Performance Benchmarking of a Software-Based LTE SGW

Stanislav Lange*, Anh Nguyen-Ngoc*, Steffen Gebert*, Thomas Zinner*, Michael Jarschel‡, Andreas Köpsel§, Marc Sune§, Daniel Raumer‖, Sebastian Gallenmüller‖, Georg Carle‖, and Phuoc Tran-Gia*

*University of Würzburg, Institute of Computer Science, Chair of Communication Networks, Würzburg, Germany
{stanislav.lange, anh.nguyen, steffen.gebert, zinner, trangia}@informatik.uni-wuerzburg.de

‖Technische Universität München, Department of Informatics, Chair for Network Architectures and Services, Germany
{raumer, gallenmu, carle}@net.in.tum.de

‡Nokia, Munich, Germany
michael.jarschel@nokia.com

§BISDN, Berlin, Germany
{andreas.koepsel, marc.sune}@bisdn.de

***Abstract*—Network Functions Virtualization (NFV) is a concept that aims at providing network operators with benefits in terms of cost, flexibility, and vendor independence by utilizing virtualization techniques to run network functions as software on commercial off-the-shelf (COTS) hardware. In contrast, prior solutions rely on specialized hardware for each function. Performance evaluation of such systems usually requires a dedicated testbed for each individual component. Rather than analyzing these proprietary black-box components, Virtualized Network Functions (VNFs) are pieces of software that run on COTS hardware and whose properties can be investigated in a generic testbed. However, depending on the underlying hardware, operating system, and implementation, VNFs might behave differently. Therefore, mechanisms for the performance evaluation of VNFs should be similar to benchmarking of software, where different implementations are compared by applying them to predefined test cases and scenarios. This work presents a first step towards a benchmarking framework for VNFs. Given two different implementations of a VNF that acts as LTE Serving Gateway (SGW), influence factors and key performance indicators are identified and a comparison between the two mechanisms is drawn.**

***Index Terms*—SDN, NFV, Benchmarking, VNF.**

I. Introduction

Specialized middleboxes are an integral part of today's network infrastructure. They provide, mostly hardware-based, solutions for a wide range of applications including firewalls, load balancers, (overlay) network management, and monitoring. Although these specialized devices bring benefits in terms of performance, several drawbacks regarding cost, flexibility, and vendor-dependence have created a trend towards Network Functions Virtualization (NFV). The core idea of NFV is to leverage virtualization mechanisms in order to migrate the functionality of the specialized middleboxes to software running on COTS hardware. In addition to reducing the costs for purchasing the hardware, network operators gain advantages like vendor independence and flexibility.

These network functions that are separated from the data plane and virtualized on COTS hardware raise questions about the performance evaluation of networks and their components that rely on NFV. As functionality is no longer necessarily co-located with forwarding elements questions arise about the performance evaluation of networks and their components that rely on NFV. We discuss the challenges in Section II. In Section III, we illustrate these challenges in a case study and provide solutions by means of an exemplary performance evaluation of an SDN/NFV-based LTE Serving Gateway (SGW) in the mobile core. Section IV concludes this work.

II. Benchmarking of VNFs

Each virtualized network function (VNF) has a specified behavior that relates to sets of performance indicators and network parameters. Therefore, a *general* framework to benchmark network functions does not exist. Nevertheless, guidelines for the performance evaluation of VNFs are essential to achieve comparable benchmarking results.

A. State of the Art

In 1994, the need for well defined benchmarks of network interconnect devices was served by RFC 2544 [1]. It defined how latency and throughput of a device under test (DuT) should be measured and, thus, developed to a de facto standard. The defined guidelines (including key performance indicators and relevant benchmarking parameters) were used for benchmarks of network devices like routers, switches, or firewalls. Subsequent documents extend the methodology by tests [2], updates, and remarks [3], [4], adapting it to keep pace with the capabilities of network devices.

The grown performance of COTS hardware aided a change towards software-based solutions that provide an additional architectural layer of abstraction compared to closed network

boxes. The performance of these systems depends on hardware and the software implementation. Due to the latter, a more fine grained analysis of performance limitations and their causes is possible. Rather than performance modeling of black box hardware, benchmarking of software has moved into the focus. The additional layer was also used to implement further metering points in it. However, results of these white box measurements have to be considered carefully, as metering on the DuT may affect the tested behavior [5].

Furthermore, a recent IETF draft [6] discusses the novel challenges that are introduced with VNF benchmarking, e.g., phenomena like shared resources between multiple VNFs and their impact on the performance of individual VNFs. In [7], the performance of DPDK-accelerated switching and routing VNFs is evaluated with respect to throughput and latency. The analysis shows that, when offered traffic to two gigabit Ethernet physical interfaces, NFV-based approaches can achieve line rate throughput as well as a latency that does not impact performance in an enterprise network. Similarly, [8] shows a DPDK-accelerated virtualized system that achieves performance levels that are close to that of non-virtualized systems in terms of latency. The authors of [9] focus on benchmarking NFV infrastructures with respect to resilience and the effects of faults on their performance. Further options for hardware acceleration for VNFs include FPGAs [10], [11] as well as NPU and GPU resources [12], [13], each resulting in different trade-offs between flexibility and performance.

[14] and [15] present platforms that enable fast packet processing on COTS hardware running VNFs. In the case of forwarding, both solutions can process packets at line rate on a 10 Gbps link while introducing latencies in the order of magnitude of 50 μs.

B. Challenges with VNFs

To benchmark the performance of a VNF, the challenges of classical benchmarking get extended by three additional problems: *complexity*, *abstraction*, and *concurrency*.

The increased system *complexity*: Software-based network devices introduce only a single layer of abstraction compared to dedicated bare metal devices. However, the use of (host) virtualization techniques and the execution in the cloud adds further abstraction layers that each introduce their own performance limitations, i.e., the hypervisor, the virtual switch, and the physical and virtual interfaces [5]. This work investigates whether hardware acceleration mechanisms, e.g., Intel's DPDK [16], improve performance or if the performance remains the same due to higher complexity.

The virtualization of a network function inherits the *abstraction* of the data plane from the executing hardware. Therefore, absolute performance values are less meaningful without an understanding of how they relate to the performance of underlying components. This, for instance, leads to the following question: if the processing power of the unit performing NFV is too low, how can it be accelerated if the employed cloud solution has doubled the amount of physical CPU cores?

In addition to increased complexity and a higher level of abstraction, the interpretation of results and application on real world setups gets even more challenging due to *concurrency*. In real setups, (virtual) network functions interact with other components, which may be other VNFs or the data plane.

III. Case Study

In order to demonstrate performance benchmarking of a VNF, we compare two implementations of a network function acting as a Serving Gateway (SGW) in the mobile core. After introducing the basic terminology used in this context, the experiment setup and details of the two alternative implementations are presented. Finally, the implementations are compared with respect to various performance indicators.

A. Network Function Under Test

The LTE Evolved Packet Core (EPC) is comprised of various specialized components. The components' responsibilities range from purely control plane related tasks, e.g., in case of Mobility Management Entities (MMEs), to combinations of control and user plane processing as in the case of SGWs. In this work, the user plane functionality of the latter is moved to a virtual network function. In particular, its task lies in transporting user data traffic through the LTE network. For this purpose, the traffic is encapsulated via the GPRS Tunneling Protocol (GTP). More specifically, the UDP-based GTP-u protocol is used. Figure 1 depicts the structure of GTP-u packets that are sent to the VNF discussed in this work. On top of UDP in the stack, a GTP header indicates the presence of an encapsulated IP packet that follows immediately. In order to identify GTP packets' membership to a particular tunnel, the GTP header includes a Tunnel Endpoint Identifier (TEID). In the following, GTP tunnels are also referred to as bearers.

Little information regarding the number of GTP-u packets a traditional SGW can handle is publicly available. According to [17], the signaling load (GTP-c) of an LTE network is around 94,000 messages per second per million smartphone users.

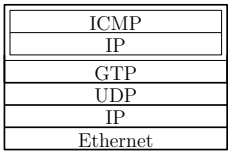

Fig. 1: Structure of the GTP-u packets considered in this work.

B. Setup

Figure 2 shows the two main components of the testbed as well as the main steps of the experiment. First, GTP-u traffic is generated via the Spirent C1[1], a dedicated hardware-based traffic generator equipped with four 1 GbE interfaces. This traffic is processed by the VNF installed on a commodity server[2] running a Linux operating system (OS)[3].

[1] http://www.spirent.com/Ethernet_Testing/Platforms/C1_Chassis

[2] Intel Xeon E5-2620 v2 CPU at 2.10 GHz, Intel I350 NICs, 32 GB of RAM

[3] 64 bit version of Debian 7.7 (wheezy, kernel version 3.2.0-4-amd64)

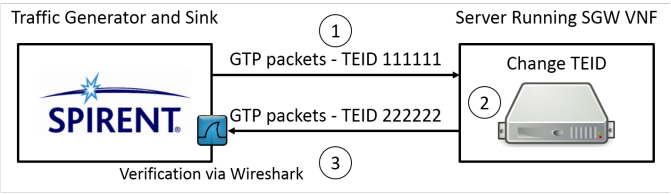

Fig. 2: Test setup

After processing, the server forwards the altered GTP-u packets to the ingress port of the Spirent chassis, where performance indicators are recorded by hardware capture cards. These include the one way delays, the received packet rate, as well as the amount of packet loss. In order to obtain the net processing times at the server, packet transmission times are deduced by performing a loopback test, i.e., transmitting packets of different sizes from the generator's sending port to its receiving port. Furthermore, capturing packets at the receiving port of the traffic generator via tools like Wireshark[4] allows verifying that packet headers are modified as expected.

To measure the baseline performance, the SGWs are configured to change the TEID as well as the IP source and destination fields of incoming GTP-u traffic according to currently present bearer entries.

Both implementations of the SGW VNF are based on the eXtensible OpenFlow data path daemon[5] (xdpd), which allows designing data path elements and supports GTP traffic. In particular, the xdpd software is responsible for matching incoming packet headers against the installed flow table entries and, in the presented case, modifying their TEID. In the xdpd version used in this work, the matching algorithm consists of a loop that goes through each flow table entry and checks for a match. Hence, the runtime of the matching algorithm is linear in the number of flow table entries.

The first SGW VNF uses MMAP-based xdpd, where packets are copied to the OS user-space and thus, are expected to have longer processing times. In case of the used Linux OS, the network I/O API that is utilized by this implementation is the *New API* [18] (NAPI). Hence, in the following, this implementation is referred to as the NAPI-based approach.

Second, a DPDK-accelerated implementation[6] is evaluated which promises faster packet processing through reduced overhead for packet I/O [16], [19]. When running DPDK-based applications, a core mask can be specified in order to change the number of cores that are used. In all presented experiments, this parameter is set to 0x03, which corresponds to utilizing a total of two cores, one core for management and one for I/O on all ports. Increasing the number of cores did not affect the results, neither with respect to the processing times nor with respect to the maximum packet rate that can be handled without packet loss. This parameter may become relevant in the context of link capacities beyond 1 Gbps.

[4]https://www.wireshark.org/
[5]http://www.xdpd.org/, version 0.7.5
[6]Version 1.7.1 of the DPDK libraries is used.

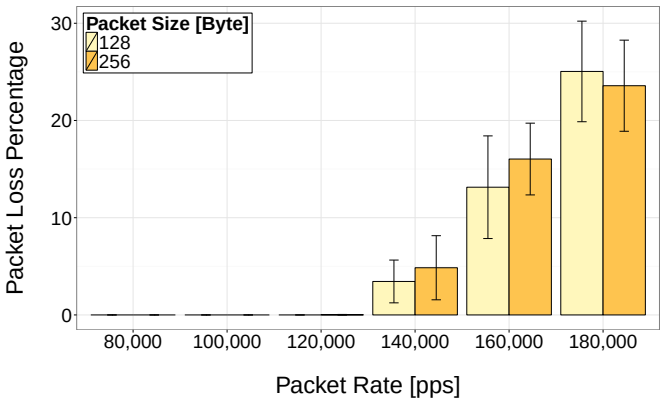

Fig. 3: Packet loss in case of small packet sizes when using the NAPI-based SGW implementation.

The systems are offered loads with varying packet sizes and rates. Each test run lasts 5 minutes and experiments for each set of configuration parameters are repeated 10 times in order to obtain confidence intervals for the key performance indicators. In order to find the performance limits for use in a practical context, the load is increased until packet loss occurs. Additionally, the influence of the number of present bearers on the resulting processing times and packet loss is investigated. When multiple bearers are registered on the server, the traffic generator sends GTP packets with corresponding TEIDs in a round robin fashion, i.e., one GTP packet for each bearer.

C. NAPI-based SGW

In Section III-B, two key performance indicators are identified for the purpose of benchmarking an SGW implementation. The occurrence of packet loss shall be minimized in order to provide a reliable service as well as fast processing of requests. When designing a network architecture containing SGW components, the operator has to consider feasible alternatives that meet the requirements of the particular use case. By utilizing the presented benchmarking methodology, it is possible to quantify the performance and limits of implementations and thereby assist the decision making process.

In nearly all practical scenarios, packet loss needs to be avoided. Thus, Figure 3 presents an analysis of the influence factors on the occurrence of packet loss as well as limits regarding the maximum load that can be handled without packet loss. While the x-axis denotes the number of packets that are sent to the server each second, the bars indicate the resulting packet loss percentage. Differently colored bars correspond to different packet sizes and whiskers represent 95% confidence intervals obtained from 10 experiment repetitions.

Low packet rates, i.e., 120,000 pps and less, are handled without packet loss. However, rates beyond roughly 130,000 pps result in a steady increase of loss. As the confidence intervals for both of the displayed packet sizes overlap and the corresponding mean values do not deviate from each other significantly, the packet rate is identified as the main influence factor on the packet loss rate. As discussed in [20],

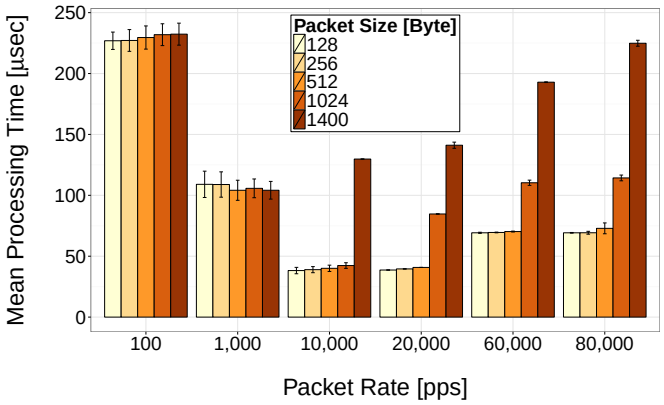

Fig. 4: Influence of packet rate and size on mean processing times using the NAPI-based SGW.

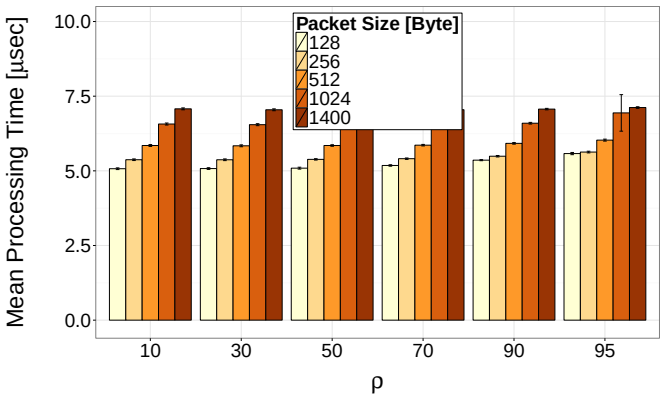

Fig. 5: Processing times for different link utilizations and packet sizes using the DPDK-enabled SGW implementation.

user space packet frameworks process packets of different sizes at an almost identical speed because only headers are copied and processed. Larger packet sizes are omitted due to the fact that these can be handled at line rate when only one bearer is installed. For example, a rate of roughly 89,000 pps corresponds to the capacity of the 1 Gbps link when a packet size of 1400 Byte is used. However, investigations regarding the processing times of packets show that not only packet rate but also packet size influences SGW performance.

After identifying the range of packet rates that can be handled by the NAPI-based implementation, benchmarking is performed with respect to processing times in this interval. Figure 4 displays mean processing times for packet sizes between 128 and 1400 Bytes, covering commonly observed values. The x and y-axis represent the packet rate and the mean processing time in microseconds, respectively. Again, 10 repetitions are performed in order to obtain confidence intervals and bar colors indicate the packet size. There are three main observations. First, very low packet rates, i.e., 100 and 1,000 pps, result in high processing times, roughly 230 and 110 μs, respectively. A possible explanation for this behavior is the interrupt mitigation mechanism that is part of the NAPI. In order to decrease the overhead that results from each individual packet causing an interrupt, this mechanism accumulates packets until either a certain amount of packets is collected or processing is initiated by a timeout. Second, the lowest processing times are observed for rates between 10,000 and 20,000 pps, corresponding to scenarios that are not affected by interrupt mitigation anymore and simultaneously do not expose the system to a high load. Finally, the processing time increases steadily when the maximum rate approaches the previously determined limits. While the packet size does not have a significant impact on the processing times observed at low packet rates, higher rates have a larger effect on the processing times of bigger packets. This can be explained by the fact that combining a given packet rate with different packet sizes results in changes to the link utilization, which also poses an influence factor on processing times.

D. DPDK-enabled SGW

While the NAPI-based SGW implementation is capable of processing packets of 1400 Bytes in size at line rate, it suffers from packet loss when receiving small packets at rates beyond 130,000 pps. In contrast to this behavior, the DPDK-enabled approach can handle even small packets, i.e., 128 Bytes in size, at line rate. This corresponds to a packet rate of around 970,000 pps, or an increase by a factor of more than 7 when compared to the maximum tolerable rate achieved by the previous user-space approach.

In addition to the performance gain in terms of maximum packet rate without packet loss, DPDK significantly accelerates the processing of packets at the SGW. Figure 5 presents the processing times measured in various conditions with respect to packet size and packet rate. In order to fit the various combinations of these two parameters into one graphic, the x-axis displays the link utilization, ρ, which is calculated as the ratio between the bandwidth that results from a particular configuration and the link capacity of 1 Gbps. The y-axis shows the mean processing times and bar colors denote different packet sizes. As processing times of the SGW implementation are very stable for the scenarios under test, the narrow confidence intervals are barely visible. The observed processing times start at roughly 5 microseconds and do not exceed 8 microseconds, corresponding to a speedup by a factor larger than 8 when compared to the processing times of the NAPI-based solution ranging from 40 to 230 μs. Furthermore, the packet size is the main influence factor on the resulting processing times, as indicated by almost constant values among configurations sharing the same packet size parameter. An increase in packet size results in an increase in the mean processing time. Nevertheless, an additional effect is visible: an increase in the packet rate also results in a slight but consistent increase of processing times.

E. Scenarios Featuring Multiple Bearers

All results presented so far are based on environments that feature only one single bearer. While these provide important

insights into the behavior of the different implementations and the influence factors on their performance, they are not sufficient for deriving practical guidelines with respect to the choice of implementation for a particular use case and dimensioning the system for a given load. Therefore, an investigation of the relationship between the number of bearers that are present in a system and various performance indicators is performed. Given a number of installed bearers, the maximum packet rate an implementation can handle without the occurrence of packet loss is empirically identified. Then, the processing times are measured for these scenarios.

Figure 6 provides an aggregated view on the packet rates that the SGW implementations discussed in this work can handle when different numbers of bearers are present. The numbers of bearers for which the rate limits are determined are between 1 and 400, as indicated by the x-axis. The y-axis shows the corresponding maximum packet rate that can be handled by a particular combination of SGW implementation and packet size, which are represented by line style and color, respectively. Solid lines denote the DPDK-enabled implementation, dashed lines denote the NAPI-based implementation. For the sake of clarity, only the two extreme values for the packet size are shown, i.e., 128 Byte and 1400 Byte.

In case of both implementations, a steady decrease of the maximum tolerable load is observed when the number of bearers is increased. The main reason for this behavior lies in the matching algorithm used by the xdpd software which both implementations have in common. As discussed in Section III-B, the matching procedure that is utilized has linear time requirements with respect to the number of flow rules in the SGW's table. Given the fact that increasing the number of bearers in the system corresponds to increasing this number of flows, it follows that a growing number of bearers also causes higher processing times. Consequently, the process of matching incoming packets becomes the system's bottleneck and affects the maximum packet rate that can be handled without the occurrence of packet loss.

When using the NAPI-based approach, both packet sizes yield almost identical values for the resulting maximum packet rate. This is consistent with the results from Section III-C, where no significant influence of the packet size on the maximum tolerable load is observed. Only for numbers of bearers below 50, the maximum rate for small packets exceeds that for big packets. This stems from the fact that the maximum packet rate is not only limited by the processing time of the server, but also by the link capacity, i.e., 1400 Byte packets are processed at line rate for up to 10 bearers.

In contrast, the gap between the two curves corresponding to the DPDK-enabled solution is significantly larger and closes only after 200 bearers are present in the system. As discussed in Section III-D, the DPDK-accelerated SGW can process packets of all considered sizes at line rate when one bearer is installed. Hence, the initial gap represents the rate limitation due to the link capacity. With an increasing number of bearers, however, the portion of the total processing time that is caused by the packet matching routine outweighs that caused by the

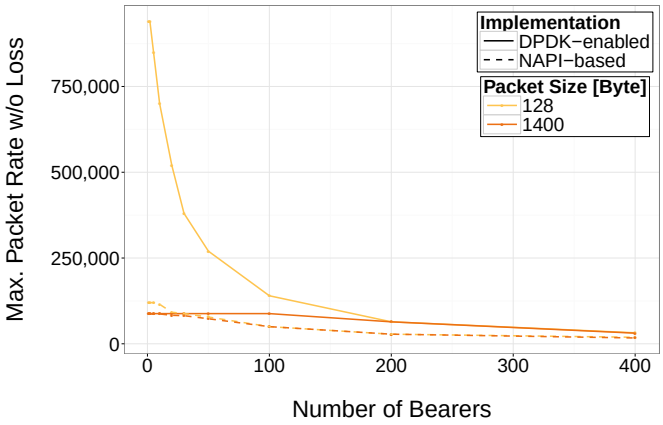

Fig. 6: Influence of the number of installed bearers on the maximum packet rate without the occurrence of packet loss.

network I/O API and the maximum rate decreases. After the maximum rate for small packets drops below the rate required for line rate with large packets, the rate limitation is not caused by the link capacity anymore and thus, both curves overlap.

For all considered packet sizes and numbers of bearers, the DPDK-enabled SGW implementation outperforms the NAPI-based approach in terms of maximum packet rate. While the advantage of the former is especially pronounced in the case of small packets and low numbers of bearers, there is still an improvement by a factor of almost two when considering the highest number of bearers. Independent of the network I/O mechanism, the maximum tolerable rate decreases due to the data plane software that is responsible for packet matching.

Although the rate limits determined in the previous paragraphs provide upper bounds for the load that can be applied to the systems without causing packet loss, the resulting processing times may not be feasible in practice. Thus, Figure 7 highlights the processing times of the SGW VNFs when facing these circumstances. Like in the previous figure, the x-axis shows the number of bearers that are present in the system. The y-axis displays the mean processing time observed when packets arrive at the maximum rate for a particular combination of network I/O API, packet size, and number of bearers. In addition to the bar color representing the aforementioned configuration, whiskers denote 95% confidence intervals that are obtained from 10 experiment repetitions.

Until a number of 100 bearers is reached, the DPDK-accelerated solution not only outperforms the NAPI-based approach with respect to the maximum tolerable packet rate, but also regarding the mean processing time of individual packets. In this interval, the former achieves processing times below 20 microseconds while the latter has processing times of roughly 100 and 250 microseconds for packet sizes of 128 and 1400 Byte, respectively. As soon as 200 or more bearers are installed, however, the processing times of the DPDK-enabled approach are higher than those of the NAPI-based implementation. The main reason for this behavior is that the packet rate and thus, the total amount of table lookups differ

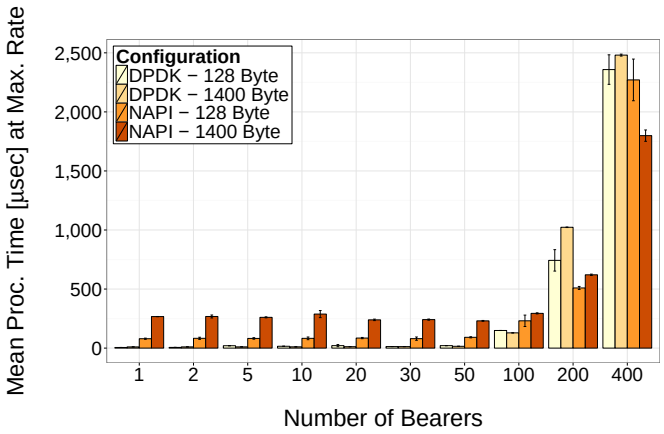

Fig. 7: Influence of the number of bearers on the processing time at the highest rate that can be handled without loss.

for the two implementations. As discussed in the context of Figure 6, the DPDK-enabled SGW is capable of processing almost two times more packets.

IV. Conclusion

This work presents a testbed and methodology for comparing the performance of two different implementations of a VNF that acts as a Serving Gateway (SGW) in the mobile core. One approach is based on the Linux NAPI, while the other uses Intel DPDK libraries. The performance is quantified by means of three indicators. First, the maximum amount of packets that can be handled per second without the occurrence of packet loss. Second, the processing time of each packet under different conditions in terms of link load. Third, the influence of the number of GTP bearers on the aforementioned measures. All measurements are performed with the Spirent TestCenter, an industry-level packet generator and test platform.

When performing the comparison with only one bearer in the system, a significant advantage of the DPDK-enabled approach is observed in terms of the maximum tolerable packet rate as well as the processing time of individual packets. In these scenarios, the DPDK-based solution achieves processing times that are orders of magnitude lower than those of the NAPI-based implementation. However, both implementations suffer from the matching algorithm used in the data plane component that constitutes the system's bottleneck in terms of processing times and, in turn, also maximum tolerable rate when multiple bearers are installed on the SGW. This common bottleneck results in a convergence of both approaches with respect to the investigated performance indicators.

There are multiple directions for future work, e.g., a performance evaluation featuring more sophisticated matching algorithms or an investigation of the behavior and limits of the implementations in the context of links whose capacity exceeds 1 Gbps. An additional aspect includes the impact on an SGW's performance when it needs to process incoming GTP-u user plane traffic and GTP-c control traffic simultaneously. Furthermore, an investigation of the influence of different combinations of VMs, hypervisors, and NICs on VNF performance in virtualized environments can aid in identifying practically feasible combinations.

Acknowledgments

This work has been performed in the framework of the CELTIC EUREKA project SASER-SIEGFRIED (Project ID CPP2011/2-5), and it is partly funded by the BMBF (Project ID 16BP12308). The authors alone are responsible for the content of the paper.

References

[1] S. Bradner and J. McQuaid, "RFC2544: Benchmarking Methodology for Network Interconnect Devices," IETF, 1999.
[2] R. Asati, C. Pignataro, F. Calabria, and C. Olvera, "RFC26201: Device Reset Characterization," IETF, 2011.
[3] S. Bradner, K. Dubray, J. McQuaid, and A. Morton, "RFC6815: Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful," IETF, 2012.
[4] "Y.1564: Ethernet service activation test methodology," ITU-T, 2011.
[5] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "Performance Characteristics of Virtual Switching," in *IEEE International Conference on Cloud Networking (CloudNet)*, 2014.
[6] A. Morton, "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure," Internet-Draft draft-morton-bmwg-virtual-net-03, 2015. [Online]. Available: http://www.ietf.org/internet-drafts/draft-morton-bmwg-virtual-net-03.txt
[7] Overture, Brocade, Intel, Spirent, and Integra, "NFV Performance Benchmarking for vCPE," Executive Summary, 2015.
[8] J. DiGiglio and D. Ricci, "High Performance, Open Standard Virtualization with NFV and SDN," White paper, Intel Corporation and Wind River, 2013. [Online]. Available: https://www.intel.eu/content/dam/www/public/us/en/documents/white-papers/communications-virtualization-snd-nfv-paper.pdf
[9] D. Cotroneo, L. De Simone, A. Iannillo, A. Lanzaro, and R. Natella, "Dependability Evaluation and Benchmarking of Network Function Virtualization Infrastructures," in *IEEE Conference on Network Softwarization (NetSoft)*, 2015.
[10] S. Byma, J. G. Steffan, H. Bannazadeh, A. L. Garcia, and P. Chow, "FPGAs in the Cloud: Booting Virtualized Hardware Accelerators with OpenStack," in *IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2014.
[11] X. Ge, Y. Liu, D. H. Du, L. Zhang, H. Guan, J. Chen, Y. Zhao, and X. Hu, "OpenANFV: Accelerating Network Function Virtualization with a Consolidated Framework in Openstack," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, 2014.
[12] Z. Bronstein, E. Roch, J. Xia, and A. Molkho, "Uniform handling and abstraction of NFV hardware accelerators ," *IEEE Network*, 2015.
[13] L. Nobach and D. Hausheer, "Open, elastic provisioning of hardware acceleration in NFV environments," in *International Conference and Workshops on Networked Systems (NetSys)*, 2015.
[14] J. Hwang, K. Ramakrishnan, and T. Wood, "NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2014.
[15] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "ClickOS and the Art of Network Function Virtualization," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2014.
[16] "Intel Data Plane Development Kit (DPDK)." [Online]. Available: http://dpdk.org
[17] "Signaling is growing 50% faster than data traffic," White Paper, Nokia Siemens Networks, 2012. [Online]. Available: http://networks.nokia.com/system/files%20/document/signaling_whitepaper_online_version_final.pdf
[18] J. H. Salim, R. Olsson, and A. Kuznetsov, "Beyond softnet," in *Proceedings of the 5th annual Linux Showcase & Conference*, 2001.
[19] 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.
[20] L. Rizzo, "netmap: A Novel Framework for Fast Packet I/O," in *USENIX Annual Technical Conference*, 2012.