KOMon – Kernel-based Online Monitoring of VNF Packet Processing Times

Stefan Geissler*, Stanislav Lange*, Florian Wamser*, Thomas Zinner*, Tobias Hoßfeld*
*Chair of Communication Networks, University of Würzburg, Germany
Email: {stefan.geissler|stanislav.lange|florian.wamser|zinner|hossfeld}@informatik.uni-wuerzburg.de

***Abstract*—The ongoing softwarization of networks comes with several advantages like cost efficiency, increased scalability, and better flexibility by migrating functionality from static, application specific hardware appliances to flexible, lightweight software solutions running on COTS hardware. In order to maximize the performance gains promised by the NFV paradigm, several challenges remain to be solved. In this work, we address the accurate acquisition of key performance indicators, specifically the packet processing time, of softwarized network functions. To this end, we present KOMon, a Kernel-based online monitoring tool to reliably measure the packet processing times of network functions through lightweight in-stack monitoring. We show that KOMon reports highly accurate values in different scenarios and discuss the applicability of the proposed mechanism for different use cases.**

***Index Terms*—Virtual Network Function, VNF Packet Processing, Online Monitoring, In-stack Monitoring, KOMon Tool.**

I. Introduction

The network function virtualization (NFV) paradigm promises greater flexibility while reducing cost and maintenance overhead by moving from dedicated hardware appliances to software solutions running on commercial off-the-shelf (COTS) hardware. In order to enable a successful transition from legacy, hardware-based solutions to virtual network functions (VNFs), multiple challenges remain to be solved. These challenges involve, among others, the placement of functions and function chains in the network, the dynamic allocation of resources to enable automatic scaling, and the performance evaluation of software implementations to replace high performance, specialized hardware appliances. All of these research aspects of the NFV paradigm eventually aim to improve the performance of a VNF deployment with the goal to provide consistent and reliable high performance networking. The IETF is currently discussing the performance evaluation of network functions as Benchmark-as-a-Service [1]. The authors of [2] discuss the integration of profiling into the NFV development cycle and analyze the components needed to generate such performance characteristics within the NFV DevOps workflow.

In this work, we propose a novel methodology to investigate packet processing times of network functions in either dedicated testing environments or live deployments. Current methodologies mostly consist of offline solutions like classical performance benchmarking. Such approaches are important tools during network dimensioning and planning. On the other hand, reliable online monitoring solutions are required in order to exploit NFV flexibility aspects most efficiently.

Reliable and accurate measurement of network function specific key performance indicators, like packet processing times, is, however, not only relevant in live environments and for monitoring purposes. Instead, the processing time can be used as an input parameter for theoretical models ahead of deployment when it comes to predicting performance under certain circumstances [3]. Additionally, the softwarization of networks is often accompanied by the application of software development paradigms during network function development. Especially in the area of continuous integration and continuous development, the availability of fast, reliable, and automatable mechanisms to obtain comparable performance metrics of a new version of a network application is required [4].

To this end, we propose KOMon, a lightweight Kernel-based online monitoring tool for measuring VNF packet processing times. After introducing the architecture of KOMon, we demonstrate its accuracy for different magnitudes of VNF processing times and different load levels. To this end, we use an industry-grade hardware traffic generator and compare the results to baseline measurements reported by the custom designed example network function used during the evaluation. The goal of the measurement is to analyze the accuracy of the proposed KOMon tool and investigate the time offset emerging from KOMon when comparing monitored values to baseline values reported by a known VNF. In particular, it is of interest to determine the influence factors on a potential accuracy bias. As a key contribution we show that the offset is independent of the VNF processing time in terms of an added artificial delay. Further, KOMon allows to measure the health of a VNF since queuing time for packets in the Kernel are included and heavy load situations lead to strongly increasing processing times, which might not be captured by measurements performed within the VNF. Finally, we show that KOMon reported values are able to closely capture the distribution of the processing time exhibited by our example VNF. Both, the KOMon Kernel module as well as the example VNF as well as the dataset used during our evaluation are available on GitHub[1].

The remainder of this work is structured as follows. In Section II an overview of different types of performance evaluation mechanisms and related work from this area is presented. Following, in Section III the architecture and monitoring

[1] https://github.com/lsinfo3/KOMon

methodology of the KOMon tool as well as the functionality of the provided example VNF is provided. Section IV describes our evaluation of the monitoring capabilities of KOMon. Finally, in Section V a discussion of applications and limitations of the proposed approach is presented before the work is concluded in Section VI.

II. Background and Related Work

As already mentioned earlier, most work in the area of VNF performance evaluation and measurements focuses on offline benchmarking and prediction of network function performance based on measurements performed in known environments. Additionally, most of these approaches rely on dedicated test beds to investigate functional and performance characteristics of virtual network functions due to large overheads and complicated test environments.

In [5], the authors propose an offline solution to gather functional as well as performance data through static code analysis without execution of the VNF itself. This approach allows the evaluation of arbitrary workload characteristics in an offline manner. To make use of this methodology, however, the codebase of a VNF needs to be accessible, making it impossible to perform blackbox testing or to evaluate closed source network functions.

The authors of [6] propose another tool for offline performance benchmarking of virtual network functions. The Gym framework enables fully automated performance benchmarking of virtually any VNF type by allowing the user to define custom test cases suitable for the VNF that is to be tested. This and the support for different underlying virtualization platforms make this approach very flexible, while still limiting it to offline, dedicated performance benchmarking.

The aforementioned approaches focus on resource utilization, like CPU time or memory usage, in order to determine VNF performance levels. However, research has shown that this information might not always be sufficient to reliably identify performance bottlenecks [9]. To alleviate this issue, the approach proposed in [7] suggests to monitor VM-to-VM communication to enable online performance monitoring of network functions. However, port mirroring of all incoming and outgoing packets is required by this solution, which imposes a significant performance overhead in a field where processing times and efficiency are crucial.

Another approach proposed by the SONATA-NFV project [8] involves using information exposed by the Linux Kernel to evaluate the performance of virtual network functions. However, the information exposed by the Kernel does not include NFV-specific metrics, like the processing time distribution of packets, but only provides generic information like interrupt counters, buffer levels and number of dropped packets. In this context, our methodology of using the */proc/* file system to extend the information provided by the Kernel can be used to increase the monitoring capabilities of the SONATA framework. Therefore, the integration of monitoring data obtained by our KOMon tool into the monitoring infrastructure of SONATA, thereby extending it by an additional VNF-specific metric, is possible.

III. The KOMon Approach

The monitoring mechanism we propose is based around the idea of *in-stack monitoring*. This novel approach eliminates the need for port mirroring and allows online monitoring with minimal overhead by hooking into the network stack implementation, which needs to be traversed by every packet destined for a hosted VNF. In the following, the architecture, monitoring logic and current implementation of KOMon as well as the example VNF are discussed. The code for both of the components can be found on GitHub[2]. In the presented use case, we use the network stack of the Linux Kernel v4.11. The KOMon approach can, however, be applied to other stack implementations such as DPDK [10] or Snabb [11].

A. KOMon Architecture

The KOMon architecture consists of two components. First, the KOMon Kernel module hooks into the network stack by injecting its monitoring code. Second, the KOMon controller runs in user space and is responsible for configuration and management tasks. Figure 1 shows this architecture and its interaction with the network stack. For reference, the right hand side of the figure shows an abstracted view of the NAPI stack, according to the ISO/OSI model.

Initialization (1+2). Before starting the monitoring procedure, the module needs to be loaded into the Kernel, thus injecting the monitoring code directly into the network stack, thereby attaching its monitoring logic to existing Kernel functions of the network stack. In this use case, we attach two monitoring probes to the network stack responsible for handling UDP traffic. In order to obtain timestamps for incoming datagrams, we inject the monitoring point into the `__udp4_lib_rcv` function that is used to process UDP datagrams before sending them to a user space application. On the other hand, in order to gather timestamps of outgoing packets, we inject a second measurement point into the `udp_sendmsg` function that is used by a UDP socket to send datagrams. After the injection process, the user space controller configures sample size and monitoring interval and thereby defines how often and how many packets are being sampled by the Kernel module. Following, the monitoring sequence consists of three main steps.

Monitoring Loop (2+3+4). The current version of the KOMon Kernel module is entirely passive until it receives an initial trigger issued by the user space controller. After the controller triggers a monitoring interval, the Kernel module is activated, samples the previously configured number of consecutive packets, and calculates their processing times. Therefor, at the first measurement point the timestamps of incoming packets destined for the VNF to be monitored are stored in a ring buffer together with a hash of the packet payload. After the packets have been processed by the VNF

[2]https://github.com/lsinfo3/KOMon

TABLE I: Overview of related work in the field of VNF monitoring approaches.

Approach	Online	VNF agnostic	Remark
SymPerf [5]	Offline	Code required	Arbitrary workload
Gym Framework [6]	Offline	Yes	Fully automated, user defined test cases
NFVPerf [7]	Online	Yes	Port mirroring required, imposing performance overhead
SONATA [8]	Online	Yes	Uses limited information exposed by Linux Kernel, may utilize KOMon reported data
KOMon	Online	Yes	Allows monitoring of VNF health and characterization of processing time distribution

and are ready to be sent out, the second measurement point monitors outgoing packets and compares payload hashes to the oldest packet still in the ring buffer. If the hashes match, the timestamps are simply subtracted and the processing time is stored as a result. After the preconfigured number of packets have been monitored, the Kernel module returns to its passive mode in order to decrease overhead. The data obtained during the active monitoring phase can then be queried by the user space controller via a *procfs* interface that returns a list of measured processing times. This mechanic of matching packets currently limits the functionality of the KOMon approach to VNFs that process packets in a FIFO manner and do not alter the payload of packets. The implications and challenges of payload altering and prioritizing network functions are discussed in Section V.

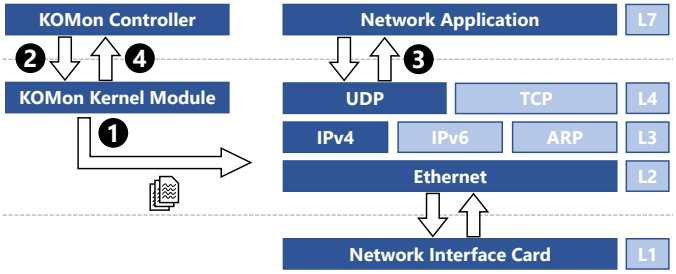

Fig. 1: Architecture and stack interaction.

Note that the highlighted nodes in Figure 1 correspond to monitoring UDP over IPv4 traffic. The Kernel module can, however, be attached to any protocol supported by the network stack with only slight modifications.

B. Example VNF Functionality

The example VNF used during the evaluation of KOMon is essentially a UDP mirror written in C that is able to relay packets to a predifined destination after inducing an arbitrary artificial delay. The delay can thereby be configured to be constant or follow a predefined distribution. The VNF currently supports negative exponential, uniform and normal distributed artificial delays, but can easily be adapted to support further processing time characteristics. Additionally, the VNF supports sampling of its own processing time, which is used in our evaluation scenario as a baseline to compare to the values reported by KOMon. Thereby, the VNF stops the time between receiving a packet after reading from the socket and sending it back out. The time it takes the Kernel to pick up the packet from the socket is thereby not included in the processing time.

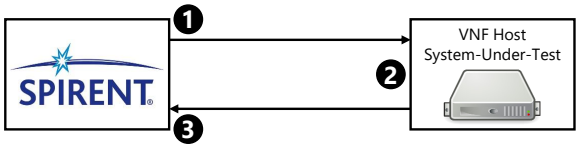

Fig. 2: Testbed setup used during the evaluation.

IV. Evaluation

In this section we present the measurement data obtained using the KOMon monitoring tool in combination with an example network function. After outlining the testbed setup, we illustrate the calibration process whose results are used as an input for identifying characteristics of VNFs' processing time distributions.

A. Testbed Setup

The results have been generated in a dedicated testbed consisting of a Spirent C1 hardware traffic generator that is directly connected to a bare metal server equipped with an Intel Xeon E5420, 16 GB RAM, and 2x 1 Gbit NICs that is running Ubuntu 16.04.3 LTS and our modified version of Kernel 4.11. The measurements have been performed using the example VNF included in the project repository on GitHub. Figure 2 shows the schematic structure of the testbed used during the evaluation.

Thereby, the Spirent Testcenter C1 (1) serves as a traffic generator as well as the traffic sink (3). It generates a continuous stream of tagged UDP datagrams that can be uniquely identified via their payload and sends them to the system-under-test hosting the network function (2) over a direct 1G copper connection. The network function is a simple UDP relay as described in Section III-B. This test setup is used throughout the evaluation presented in this work. In the following, we evaluate the accuracy of the KOMon monitoring tool in different scenarios and evaluate the influence of various parameters on its performance. The data used in this evaluation is provided in the public GitHub repository.

B. Calibration

The first performance metric of the KOMon monitoring tool is presented in Figure 3. The figure shows the processing time along the y-axis. The x-axis shows an increasing artificial delay as it is added by the used network function. Thereby,

values in red represent the data measured using the KOMon monitoring tool, while the blue data points show the baseline data reported by the example VNF. The black markers in this plot mark outliers whose distance to the mean is either smaller than $Q1 - 1.5 \cdot \text{IQR}$ or larger than $Q3 + 1.5 \cdot \text{IQR}$, where IQR is the inter quartile range, Q1 is the 25% quantile and Q3 is the 75% quantile. The data presented in this figure has been obtained by generating a continuous stream of 1 000 UDP datagrams per second of size 128 byte that are processed by the VNF and consequently sent back to the traffic generator. In the process, the VNF induces a predefined artificial delay before sending back packets. In this scenario, both the VNF as well as KOMon sample 10 packets in negative exponentially distributed intervals with a mean of 0.2 seconds.

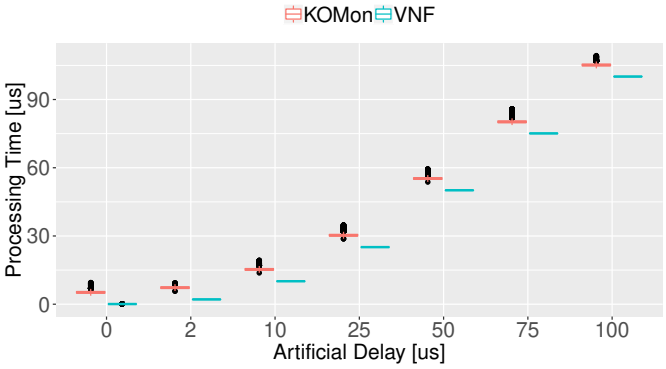

Fig. 3: Comparison of VNF reported values and values observed using KOMon for different artificial delays.

It can be seen that values monitored using KOMon exhibit a roughly constant offset over the baseline values reported by the VNF for all artificial delay levels. The differences in outliers are explained by the fact that both the VNF as well as KOMon obtain the values through packet sampling and it cannot be guaranteed that both tools sample the exact same packets. The general presence of these outliers is explained by the general purpose operating system used to host the VNF that performs task scheduling that may affect a time measurement in the microsecond realm.

As already explained in Section III, the measurement points created by KOMon are located in Kernel space. Thus, the processing time observed by the monitoring tool includes an offset formed by operations happening between the measurement points and the processing performed by the VNF, e.g. copying packet data from Kernel space to user space. This explains the slightly higher variance exhibited by KOMon reported values over the baseline data. In order to quantify this offset as well as validate the observation of the measurement accuracy being independent from the total processing time of the VNF, Figure 4 shows the bootstrapped difference of means for different levels of artificial delay. Thereby, 1 000 random samples, each consisting of 10 subsequent packets, have been taken from both the VNF-reported as well as the KOMon generated data set, respectively. In order to soften the impact of scheduling on a general purpose operating

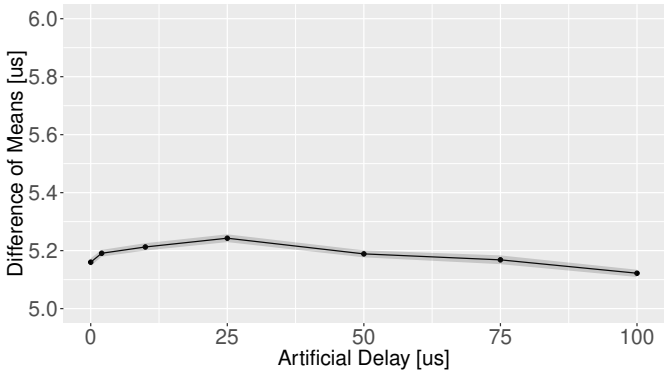

Fig. 4: Bootstrapped difference of means for different artificial delays.

system, the following figures only contain the 99% quantile of the dataset. For each of the pairs of samples, the difference of means has been calculated before finally determining the 95% confidence interval for the resulting distribution of the difference of means.

Figure 4 shows the mean difference of means after 1 000 repetitions along the y-axis with the opaque area depicting the 95% confidence interval. The x-axis shows again the different levels of artificial delay. The first observation made in this figure is the fact that KOMon reported values are within a range of 0.2 us. Considering the fact that these values have been obtained by the means of software based measurements, this difference falls well within the expected accuracy of our methodology. Additionally, none of the 95% confidence intervals exceeds a width of 0.03 us, further supporting the fact that KOMon exhibits a constant offset over the baseline values. Based on this values, we consider 5.2 us to be the overhead of KOMon reported values over the baseline.

The second performance metric of the KOMon tool we investigate is the influence of different load levels on its monitoring capabilities. Therefor, Figure 5 shows the mean as well as 95% quantiles of processing time values monitored using KOMon as well as reported by the VNF. Thereby, the x-axis shows the different load levels in packets per second. The y-axis shows the observed processing time in microseconds. Values obtained by KOMon are again reported in red while the blue values show the values reported by the VNF. In addition, the dotted lines show the mean value while the solid curve depicts the 95% quantile. Similar to the scenario investigated before, the traffic generator produces a continuous stream of UDP datagrams of size 128 bytes. Both the VNF as well as KOMon sample batches of 10 consecutive packets in negative exponentially distributed intervals with a mean of 0.2 seconds. Instead of adding an artificial delay, the VNF is configured to flood out packets as fast as possible this time.

Two observations can be made in this figure. On the one hand, it can be seen that the mean processing time reported by the VNF is nearly constant for all evaluated load levels between 1 000 and 175 000 packets per second with 150 000

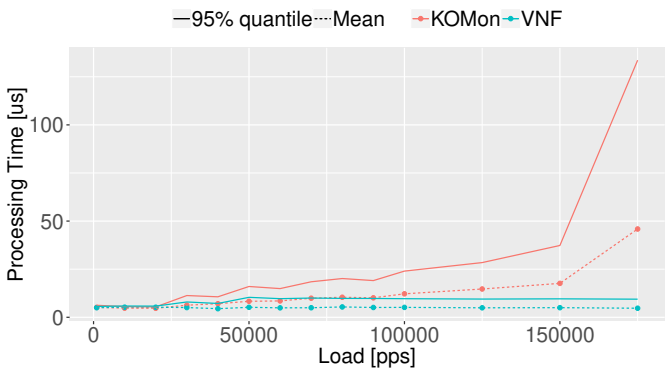

Fig. 5: Comparison of VNF reported values and values observed using KOMon for different load levels.

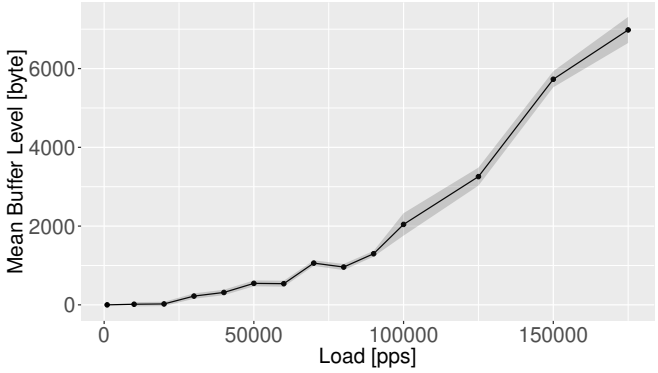

Fig. 6: Socket buffer fill levels for different load levels.

packets per second being the non-drop rate for the test setup as any higher load leads to significant packet loss and is therefore not included in this evaluation.

The mean of the VNF reported values is mostly constant for all load levels. This observation is quite intuitive as the VNF is processing packets as fast as possible for all load levels and, since no packet loss occurs even at 150 000 packets per second, it can even do so at high loads. The 95% quantile is after a slight increase at around 30 000 packets per second also mostly constant. On the other hand, the mean values obtained by KOMon show a continuous growth with the 95% quantile even exhibiting exponential growth behavior. This difference in observed processing times is once again attributed to the operations taking place between the VNF and the monitoring points used by KOMon. Packets processed by the Linux Kernel network stack are in general queued two times on their way from the network interface card (NIC) to a user space application. Once at the NIC itself in a process called interrupt mitigation [12] that aims to decrease the overhead of sending an interrupt to the Kernel for every incoming packet, thus avoiding livelocks. Then, after having traversed most of the network stack, packets are queued a second time in the socket buffer of the socket opened by a user space application. This second buffer is located between the KOMon measurement points and the VNF and is filled to a different extent for different load scenarios. Figure 6 shows the mean buffer fill levels and 95% confidence intervals recorded during the load test presented in Figure 5. The values have been obtained from `/proc/net/udp`.

The figure shows the different load levels along the x-axis and the mean buffer fill level along the y-axis. The opaque area depicts the 95% confidence interval. It can be seen that the buffer fill level develops similarly to the 95% quantile of the monitored values in Figure 5, thereby exhibiting a correlation of 0.85.

The combination of the information presented in Figures 5 and 6 shows that KOMon is able to take the queuing of packets into account that not even the VNF itself can report and can thus be used to trigger measures like scale-up or scale-out ahead of time, thereby avoiding packet loss. This shows that KOMon can be used in live scenarios to monitor the current performance of network functions and is able to detect performance bottlenecks.

Additionally, as has been observed in Figures 3 and 4, the monitoring values can be used to determine the packet processing time of a network function. The values obtained during the calibration show that the offset is statistically independent from the total processing time of the VNF and can be used to infer its real processing time.

C. Estimation of Processing Time Distributions

We demonstrate the applicability of the proposed approach by using our KOMon tool to determine the processing time distribution of VNFs. To this end, we configure our example VNF in such a way that its processing time follows one of four distributions with configurable mean processing times μ. These distributions include a negative exponential distribution, a uniform distribution that ranges from 0 to 2μ, as well as two normal distributions whose coefficients of variation are equal to 1 and 0.2, respectively. Furthermore, we vary the mean processing time of the VNF from 2 to 100 microseconds.

We first provide qualitative results of the evaluation in Figure 7. Each subfigure corresponds to one of the four processing time distributions and displays the empirical cumulative distribution function (ECDF) of the values that are reported by the KOMon tool and the VNF, respectively. While the data source is represented by the line type, differently colored curves denote different mean processing times μ as listed in the annotation in the first subfigure.

Two main observations can be made. First, the shape of the distributions that are obtained by means of the KOMon tool as well as the VNF are very similar w.r.t. to both their shape and values in all considered scenarios. This phenomenon indicates that KOMon is capable of capturing general properties of the VNF processing time, in particular characteristics beyond the first moment. Secondly, an almost constant offset between the curves that correspond to KOMon and the VNF can be identified. This offset can be explained by the fact that the values which are reported by KOMon include the overhead that is caused by operations that are performed in the Kernel.

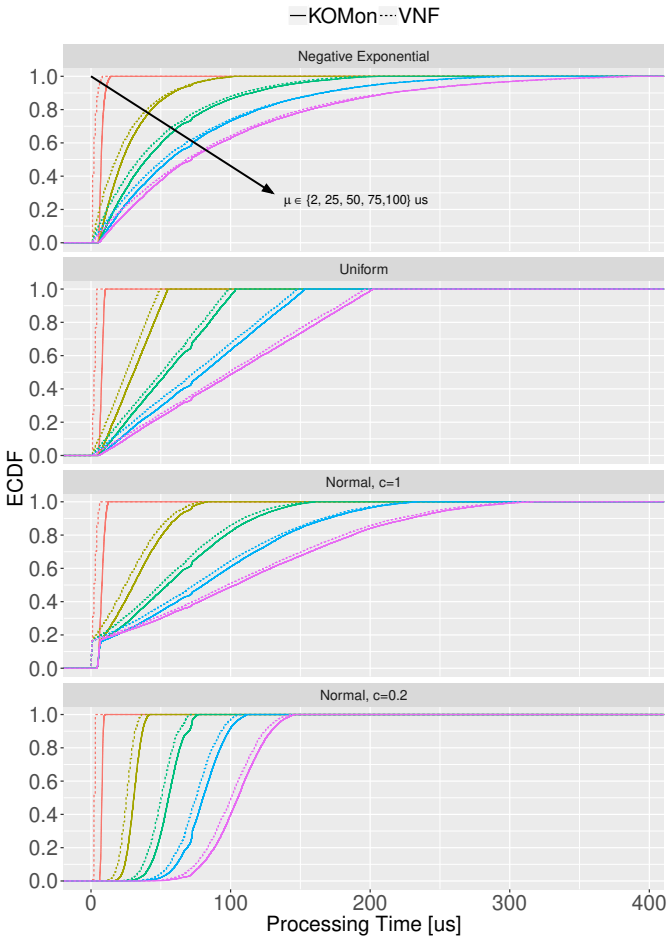

Fig. 7: Empirical CDFs for different delays and different processing time distributions as sampled by the VNF as well as KOMon.

By using the difference of means that is obtained during the calibration phase in conjunction with the KOMon-based values, we can correct this shift and estimate the mean processing time of the VNF as well as its coefficient of variation.

Results of this estimation are presented in Table II. We show the mean values as well as the coefficient of variation of processing times that are reported by the VNF itself as well as the KOMon-based estimates. The latter are obtained by subtracting the calibration offset from KOMon-reported values. In the presented case, this offset is equal to 5.2 microseconds.

As evidenced by almost identical values in the context of all processing time distributions and average processing times, mean values can be estimated in a reliable manner. While this is also true for most scenarios in the case of the coefficient of variation, significant deviations are observed for $\mu = 2$ microseconds. This behavior can be explained by the fact that in these scenarios, the offset is larger than the mean processing time and the small values are close to the resolution of software-based measurements.

In summary, we have demonstrated that KOMon can be used to reliably reproduce the shape and estimate the mean as well as the coefficient of variation of a wide range of VNF processing time distributions. In particular, this estimation can be performed *without* VNF-reported values or access to the VNF code. It only requires a system-specific constant which can be obtained by means of a single calibration run that needs to be performed once prior to deployment.

TABLE II: Mean and coefficient of variation for different artificial delays and different processing time distributions.

Distribution	Delay μ	Mean VNF	Mean KOMon	Coeff. of Variation VNF	Coeff. of Variation KOMon
Negative Exponential	2	2.75	2.68	1.43	1.75
	50	50.91	50.59	0.99	0.99
	100	100.53	101.37	0.99	1.00
Uniform	2	2.73	2.61	1.39	0.87
	50	50.77	51.05	0.59	0.59
	100	100.69	101.02	0.58	0.58
Normal $c_v = 1$	2	2.89	2.79	1.59	1.49
	50	55.17	55.01	0.80	0.81
	100	109.74	109.57	0.80	0.80
Normal $c_v = 0.2$	2	2.73	2.62	1.22	0.71
	50	51.05	50.84	0.25	0.25
	100	100.88	101.05	0.22	0.22

V. Discussion and Guidelines

In this section, we briefly discuss the packet matching problem and its impact on the proposed methodology and provide an outline on how it can be applied to different use cases.

The Packet Matching Problem. As described in Section III, KOMon uses the hash value obtained from the packet payload for packet identification. This limits the functionality of the methodology in its current state to network functions that do not alter the payload in any way. In addition, in order to decrease the overhead induced by the monitoring logic as far as possible, we currently compare outgoing packets only to the oldest packet still in the incoming ring buffer, thus eliminating the possibility of per flow prioritization. The second issue of supporting non-FIFO network functions could be solved by comparing all packets currently in the system whenever a packet is sent out. This would lead to a slight increment in monitoring overhead while at the same time enabling more functionality. The initial problem, however, still remains. Network functions that alter payloads, drop, aggregate, or split up packets can, currently not be monitored. This problem could be worked around by providing a VNF policy description ahead of time. The policy description can then be used to predict how the VNF is going to behave and KOMon can monitor for the expected result. This functionality, however, strongly depends on the type and functionality of the network function and is thus not part of the generic methodology proposed in this work.

Application in Performance Modeling. One of the outcomes of the evaluation performed in Section IV is that, for

low loads, the reported monitoring values are very close to the real processing time of the VNF. In addition, we have shown that the offset included in the measurement is statistically independent from the magnitude of the processing times of the network function. This allows for KOMon to be used to determine the distribution of processing times as it is needed for theoretical performance models such as [3]. In addition, this offset can be eliminated from the values as it is possible to calibrate the system by performing measurements involving a network function of which the processing time is known. Hence, the offset, that depends on the hardware or virtual environment, can be calculated and taken into account by comparing the reported monitoring values to baseline measurements reported by a known network function.

Application during Network Function Development. Similar to the application of the approach in theoretical models, it can be applied during the development phase of a network function. Especially in the realm of continuous integration and continuous delivery (CI/CD) [13], automatable evaluations of application specific performance characteristics are crucial. To this end, KOMon can be seamlessly integrated into a build and evaluation pipeline to perform automated performance evaluations of new versions of an application before its deployment, thereby ensuring no performance degradation, e.g., through added features. How the CI/CD paradigm can be applied to the networking realm is discussed in [4].

Application in Network Function Monitoring. Finally, the KOMon tool can be applied to monitor the performance of network functions in live environments by continuously gathering and evaluating samples. This allows for policing of the remaining resources of a network function instance. Upon reaching a certain threshold, scaling mechanisms can be triggered to provide additional resources before packet loss or processing time explosion can occur. As was observed in Figure 5, the 95% quantile could serve as a suitable indicator in most scenarios. In some cases other metrics extracted from the processing time observations, e.g. entropy, might provide better results.

Guidelines for using KOMon

Repository: https://github.com/lsinfo3/KOMon. The following guidelines briefly describe how KOMon can be used for different use cases. Installation, configuration and general usage instructions can be found in the repository.

Characterization of processing time distribution

1. Use VNF with fix, known processing time
2. Use KOMon to measure reported values
3. Compute overhead included in KOMon values as the average difference of measurements

Continuous monitoring of VNF health

1. Perform load tests in controlled environment
2. Calibrate evaluation by continuously increasing load until VNF is overloaded, using KOMon to measure reported processing time values
3. Select metric to monitor as indicator for impending performance bottleneck (e.g. 95% quantile, entropy, maximum, ...)
4. Define threshold depending on specific use case and previously selected metric
5. Monitor selected metric using KOMon in live environment with unknown load profile

VI. Conclusion

Reliable, low overhead online monitoring of VNF performance and accurate characterization of application specific performance metrics are key to leveraging the benefits of NFV in terms of flexibility and scalability. To this end, we propose KOMon, a lightweight mechanism for accurate Kernel-based online monitoring of VNF packet processing times. KOMon allows the accurate measurement of processing times when used in dedicated environments while enabling accurate, low overhead online monitoring in live deployments. In order to substantiate these claims, experiments with an industry-grade hardware-based traffic generator have been performed. Our evaluation has shown that the values reported by KOMon in controlled environments are valid and close to baseline values, independent of the magnitude of processing times. In addition, we have shown that KOMon can be used to monitor the health of a network function instance as the reported values include the queuing time for packets before being processed by the network function. Hence, performance bottlenecks can be detected ahead of time and corresponding actions can be triggered before severe performance degradation occurs. Furthermore, our experiments demonstrate that KOMon can be used to reliably estimate key characteristics of processing time distributions, such as mean and coefficient of variation.

Finally, the packet matching problem and current limitations of the methodology have been discussed and a brief outline of possible applications for the proposed approach has been presented.

Future work in this area includes the introduction of VNF policy descriptions to counter the packet matching problem and the investigation of sampling of heavy-tailed processing time functions with arbitrary workload characteristics. Sampling in the case of heavy-tailed processing time distributions proves to be difficult due to the rarity of events with high impact. In order to overcome limitations regarding performance and accuracy, more sophisticated measurement algorithms and data structures need to be revisited [14].

Acknowledgment

This work has been performed in the framework of the CELTIC EUREKA project SENDATE-PLANETS, and it is partly funded by the German BMBF. The authors alone are responsible for the content of the paper.

REFERENCES

[1] R. Rosa, C. Rothenberg, and R. Szabo, "Vnf benchmark-as-a-service," IRTF NFVRG: Internet draft, Tech. Rep., 2015.

[2] M. Peuster and H. Karl, "Understand your chains: Towards performance profile-based network service management," in *European Workshop on Software-Defined Networks (EWSDN)*, 2016.

[3] T. Zinner, S. Geissler, S. Lange, S. Gebert, M. Seufert, and P. Tran-Gia, "A discrete-time model for optimizing the processing time of virtualized network functions," *Computer Networks*, 2017.

[4] S. Gebert, C. Schwartz, T. Zinner, and P. Tran-Gia, "Continuously delivering your network," in *International Symposium on Integrated Network Management (IM)*, 2015.

[5] F. Rath, J. Krude, J. Rüth, D. Schemmel, O. Hohlfeld, J. A. Bitsch, and K. Wehrle, "SymPerf: Predicting Network Function Performance," in *Proceedings of the SIGCOMM Posters and Demos*, 2017.

[6] R. V. Rosa, C. Bertoldo, and C. E. Rothenberg, "Take Your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking," *IEEE Communications Magazine*, 2017.

[7] P. Naik, D. K. Shaw, and M. Vutukuru, "NFVPerf: Online performance monitoring and bottleneck detection for NFV," in *IEEE Conference on NFV and SDN*, 2016.

[8] "SONATA-NFV Project," https://github.com/sonata-nfv/son-monitor-probe/wiki/VNF-monitoring, accessed: 2019-04-24.

[9] A. S. Rajan, S. Gobriel, C. Maciocco, K. B. Ramia, S. Kapury, A. Singhy, J. Ermanz, V. Gopalakrishnanz, and R. Janaz, "Understanding the bottlenecks in virtualizing cellular core network functions," in *IEEE International Workshop on Local and Metropolitan Area Networks*, 2015.

[10] "Data Plane Development Kit," http://dpdk.org/, accessed: 2019-04-26.

[11] "SnabbCo, "Snabb: Simple and fast packet networking," https://github.com/snabbco/snabb, accessed: 2019-04-26.

[12] K. Salah and A. Qahtan, "Implementation and experimental performance evaluation of a hybrid interrupt-handling scheme," *Computer Communications*, 2009.

[13] M. Fowler and M. Foemmel, "Continuous integration," *Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf*, 2006.

[14] O. Alipourfard, M. Moshref, Y. Zhou, T. Yang, and M. Yu, "A comparison of performance and accuracy of measurement algorithms in software," in *Proceedings of the Symposium on SDN Research*. ACM, 2018.