DROPv2: Energy-Efficiency through Network Function Virtualization

R. Bolla[*‡], R. Bruschi[*], C. Lombardo[‡], S. Mangialardi[*]
[*] CNIT - Research Unit of the University of Genoa - Genoa, Italy
[‡] DITEN - University of Genoa - Genoa, Italy

Abstract—**Future Internet devices and network infrastructures need to be significantly more energy-efficient, scalable and flexible in order to realize the extremely virtualized and optimized ICT/network infrastructures. In this respect, this paper presents a recent extension of an open source software framework, the Distributed Router Open Platform (DROP), to enable a novel distributed paradigm for Network Function Virtualization (NFV) through the integration of Software Defined Network (SDN) and Information Technology (IT) platforms, as well as for the control/management of flexible IP-router platforms. To answer to the need for increased energy efficiency of the NFV paradigms, DROP includes sophisticated power management mechanisms, which are exposed by means of the Green Abstraction Layer (GAL), under consideration for standardization in ETSI (the European Telecommunications Standards Institute). Moreover, the DROP architecture has been specifically designed to act as "glue" among a large number of the most promising and well-known open-source software projects, providing network data- or control-plane capabilities.**

Keywords—SDN; NFV; open-source; energy efficiency.

I. INTRODUCTION

As underlined by ITU-T [1], a radical change in core networking technologies, devices and network architectures is necessary to achieve the performance scalability levels needed to cope with the expected traffic increase, while significantly reducing the energy consumption and footprint of next-generation Internet service infrastructures. Major Telecom operators worldwide [2] also confirmed that future Internet devices and network infrastructures need to be significantly more *energy-efficient*, *scalable* and *flexible* in order to realize the extremely virtualized and optimized Information Technology (IT)/network infrastructures to adequately support (on the basis of these new scenarios) a very large number of heterogeneous and rapidly changing user-led services [1].

Some of the most promising and recent research and standardization trends [2][3] combine technologies like SDN and NFV to meet Future Internet scalability and flexibility requirements. Such approaches are based on the distribution of network functionalities among SDN nodes and IT devices.

SDN nodes are designed to perform high speed customizable switching operations. To this aim, these nodes are generally realized with special-purpose hardware components (e.g., packet processors, TCAM [4], etc.) that allow high performance but limited programmability in customizing switching operations on a per-flow basis. In this respect, SDN protocols, like OpenFlow [5], are devoted to providing standard interfaces for the customization of per-flow switching operations.

More complex networking operations (e.g., accounting, deep packet inspection, anomaly detection, etc.) are delegated to IT devices, which combine the advantages and the cheapness of component-off-the-shelf hardware platforms with the intrinsic flexibility and programmability levels, as well as the advanced network support of modern Operating Systems (OSs). This architectural solution is often referred to as Network-as-a-Service (NaaS). Moreover, when open-source projects (and especially the Linux environment) are applied/integrated to IT nodes, programmability becomes much easier and constraint-less, since every part of the system is open and customizable by everyone.

Notwithstanding the potential benefits in network flexibility and scalability, energy efficiency remains a serious concern for these novel network paradigms. It especially arises from (*i*) the joint use of multiple hardware platforms to realize functions that today are embedded inside single legacy network nodes, and (*ii*) the use of general-purpose systems instead of more hardware-optimized special-purpose ones. Both these aspects contribute to the increase of the overall network energy requirements.

A possible solution to increase the energy efficiency of SDN/NFV systems can reside in the use of advanced power management schemes. Exploiting the increased modularity and virtualization of SDN/NFV network architectures, SDN or IT hardware platforms may be switched on and off and virtual functionalities migrated, or the maximum transmission/elaboration capacities of devices can be tuned in order to meet current traffic volumes and service requirements [6].

Taking the flexibility, scalability and energy efficiency aspects of aforementioned network paradigms as the cornerstone of our research objectives, we extended the open-source software framework DROP (Distributed Router Open Platform) toward NFV/SDN architectures and power management capabilities.

This work has been supported by the ECONET (low Energy Consumption NETworks) project co-funded by the European Commission under the 7th Framework Programme (FP7).

DROP was originally designed as a middleware for realizing extensible multi-chassis Linux software routers on top of Component-off-the-shelf hardware platforms [7], and for transparent integration of Linux network control-plane applications like Quagga and Xorp. DROP aimed at aggregating and autonomously coordinating all these building blocks into a single logical IP node, hiding the complexity of its modular architecture to control-plane applications and system administrators. In its basic conception, DROP is OS-independent. However, since Linux already includes many solutions and support that are common to other modern operating systems optimized for multi-core HW, our current implementation as described in the following considers Linux native interfaces. By all means, no limitations would be envisaged to making it OS agnostic.

In the new version, introduced in this paper, DROP has been extended

(i) to natively interface and integrate with OpenFlow switches within its modular architecture;
(ii) to provide an enhanced and even more transparent integration environment to native Linux control-plane applications and administration command-line tools (e.g., "ip");
(iii) to implement advanced power management strategies by means of the Green Abstraction Layer (GAL), a standard interface under consideration for standardization in ETSI;
(iv) to easily integrate novel open-source frameworks for data-plane processing in the Linux user-space, like, among others, the Intel Data-Plane Development Kit (DPDK) [8] and Netmap [9], which are often applied to realize custom packet forwarding chain operations at very high speeds on general-purpose processors.

In few words, the DROP architecture has been specifically 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.

To this purpose, DROP implements a number of "standard" interfaces native to the Linux operating system (e.g., Netlink, which is the standard Linux interface to notify configuration changes among control processes and the Linux kernel), to SDN devices (Openflow), and to the network device power management (i.e., GAL). These interfaces are used in the DROP southbound to transparently interact with multiple data-plane elements, as well as to provide a simplified device view at the control-plane processes in the northbound. Between north- and southbound interfaces, DROP implements a number of mechanisms to manage the internal topology, to (dis-) aggregate network configuration data, like the Forwarding Information Database (FIB) or the power management configurations of elements, and to support slow-path communications (i.e., delivery of signaling traffic to and from the control-plane).

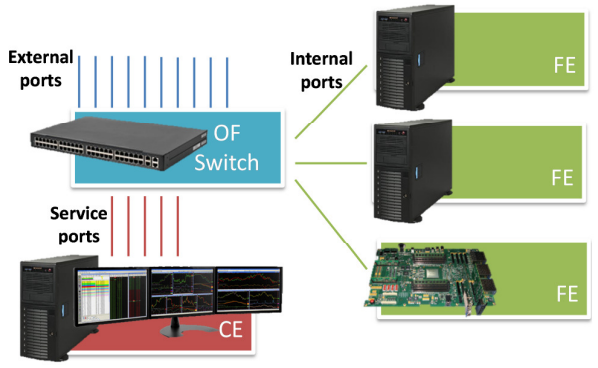


Figure 1. The base DROP setup architecture used in the experimentation.

The remaining of the paper is organized as follows. Section II presents an overview of the DROP architecture, while the technical details about control and forwarding elements are in Sections III and IV, respectively. Some details about the behavior of the power management capabilities can be found in Section V. Testbed and results relative to the current DROP implementation can be found in Section VI. Finally, conclusions are offered in Section VII.

II. THE DROP ANATOMY

The DROP architecture is composed by three main kinds of building blocks, as shown in Figure 1:

- *Forwarding Elements* (FEs);
- *Control Elements* (CEs);
- *Interconnection Elements* (IEs).

IEs are realized by OpenFlow switches, and are meant to interconnect the other DROP elements. The external ports of the DROP router (i.e., the links toward other network nodes) reside on the IEs, which are also meant to provide simple and customizable traffic switching operations to/from internal elements or external ports.

If more complex data-path operations are required, the traffic is sent to FEs for further processing. FEs are realized on top of "general purpose" IT platforms with the Linux operating system, and consequently allow an extremely high level of programmability. Forwarding operations available at FEs can be customized and differentiated among the different elements.

CEs, realized on top of Linux IT servers just as FEs, are devoted to orchestrating the operations of IEs and FEs by maintaining and updating the router FIB, and to provide control-plane services to the router. To this purpose, the DROP framework is composed by two main kinds of applications, namely the Control Element Controllers and Forwarding Element Controllers (CECs and FECs, respectively). These applications interact to realize the distributed management of the FIB data, the internal topology and organization, and the slow and fast paths in a transparent and autonomic way. The CEC is also responsible for the direct connection to the IEs by means of the OpenFlow protocol.

DROP supports router virtualization, in the sense that multiple logical router instances can be created on the same set of hardware elements. Each FE or IE can be shared by more than one router instance and internal resources (e.g., ports, traffic processing engines, etc.) can be assigned to a specific router instance or shared. A "master" CEC is needed for each active DROP instance, and further CECs in the same logical

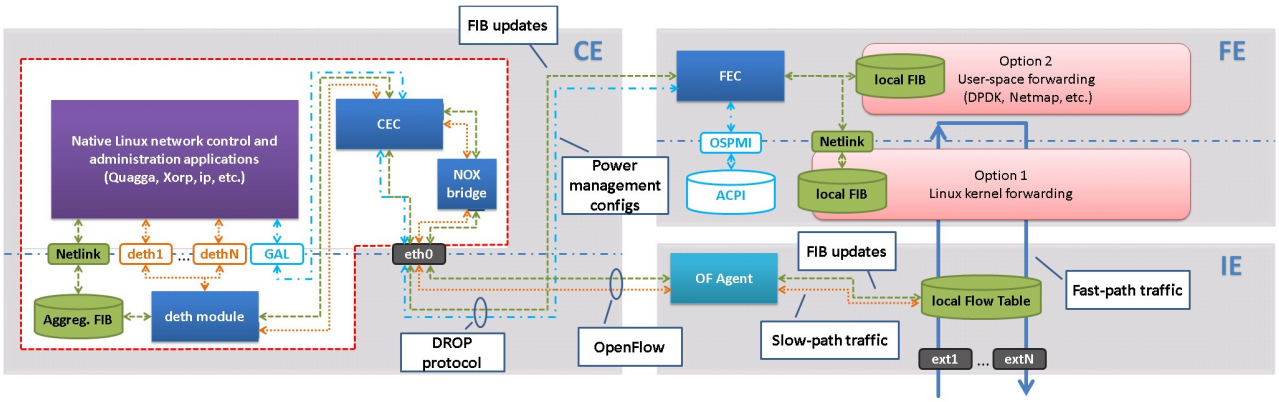


Figure 2. The DROP software structure including the main building blocks, interfaces, and slow- and fast-paths.

router are used as backup for fault recovery of the master copy. CECs/FECs of different logical nodes can run on the same CE/FE hardware platform.

Although DROP allows large-scale and complex set-ups, in this paper we decided to use a base router configuration with a limited set of elements and enabled network functionalities. This choice helps us to better focus on the internal router design, and to highlight how DROP acts as glue among various open-source networking tools by fostering programmability, scalability and energy-efficiency. The selected DROP configuration is shown in Figure 1: it is based on a simple star topology with one OF switch acting as IE at the star center. The IE is capable of connecting one CE and up to 10 FEs.

Regarding the enabled functionalities, DROP is configured to act as a "legacy" router doing IP forwarding[1]. The IE is responsible for balancing the traffic load coming from the external ports to the FEs, for collecting processed packets from FEs, and for delivering them to the corresponding external ports. FEs perform standard IP look-up operations.

III. THE CONTROL-PLANE

The CE consists of a general-purpose Linux PC running a number of CEC applications (one for each configured logical router). The CEC is responsible for:

- managing and configuring all IEs and FEs belonging to the same logical router;
- considering the internal physical organization and topology, aggregating the FIBs of all elements into a single "aggregated" FIB representing the logical node as a whole;
- exposing this information toward control processes, receiving their feedbacks and, consequently, applying configuration changes (e.g., a new route);
- managing the delivery of signaling traffic among control processes and router external ports.

In order to provide a seamless integration with Linux native control applications (like Quagga and Xorp, or simple command line tools like "ip"), as shown in Figure 2, the logical router control-plane is composed by a number of other software objects and tools (depicted with the blue blocks in the figure), of which the CEC is the core component.

All the control-plane processes of each logical router run inside a Linux *network namespace*, which is a very lightweight form of OS virtualization. Within the DROP architecture, the namespace creates a standard virtual environment, where the Linux operating system works as an abstraction layer, and hides the complexity of the modular nature of DROP to Linux-native applications. For example, the namespace exposes only the "aggregated" FIB provided by the CEC (the one representing the router as a whole in terms of router interfaces, routing tables, etc.) through the usual Linux interfaces. This virtual environment constitutes the northbound interface of the CEC, and allows a seamless integration of DROP with any networking tools available in the Linux OS. The only integration requirement is to run these tools inside the aforementioned namespace. If more than one control plane of logical routers are hosted on the same CE, multiple namespaces are allocated.

Inside the same namespace, the CEC creates a number of virtual interfaces ("*deth1*,…, "*dethN*" with reference to Figure 2), one for each DROP external port, which resides on the IE. These virtual interfaces deliver signaling traffic from control processes to external ports, and vice versa. The delivery is possible thanks to a kernel module, named "*deth*," which was specifically designed to enable the CEC to receive and to transmit packets from/to the virtual interfaces.

Moreover, the CEC application also acts as a Netlink "proxy," in the sense that it maintains the aggregated FIB of the Linux namespace aligned with its own copy, and, consequently, with the local FIBs of FEs and IEs. Thanks to this Netlink proxy, the CEC can wait for routing or interface configuration updates from control processes and apply them to the DROP elements; in addition, it can notify events related with status changes (e.g., connectivity failure of router interfaces) to these processes. In addition to the above aspects, a configuration interface for the router power management is also exposed by means of the GAL (see section V).

[1] The choice of using legacy IP forwarding is driven by the need of providing "absolute" performance indexes, which can be compared to commercial products.

On the CEC southbound side, connections toward other DROP elements are maintained to synchronize the local FIBs. In detail, the CEC connectivity toward FEs is performed by the DROP protocol, a simple protocol we designed to encapsulate Netlink and GAL messages and to deliver them to the network elements.

Regarding IEs, we decided not to integrate an OF interface inside the CEC application, but to use an existing open-source and well-known controller, named NOX [10]. The clear advantage of this choice is the possibility of exploiting a widely used and rapidly evolving OF software library. A bridge application was integrated inside NOX in order to bind the OF messages with CEC events.

IV. THE DATA-PLANE

As previously sketched, the DROP data-plane is composed by IEs and FEs. IEs provide the router external ports and perform simple switching operations. FEs act as internal packet processing engines, whose operations can be fully customized by means of software programming.

In the simple set-up used in this paper, where DROP acts as a legacy IP router, a single IE acts as a "first-stage" load balancer among external ports and internal FEs devoted to performing IP lookup operations. Other and more complex setup/application scenarios are obviously possible, but fall out of the scope of this contribution.

The IE is configured through the OpenFlow protocol. Two sets of OF rules are associated to each external port: the first set is related to the fast-path forwarding, and the second one to the slow-path delivery. In detail, fast-path rules simply redirect all the traffic coming from an external port to an internal one connecting a specific FE. The destination MAC address of incoming packets is updated from the one of the external port, to the one of the FE. As explained in section V, the configuration of the load-balancer is updated only when new power management configurations are applied.

Slow-path packet reception rules are dynamically configured by the CEC, which listens for "server-side" sockets opened by control applications into the Linux namespace. When one of such sockets is opened the CEC installs one or more OF rules into the IE in order to filter and to capture the packets destined to the new socket. This procedure happens for both TCP/UDP "server" sockets and "raw" sockets listening for L2/3 traffic with unicast/multicast addresses. The transmission direction of the slow-path is simpler, since OF permits to send L2 frames to specific ports or groups of ports.

The FEs are based on high-end multi-core hardware platforms, running the Linux OS. The following two types of hardware platforms have been used:

- Component-off-the-shelf servers equipped with two Intel Xeon E-5640 quad-core processors and a Mellanox ConnectX-3 network adapter with two 10 Gigabit Ethernet ports;
- Evaluation boards of the Broadcom XLP832 System-On-Chip [11], a MIPS processor with 8 cores and various network accelerators.

From the software perspective, FECs allow fast-path processing with two main options (Figure 2):

- the seamless integration with the local kernel networking chain, thanks to direct Netlink communications;
- a set of simple APIs to integrate with custom user-space forwarding chain, based, for instance, on the Intel DPDK, Netmap, or the Broadcom HyperExec libraries.

Despite the easiness in customizing forwarding chains, 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 [12].

V. POWER MANAGEMENT

As previously sketched, a further innovation aspect of DROP is represented by the inclusion of the first (and open-source) complete realization of the GAL. The GAL is an abstraction interface for power management in network devices [13], which is under standardization in ETSI.

The GAL is organized as a multi-layered and hierarchical interface, acting inside network devices to drive power management primitives, and hiding the details of their hardware implementations. At the top layer, it allows associating a set of "energy-aware" states to each router interface. Energy-aware states allow representing not only the device energy consumption, but also the performance provided under a certain configuration.

At the bottom layer, a convergence layer is used to access the power management hardware primitives available in the network node. Intermediate layers are needed to drive the optimization of device energy configurations at different levels of the internal device architecture. To this purpose, intermediate layers are associated with Local Control Policies (LCPs).

As shown in Figure 3, DROP implements a 4-layer realization of the GAL including two types of LCPs: a first LCP acting at the router layer and a group of LCPs optimizing the behavior of DROP elements (and more specifically of FEs). The top layer is terminated into the CE Linux namespace (Figure 2), and it exposes the available energy-aware states associated with each "*deth*" interface to control plane processes. The bottom level is used to interface the GAL with the Linux OS Power Management Interface (OSPMI) of the FEs, which already includes the drivers for configuring the power setting of many hardware components.

The intermediate GAL layers host the aforementioned two LCPs. Although these LCPs work at different levels, they are both based on the same optimization mechanism introduced in [14].

At the router level, on the basis of the estimated traffic load to be received, the LCP periodically calculates the number of FEs to be put into standby states, as well as the speed (and then the power settings) and the share of incoming traffic to be configured on the active FEs; load balancing rules at the IE are updated accordingly.

At the FE level, LCPs behave in a similar way: by knowing an estimate of traffic load, they calculate the optimal settings for forwarding operations with minimal energy consumption and a desired performance level. The optimal settings are obtained by putting some cores into standby mode, and by tuning the working frequencies of the ones remaining active. The ConnectX3 adapter on the server platform or the network accelerators of the XLP System-on-Chip are used to steer the incoming traffic to active cores, as well as to adjust the traffic volumes according to the core frequencies. Additional details can be found in [15].

VI. Performance Evaluation

The scalability of the DROP data- and control-plane are evaluated in section VI.A. Results related to the energy efficiency are reported in Section VI.B.

A. *Scalability Tests*

We consider a DROP router set-up similar to the one depicted in Figure 1. The number of FEs varies throughout the test sessions, and includes both the Xeon servers and the XLP evaluation boards. All platforms are equipped with two 10 Gbit Ethernet ports. The DROP external ports on the IE are connected to an Ixia tester to generate traffic and measure the router performance. Packets have been generated according to an IMIX distribution at the maximum line rate.

Figure 4a shows the maximum throughput provided by DROP according to the number of FEs and their hardware architectures. In detail, we deployed up to 10 FEs with the following combinations: (*i*) only Xeon-based FEs performing kernel-level forwarding (*ii*) only Xeon-based FEs performing user space-level forwarding by means of DPDK (*iii*) only XLP Systems-on-Chip performing forwarding by means of the user-space "HyperExec" environment. As expected, performance levels significantly increase when user-space forwarding is applied for both Xeon servers and XLP platforms. Moreover, all the results show a linear dependency between the number of FEs and the maximum throughput. This linearity is a clear indication of the scalability of the data-plane reference design.

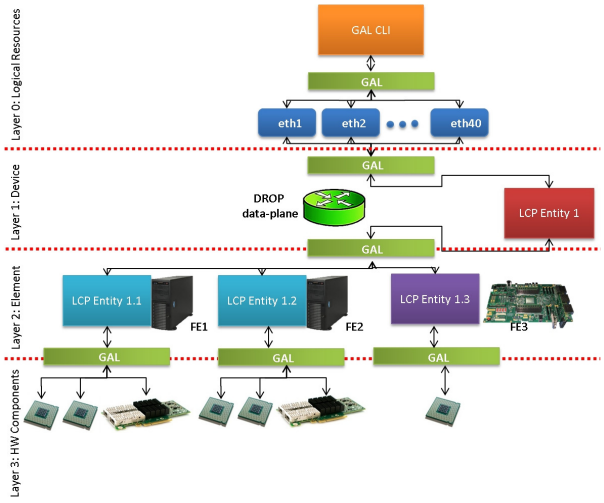
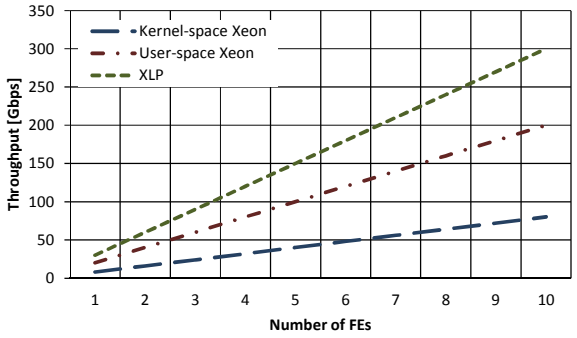
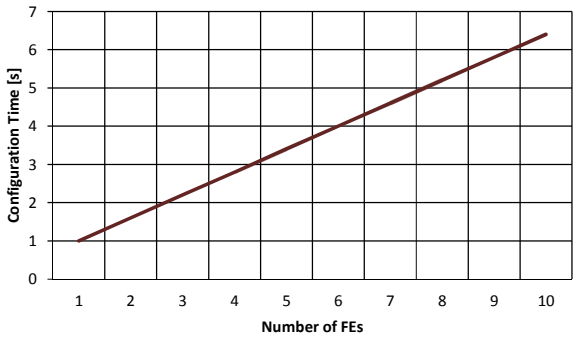


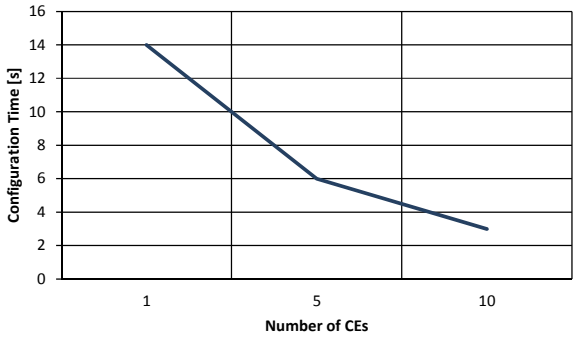
Figure 3. The interactions between the GAL and the LCPs.



(a) Throughput



(b) FEs



(c) CEs

Figure 4. Test results depicting (a) throughput at varying number of FEs (b)FE configuation times and (c) CE configuration times.

Figure 4b shows the time needed to configure the entire router upon addition of an increasing number of FEs. In other words, this time represents the interval passing from the new FE notification to the instant when it is ready to start forwarding traffic. The time needed for activating the first

element is approximately 1s, while each further FE provides an additional delay of only 0.6 s.

Figure 4c takes the general idea of the previous test, but it analyzes the CE configuration times when multiple logical routers are configured. Hence, in this test we considered the configuration times of (*i*) one router including 10 FEs (*ii*) two logical routers, each one including 5 FEs, and (*iii*) ten routers, each one with only one CE and one FE. Starting from the previous results on FE configuration times (Figure 4b), an additional delay of 2s is required for activating each CE instance. Results in Figure 4c also show that a "smaller" DROP set-up (lower number of elements) requires shorter configuration times.

B. DROP Energy Efficiency

The tests here reported have been performed during a live demonstration at the 2nd year review meeting of the ECONET project, held in Turin in March 2013, in front of European Commission delegates.

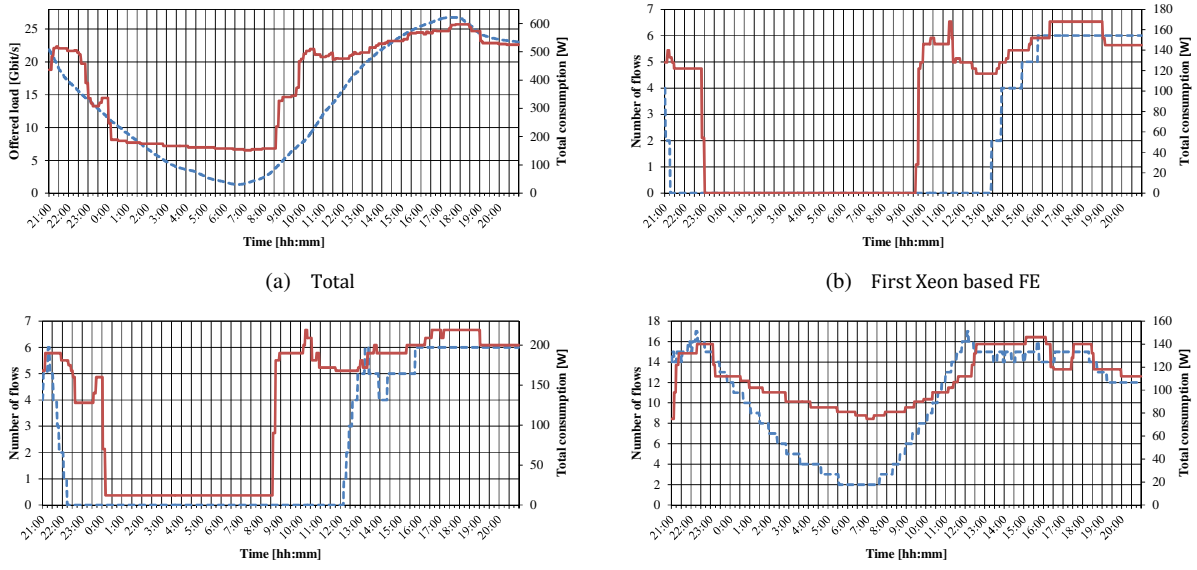
The DROP architecture exploited for this test session includes one CE (with only one CEC), an IE, and three FEs, namely two Xeon servers and one XLP board. The Broadcom FE has only the capability to adjust its maximum throughput (and then its consumption) through voltage and frequency scaling primitives. Xeon-based FEs can scale their maximum throughput and can also enter standby modes. The GAL architecture and the LCPs described in Section V have been applied.

We fed DROP with a realistic daily traffic profile with typical night and day fluctuations, which was obtained from the measurement campaign performed by the ECONET consortium [15]. The traffic profile is represented by the dotted line in Figure 5a, and was reproduced at accelerated time during the live experimentation. In detail, a profile day was reproduced in 15'. The overall load offered to DROP varies between 2 and 28 Gbit/s throughout the trace, and enters the router through the 40 external Gigabit Ethernet ports on the IE.

The solid line in Figure 5a shows the total power consumption of the DROP router detected during an emulated day. The energy proportionality achieved by using the GAL and the LCPs is evident. The joint use of green capabilities on the three FEs, in terms of power scaling and standby, allows an average energy saving of approximately 37%. This result is even more convincing considering that no considerable service degradation has been experienced during the tests (i.e., zero losses, and packet latency increase lower than 20% with respect to the case with no power management).

The slight increase of consumption experienced between 9.00 and 11:00 is due to the wake-up phase of the two Xeon-based FEs, whose power consumptions with respect to the received number of 1-Gbit/s flows is plotted in Figures 5b and 5c. A peak in consumption appears right after the wake-up, corresponding to the system boot phase. Of course, the increase is noticeable only because the traffic trace lasts less than 15 minutes: it would go completely unnoticed in a 24-hour-long scenario. Finally, it is worth mentioning that, thanks to the low power idle and smart standby capabilities of the Xeon-based platforms, the two FEs account for a power saving of 54% and 43%, respectively.

The XLP does not have standby capabilities. Thus, when the traffic is low, the router-level LCP redirects all flows to the XLP FE, and puts to sleep the remaining FEs. Although the amount of energy savings is lower (25%) with respect to the Xeon-based FEs, the XLP guarantees higher performance and a more than convincing proportionality to the incoming load. Figure 5d shows the trend of the power consumption with respect to the received flows. The traffic falling edge between 11:00 PM and 6:00 is accurately followed by the power



(a) Total

(b) First Xeon based FE

(c) Second Xeon based FE

(d) XLP

Figure 5: Power consumed by (a) the total system (b) the first Xeon-based FE (c) the second Xeon-based FE and (d) the XLP over the traffic trace. The dotted lines represent the incoming traffic load, and the solid ones the energy consumption.

consumption thanks to the XLP adaptive rate capabilities and to the LCP, which in this time slice reduces the frequency of one core at a time and then increases it as the traffic starts growing again.

VII. CONCLUSIONS

This paper describes the most recent extension of the DROP project, which now supports advanced NFV paradigms and power management primitives by means of the GAL interface. DROP is conceived for aggregating and coordinating different building blocks like SDN and IT devices, acting at data- and control-plane, in a single logical IP node.

Its architecture acts as "glue" among different open-source networking tools, generally available in the Linux OS. Thanks to its software design, DROP allows seamless integration of any Linux native application for network control and administration (from the well-known Quagga and Xorp suites to simple command-line tools like "ip"). SDN devices have been integrated exploiting the open-source NOX controller. Forwarding elements can run both standard networking functionalities of the Linux kernel and new-generation user-space forwarding chains based on libraries like Netmap, the Intel DPDK or the Broadcom HyperExec.

Test experimental results attested the DROP scalability from both the data- and control-plane point of view. In addition, the power management capabilities guarantee energy proportionality without affecting network performance in a real-world emulation scenario.

REFERENCES

[1] D. Matsubara, *et al.* "Toward Future Networks: A Viewpoint from ITU-T," IEEE Communications Magazine, vol. 51, no. 3, pp. 112-118, Mar. 2013.

[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] S.H. Yeganeh, *et al.* "On Scalability of Software-Defined Networking," IEEE Communications Magazine, vol.51, no.2, pp.136-141, Feb. 2013.

[4] V.C. Ravikumar, R. N. Mahapatra. "TCAM Architecture for IP Lookup Using Prefix Properties," IEEE Micro, vol. 24, no. 2, pp. 60–69, March/April 2004.

[5] N. McKeown, *et al.* "OpenFlow: Enabling Innovation in Campus Networks", SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69-74, Mar. 2008.

[6] R. Bolla, *et al.* "Cutting the Energy Bills of Internet Service Providers and Telecoms through Power Management: an Impact Analysis ," Computer Networks (COMNET), Elsevier, vol. 56, no. 10, pp. 2320-2342, July 2012.

[7] 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/.

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

[9] L. Rizzo, "Netmap: a novel framework for fast packet I/O," Proc. of the 2012 USENIX Conf., June 2012.

[10] The NOX OpenFlow Controller. http://noxrepo.org/wp/

[11] http://www.broadcom.com/products/Processors/Data-Center/XLP800-Series.

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

[13] R. Bolla, *et. al.* "The Green Abstraction Layer: A Standard Power-Management Interface for Next-Generation Network Devices," IEEE Internet Computing vol. 17, no. 2, pp. 82-86, March-April 2013.

[14] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, "Green Networking with Packet Processing Engines: Modeling and Optimization," IEEE/ACM Transactions on Networking, to appear, available in pre-printing.

[15] R. Bolla, *et al.* "EE-DROP: An Energy-Aware Router Prototype," Proc. of TIWDC, Genoa, Italy Sept. 2013.