Intra-node Resource Isolation for SFC with SR-IOV

Simon Bauer, Daniel Raumer, Paul Emmerich, and Georg Carle
Technical University of Munich (TUM), Department of Informatics
Chair for Network Architectures and Services, Garching b. München, Germany
{bauersi|raumer|emmericp|carle}@net.in.tum.de

***Abstract*—Single Root I/O Virtualization (SR-IOV) is intended to provide simultaneous native access to network interface cards (NIC) from multiple virtual machines or applications. For this, SR-IOV offloads packet switching from software to hardware. We use SR-IOV outside its original purpose and establish a chaining infrastructure between virtual Service Functions. Thus, resources to provide chaining are isolated by the NIC.**

We compare Service Function Chains based on SR-IOV to fully software-based Service Function Chains and survey how shifting workload from the CPU to the NIC affects performance. Furthermore, we analyze the impact of virtual PCIe functions, which are required for the use of SR-IOV, on performance. Our study provides a detailed performance evaluation of Service Function Chains implemented with Open vSwitch and DPDK. The performance evaluation is based on comparative measurements on commodity hardware including profiling of the CPU and PCIe bus. We model the resource constraints of both implementation approaches to specify performance bottlenecks and to determine a scenario's maximum throughput.

***Keywords*—SFC, SR-IOV, packet I/O, virtual PCIe function, performance guarantees, resource isolation**

I. Introduction

Service Function Chaining (SFC) is proposed to provide flow-specific packet processing by interconnecting different Service Functions (SF), while complying high-performance processing requirements. For example, these requirements apply to data centers, mobile networks, or, cloud infrastructures. The establishment of an efficient chaining infrastructure between several SFs is one challenge of Service Function Chaining.

In this paper, we examine intra-node Service Function Chaining provided via the NIC, i.e., with Single Root I/O Virtualization. We contribute performance measurements of Service Function Chains implemented with Open vSwitch (OvS) and the Data Plane Development Kit (DPDK). We survey how SR-IOV-based Service Function Chains can be implemented, which performance characteristics they provide, and, how they differ from fully software-based Service Function Chains. We consider metrics like throughput and end-to-end latency. In addition, we profile the Device-under-Test's (DUT) CPU usage, memory accesses, and PCIe bus utilization. Our methodology parameterizes packet size, packet rate, and the number of chained SFs. We discuss the impact of offloading chaining from the CPU to the NIC and the impact of isolating chaining resources. To survey the performance of Service Function Chains from a theoretical point of view, we contribute a model of the load on performance bottlenecks of virtual packet I/O for both implementation approaches. Scripts to reproduce our results are available online [1].

978-1-5386-6831-3/18/$31.00 ©2018 IEEE

The remainder of this paper is structured as follows: First, we dive into the paper's background regarding SFC and SR-IOV in Section II. Then, we introduce the concept of SR-IOV-based Service Function Chains in Section III. We evaluate the performance of virtual PCIe functions and SR-IOV-based Service Function Chains in Section IV. Section V presents a model to determine the maximum throughput of Service Function Chains. We discuss observed trade-offs in Section VI and introduce related work in Section VII. To conclude this paper, we summarize our results in Section VIII.

II. Background

This section provides background information regarding the performance of Service Function Chains and Single-Root I/O Virtualization.

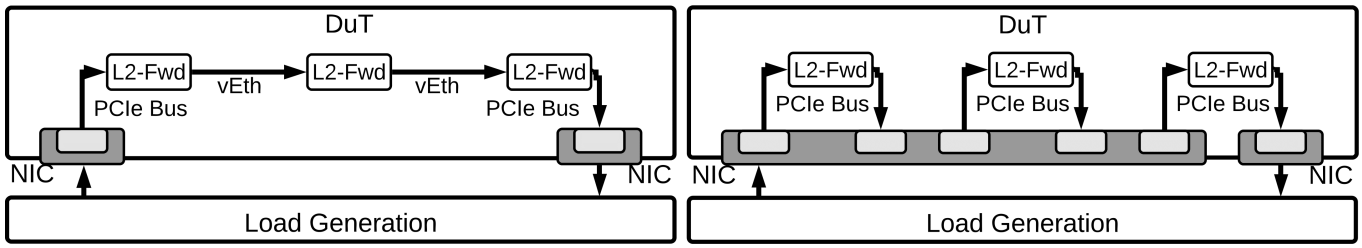
A. Performance of Service Function Chains

SFC is an approach to provide flow-specific packet processing across several SFs in virtual or para-virtual environments. A Service Function Chain consists of several interconnected SFs. Typically chained entities are middleboxes like switches, routers, or firewalls. RFC 7665 [2] describes architectural concepts and requirements to provide Service Function Chaining.

In previous research, we survey the performance characteristics of Service Function Chains [3] based on software-links (SWlink-based) and determine the CPU as performance limitation. This limitation results in decreasing achievable throughput with increasing chain length n, while chain length refers to the number of chained SFs. Furthermore, we determine software (SW) drops by the operating system in SWlink-based chains to be a significant impact on performance. In contrast to traffic discarded by the NIC, SW drops require additional computational resources from the CPU. Therefore, SW drops intensify the shortage of computational resources with increasing packet rate, as more packets get dropped by the OS. This decreases throughput as soon as the DuT is overloaded.

B. SR-IOV and Virtual PCIe Functions

Single Root I/O Virtualization is motivated by offloading packet switching from software to hardware, i.e., the NIC. SR-IOV enables a physical PCIe device to share resources between multiple virtual PCIe functions (VFs) [4]. In case of network packet I/O, SR-IOV enables to address virtual



(a) based on vEth interfaces

(b) based on SR-IOV

Fig. 1. Exemplary Service Function Chains of length $n = 3$

PCIe functions via a virtual Layer-2 switch. This switch is added to the receive path , respectively transmit path, of the NIC. With SR-IOV enabled, received packets are categorized by the NIC and steered towards the appropriate VF.Available filters for the categorization depend on the NIC model, ranging from simple basic switch functionality like matching VLAN tags and destination MAC addresses to complex filters (e.g., Intel Flow Director). The receiving VF transfers packets via DMA to main memory or CPU cache where it is handled by the driver. A prerequisite of SR-IOV are virtual PCIe functions. Each VF is related to a physical PCIe function (PF), respectively to a physical PCIe device. This implies trade-offs between overhead and functionality. While VFs support data movement operations like receiving or sending network traffic, configuration possibilities are limited compared to physical PCIe functions. Possible limitations are fixed buffer sizes, a limited number of RX and TX queues, or a limited set of filters that can be configured by the VF driver.

III. SR-IOV-based SFC

The approach to implement a chaining infrastructure between SFs with SR-IOV significantly differs from providing chaining with SW links or SW queues. In case of SR-IOV-based chains, packet forwarding between SFs is done by one virtual switch provided by the NIC instead of multiple direct connections between the SFs.

To implement SR-IOV-based Service Function Chains, SFs have to be configured to modify packet headers. This is required to address the next SF in the chain. The implementation of SWlink-based chains does not require packet modification in general. However, the most significant difference between both implementation approaches is the chaining medium. SWlink-based chains require several independent links to interconnect exactly two SFs at a time. I.e., a link is required for each connection between any pair of SFs. In case of SR-IOV-based chains, one medium, the virtual switch in the NIC, is shared to interconnect all SFs. This approach requires $2 \times n$ connections between the SFs and the NIC. Figure 1(a) and Figure 1(b) show the packet processing path of an exemplary Service Function Chain for both implementation approaches, as used for our evaluation. SWlink-based chains keep received packets in software until all processing steps are done. In contrast, SR-IOV-based chains require to transmit packets between software and the NIC after each SF.

IV. Performance Evaluation

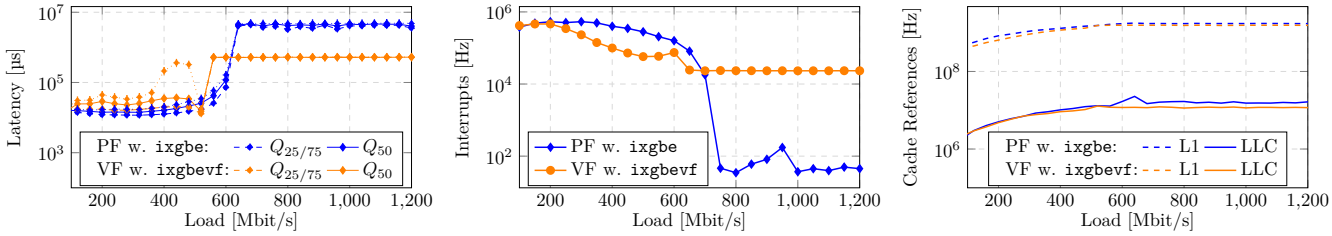
A. Hardware and Software Setup

All measurements are performed on two physical hosts: a DuT and a host dedicated to load generation and measurement. The DuT is equipped with a Xeon E5-2640 v2 8-core processor, clocked at 2.0 GHz, and one 10 GbE Intel X520-T2 dual-port NIC [5]. The NIC is connected to the CPU via 8 PCIe v2.0 lanes. This implies a bidirectional PCIe bandwidth of 32 Gbit/s between the NIC and the CPU [6]. The CPU's cache line size is 64 B. Our DuT runs Debian Linux with kernel version 4.9. Setups are implemented with Open vSwitch [7] (v2.8.1), DPDK [8] (commit 45d3635), and the DPDK wrapper libmoon [9] (commit aba8e12). Next to named processing frameworks, virtual Ethernet (vEth) interfaces, included in the kernel, are used to implement SWlink-based Service Function Chains. Profiling of the DuT is done with the Linux kernel profiling tool perf. We use the Intel PCM tool [10] to measure utilization of the the PCIe bus between NIC and CPU. Load is generated with the packet generator MoonGen [11] (commit 5388192). In addition, we use MoonGen to measure throughput and latency. The used drivers are as follows: For the kernel-based framework OvS we used the 10 GbE NIC driver `ixgbe` [12] and its pendant for virtual PCIe functions `ixgbevf` [13]. Measurements of DPDK and libmoon are based on DPDK's userspace drivers `net_ixgbe` and `net_ixgbevf`. Drivers are used in default configurations.

B. Performance of Virtual PCIe Functions

The usage of SR-IOV requires the usage of VFs. Therefore, we analyze how the performance of PFs and VFs differ to determine whether the SR-IOV-based approach suffers under the impact of less efficient PCIe functions. We evaluate a single forwarding SF in interplay with PFs and with VFs. As these setups only differ in the used PCIe function and driver, performance deltas can be traced back to the used PCIe function and driver. We consider three different SFs: (1) an Open vSwitch bridge, (2) a libmoon Layer-2 forwarder, (3) the DPDK L2FWD example. The tested OvS bridge and libmoon forwarder are configured to modify packets' destination MAC address, as required to provide SR-IOV-based chains.

First, we are interested whether VFs and VF drivers influence maximum achievable throughput of Service Functions. Throughput is measured with minimum packet size, i.e., 64 B, as small packet sizes imply the heaviest load for virtual packet



(a) Latency (b) Interrupts (c) Cache accesses

Fig. 2. Performance of physical and virtual PCIe functions

TABLE I
MAXIMUM THROUGHPUT IN MPPS

Service Function	1.2 GHz	1.6 GHz	2.0 GHz
OvS bridge w. PF*	0.68	0.74	1.07
OvS bridge w. VF*	0.59	0.63	0.98
libmoon forwarder PF*†	11.16	12.06	14.88
libmoon forwarder VF*†	11.35	12.31	14.88
DPDK L2FWD w. PF	14.88	14.88	14.88
DPDK L2FWD w. VF	14.88	14.88	14.88

* SF modifies Layer 2 destination address
† Driver uses offloading-enabled data-path

I/O [14]. We consider different CPU clock frequencies to determine dependencies between computational power and throughput. The tested DPDK L2FWD example is capable to process 14.88 Mpps with both kinds of PCIe functions and all tested clock frequencies. The tested libmoon forwarder only achieved line rate for measurements with 2.0 GHz CPU clock frequency. Measurements with 1.6 GHz, respectively 1.2 GHz, show that the setup with VFs is slightly more efficient. This can be explained by the used driver configuration. By default, libmoon optimizes the used driver to use all offloading features. This results in more complex RX and TX paths in case of the PF driver as the VF driver supports less features. Our measurements with OvS and the kernel-based drivers show that a VF with `ixgbevf` implies an CPU cycle overhead between 10-25% per packet compared to a PF with `ixgbe`. Table I shows the maximum throughput of all tested SFs and drivers.

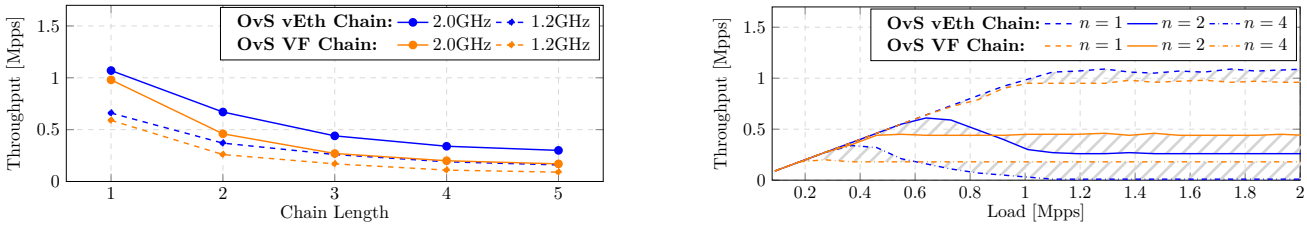
Next, we analyze quartiles of latency to survey the impact of VFs on latency. Our measurements shows that the VF driver `ixgbevf` causes significant higher latency as `ixgbe`, short of overloading loads. Figure 2(a) shows measured latency of an OvS bridge for both drivers. We profile the DuT for interrupts and cache operations and find that `ixgbevf` lacks the dynamic interrupt-throttle rate (ITR) adaption feature which is supported by `ixgbe` [15]. Hence, `ixgbevf` stays at a high interrupt rate even under full load. In addition, our measurements reveals that VFs can only use a partition of available NIC buffer, as they result in lower latency under overload. As expected, cache accesses correlate to the number of processed packets per second, independent of the used driver. Figure 2(b) and Figure 2(c) show results of our profiling analysis of interrupts and cache access. Regarding the DPDK poll mode drivers `net_ixgbe` and `net_ixgbevf`, the latency of PFs and VFs does not differ significantly. The libmoon forwarder in interplay with VFs is slightly more efficient and results in lower latency as its physical function pendant. Higher latency of the PF based setup can be traced back on the active offloading features of libmoon's PF driver, again.

C. Performance Characteristics of SR-IOV-based Chains

In previous work [3], we study the performance characteristics of Service Function Chains based on software links. We implement SW links with pairs of vEth interfaces provided by the Linux kernel. A chain is established by assigning each interface of a vEth pair to a different SF. A service Function Chain of length n requires $n-1$ vEth pairs. For the comparison of SWlink- and SR-IOV-based chains we implement Service Function Chains by interconnecting OvS bridges.

For our SR-IOV-based chains, we connect two VFs to each SF. To modify the destination MAC address of each forwarded packet, we use OpenFlow's action feature. For our measurements all OvS bridges are pinned to the same CPU core.

We measure maximum throughput of OvS chains with different chain lengths, i.e., length $n = 1$ up to length $n = 5$. The measured maximum throughput for all measured lengths is shown in Figure 3(a). The figure shows that SWlink-based chains result in higher maximum throughput. We observe that the CPU is the limit for all measured lengths. The determined lower maximum throughput of SR-IOV-based chains can be explained with a) the less efficient driver for VFs and b) the additionally required I/O operations. The impact of additional I/O operations depends on the chain length. In previous research [3], we found that SWlink-based implementations suffer under performance loss due to software drops, as soon as the DuT is overloaded. SW drops result in decreasing throughput, as packets are dropped by computational resources of the CPU. In case of SR-IOV-based chains, we do not observe packet drops by software and found stable maximum throughput for all offered loads. This confirms that resource isolation between SFs and chaining infrastructure provides guaranteed maximum throughput independent of the offered load. Therefore, SR-IOV-based chains process significantly more packets per second regarding overload. Figure 3(b) presents measured throughput for chain lengths $n = 1$, $n = 2$, and $n = 4$ for varying offered load. The shaded areas show throughput deltas of both implementation approaches.



(a) Maximum throughput

(b) Throughput to increasing load

Fig. 3. Performance comparison between chains implemented with SR-IOV and vEth interfaces

However, our evaluation shows that SR-IOV-based chaining implicates an interesting advantage: the isolation of resources. Each VF has a certain set of resources for its disposal.Isolating resources antagonizes performance loss in case of overloading rates, as no chaining element is able to claim more resources as others and there is no competition for resources between SFs and the chaining infrastructure. This does not apply on SWlink-based chains, where all links share the resources of the same CPU core.

D. Bottlenecks of SR-IOV-based Chains

The per packet processing costs of user space packet processing frameworks are significantly lower compared to kernel-based frameworks [16]. This higher number of processed packets implies higher load for other components, i.e., for the NIC and PCIe bus. For SR-IOV-based chains the load on the NIC and PCIe bus intensifies with increasing chain length, as each packet requires two transmissions between NIC and CPU per chained SF. In addition, each SF implies a switching decision by the NIC to forward packets to the next SF. To survey the mentioned impact we chain several libmoon forwarder via SR-IOV. Each forwarder modifies the destination MAC address of each processed packets.

We measure libmoon chains of lengths $n = 2$ and $n = 4$ and profile the PCIe utilization, i.e., the DMA writes and DMA reads sent via the PCIe bus. To stress the DuT we offer a load of 10 Gbit/s and increase packet size s from minimum (64 B) up to maximum (1518 B) in steps of 1 B. We observe that small packet sizes imply throughput fluctuations. These fluctuations are neither caused by the NIC's line rate nor the PCIe bus' bandwidth. Therefore, we argue that small packets result in more operations per second than the NIC is capable of. For example, a libmoon chain of length $n = 4$ confronted with 10 Gbit/s of 64 B packets, i.e., 14.88 Mpps, requires $14.88 \times 10^6 \times 4 = 59.52 \times 10^6$ switching decisions per second. Figure 4(a) shows the throughput of a libmoon chain of length $n = 4$ as data rate and packet rate. Mentioned fluctuations can be found for packet sizes $s < x_0$. For packet sizes $s > x_0$, throughput gets reliable. Profiling the PCIe bus utilization shows, that both the DMA writes and DMA reads correlate to the processed data rate. A single PCIe DMA transaction can transfer up to 512 B of data, but the NIC also needs to transfer a so-called DMA descriptor (16 B for the NIC used here) containing meta-data and a pointer to the packet data. These descriptors are usually fetched and written back in batched transactions of at least four descriptors. The resulting overhead is especially important for small packets. However, DMA transactions are transfered as full cache lines (64 B). The effect of cache line padding results in a saw-toothed course of PCIe utilization under increasing packet sizes as shown in Figure 4(b). Measurements with a reduced CPU clock frequency (1.2 GHz instead of 2.0 GHz) show that throughput is not limited by computational resources of the CPU, as the same throughput is achieved.

V. Resource Constraints

In the following we demonstrate how to apply results of our analysis to performance modeling. We use a simple approach called resource constraints (RC) modeling. A RC model is a directed graph which represents the flow of traffic through a system. Each node has a limited capacity that is gained from a systematic system analysis; cf. [17]. The graph represents the order in which the components are processing packets. The actual maximum throughput of the system is then defined by the maximum flow from the start to the end node.

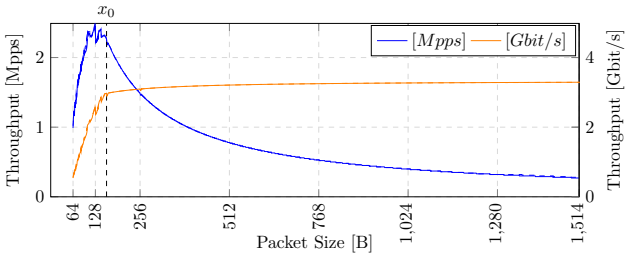
To model maximum throughput of service function chains we determine potential performance limitations. From previous research [3], we know that the CPU can limit a chain's maximum throughput. During our evaluation of SR-IOV-based libmoon chains, we found that the NIC reaches its computational limit for high packet rates.In addition, a chains throughput is limited by the bandwidth of all involved components. We consider chain length, average packet size and packet processing costs (in CPU cycles) for our model. Table II lists all parameters and variables involved in our model.

SWlink-based chains keep a received packet in software, respectively in the CPU, as long as all processing steps are done. Therefore, each packet has to be received and transmitted by the NIC only once. The CPU has to provide resources to process each packet by all n SFs and to process each packet via $(n-1)$ SW links. These considerations result in the following per packet processing costs:

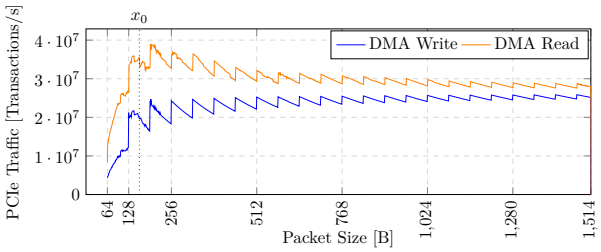
$$C_{NIC,SW} = C_{NIC,RX} + C_{NIC,TX}$$

$$C_{CPU,SW} = C_{CPU,RX} + n \times C_{SF} + (n-1) \times C_{SW} + C_{CPU,TX}$$

For SR-IOV-based chains, chaining is offloaded to the NIC. This unburdens the CPU from providing SW links, but makes it necessary to receive and transmit a packet several times.Load on the NIC increases compared to the SWlink-based approach.Consider that $C_{NIC,switch}$ comprises the tasks of receiving a packet, switching it, and transmitting it back to



(a) Throughput



(b) PCIe traffic

Fig. 4. Evaluation of a SR-IOV-based chain consisting of four libmoon forwarders

TABLE II
EXPLANATION OF MODEL PARAMETERS

Parameter	Symbol	Unit
chain length	n	SFs
average packet size	$\overline{s}$	bit
overhead implied by X	δ_X	bit
CPU's computational resources	f_{CPU}	cps*
NIC's computational resources	f_{NIC}	cps*
NIC line rate	lr_{NIC}	bit/s
PCIe bandwidth	bw_{PCIe}	bit/s
max. throughput of X	tp_X	bit/s

*cps = cycles per second

the CPU. Finally, SR-IOV-based implementations result in the following per packet costs:

$$C_{NIC,SR-IOV} = C_{NIC,RX} + (n+1) \times C_{NIC,switch} + C_{NIC,TX}$$

$$C_{CPU,SR-IOV} = n \times (C_{CPU,RX} + C_{SF} + C_{CPU,TX})$$

With the introduced formulas, maximum throughput for each component can be determined. The NIC's throughput is limited by its line rate and by computational resources of the NIC. Line rate limits maximum packet rate to $\frac{lr_{NIC}}{\overline{s}}$ for both implementations. The NIC's computational resources limit the number of processed packets per second to $\frac{f_{NIC}}{C_{NIC,total}}$. For our RC model, it is sufficient to consider the minimum of both limitations. The PCIe bus limits throughput by bandwidth. Next to transmitted network packets, the bus has to transmit meta-data, like DMA descriptors and acknowledgments. This implies an overhead δ_{PCIe} per packet. The limitation by the PCIe bus differs for considered implementations:

$$tp_{PCIe,SW} = \frac{bw_{PCIe}}{\overline{s} + \delta_{PCIe}}$$

$$tp_{PCIe,SR-IOV} = \frac{bw_{PCIe}}{n * (\overline{s} + \delta_{PCIe})}$$

The CPU limits throughput by the number of available CPU cycles per second. With the compositions of per packet costs introduced above, maximum throughput of the CPU is limited to $\frac{f_{CPU}}{C_{CPU,total}}$. The maximum throughput of an implementation approach is defined as the minimum of the throughputs of all components, i.e., the NIC, PCIe bus, and the CPU.

VI. Discussion

Our performance evaluation shows that SR-IOV-based service function chains are not able to achieve better performance compared to SWlink-based implementations. This can be explained by increased I/O costs in case of SR-IOV-based chains. Due to increased I/O costs, the CPU is not unburdened by offloading the chaining infrastructure.

However, our evaluation shows another interesting characteristic of SR-IOV-based chains: reliable throughput independent of the offered load. In contrast to SWlink-based chains, implementations with SR-IOV provide stable throughput independent of the load. This can be explained by the isolation of chaining resources by the NIC, i.e., by different virtual PCIe functions. This characteristic is interesting for network operators that provide services to several customers on the same hardware. Resource isolation prevents that a resource demand of a customer impairs another customer's performance. As network operators and service providers try to give performance guarantees to customers, it might be reasonable to renounce maximum performance in order to ensure reliable network performance. In case of SFC, this reliability can be achieved with SR-IOV.Such a use case is network slicing in mobile networks, e.g., regarding 5G as described by Foukas et. al. [18].

VII. Related Work

The problem of managing resources of virtual network functions (VNF) and Service Function Chains is seen as a related topic. Efficient placement of SFs (referred to as network functions) is surveyed by Mehraghdam et al. [19]. Authors introduce an approach to model and specify chains of VNFs and propose a program to place network functions as efficient as possible. Furthermore, the efficient placement of VNFs is surveyed by Moens and Turck [20], who considered a hybrid scenario, including physical and virtual network functions. The prevention of performance loss by backpressure is examined by Kulkarni et al. [21]. The problem of wasted resources in Service Function Chains results in NFVnice, a user space framework to schedule and manage service chains. NFVnice monitors load and improves performance by preventing the waste of resources in case of a congestion in the service chain.

SR-IOV has been related to the topic of high-performance packet I/O in the past.Musleh et al. surveyed the tuning of interrupt moderation in InfiniBand scenarios virtualized with

SR-IOV [22], while Jose et al. [23] surveyed performance benefits of SR-IOV in InfiniBand networks in general. A performance evaluation of SR-IOV and NetVM was published by Hwang et al. [24]. The results of the presented performance analysis show that NetVM has been more efficient for considered measurements. The use of SR-IOV for SFC was considered by Intel before. Intel presented practical considerations and performance measurements regarding SR-IOV in 2017 [25]. Among others, the technical brief surveys an east-west traffic pattern, similar to our SR-IOV-based Service Function Chains. However, Intel does not provide a detailed performance analysis nor a comparison to other SFC approaches.

In this paper we apply SR-IOV to SFC to face performance loss by software drops and guarantee performance of Service Function Chains. Our study is motivated by SR-IOVs application to high-performance packet I/O, the currentness of SFC, and the trend to offload functionality from the CPU.

VIII. Conclusion

We presented an extensive study of SR-IOV in the field of SFC. With SR-IOV, chaining between different SFs is offloaded to L2 switching functionality in the NIC. Therefore, services have to be connected to virtual PCIe functions offered by the NIC via a virtual function driver. We show that SR-IOV-based chains provide isolation and mitigate software drops inside chains, which are a critical characteristic of SW-linked SFs in regard to overload. Nevertheless, this isolation comes at cost of new potential bottlenecks, i.e., the PCIe bus bandwidth and the NIC's computational capabilities. We addressed these with a resource constraints model. The model is purposed as a guide to implement efficient Service Function Chains and to illustrate dependencies between chain length and workload for potential performance bottlenecks. We showed that NICs become bottlenecks of SR-IOV-based Service Function Chains, if packet processing by the SFs is efficient enough, e.g., due to efficiency provided by DPDK. This is furthered by increased switching demands with an increasing chain length. In result, our SR-IOV-based chains were not able to exceed the throughput of SW-link-based chains. With time, PCIe bandwidth and limits by the NIC will increase while offloading efforts to the NIC continue. However, our demonstrated methodology regarding measurement, modeling, and evaluation is compatible with the analysis of updated setups.

We provide a repository [1] with all sources to reproduce our performance measurements.

Acknowledgments.: This work has been supported by the German Federal Ministry of Education and Research, project SENDATE-Planets (KIS2ITS001) and the German-French Academy for the Industry of the Future.

References

[1] "SFC-with-SRIOV-performance-measurements." https://github.com/bauersi/SFC-with-SRIOV-performance-measurements.git. Last visited 2018-03-23.

[2] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture." RFC 7665, Oct. 2015.

[3] D. Raumer, S. Bauer, P. Emmerich, and G. Carle, "Performance Implications for Intra-node Placement of Network Function Chains," in *IEEE 6th Int. Conf. on Cloud Networking (CloudNet'17)*, 2017.

[4] "Using PCIe SR-IOV Virtual Functions." https://docs.oracle.com/cd/E35434_01/html/E23807/usingsriov.html. Last visited 2018-03-23.

[5] "Intel Ethernet Server Adapter X520-T2." https://www.intel.com/content/dam/doc/product-brief/ethernet-server-adapter-x520-t2-brief.pdf. Last visited 2018-03-23.

[6] "PCIe - Express base specification revision 2.0," 2006.

[7] "Open vSwitch - Production Quality, Multilayer Open Virtual Switch." http://openvswitch.org/. Last visited 2018-03-23.

[8] "DPDK (github)." https://github.com/emmericp/dpdk/tree/45d3635fbc1cdf868d9f58608f157cec3198def8. Last visited 2018-03-23.

[9] "Github/libmoon." https://github.com/libmoon/libmoon. Last visited 2018-03-23.

[10] "Processor Counter Monitor." https://github.com/opcm/pcm. Last visited 2018-03-23.

[11] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," in *15th ACM SIGCOMM Conference on Internet Measurement (IMC'15)*, 2015.

[12] "Intel Network Adapter Driver for PCIe* Intel 10 Gigabit Ethernet Network Connections Under Linux*." https://downloadcenter.intel.com/download/14687/Intel-Network-Adapter-Driver-for-PCIe-Intel-10-Gigabit-Ethernet-Network-Connections-Under-Linux-. Last visited 2018-03-23.

[13] "Intel Network Adapter Virtual Function Driver for Intel 10 Gigabit Ethernet Network Connections." https://downloadcenter.intel.com/download/18700/Intel-Network-Adapter-Virtual-Function-Driver-for-Intel-10-Gigabit-Ethernet-Network-Connections. Last visited 2018-03-23.

[14] L. Rizzo, "netmap: a novel framework for fast packet I/O," in *USENIX Annual Technical Conference*, 2012.

[15] P. Emmerich, D. Raumer, A. Beifuß, L. Erlacher, F. Wohlfart, T. M. Runge, S. Gallenmüller, and G. Carle, "Optimizing latency and cpu load in packet processing systems," in *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Spects '15, Society for Computer Simulation International, 2015.

[16] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle, "Comparison of Frameworks for High-Performance Packet IO," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2015.

[17] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "Assessing Soft- and Hardware Bottlenecks in PC-based Packet Forwarding Systems," in *Fourteenth International Conference on Networks (ICN 2015)*, 2015.

[18] X. Foukas, A. Elmokashfi, G. Patounas, and M. Marina, "Network slicing in 5g: Survey and challenges," 5 2017.

[19] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Oct 2014.

[20] H. Moens and F. De Turck, "Vnf-p: a model for efficient placement of virtualized network functions," in *2014 10th International Conference on Network and Service Management (CNSM)*, 2014.

[21] S. G. Kulkarni, W. Zhang, J. Hwang, S. Rajagopalan, K. K. Ramakrishnan, T. Wood, M. Arumaithurai, and X. Fu, "Nfvnice: Dynamic backpressure and scheduling for nfv service chains," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, ACM, 2017.

[22] M. Musleh, V. Pai, J. P. Walters, A. Younge, and S. Crago, "Bridging the virtualization performance gap for HPC using SR-IOV for InfiniBand," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, IEEE, 2014.

[23] J. Jose, M. Li, X. Lu, K. C. Kandalla, M. D. Arnold, and D. K. Panda, "SR-IOV Support for Virtualization on InfiniBand Clusters: Early Experience," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 05 2013.

[24] J. Hwang, K. K. Ramakrishnan, and T. Wood, "NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation(NSDI'14)*, USENIX Association, 2014.

[25] Intel Network Division, "SR-IOV for NFV Solutions - Practical Considerations and Thoughts," Tech. Rep. Revision 1.0 335625-001, 12 2017.