Demonstrating the Cost of Collecting In-Network Measurements for High-Speed VNFs

Leonardo Linguaglossa[*], Fabien Geyer[†‡], Wenqin Shao[§], Frank Brockners[¶], Georg Carle[†]
[*] *Telecom ParisTech – Paris, France – first.last@telecom-paristech.fr*
[†] *Technical University of Munich – Munich, Germany – last@net.in.tum.de*
[‡] *Airbus – Munich, Germany – first.last@airbus.com*
[§] *Cisco Systems Inc. – Paris, France – first.last@cisco.com*
[¶] *Cisco Systems Inc. – Cologne, Germany – fbrockne@cisco.com*

***Abstract*—Recent advances in the state-of-the-art of software packet processing along with the incarnation of SDN and NFV in networking brings the utility of software switches in production to a high level. Accompanied with the wide deployment of the latter, comes the practical and urgent need of monitoring networks that are composed of software forwarders/switches. On the one hand, this may provide new types of very fine-grain operational data that can be collected, thus bringing the opportunity for network managers to get a deeper understanding of the underlying network state and performance. On the other hand, this massive data availability comes at a cost: software measurements can highly affect the measured values, thus biasing the collected data. The intensity of this bias becomes stronger when measurements are taken close to the data path. We believe that this trade-off should be explored more in detail, since the availability of fine-grained data offers new opportunities to apply machine learning techniques to infer changes in the network state, to forecast the evolution of some performance metrics or to automatically respond to event triggers without the human intervention. While our long-run objective[1] is a full framework for performing automated test on software routing platforms, in this demonstration we focus on two key points that are prerequisite for our approach: (i) we showcase the impact of collecting the desired data within a Virtual Network Function and (ii) we setup a simple environment for data visualization on the same physical device.**

***Index Terms*—NFV, High-Speed Software Routers, Performance Evaluation, Measurements**

I. Context

Network telemetry refers to a set of technologies devoted to gathering and transferring information related to a network's state. This typically implies measuring values of some variable, logging, detecting anomalies and so on.

Many enablers for telemetry are commonly adopted, but the choice is still constrained by the protocol used or by the amount of resource needed. The Simple Network Management Protocol [3] can be used to collect a pre-defined set of information useful for network operation. The IPFIX protocol [4] or NetFlow-based solutions [6] are still heavily used, at the cost of trading off the amount of resources devoted to the metering procedure for an extended set of measurable features. Telemetry data is usually collected out of band, but other approaches advocate to make data available directly in-situ, i.e. together with and as part of the live traffic. This can be achieved by adopting novel protocols such as the In-situ Operation Administration and management (IOAM) [2], or by coupling the implementation of both forwarding and telemetry within a software switch [8].

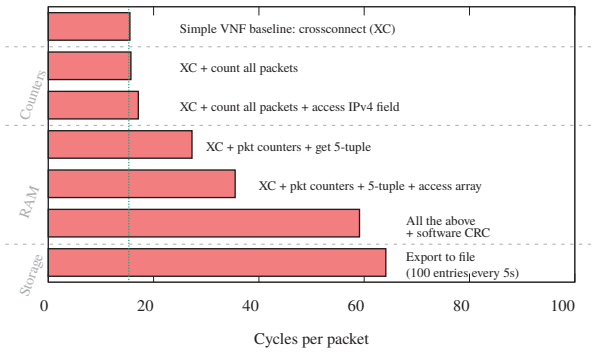

Fig. 1: Sample of network function and the export capabilities

In the software domain, the cost of measuring and making this data available (present also for hardware solutions), is shifted from dedicated/specific hardware to generic computation resources. Several operations have to be performed on a per-packet basis, some of them keep state, and some others export data to external storage. Regular switches and routers typically perform these operations on specific components especially designed for the measurement process. Software switches, instead, implement the measurement process as a piece of code that needs to be executed on the target device.

Recent approaches of high-speed and low-impact flow monitoring [9] [5] open scenarios where all devices can get some measurements with low-impact[2]. This raises a first question: what is the impact of measuring these items? We provide in Fig. 1 an example of a simple VNF that performs basic forwarding (cross-connect, or XC). We observe that the cost, measured in cycles per packet, for the simple VNF increases with the complexity of the measurement. If counting packets does not significantly affect the performance, accessing data which is off-chip (e.g. RAM memory) or exporting to external storage has a significant impact, even when sampling is used. We identify three categories of costs with increasing weight:

[1] http://www.industry-of-the-future.org/ai4perf/demo/

[2] This can also open scenarios where devices spontaneously provides measurements when needed.

978-3-903176-17-1 / © 2019 IFIP

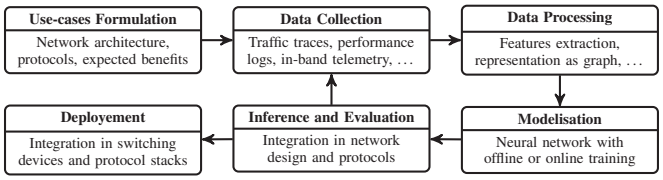

Fig. 2: Block diagram of the proposed framework

a small cost that occurs whenever accessing or de-referencing variables in software; a medium-impact cost due to writing data to the RAM (e.g. saving state to a hash-table); a high-impact cost occurring when writing data to external storage.

This raises a trade-off to be considered while deploying a telemetry application: while out-of-band data collection minimizes the impact of the measurements w.r.t. in-situ solutions, the former requires more resources than the latter. This trade-off applies also to the processing step, which can be implemented *online* or *offline*: online processing of the data allows the application to quickly respond to events such as anomalies, but requires additional complexity and has a higher impact in the computational resources than offline processing, which in turn can rely on a bigger time scale and thus perform more complex processing on a possibly bigger set of data.

II. Methodology

Our goal is to perform smart tests, measure test data efficiently, process the measured data, and finally interpret the observations using machine intelligence, with the goal of minimizing the impact of this additional processing on the forwarding performance of the network. Our approach aims at targeted and reproducible measurements in combination with proper modeling of the complete system for improving performance in the relevant metrics such as latency. The components of our framework are shown in Fig. 2.

We envisage the following use-cases: *what-if analysis* for predicting the impact on performance in case of changes such as increased number of users, deployment of additional or more powerful devices or virtual machines, modification of routing, or update of virtualization technology; *root-cause analysis* and *bottleneck discovery* to find where changes can be made to improve performance in the relevant metrics (e.g. Quality-of-Service / Quality-of-Experience); *detection of performance anomalies* and correlation with misconfigured or misbehaving devices. The Data collection block refers to the activity of gathering the required measurements from the devices under test. Existing software routers have already some built-in functions to export clock cycles spent on specific functions, which are being used to do performance evaluation [7]. After the data is collected, we can perform iteratively processing, apply machine learning algorithms and obtain the results for our desired use-case. This is represented in the loop of Data collection, processing, modelisation and inference in Fig. 2. As a final step, we aim to provide an approach that can be easily deployed in network devices.

III. Testbed setup and experimental evaluation

For this demonstration we propose to showcase the costs and the benefits of software telemetry solutions with in-situ and out-of-band measurements in combination with online and offline processing on the measurements. Additional details about the demo are shown in appendix A to C. Our hardware setup consists in a COTS server with 2× Intel Xeon Processor E5-2690, each with 12 physical cores running at 2.60 GHz. The server is equipped with 2× Intel X520 dual-port 10 Gbps NICs. The hardware is remotely accessed for both bare-metal and container experiments (a ssh connection is required). We use VPP as state-of-the-art software router [1]. The code is modified to retrieve network measurements (cfr. Fig. 1) and export the data to external storage.

The demonstration focuses on a virtual network function to be monitored. We show (i) the cost of running a telemetry solution and (ii) some possible usage of telemetry data. We first execute the router with a simple function that receives and forward some L2 traffic. We then progressively increase the complexity of the monitoring step, using both in-situ and out-of-band telemetry. We finally collect the data and interactively visualize the collected measurements. For the sake of comparison, we also show the process of getting the same measurements on a hardware switch connected to an interactive dashboard. This scenario can be reproduced for a number of more complex network functions, including IPv4/IPv6 routing and mixed L2-L3 scenarios. We then demonstrate how to use the collected data to adapt the frequency of the CPU running the VNF to better cope with the incoming input rate. We show how online and offline solutions (based on decision trees) deal with the detection of the current behavior of the network function. The demonstration will be providing data visualization and interactivity.

Acknowledgments

This work is funded by the German-French academy for the industry of the future, a collaboration between Telecom ParisTech (France) and Technical University of Munich (Germany). This work also benefited from support of NewNet@Paris, Cisco's Chair "NETWORKS FOR THE FUTURE" at Telecom ParisTech (http://newnet.telecom-paristech.fr).

References

[1] Vector Packet Processing. https://fd.io/.
[2] F. Brockners, S. Bhandari, et al. Data Fields for In-situ OAM. Draft at https://tools.ietf.org/html/draft-ietf-ippm-ioam-data-05.
[3] J. Case, M. Fedor, and other. A Simple Network Management Protocol (SNMP). RFC 1157 at https://tools.ietf.org/html/rfc1157, 1990.
[4] B. Claise, B. Trammell, and P. Aitken. RFC 7011: Specification of the IPFIX protocol for the exchange of flow information, 2013.
[5] P. Emmerich, M. Pudelko, Q. Scheitle, and G. Carle. Efficient Dynamic Flow Tracking for Packet Analyzers. In *CloudNet*, Oct. 2018.
[6] Y. Li, R. Miao, C. Kim, and M. Yu. FlowRadar: A Better NetFlow for Data Centers. In *NSDI*, pages 311–324, 2016.
[7] L. Linguaglossa, D. Rossi, et al. High-speed data plane and network functions virtualization by vectorizing packet processing. *Computer Networks*, 149, 2019.
[8] P. Tammana, R. Agarwal, and M. Lee. Distributed Network Monitoring and Debugging with SwitchPointer. In *USENIX NSDI*, 2018.
[9] T. Zhang, L. Linguaglossa, et al. FlowMon-DPDK: Parsimonious Per-Flow Software Monitoring at Line Rate. In *IFIP TMA*, June 2018.