Performance Implications for Intra-node Placement of Network Function Chains

Daniel Raumer, Simon Bauer, Paul Emmerich, Georg Carle
Technical University of Munich, Department of Informatics, Chair of Network Architectures and Services
{raumer|bauersi|emmericp|carle}@in.tum.de

***Abstract*—With virtual switching, a chain of virtual network functions can be realized on a single physical host. This promises higher flexibility and reduction of costs for network operators due to the usage of commodity hardware instead of expensive hardware appliances. On the down side this may cause interferences that affect the performance. Therefore, we survey the performance impact of intra-node placement of network functions. The applied metrics include throughput, latency, and CPU utilization. Our analyzed setups include Open vSwitch, Linux network namespaces, and the DPDK IP Pipeline as different chaining technologies. We introduce a model to estimate the maximum achievable throughput with respect to placement of network functions considering chaining across several computational resources.**

I. Introduction

Network function virtualization (NFV) is designed to replace hardware appliances, which provide a specific network function, by virtualized network functions (VNF). NFV enables network operators to implement network functions in software, with the ability to run on commodity hardware that is less expensive compared to specialized hardware appliances. Next to lower asset- and operating costs, NFV provides higher flexibility and scalability, which enables network operators to adapt the maintained network to their customers' requirements. Networking tends towards differentiated processing chains for different kind of traffic via flow-specific processing paths of interconnected VNFs. This is called service or network function chaining (NFC respectively SFC). Network function chains are a recent research topic, comprising different research directions and needs of specification and standardization.

This paper determines performance implications caused by the placement of network function chains on a single physical host with limited resources. To determine implications for the implementation and architecture of the NFC, we measured chains of network functions restricted to VNFs performing port-to-port packet forwarding. It includes novel measurements to show the influence of intra-node placement of chained network functions regarding throughput, latency, and costs per packet. The network functions were chained as an ordered set while each packet is processed by each chain element. Figure 1 illustrates our baseline performance tests. The questions to be answered concern the influences of chained network functions placed on a physical host. We describe how different resource-to-VNF bindings improve or harm NFC performance. Modeling of the resource requirements of NFC and its performance with respect to different resource-to-VNF bindings is the second aspect.

The remainder of this paper is structured as follows: In Section II, we discuss related work. Section III describes a model purposed to approximate maximum throughput of network function chains, depending on the per packet processing costs. Next we analyze our measurement results for network function chains implemented with Open vSwitch (OvS) and Linux Namespaces in Section IV. Chains based on Data Plane Development Kit (DPDK) are measured in Section V. We sum up our results in Section VI.

II. Related Work

NFC has found its way into carrier-grade networks. In 2015, the conceptual architecture of NFC was specified by RFC 7665 [1], titled "Service Function Chaining Architecture". Since then, different use cases and its individual benefits as well as research directions of service chaining have been discussed [2], [3]. Research directions include optimization strategies for decomposition and aggregation of services and service blocks, strategies for service instance deployment, continuous network service delivery, and security aspects [3]. Mehraghdam et al. [4] describe a context-free language to formalize the chaining of network functions. They developed a mixed integer quadratically constrained program, with the purpose to find a high-performance placement of network functions. The authors consider data rate, number of network nodes, and latency as metrics of interest to optimize VNF placement.

Although NFC is a frequently studied topic, performance analysis of chains are rare and VNF placement is analyzed from a rather theoretical point of view.

However, several studies of NFC performance and relevant methodology for NFC benchmarking exist. In 2009, Dobrescu et al. [5] built a Click router that runs

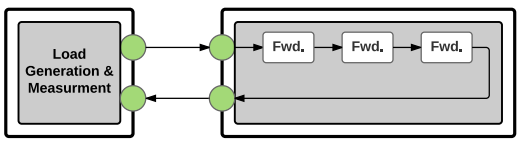


Figure 1: Illustration of the forwarding chain setups

978-1-5090-4026-1/17/$31.00 ©2017 IEEE

Table I: Explanation of model parameters

type of costs (per packet)	expression
total costs of a NF chain of length n	$C_{total,n}$
processing costs by i-th VNF	C_{VNF_i}
processing costs by n VNFs	$C_{VNF,n}$
receiving costs via a physical interface	C_{RX}
transmitting costs via a physical interface	C_{TX}
costs for passing packets from VNF_i to VNF_j	$C_{link_{i,j}}$
costs for chaining between n VNFs	$C_{chaining,n}$
costs for packets dropped in software	$C_{SWdrops}$
costs of other tasks, done by the OS	C_{other}
available computational power per core in $\frac{cycles}{sec}$	f_{CPU}

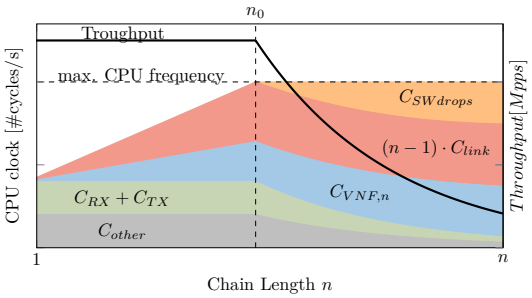


Figure 2: Composition of total costs per packet

parallel on several cores and analyzed different strategies for chaining of the Click modules. Today, measurement results of their study are overtaken by changed hardware and software architectures. Emmerich et al. describe a quantitative and qualitative study on the performance characteristics of Open vSwitch [6]. They consider the influence of physical and virtual network interfaces but did not study the effect of chaining. Livi et al. [7] study chains of Linux namespaces and different ways to connect them. They describe the decreasing throughput t_{out} due to an increasing chain length n and the percentage of packet loss per chain element l to be similar to the power law $t_{out} = (1-l)^n \cdot t_{in}$. Livi et al. [7] measure throughput with packet sizes up to 70 kB but neglect measurements with small packet sizes which are known to be demanding scenarios of packet processing in software [8]. Martins et al. [9] show the chaining performance of their proposed NFC framework ClickOS. Panda et al. [10] proposed NetBricks and compare its chaining performance with containers and VMs chained with two different switches. The throughput and latency tests that still apply today were already defined in 1999 by RFC 2544 [11]. In 2016, Kim et al. [12] proposed a draft to the Benchmarking Methodology Working Group of the IETF that describes methodology to benchmark NFC setups. The proposed performance metrics are end-to-end (E2E) latency, E2E packet loss rate, and E2E bandwidth. We consider these in our paper. Additional IETF drafts related to NFC exist: Kumar [13] describe the use cases of NFC in data centers. NFC in mobile networks is addressed by Haeffner et al. [14]. Conveying information between NFC control elements and data plane functional elements was proposed by Boucadair [15]. Scalability of NFC was addressed by Ao and Mirsky [16].

III. Modeling NFC processing costs

To explain the throughput of chained network functions, we introduce a model, based on the processing costs per packet, expressed as required CPU cycles. The underlying assumption is that CPU cycles are the main limiting factor which leads to the following constraint for maximum throughput: $Throughput[pps] \leq \frac{f_{CPU}}{C_{total,n}}$. All parameters are listed in Table I.

The total costs $C_{total,n}$ are constituted by different components: On its way each packet has to be processed by n VNFs, while each VNF_i causes costs C_{VNF_i}. The processing costs $C_{VNF,n}$ for the first n VNFs are calculated as: $C_{VNF,n} = \sum_{i=1}^{n} C_{VNF_i}$. To be processed, each packet has to be received and transmitted via a physical interface and has to be forwarded via $n-1$ links between the n VNFs. This leads to the following formulas:

$$C_{chaining,n} = C_{RX} + \sum_{i=2}^{n} C_{link_{i-1,i}} + C_{TX} \quad (1)$$

$$C_{total,n} = C_{chaining,n} + C_{VNF,n} + C_{other} + C_{SWdrops} \quad (2)$$

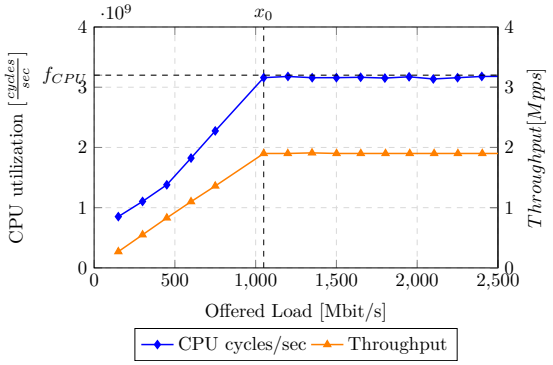
Costs that are not clearly related to a certain task or element, e.g. periodic statistics collection, are expressed by C_{other}. These are neither per packet costs nor independent of the load. The costs for dropped packets $C_{SWdrops}$ are explained in more detail in Section IV. If a network function chain gets overloaded, i.e., there are not enough resources to process all received packets, packets are dropped. Packets are either dropped in hardware, i.e., by the NIC without occupying computational resources or by the software. Software drops require computational resources as the task of dropping has to be done by the CPU. Therefore, SW drops increase the costs per packet for overloaded network function chains significantly.

Figure 2 shows a qualitative visualization of the costs for increasing chain lengths for an idealized chaining technology with constant C_{VNF_i} and a fixed offered load. It shows how the costs to run the VNFs and the costs to provide the chaining grow linear with the chain length n. $C_{SWdrops}$ occur for longer chains starting from n_0 (the point when $\frac{f_{CPU}}{C_{total,n} \cdot Throughput[pps]} = 100\%$. $C_{SWdrops}$ and the processing costs that are wasted for processing until the drop at hop n_{drop} (i.e. the increased costs per successfully processed packet in C_{RX}, $\sum_{i=2}^{n_{drop}-1} C_{link_{i-1,i}}$, and $C_{VNF,n_{drop}}$) consume further CPU cycles. Therefore, packet drops in software have an amplifying effect that leads to a decrease in throughput.

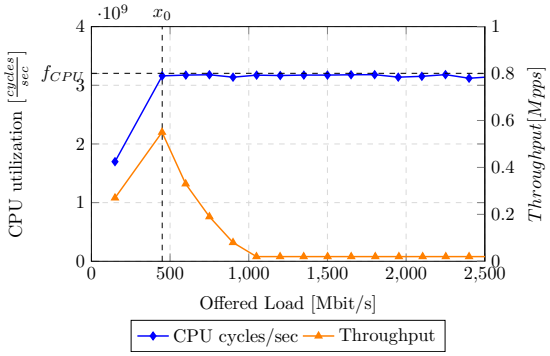
IV. Basic Performance Characteristics

The measurements in this section are conducted with Linux network namespaces, Linux kernel virtual Ethernet (vEth) interfaces and Open vSwitch (OvS). Although less efficient in terms of maximum throughput, these classical technologies are part of the operating system and are considered technologies for NFC. They are matured, provide a broad range of functionality and operators are already familiar with them.

OvS [17] is an open-source virtual switch that is frequently used for NFV or for SDN due to its support



(a) Length $n = 1$



(b) Length $n = 5$

Figure 4: Correlation between used CPU cycles and Throughput of OvS Chains

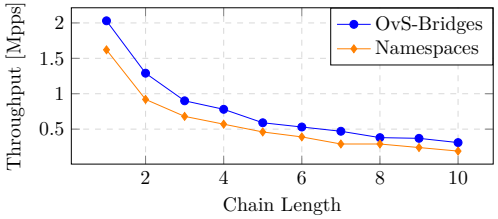


Figure 3: Max. throughput of OvS and namespace chains

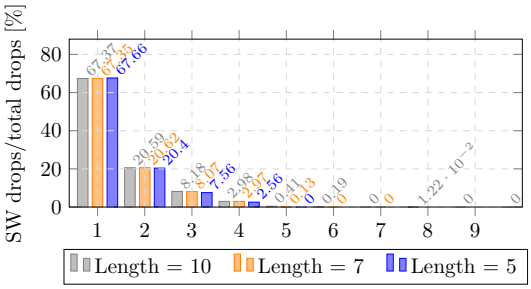


Figure 5: Dropped packets in each chain element

for OpenFlow [18]. To implement OvS forwarding chains, OvS bridges are interconnected by vEth pairs. Two vEth interfaces are added as ports to each bridge, while the bridge is configured to forward traffic from the one port to the other via OpenFlow rules. Livi et al. showed that the vEth pairs are not a bottleneck when measuring throughput in terms of Bytes per second [7].

Linux namespaces provide independent network stacks [19]. Known applications are Linux containers, like Docker or LXC. Each namespace maintains a routing table, an ARP table, and network interfaces. To implement our forwarding chains, each namespace is connected to one interface of a vEth pair while the other interface of the pair is connected to the next namespace. Each namespace is configured to forward incoming packets via a default route. Therefore, routing table entries and ARP table entries are created in each namespace.

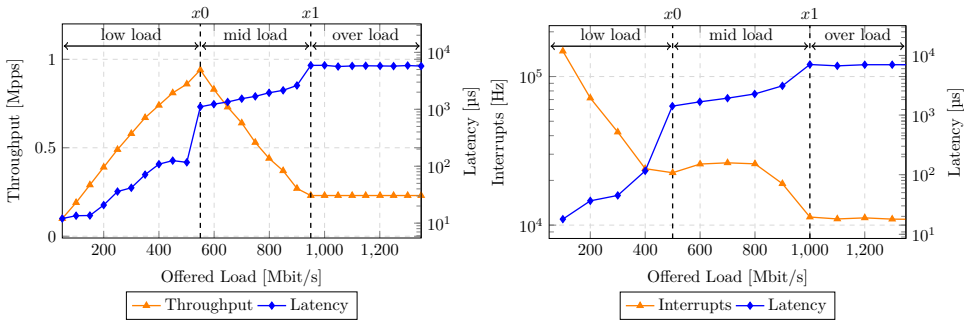
A. Test Setup

Our test setup for OvS bridge- and namespace chains consists of two physical hosts, a load generator and a device under test (DuT) as shown in Figure 1. The DuT has an Intel Xeon E3-1230 processor with four physical cores operating at 3.3 GHz and a dual-port 10 Gbit/s Intel X520-T2 NIC. Dynamic frequency scaling of the CPU was disabled to avoid measurement artifacts. Both machines run a Debian Linux with kernel version 3.16. and Open vSwitch 2.3.0. Profiling of the DuT is done with *perf*. The load is generated with the packet generator MoonGen [20]. All graphs show measurements under constant load of 64 B packets. By testing with bigger packets, we confirmed that packet size has no effect on the costs per packet. For the measurements on multi-CPU systems, we use a system with two Xeon E5-2640 processors operating at 2.0 GHz and two Intel X520-T2 dual-port NICs.

B. Throughput Measurements

First chain lengths from $n = 1$ up to $n = 10$ are measured, for OvS chains and for namespace chains. Figure 3 shows the maximum **throughput** determined by our measurements. OvS bridges and namespace forwarding cause different per packet costs $C_{VNF,i}$ and accordingly achieve different throughputs. Emmerich et al. [6] already found these differences between Linux forwarding and an OvS for a chain length of 1. While $C_{VNF,i}$ is different for both kinds of chains, the links are implemented with vEth pairs in both cases. Therefore, $C_{chaining,n}$ is equal for the OvS and for namespace chains of the same length. In the following we discuss effects at the example of OvS chains but results also apply to namespaces.

Figure 4 reveals another important aspect of chained setups. In an OvS setup with a chain of length $n = 1$ throughput and CPU cycles per packet are not affected by load above the maximum throughput marked by x_0 in Figure 4(a). Figure 4(b) shows the throughput of an OvS bridge chain of length $n = 5$. First, the throughput increases with the offered load until the maximum throughput, marked with x_0, is reached. For rates $r_0 > x_0$ the throughput decreases again until a minimum throughput, depending on the chain length n, is reached. This effect occurs for all chain lengths $n > 1$. It is explained by CPU cycles consumed by packets that get dropped in software. Different to drops done by the NIC, these waste CPU cycles (cf. Section III) and decrease the throughput. Additionally, drops early in the chain waste fewer cycles, compared to packets dropped later in the chain as $\forall n < m$:



(a) Length $n = 3$ (b) Length $n = 3$

Figure 6: Correlation between Latency and (a) Throughput, and (b) Interrupts/sec of OvS Chains

Table II: Placement dependent transmit and receive costs

remote NIC	throughput [Mpps]	ΔC [cycles]
–	1.18	–
TX	1.08	$\Delta C_{TX} = 157$
RX	1.16	$\Delta C_{RX} = 29$
TX+RX	1.09	$\Delta(C_{RX} + C_{TX}) = 140$

$C_{chaining,n}+\sum_{i=1}^{n} C_{VNF,i} << C_{chaining,m}+\sum_{i=1}^{m} C_{VNF,i}$. To investigate this, we analyze drop statistics for chains of lengths $n = 5$, $n = 7$ and $n = 10$. Figure 5 shows the distribution of drops in software with an offered load of 1.4 Mpps (64 B packets); each chain element in Figure 5 drops approx. $p_{SW} = 67\%$ of its traffic and only $p_{NIC} = 30\%$ were dropped by the NIC. SW drops and their distribution in the chain depend rather on the offered load and show no significant difference for the chain length. The average wasted cycles C^* per dropped packet is as follows: $\overline{p_{NIC}}*(C_{RX}+C_{SW drops}+\sum_{i=0}^{n-1}(\overline{p_{SW}})^i*p_{SW}*C^*_{chaining,i})$, with $\overline{p_{NIC}} = 1 - p_{NIC}$.

To analyze the costs for transmitting and receiving packets, we used the dual CPU server as DuT. We fixed the OvS chain to length $n = 1$ and measured the throughput in four different scenarios. The baseline performance was measured with the TX and RX NIC connected to the processing CPU. This was compared to setups where RX, TX, and both RX+TX is not directly connected to the processing CPU but remotely via the other CPU. The performance differences ΔC_{RX} and ΔC_{TX} listed in Table II are caused by NUMA for receiving or transmitting packets. The difference between $\Delta(C_{RX} + C_{TX})$ and $\Delta C_{RX} + \Delta C_{TX}$ is caused by computations taken over by the CPU that is in direct responsibility of the NIC.

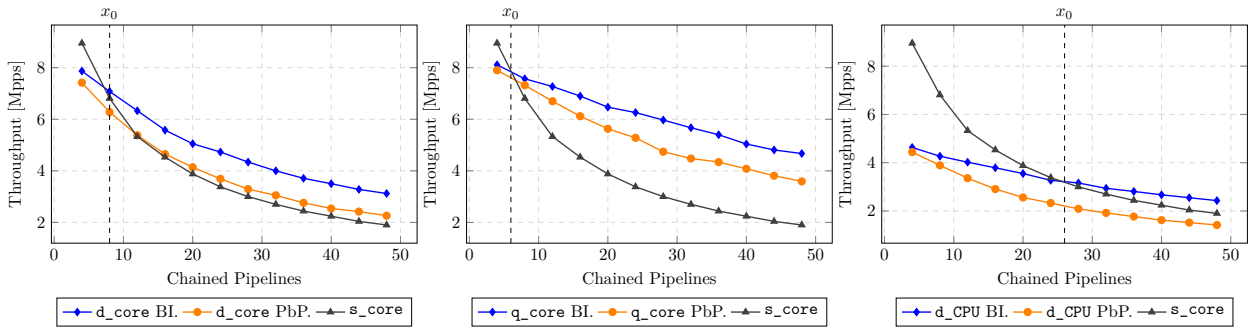
C. Latency measurements

Latency changes with the CPU cycles per packet and its waiting times. While CPU cycles can be measured, waiting cycles can only be derived indirectly by the knowledge that a full queue leads to packet drops. Figure 6(a) illustrates throughput in Mpps and latency in microseconds for different loads measured for an OvS chain with length $n = 3$. For all OvS and namespace chains with length $n > 1$, three ranges of offered loads can be identified. The boundaries decrease with increased chain lengths. The first range concerns rates with $r_0 \in [0, x_0]$. No packet loss occurs and latency increases moderately with the offered load until the threshold rate x_0 is reached. At x_0 buffers fill up due to overload and latency increases significantly. For rates $r_1 \in [x_0, x_1]$ throughput decreases and packets get dropped. The moderate increase of latency in that range is caused by delayed interrupts, i.e., the Interrupt Throttling Rate (ITR) (cf. [21]). This matches to the increasing latency for chains stressed with mid loads and correlates with our measurements of executed interrupts per second. Figure 6(b) shows the effect of interrupts towards latency. Last but not least, rates $r_2 \in [x_1, \infty]$ overload the NFC under test. This gets visible via a minimized number of interrupts per second and maximized latency in Figure 6(b).

V. Chaining Strategies

Software for packet processing like the Data Plane Development Kit (DPDK), Snabb [22], or netmap [8] is designed for high data and packet processing efficiency. Therefore, they bypass the networking functionality of the OS and the driver. NFC frameworks use these, e.g. Netbricks [10] uses DPDK and ClickOS [9] uses netmap. In the following, we use the IP pipeline [23] application that is provided by DPDK. Each DPDK IP pipeline element can be bound to a computational resource, i.e., a CPU socket, CPU core, or virtual CPU core. We use DPDK pass-through pipelines with various lengths and CPU core bindings to analyze the linkage costs $C_{link_{ij}}$ (cf. Section III) with respect to the node internal placement of the network function i and j. The used DPDK IP pipelines are based on DPDK v16.11.1.

To analyze chaining costs, we use two patterns to distribute pipeline elements to the available CPU resources. Block-Interleaving (BI) represents the distribution scheme with the minimal number of core switches, respectively socket switches. BI distributes blocks of subsequent pipeline elements with evenly distributed length to each of the CPU resource. The second configuration is called Pipeline-by-Pipeline-Interleaving (PbP). PbP requires a packet to change its computational resource with every element of the chain. It implicates highest chaining costs per packet, the core, respectively socket, is switched after each processing element. We applied the BI and PbP pattern to two (`d_core`) and four (`q_core`) CPU cores on the same CPU and to two cores on different CPU sockets (`d_CPU`). For comparison we deployed each chain to a single core (`s_core`).



(a) d_core (b) q_core (c) d_CPU

Figure 7: Maximum throughput of DPDK IP Pipeline chains with different lengths

Figure 7 shows the maximum throughput of DPDK IP pipeline chains with a length of 4 up to a length of 48, for the `s_core`, `d_core`, `q_core`, and `d_CPU` setup with PbP and BI. It shows that link costs differ for three fundamental configurations:

- Intra-core links connect two chain elements, both placed on the same physical core.
- Inter-core links connect two chain elements, while both elements are bound to two different physical cores, that are located on the same CPU. (Figure 7(a), Figure 7(b))
- Inter-CPU links, that interconnect chain elements, while both elements are bound to different CPUs. (Figure 7(c))

The throughput measured on one single core decreases exponentially, while the throughput of the other setups decreases significantly slower for two cores and even slower for four cores. Commencing with our setups with two cores on one CPU socket, the influence of core switches on maximum throughput was quite moderate. The overhead implicated by switching the CPU socket is significantly higher as the overhead implicated by switching the core inside a CPU socket. For short chain lengths, the overhead of the core switches can not be compensated by the increase of computational resources. For chain lengths $n \leq n_0$, the BI approach achieves higher throughput. For all chains above the break even point, the computational demand of the whole chain increases significantly while the number of core switches does not change. The break even point for setups with more core switches (PbP) is reached later, for longer chains that require higher overall computation. For switches of the CPU socket the break even point is reached for even longer chains at $n_0 = 26$ as it can be seen in Figure 7(c). The decision problem of an optimal NFC configuration is as followed:

$$max(tp_n^{s\_core}, tp_n^{d\_core/CPU}) = \begin{cases} \frac{f_{CPU}}{C_{total,n}^{s\_core}} & \text{for } n \leq n_0 \\ \frac{2 \cdot f_{CPU}}{C_{total,n}^{d\_core/CPU}} & \text{for } n > n_0 \end{cases} \quad (3)$$

The placement of NFC elements also impacts the latency. In Figure 9(a) and Figure 9(b) latencies for a chain of length $n = 24$ are measured. In case of overload, latency is dominated by the time that packets have to wait

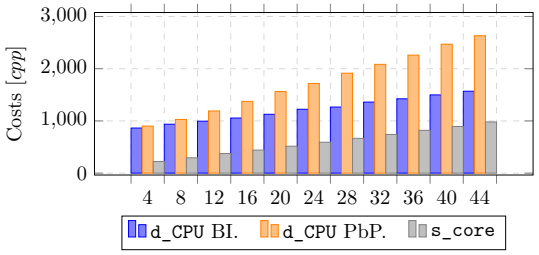


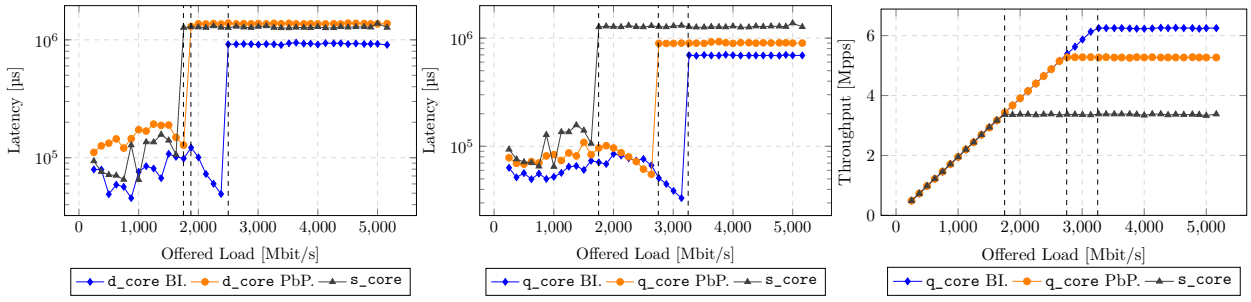
Figure 8: Cycles per packet, DPDK IP Pipelines, `d_CPU`

in buffers to be processed. Depending on the position of the bottleneck in the chain the congestion only fills up buffers of previous elements. The second observation is that each switch of a core adds processing delays by itself. This delay is orders of magnitude lower then the waiting delays and gets visible for latencies below overload when we compare BI with PbP. The numerous core switches with PbP are nearly as expensive in terms of latency as the congestion in case of the single core setup. As the load of core switches in case of the PbP pattern is split across four inter-core links, the latency decreases compared to the latency of the two core PbP setup, due to a shorter period a packet is buffered at the bottleneck. The BI pattern improves latency compared to the two core setup, too. The congestion caused by the first bottleneck is already triggered after $n/4$ pipeline elements. As the dashed lines in Figure 9(b) and in Figure 9(c) show the points when the system is overloaded are recognizable by the latency as well as by the throughput.

For chain lengths $n > 24$, the approach with two CPUs was more performant as the approach with one single core, because of the distribution of overhead on a high demand of computational power by a high number of chain elements. Figure 8 shows the costs per packet calculated by the cycles offered by the core(s) and the measured maximum throughput. Comparing the costs of the single core measurement to the costs of the BI two socket measurement, the significant overhead can be determined, caused by one CPU switch.

VI. Conclusion

We have done extensive analysis of NFC with classical technologies and advanced high-speed packet processing frameworks. Discovered limitations are expressed



(a) median latency for `d_core` (b) median latency for `q_core` (c) throughput for `q_core`

Figure 9: Measurements for chain length $n = 24$

by equations and optimization problems for NFC design. Analyzed technologies like OvS and Linux namespaces are used by container technologies, which are seen as a key component of NFC setups. The innovation potential shown at the example of DPDK for implementations of NFC is representative for a zoo of technologies that can be used for acceleration of NFC like Snabb [22], Netbricks [10], or ClickOS [9]. Based on our analysis, we derive a few guidelines: First, the earlier a bottleneck occurs in a chain, the lower is the impact of congestion in the buffers. NFC optimized software helps to avoid negative knock-on effects: with the wrong chaining technology or configuration the increase in latency and decrease in throughput caused by a VNF may be amplified by several orders of magnitude. We also quantified the decision when a VNF should be placed on a new CPU core. Generally, core switching within a chain should be avoided if possible, e.g., in favor of distributing packets equally to multiple parallel chains.

References

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

[2] G. Brown and H. Reading, "Service Chaining in Carrier Networks," Feb. 2015. White Paper.

[3] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research Directions in Network Service Chaining," in *IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov. 2013.

[4] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and Placing Chains of Virtual Network Functions," in *IEEE 3rd Int. Conf. on Cloud Networking (CloudNet)*, 2014.

[5] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: Exploiting Parallelism to Scale Software Routers," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, SOSP, 2009.

[6] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "Performance Characteristics of Virtual Switching," in *IEEE 3rd Int. Conf. on Cloud Netw. (CloudNet)*, 2014.

[7] S. Livi, Q. Jacquemart, D. L. Pacheco, and G. Urvoy-Keller, "Container-Based Service Chaining: A Performance Perspective," in *IEEE 5th Int. Conf. on Cloud Netw. (CloudNet)*, 2016.

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

[9] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "ClickOS and the Art of Network Function Virtualization," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI'14)*, 2014.

[10] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, "NetBricks: Taking the V out of NFV," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016.

[11] S. Bradner and J. McQuaid, "RFC 2544: Benchmarking Methodology for Network Interconnect Devices," 1999.

[12] T. Kim, H. Yu, C. Jeong, Y. Han, and E. Paik, "Internet-Draft: Benchmarking Methodology for Service Function Chain Performance," Internet-Draft draft-kim-bmwg-sfc-benchmark-meth-00, Internet Engineering Task Force, Oct. 2016.

[13] S. Kumar, C. Obediente, M. Tufail, S. Majee, C. Captari, and S. Homma, "Internet-Draft: Service Function Chaining Use Cases In Data Centers," Internet-Draft draft-kumar-sfc-dc-use-cases-06, Internet Engineering Task Force, Feb. 2017.

[14] W. Haeffner, J. Napper, M. Stiemerling, D. Lopez, and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks," Internet-Draft draft-ietf-sfc-use-case-mobility-07, Internet Engineering Task Force, 2016.

[15] M. Boucadair, "Service Function Chaining (SFC) Control Plane Components and Requirements," Internet-Draft draft-ietf-sfc-control-plane-08, Internet Engineering Task Force, Oct. 2016. Work in Progress.

[16] T. Ao and G. Mirsky, "Analysis of the SFC scalability," Internet-Draft draft-ao-sfc-scalability-analysis-02, Internet Engineering Task Force, Mar. 2017.

[17] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The Design and Implementation of Open vSwitch," in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI'15)*, 2015.

[18] "Why Open vSwitch?." https://github.com/openvswitch/ovs/blob/master/Documentation/intro/why-ovs.rst.

[19] "IP-NETNS(8)." http://man7.org/linux/man-pages/man8/ip-netns.8.html.

[20] 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.

[21] 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 *Int. Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2015)*, July 2015.

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

[23] "34. Internet Protocol (IP) Pipeline Application." http://dpdk.org/doc/guides/sample_app_ug/ip_pipeline.html.