel 10 Gbit X540-T2 NIC. The device under test runs Debian Linux (Kernel 3.16) and OVS version 2.3.0 (classical SFC) or Snabb 2016.11 (compound SFC). The traffic generator runs MoonGen [24], a software-based load generator that achieves high accuracy by relying on NIC hardware features. The traffic generator applies constant bitrate traffic at a fixed frame length of 64Bytes. As the compound SFC implementation based on Snabb implies running the entire chain on a single CPU core, for comparison reasons the classical SFC was implemented in both ways by pinning the entire chain to a single CPU core as well as having the NFs distributed over multiple CPU cores. Furthermore, the experiments include a variety of NFs ranging from simple forwarding functions, real NFs (e.g. IwAFTR Carrier-Grade NAT), to synthetic NFs that mimic the resource consumption of real NFs while having the advantage of tunable parameters (e.g. L3 cache memory consumption).

The focus of the experiments is on classical vs. compound SFC performance in terms of latency and resource contention in order to get an insight of their suitability to the 5G network use case. The following sections give answers to the two research questions posed in the introduction section discussing the measurement results on the chaining overhead and resource consumption of NF.

IV. Discussion of the Chaining Performance

This section is concerned with the overhead introduced by the communication between NFs and not the performance of NFs themselves. If we assume that the chained NFs do not consume any resources, the overhead introduced by a single chain element can be measured by extending the measured SFC by one additional element. In practice, a NF at least needs to forward traffic, which means that only an upper bound for the chaining overhead can be measured and not its exact value. The measurements presented in this section were carried out using NFs that only provide minimal forwarding.

The first set of measurements was conducted to compare the chaining overhead present in the classical SFC and compound SFC. All measurements were run on a single CPU core to ensure that the test results are comparable. In the case of classical SFC, the different VMs each implementing a single NF are all running on the same CPU core. The compound SFC

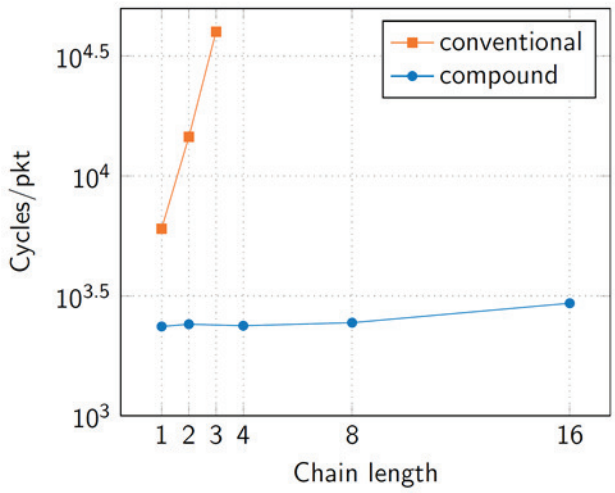

Fig. 3. Performance comparison between classical SFC and compound SFC in terms of cycles per packet with respect to the number of NFs in the chain. (chain length 1 equals 2 NFs – 1 per direction).

is implemented in a single VM, on a single CPU core and contains multiple NFs. Fig. 3. shows the results of the performance comparison of compound and classical SFC measured in CPU cycles per packet as a function of the chain length.

In both, the classical and compound SFC cases, for one NF already a considerable number of cycles is necessary for a packet to enter and leave the NF. This includes the processing of the NIC driver and the virtual switch performance that is involved in both cases. At chain length one there is no difference between the classical and compound scenarios. This allows the performance comparison of OVS and Snabb, where Snabb achieves two times the performance of OVS.

In the compound implementation, a linear correlation of performance and chain length can be observed. The calculation shows that the communication overhead between two Snabb NFs requires only 35 cycles per packet. In other words, each additional NF in a compound SFC solution adds a constant overhead of 35 cycles per packet. The cycles can be converted into time by applying the 3.20 GHz CPU clock: 35 cycles equals 0,01 µsec. In the classical SFC implementation, an exponential growth of the consumed processing time can be observed with increasing chain length.

As performance of the classical SFC might be related to the particular distribution of NFs over available CPU cores additional measurements of the classical SFC setup were performed. For optimal performance, OVS was allocated to a dedicated core and the chained VMs are distributed among other cores of the processor (up to three NFs in a four core CPU). The results of this evaluation of the classical SFC are presented in Fig.4. For comparison reasons the results for classical SFC performance when implemented on a single core and when NFs are distributed over multiple cores are provided. Although the performance of the classical SFC approach has improved in the multicore implementation the compound SFC approach still shows significantly better performance as shown in Table 1. Table 1 shows the cycles per packet on the bottleneck CPU core for the classical SFC when the NFs are located on different cores compared to compound SFC.

Table 1 Comparison between classical SFC and compound in terms of cycles per packet on the bottleneck CPU core and the number of NFs in the chain

n NFs	1	2	3	4	16
Classical SFC - OVS (n+1 cores)	5466	6652	11800	N/A	N/A
Classical SFC - OVS (single core)	6035	14545	40000	N/A	N/A
Compound SFC - Snabb (single core)	2357	2409	N/A	2374	2946

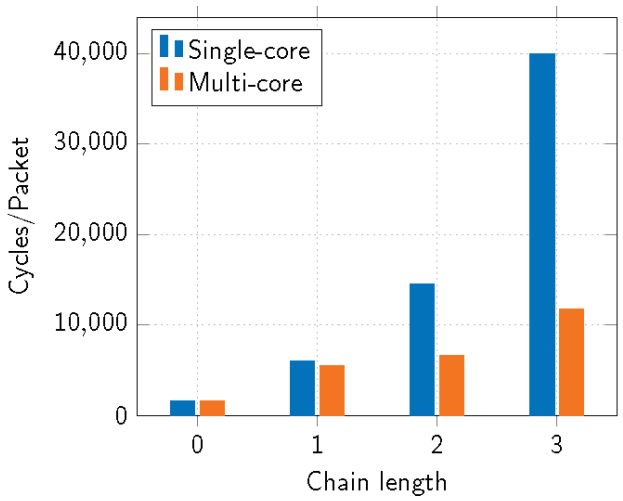

Fig. 4 Performance evaluation of single and multicore implementations of classical SFC approach.

For both, the classical SFC with OVS and the compound SFC with Snabb, the cycles required to get the packet in and out of the VM dominated the required cycles. IO of the VM requires ~4k cycles per packet for OVS+KVM and ~2.3k cycles per packet for more efficient compound SFC with Snabb and KVM.

In addition to the measurement of processing time, the overall latency was also measured based on precise hardware time stamping. Apart from the processing overhead for chaining NFCs, the latency is more significantly impacted by buffering. Packets face waiting times in buffers on the NIC, within the batches that are used to transfer packets between the chain elements, and at VM interfaces.

Table 2 shows the results of experiments measuring the latency of the whole system where different numbers of NFs have been chained together in a classical (single and multicore implementation) and compound SFC. OVS comes with a dynamic batch size algorithm that allows for lower latency for a small number of NFs. In Snabb the batch size is fixed, which results in an unchanged latency for 1 – 4 NFs.

Table 2 Latency measurements with different chain lengths

# NFs	1	2	3	4
Latency of OVS in µsec (under low load, 100 Mbit/sec)				
all NF on one core	39.7	67.5	195.5	833.4
Dedicated cores	39.7	74.6	151.4	491.2
Latency for Snabb in µsec				
Snabb	95.7	80.9	96.0	95.9
w/o power saving	16.6	59.8	63.2	60.6

Latency can be reduced in Snabb on the cost of energy efficiency: In busy mode, Snabb consumes the computing resources completely by busy waiting for new packets (polling) for shortest latency. This is an extreme case as the simple forwarder app provides the baseline of maximum achievable performance. Higher workload in the NF will increase the Snabb efficiency as the batch size will increase and Snabb will fetch more packets per polling cycle.

Due to highly efficient Snabb chaining forwarding (0.01µsec) the latency of the overall Snabb chain is nearly independent of the number of NFs. The latency mainly depends on the workload of the NF, the batch size and the use of power saving modes.

In contrast, in an OVS-based chaining the latency depends also on the number of NFs significantly. The forwarding

overhead increases exponentially in particular when NFs are hosted on the same core. Also in case of dedicated cores for the OVS and NFs the overhead for the first additional chain element is already 1186 cycles = 0.37 µsec and for the second additional chain element 5148 cycles = 1.6 µsec, see Table 1.

As investigated by Emmerich et al. [25] the performance of OVS can be increased up to the factor of 6 by DPDK. Taking this finding from Emmerich et al. [25] and the results illustrated in Fig.4 it can be concluded that the Snabb overhead in processing is still around 8 times smaller even compared to a DPDK accelerated OVS. Therefore, although this paper does not contain measurements for a comparison with DPDK accelerated OVS it can be assumed that Snabb still performs better than DPDK accelerated OVS for more than 3 NFs in a chain.

V. Evaluation of Impacts of Resource Limitations

In case several NFs run in a Snabb environment to constitute a compound SFC they need to share the resources of a single CPU core they are running on. As mentioned in the introduction the CPU computing can be reduced by a modified load balancing or parallel computing schema compared to the classical SFC. More critical for the SFC performance in the compound SFC implementation is the available cache size on the single CPU core where the SF chain is running.
Computing and memory resources differ among the NFs. Table 3 shows two typical examples of NFs with IPsec more demanding in terms of both computing and cache resources than IP routing, see [26, 27].

Table 3. Resource consumption of NFs

	CPU-Load	Cache-Load	workingSetSize
routing	250	100	4MB
IPsec	9000	1000	100MB

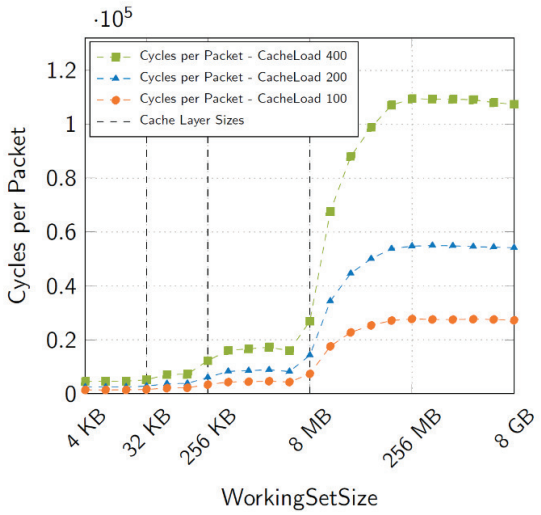

Fig. 5. Impact of cache size to NF performance

To evaluate the concepts of compound SFCs, a synthetic NF was constructed based on Snabb that is configurable with respect to the resources it consumes. It emulates in this way the performance characteristic of real NF, similarly to the examples in Table 3. Configurable parameters are the number of CPU cycles per packet (CPU load), the size of a data set and the number of accesses on the data per packet (Cache load). The NF accesses the working set randomly to accommodate for worst-case behavior.

Fig. 5 shows the cycles per packet from the measurements of forwarding performance by a native Snabb with the synthetic NF (simple forwarder) with a synthetic cache load (100, 200, and 400 accesses per packet) and a varying working set size (ranging from 4KB up to 8GB). When the working set size exceeds the cache size of the L1, L2 and L3 cache this results in a significant decrease of performance (by increasing the number of CPU cycles needed for packet processing). While in the classical SFC approach the NFs on different cores share the same L3 cache but have own L1 and L2 cache (Intel processor) in a compound SFC approach implemented by Snabb also the L1 and L2 cache is shared among NFs. In this experiment, the L1 cache size is 32KB, L2 cache size is 256KB and L3 cache size is 8MB. As illustrated in Fig. 5 L1 and L2 cache misses result in about each time doubling the number of cycles per packet when data on the next level cache are accessed.

Nevertheless, the biggest decrease in performance is caused by L3 cache misses. The risk of performance degradation due to L3 cache misses is present also in classical SFC as the NFs share L3 cache.

VI. Conclusions

In this work, different implementation options for SF Chaining have been evaluated: the classical SFC approach where all SFs are implemented in separate VMs and the compound SFC approach where SFs are implemented in a single VM. The compound solution based on Snabb shows significant benefits for chaining performance in terms of communication overhead and latency especially for chains with more than two NFs. On the other hand the cache miss rate and its impact to overall performance needs to be considered. The measurements indicate the main impact factors as well as the extent to which they might influence the chaining performance. It can be envisioned that for microservices that process different tasks on the same work load the compound function chaining approach can be well suited. A higher number of NFs might not increase the cache misses dramatically.

Thus, compound SFCs provides a framework for services that can be reconfigured or re-programmed flexibly to adapt to varying traffic patterns in telecommunication networks. As such flexibility is one of the imperatives of 5G networks, the compound SFC approach might be suitable for implementing agile 5G services. Furthermore, due to its good performance in terms of communication overhead and latency the compound SFC framework can be applicable to low latency 5G services as well. As a next step, more detailed measurements and quantitative evaluations of the possible approaches and the impact of co-locating NFs on common hardware will be carried

out taking into consideration the performance characteristic of the NFs.

VII. ACKNOWLEDGEMENTS

This work has been performed in the framework of the H2020-ICT-2014-2 project 5G NORMA. This information reflects the consortium's view, but the consortium is not liable for any use that may be made of any of the information contained therein.

VIII. REFERENCES

[1] SnabbCo, "Snabb: Simple and fast packet networking," [Online]. Available: https://github.com/snabbco/snabb.

[2] 3GPP, "TR23.718: Architecture Enhancement for Flexible Mobile Service Steering (Release 13)," 2014.

[3] J. Halpern und C. Pignataro, RFC7665: Service Function Chaining (SFC) Architecture, IETF, 2015.

[4] ONF Open Networking Foundation, „Software Defined Networking: The New Form for Networks," White paper, April 2013.

[5] „EU H2020-ICT-2014-2, 5G NORMA," 2015. [Online]. Available: https://5gnorma.5g-ppp.eu.

[6] B. Han, V. Gopalakrishnan, L. Ji und s. Lee, "Network function virtualization: Challenges and opportunities for innovations," IEEE Communications Magazine , February 2015.

[7] W. Hahn und B. Gajic, "Compound implementation of chained network functions and virtual resource management performance evaluation," IEEE/IFIP Network Operations and Management Symposium (5GMAN workshop), 2016.

[8] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Faragano, C. Cui, H. Deng, J. Benitez, U. Michel, H. Damker, K. Ogaki, T. Matsuzaki, M. Fukui, K. Shimano, D. Delisle, Q. Loudier, C. Kolias, I. Guardini, E. Demaria, R. Minerva, A. Manzalini, D. López, F. Salguero, F. Ruhl und P. Sen, „Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action," Whitepaper at the SDN and OpenFlow World, 2012.

[9] J. Whiteaker, F. Schneider und R. Teixeira, "Explaining Packet Delays under Virtualization," ACM SIGCOMM Computer Communication Review, January 2011.

[10] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer und G. Carle, "Comparison of Frameworks for High-Performance Packet IO," Proceedings of the 11th ACM/IEEE symposium on Architectures for networking and communications systems (ANCS), 2015.

[11] J. García-Dorado, F. Mata, J. d. Santiago, P. d. Río, V. Moreno und J. Aracil, "High-Performance Network Traffic Processing Systems Using Commodity Hardware," Data Traffic Monitoring and Analysis, 2013.

[12] Intel, "GitHub DPDK vSwitch," [Online]. Available: https://github.com/01org/dpdk-ovs.

[13] L. Rizzo und G. Lettieri, "VALE, a switched ethernet for virtual machines," ACM International Conference on emerging Networking EXperiments and Technologies, CoNEXT, 2012.

[14] I. Cerrato, G. Marchetto, F. Risso, R. Sisto und M. Virgilio, „An Efficient Data Exchange Algorithm for Chained Network Functions," IEEE 15th International Conference on High Performance Switching and Routing (HPSR), 2014.

[15] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda und R. Bi, "ClickOS and the art of network function virtualization", Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, NSDI, 2014.

[16] M. Paolino, N. Nikolaev, J. Fanguede und D. Raho, „SnabbSwitch user space virtual switch benchmark and performance optimization for NFV," Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), 2015.

[17] Y. Luo, E. Murray und . T. L. Ficarra, "Accelerated Virtual Switching with Programmable NICs for Scalable Data Center Networking," Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, 2010.

[18] VM Ware, "Network I/O Latency on VMware vSphere 5 – Performance Study", Technical White Paper, 2012.

[19] Intel, „PCI-SIG SR-IOV Primer: An Introduction to SR-IOV Technology," White Paper, 2011.

[20] Intel Corporation and Wind River, "High Performance, Open StandardVirtualization with NFV and SDN", Joint Technical White Paper, 2015.

[21] E. Kohler, R. Morris, B. Chen, J. Jannotti und M. F. Kaashoek, "The Click modular router," ACM Transactions on Computer System, 2000.

[22] T. Barbette, C. Soldani und L. Mathy, "Fast userspace packet processing," Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for networking and communications systems, 2015.

[23] L. Rizzo, "netmap: a novel framework for fast packet I/O," in USENIX, Boston, 2012.

[24] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart und G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator. In Proceedings of the 15th", ACM SIGCOMM Conference on Internet Measurement (IMC), October 2015.

[25] P. Emmerich, D. Raumer, F. Wohlfart und G. Carle, "Performance Characteristics of Virtual Switching," In 3rd IEEE International Conference on Cloud Networking, CloudNet, October 2014.

[26] D. Raumer, S. Gallenmüller, P. Emmerich, L. Märdian und G. Carle, "Efficient serving of VPN endpoints on COTS server hardware," In 5th IEEE International Conference on Cloud Networking, CloudNet, 2016.

[27] S. Govind und R. Govindarajan, "Performance Modelling and Archieture Exploration of Network Processors," International Conference on Quantitative Evaluation of SysTems, 2015.