OpenVolcano: An Open-Source Software Platform for Fog Computing

R. Bruschi[*], P.Lago[*‡], G. Lamanna[§], C. Lombardo[*‡], S. Mangialardi[*]
[‡] DITEN - University of Genoa - Genoa, Italy
[*] CNIT - University of Genoa Research Unit - Genoa, Italy
[§] INFOCOM S.R.L. - Genoa, Italy

***Abstract*—In order to overcome the cloud service performance limits, the INPUT Project aims to go beyond the typical IaaS-based service models by moving computing and storage capabilities from the datacenters to the edge network, and consequently moving cloud services closer to the end users. This approach, which is compatible with the concept of fog computing, will exploit Network Functions Virtualization (NFV) and Software Defined Networking (SDN) to support personal cloud services in a more scalable and sustainable way and with innovative added-value capabilities. This paper presents OpenVolcano, the open-source software platform under development in the INPUT Project, which will realize the fog computing paradigm by exploiting in-network programmability capabilities for off-loading, virtualization and monitoring.**

Keywords-fog computing; SDN; NFV; personal cloud services.

I. INTRODUCTION

The present-day Internet is characterized by a number of devices and communication media that certainly were not envisaged in the original network design. In particular, the advent of the Internet of Things (IoT) will further extend the plethora of available connected objects, bringing their number, according to Ericsson, up to 50 billion in 2020 [1]. Since such devices are characterized by limited storage and computational capabilities, their diffusion has led to the development of a significant number of cloud services to provide the support needed by smart devices. As a consequence, new networking and IT architectures are required that also take into account a more energy/cost efficient and environmental friendly approach.

The joint adoption of the cloud computing supported by Network Functions Virtualization (NFV) [2] and Software Defined Networking (SDN) [3] architectures is considered a promising solution to promote flexibility and foster the introduction of new services otherwise unfeasible. This solution represents the foundation to the Infrastructure-as-a-Service (IaaS), a realization of the cloud computing paradigm that provides virtualized computing resources over the Internet. However, the proper deployment of similar paradigms is limited by the performance level currently offered by datacenters, which are located near the core networks and present end-to-end latencies around two orders of magnitude higher than the required values.

To address this issue, fog computing has been proposed [4] by CISCO with the goal of deploying storage, computing and configuration features closer to the end-user instead of inside datacenters. This deployment scheme has three main advantages: *i*) the reduced distance of the user from his/her services reduces the end-to-end latency, hence improving overall Quality of Services and Experience (QoS and QoE); *ii*) the deployment of services inside the edge network allows for better control on the allocation of physical and logical resources, with the possibility of applying economies of scale and power management schemes, and improved privacy; *iii*) additionally, the telecom providers can hide the complexity of the underlying network infrastructure, and hence promote simplified applications design.

In this respect, the INPUT Project [5] is devoted to foster Future Internet infrastructures beyond the typical IaaS-based service models by moving computing and storage capabilities, from both cloud services and user devices, to the edge network. This approach will exploit the ability to directly access network primitives, and will improve scalability in the interactions of the network with users and datacenters.

In order to provide scalable and virtualized networking technologies able to natively integrate cloud services, both personal and federated, the INPUT Project is realizing OpenVolcano, an open-source software platform for fog computing, which will exploit in-network programmability capabilities for off-loading, virtualization and monitoring.

The paper is organized as follows. Section II focuses on the deployment of the personal cloud services. Section III reports the rationale behind the OpenVolcano architecture, while its control and data plane building blocks are described in Sections IV and V, respectively. Section VI provides further details and performance evaluation on *quake*, the internal virtual switch. Finally, conclusions are drawn in Section VII.

II. DEPLOYMENT OF PERSONAL CLOUD SERVICES

The INPUT technologies will enable next-generation cloud applications to go beyond classical service models and even to replace physical Smart Devices (SD), usually placed in users' homes (e.g., network-attached storage servers, set-top-boxes, video recorders, home automation control units, etc.) or deployed around for monitoring purposes (e.g., sensors), with their "virtual images," providing them to users "as a Service". A virtual image is defined to be a software instance that dematerializes a physical network-connected device, and that provides its virtual presence in the network and all its functionalities. Virtual images are meant to realize smarter, always and everywhere accessible, performance-unlimited virtual devices into the cloud. SDs can be fully or partially virtualized depending on their constraints on physical components: sensors cannot be entirely virtualized, since they still need the hardware to acquire measures and to transfer them elsewhere. On the contrary, entertainment appliances can be fully virtualized, and interfaced with visualization devices (e.g., a television or a tablet) through standard and well-known protocols. A breakdown of the

carbon footprint reduction in presence of partial, full or not virtualized SDs can be found in [6].

These SDs will be made available to users at any time and at any place by means of virtual cloud-powered Personal Networks, which will provide users with the perception of always being in their home Local Area Network with their own (virtual and physical) SDs, independently from their location.

A. The Personal Network

A Personal Network (PN) is a secure and trusted virtual overlay network composed of standard L2 protocols and operations equivalent to the ones presently available in the user's home network, independent of their location. The correct routing of the L2 data and signaling packets is guaranteed by the OpenFlow's matching/action rules.

PNs are realized by virtualizing typical Network Functions provided by the user's home gateway, and transferring these Virtual Network Functions (VNFs) into software instances running in commodity computing facilities deployed in the telecom provider edge network.

As defined by ETSI [7], VNFs can be provisioned either in a Virtual Machine (VM) environment or via bare metal. The choice of one solution over the other is driven by the resources required for a specific network function; moreover, migration can be less efficient in the presence of bare metal virtualization because of the need to power on physical machines, which implies a latency overhead.

Hence, for the design of the VNFs, the evaluation of whether to implement in a VM or on bare metal will be made on a case by case basis.

Physical SDs typically connected to the user's PNs are fully or partially virtualized through software instances running at different levels of the edge network infrastructure. As the use of VMs allows a higher level of isolation, they will probably be the best candidate for this deployment.

III. The Open Volcano Conception

In order to allow the deployment of the personal cloud services as described in the previous section, the edge network nodes need to be re-thought and re-designed in order to provide the support for the new cloud services, including their communication and information exchange. Among the architectural and logical extensions that they will undergo, an important role will be played by in-network programmability, hardware off-loading and power management capabilities. The architecture that the INPUT Project has selected as a starting point for the development of the edge network nodes is the Distributed Router Open-Source Project (DROP) [8].

DROP was originally designed as a middleware for realizing extensible multi-chassis Linux software routers on top of Component-off-the-shelf hardware platforms [9]. During the ECONET Project [10], DROP has been extended to a SDN/NFV-enabled modular router, with fine-grained in-network programmability and power management capabilities exposed by means of the Green Abstraction Layer (GAL) [11]. Because of these characteristics, DROP represents a favorable starting point for the INPUT research activities, which aim to vertically integrate the networking functionalities with the personal cloud services support in a scalable way.

In its basic conception, the DROP architecture had been designed to act as "glue" among a large number of the most promising and well-known open-source software projects, providing novel data- or control-plane capabilities. Throughout its evolution, this characteristic has been maintained by incorporating the protocols and libraries that are promisingly contributing to the innovation of the networking and computing scenarios, such as OpenStack [12], KVM [13] and libvirt [14].

Thanks to these new features, the architecture has made a leap from a Linux software router to an open-source software platform for fog computing. For this reason, the original name was not suitable anymore for a platform that has become so different from its starting point. The OpenVolcano will represent the prototype platform that will be exploited throughout the INPUT Project lifetime and beyond to provide scalable and virtualized networking technologies able to natively integrate cloud services, both personal and federated.

It is worth noting that this platform presents some similarities with OpenStack, which in fact is one of the open-source software projects included in this platform as explained in the following sections. However, the main difference is that OpenVolcano is natively conceived to operate inside fog environments, which guarantees that service instances will always be physically located in the proximity of the user, and be deployed and dynamically migrated according to the user location. In addition, the performance evaluation is carried out both on the service QoS/QoE requirements and on energy efficiency.

The main logical building blocks composing the OpenVolcano architecture, as depicted in Figure 1, are the *StratoV* and the *Caldera*, which represent the control and the data plane of the architecture. Physically, these building blocks, and the elements composing them, can be located in multiple machines spread inside the telecom provider edge network, in order to guarantee the best allocation of the applications and functions needed to realize the personal cloud services. The communication among the various modules and the stakeholders (users, telecom operator and cloud service provider) is mainly performed through RESTful APIs and web interfaces, with additional OpenFlow (OF) connections and proprietary protocols when needed.

IV. The OpenVolcano Control Plane: *StratoV*

The *StratoV* includes the features typically found at the control plane of an IP router, but it is not limited to them. As shown in Figure 1, from the functionality point of view, the *StratoV* can be divided into three layers, which are devoted to the collection of the data and their configuration, the elaboration of the resource allocation and forwarding rules, and the commit of such rules.

The *network manager* is physically located in a remote/separate server, but it belongs to the control plane architecture in every respect, and more specifically to the Data Collection and Configuration layer. The *network manager* is in charge of the long term configuration/optimization of the available physical and logical resources to properly satisfy the bandwidth and quality levels required by the different cloud services instantiated over time. In more detail, it has the ability to collect the monitored data coming from the elements in an

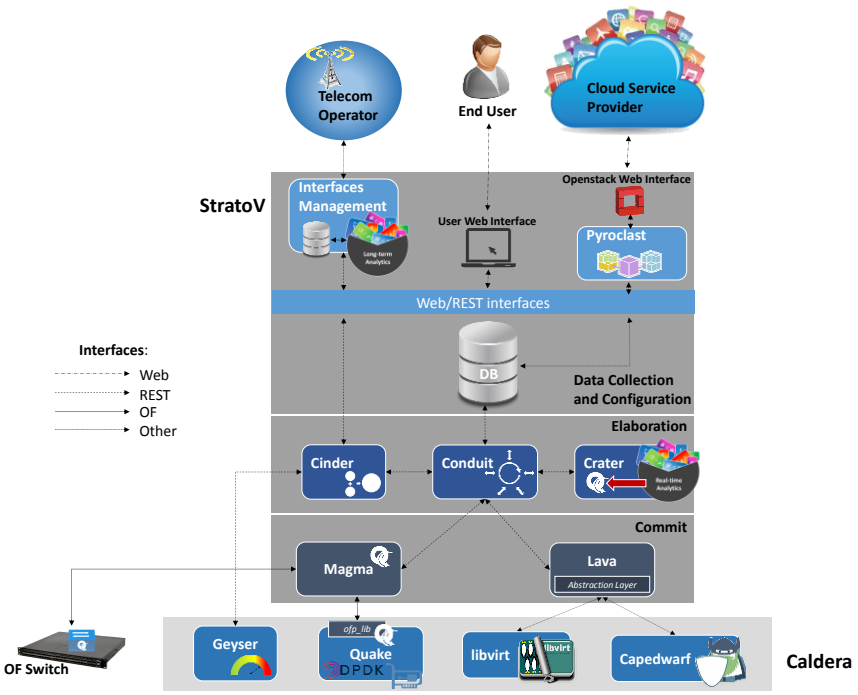

Figure 1. The OpenVolcano Architecture.

internal database, which provides a full view of the deployed services and network states. Thanks to this information, the Long-term Analytics task allows predicting service demand, planning resource provisioning, preventing congestion and possible failures, and maximizing energy saving.

The *database* collects the information provided by OpenStack that regards the users and their subscribed services. The *database* has also the ability to detect modifications in the services subscriptions and notify them to the control functionalities through a REST interface.

The *pyroclast* block presents the same functionalities provided by the OpenStack Nova and Neutron modules. In more detail, it is exploited by the cloud service provider to manage the service chains, including the computation and networking aspects needed for a proper service support.

The Elaboration layer comprehends all of the functionalities needed to aggregate, elaborate and deliver the updated physical and logical configuration for the data plane devices. It is composed of three main modules.

The *cinder* module is in charge of aggregating the monitored data and detecting possible faults. This module can perform synchronous requests for information to the data plane devices and the *network manager* and verify the proper behavior of all network components. The *conduit* module can be considered the center of the control plane, as it is responsible for receiving the information related to monitoring and services, forwarding it for further elaboration, and then provide the obtained configuration to the second control stage.

The *crater* is in charge of the real-time configuration of the logical and virtual resources, which can be triggered by the current users' location and the monitoring of parameters like the power consumption or CPU utilization. The data used to feed this engine come from both the aggregated monitored information provided by the *cinder* and the deployed services as reported by the database. Exploiting these data, a number of Consolidation and Orchestration algorithms calculate the optimal resource allocation, according to the required QoE/QoS and the estimated workload/traffic volumes, and the OpenFlow rules.

In the case of faults, *cinder* signals the presence of a device or sub-element that is not behaving properly to the *conduit*, which sends a request to the *crater*. The device is then removed from the graph of the available resources and a new configuration is computed accordingly.

The communication of the re-calculated OF rules and allocations is managed again by the *conduit*, which transmits them to the Commit stage to be further made available to the data plane elements. Specifically, the OF rules computed by *crater* are sent to *magma*, which provides the typical OF controller capabilities and is responsible for updating the flow tables of the elements devoted to forwarding (e.g., physical and virtual OF switches).

Finally, *lava* has the ability to abstract the virtualization libraries deployed at the data plane in order to make them transparent to the computed resources allocation. In this way, the rules can be computed by *crater* without the knowledge of the virtualization platform (libvirt, Capedwarf, etc.) deployed on each server.

V. THE OPENVOLCANO DATA PLANE: *CALDERA*

The *Caldera* of OpenVolcano represents the data plane and can be composed of up to four modules. As shown in Figure 2, the servers in the data plane are connected among themselves and to the *StratoV* through OpenFlow switches.

The servers composing *Caldera* are also built on top of a Linux architecture. Netlink is used for communication of the kernel with the user-space, where most of our implementations take place. In fact, user-space libraries have been demonstrated to provide massive performance improvements, at the price of a reduced number of already available network functionalities with respect to the kernel-level forwarding [15].

Aside from Netlink, which is mainly used for the exchange of generic data between the kernel and user-space,

like requests for acknowledgement, echo or FIB updates, the other interfaces between kernel and user-space are the `/proc` and `/dev` filesystems. The ACPI [16] standard is used to represent the power management capabilities of the hardware components (packet processing engines, network cards, cores, etc.).

Packets incoming through the NICs are managed by high performance data packet applications based on DPDK [17]. All packets received through the NICs are then taken in charge by the *quake* module, a software OpenFlow switch implemented in the user-space according to the OF 1.3 specification, which will be described in more detail in Section VI. KVM [13] has been chosen to implement the hypervisor in charge of handling the VMs according to the directives sent by *conduit*, as sketched in the previous section.

The *geyser* module provides the data related to the server current status to the *crater* module after the aggregation operated by *cinder*. Such data include the current energy state (according to the ACPI standard), CPU load and memory, which can be monitored both globally and at single process level. The communication is managed through the GAL REST interface.

The *geyser* and *quake* modules are present in every server composing the data plane, while additional libraries can be available depending on the deployed service. In the example reported in Figure 2, libvirt is used to manage VM migrations, while other libraries such as Capedwarf allow the management of PaaS instances.

VI. The Quake Module

The *quake* module included in the *Caldera* has been designed to manage the received traffic and direct it to the correct function located in that server. *Quake* has been implemented according to OpenFlow Version 1.3, and provides the protocol's basic functionalities, such as echo request/reply, features request/reply, and hello messages.

Quake is an original design of the project, and it has been implemented by exploiting the DPDK libraries available for packet capture and for the design of the packet buffers.

In order to overcome the rigid resource allocation of DPDK, in which the number of threads and their affinity are set at boot and cannot be modified at runtime, the introduction of a more flexible affinity has been realized by exploiting the worker threads: during the initialization phase, a dispatcher thread, associated with the main core (usually Core 0), and a number of worker threads are started. The affinity of the thread is fixed, but the number of active worker threads can be changed at runtime. The dispatcher will not use the inactive worker threads.

The internal structure of the virtual switch is shown in Figure 3. After the initialization, the dispatcher cyclically queries the Rx queues of the physical and logical ports waiting for incoming packets. Upon packet arrival, the dispatcher retrieves the packet from the queue and decides at runtime the worker thread that will receive the packet. The designated worker thread is then in charge of matching the packet header against the OF table using a specific OF function and determines where to forward the packet. The possible destinations are the transmission queue of a physical port (as shown in Figure 3, there is a dedicated Tx queue for each worker thread to avoid the need for some synchronization mechanism), or the transmission queue (in this case, multi-producer) of a logical port. The latter case regards packets that are directed to the virtualized images of the personal cloud services, as will be described in the next section.

The ofp_lib module guarantees the communication with the *magma* module located in the *StratoV*, generating/receiving the packets and updating the flow tables.

A. Performance Evaluation

In order to characterize the performance level of the virtual switch under different traffic conditions, and to identify current issues and room for improvement, some tests have been run on our prototype and are reported in this section. The same tests have also been performed on another device equipped with Open vSwitch to provide a term of comparison.

Both devices are built on an Intel Xeon E5-2643 v3 processor with two CPUs and six cores with a working frequency up to 3.40GHz, and 8 x 8 GB DDR4 2133 ECC

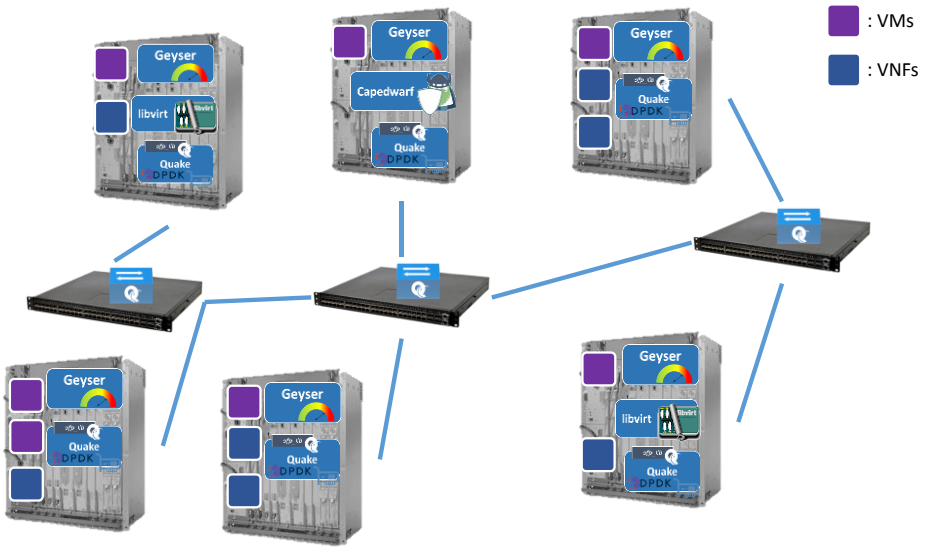

Figure 2. The data plane deployment.

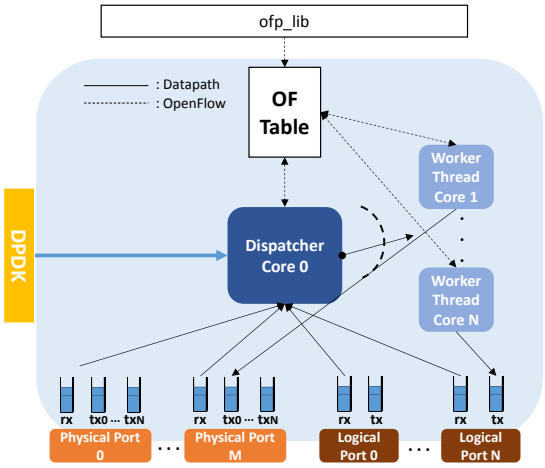

Figure 3. The *quake* module internal structure.

RAM memory. The operating system is Linux version 4.2.6-3 x86_64.

Tests are performed by transmitting traffic to the tested devices and measuring its throughput using an OptIxia router tester [18] connected by a couple of 10 Gigabit Ethernet links, one in transmission and one in reception. Continuous traffic is transmitted at different offered loads, growing from 1 Gbit/s to 10 Gbit/s.

In the first test case, the throughput is measured on the device equipped with Open vSwitch (results labelled as "OVS" in Figure 4. Throughput percentage measured in presence of varying packet sizes on the two devices under test.) and our prototype exploiting the DPDK libraries (results labelled as "DPDK" in Figure 4).

As expected, the results in Figure 4 show that both devices present better performance in the presence of longer packets, with zero-loss throughput for 1518-Bytes packets. Regarding the device based on Open vSwitch, the decay in the throughput for packets of 64 Bytes begins immediately, while for packets of 570 Bytes losses begin at 40% of the offered load (e.g., 4 Gbit/s) and continue with the same trend of the smaller sized packets. This trend shows that the OVS switch can handle a certain number of packets, discarding all of the traffic above that threshold. This behavior is confirmed by the results represented in terms of Gbit/s, as reported in Figure 5, where it can be seen that the throughput stays constant once packet loss begins.

Regarding our prototype based on DPDK, it outperforms the OVS-based device for the respective packet sizes. Considering the results in Figure 4, it can be seen that the prototype loses visible traffic between 20% and 30% of the offered load, then throughput starts improving again. This behavior is due to the fact that the prototype enters a deep idle state when packets are too separated from each other. This aspect, however, does not prevent the switch from outperforming the reference device and will be solved in the near future.

In the second test case, traffic is not composed of fixed size packets, but it follows an Internet Mix (IMIX) distribution [19]. Traffic is also divided in a number of flows going from 1 to 256, in order to trigger matches in the switches flow tables. The results of these tests are reported in Figure 6.

Throughput decreases for both tested devices as the number of flows grows. However, the differences for the

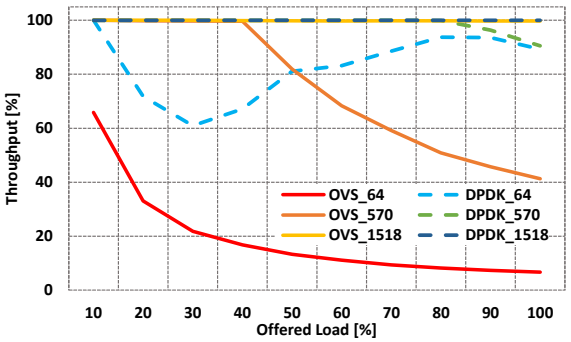

Figure 4. Throughput percentage measured in presence of varying packet sizes on the two devices under test.

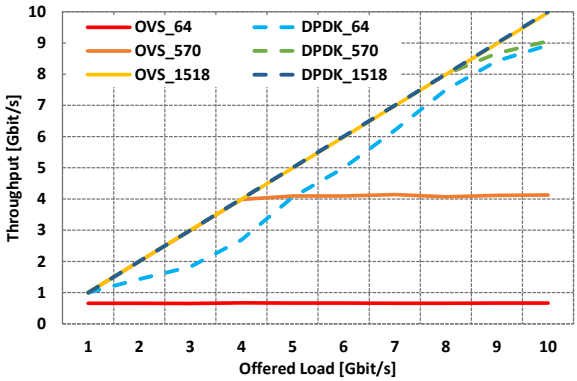

Figure 5. Throughput expressed in Gbit/s measured in presence of varying packet sizes on the two devices under test.

OVS switch are not very significant, showing that the losses are more driven by the volume of received traffic, hence preceding the classification phase. It is worth noting that the OVS-based switch performs a hash search in the entry table, hence the time complexity is $O(1)$ and independent from the number of entries. On the other hand, for the prototype, losses appear to be proportional to the number of flows, due to a different choice of the flow table search algorithm that leads to a time complexity of $O(n)$, and become very severe above 128 flows, because above that threshold the flow table becomes too big to be kept in the cache memory.

This behavior is further confirmed by a final measurement that has been performed under the same conditions of Figure 6 for 64 flows, but with the addition of a rule in the flow table that modifies the MAC destination address. The results can be seen in Figure 7, and show again that the increased number of operations on the flows impacts the performance of the prototype but not those of the OVS-based switch.

VII. CONCLUSIONS

The European Commission-funded INPUT Project is devoted to foster future Internet infrastructures by moving computing and storage capabilities to the edge network. The virtualization of home entertainment appliances and of sensors, which will be made available to the users as cloud services, will add potentially infinite smartness and capacity to devices with performance- and functionality-constrained hardware platforms.

This paper has described OpenVolcano, the prototype that will be exploited throughout the INPUT Project lifetime and beyond to provide scalable and virtualized networking technologies able to natively integrate cloud services, both

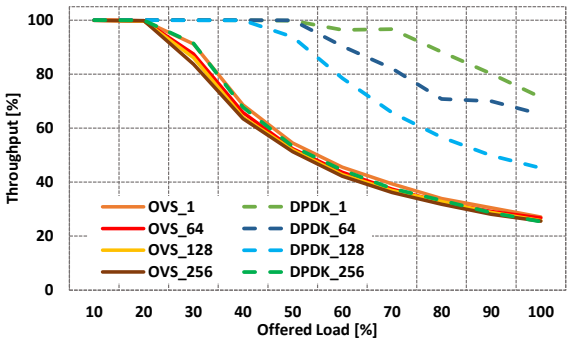

Figure 6. Throughput measured in presence of a varying number of flows on the two devices under test.

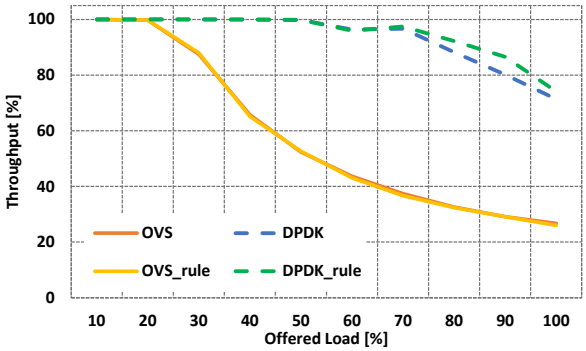

Figure 7. Throughput measured in presence of a varying number of flows sizes on the two devices under test.

personal and federated. OpenVolcano is an open-source software platform for fog computing, which exploits the most promising and well-known open-source software projects to introduce in-network programmability capabilities for off-loading, virtualization and monitoring. Tests performed on the *quake* module have demonstrate the high performance level that is achieved through OpenVolcano.

ACKNOWLEDGMENT

This work has been supported by the INPUT (In-Network Programmability for next-generation personal cloUd service supporT) project funded by the European Commission under the Horizon 2020 Programme (Call H2020-ICT-2014-1, Grant no. 644672).

REFERENCES

[1] Ericsson, "More than 50 Billions Connected Devices,"white paper, Feb. 2011.

[2] M. Chiosi, et al., "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call For Action", ETSI White Paper, Oct. 2012, URL: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[3] "Software-Defined Networking: The New Norm for Networks, Open Networking Foundation (ONF)," White Paper, Apr. 2012.

[4] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, "Fog Computing and Its Role in the Internet of Things," Proc. of ACM MCC 2012, Helsinki, Finland, August 2012.

[5] "The In-Network Programmability for next-generation personal cloUd service supporT (INPUT) Project," URL: http://www.input-project.eu/.

[6] R. Bolla, R. Bruschi, F. Davoli, C. Lombardo, L. Masullo, "The Expected Impact of Smart Devices Virtualization ," 1st Internat. Workshop on Sustainability, Implementation and Resilience of Energy-Aware Networks (SIREN 2016), Kauai, Hawaii, USA, Feb. 2016.

[7] ETSI GS NFV, "Network Function Virtualization; Virtual Network Functions Architecture," GS NFV-SWA 001 v1.1.1.

[8] R. Bolla, R. Bruschi, C. Lombardo, S. Mangialardi, "DROPv2: Energy-Efficiency through Network Function Virtualization," IEEE Network, Special Issue-Open Source for Networking: Development and Experimentation, vol. 28, no. 2, pp. 26-32, Apr. 2014.

[9] R. Bolla, R. Bruschi, "An Open-Source Platform for Distributed Linux Software Routers," Computer Communications (COMCOM), Elsevier, vol. 36, no. 4, pp. 396-410, Feb. 2013. DROP source code available at https://svn.econet-project.eu/svn/.

[10] "low Energy Consumption NETworks" (ECONET) Project, URL: http://www.econet-project.eu.

[11] ETSI Environmental Engineering (EE), "Green Abstraction Layer (GAL) - Power management capabilities of the future energy telecommunication fixed network nodes," ES 203 237 V1.1.1, URL: http://www.etsi.org/news-events/news/822-2014-09-news-green-abstraction-layer-standard-to-manage-energy-consumption-of-telecom-networks.

[12] OpenStack cloud computing software platform URL: http://www.openstack.org/.

[13] Linux Kernel Virtual Machine, http://www.linux-kvm.org/.

[14] libvirt Virtualisation APIs, URL: http://libvirt.org/.

[15] R. Bolla, R. Bruschi, "PC-based Software Routers: High Performance and Application Service Support," Proc. of ACM SIGCOMM PRESTO, Seattle, WA, USA, August 2008.

[16] The Advanced Configuration & Power Interface (ACPI) Specification - Revision 5.0, http://www.acpi.info/.

[17] The Intel "Data-Plane Development Kit," URL:http://www.dpdk.org.

[18] The Ixia XM2 router tester, URL: http://www.ixiacom.com/products/display?skey=ch_optixia_xm2.

[19] "IMIX (Internet MIX)," Test Methodology Journal, Spirent Communications, March 2006, URL: http://spcprev.spirentcom.com/documents/4079.pdf