Green Extension of OpenFlow

Roberto Bruschi
CNIT – Research Unit of the University of Genoa
Genoa, Italy

Alfio Lombardo, Giacomo Morabito, Vincenzo Riccobene
CNIT – Research Unit of the University of Catania
Catania, Italy

Raffaele Bolla, Franco Davoli, Chiara Lombardo
CNIT – Research Unit of the University of Genoa
DITEN – University of Genoa
Genoa, Italy

Abstract— **Today's network architectures are not able to cope with the Future Internet requirements, as they are too inefficient, power hungry, and ossified on the TCP/IP paradigm. In order to promote a viable evolution towards new protocols and paradigms, modern network routers should become programmable to allow a flexible management of both traffic flows and power states. In this respect, Software Defined Networking (SDN) can represent the perfect support for the allocation of traffic flows and, at the same time, the management of the power states of the single nodes. To this purpose, this paper proposes to use the Green Abstraction Layer (GAL) to integrate the exchange of power management data among network elements and the application of control strategies inside the SDN paradigm. Moreover, in order to clarify how our study would help overcoming scalability issues and promote energy efficiency at the same time, this paper defines and solves a consolidation problem where the network is described in terms of allocation of packet handling resources and power states.**

Keywords—SDN, power management, network programmability

I. INTRODUCTION

The evolution of network and networked device architectures are two of the most important aspects of the Internet as we know it today, since they directly reflect the main open issues and the needs of current and upcoming network technologies and services.

The first aspect that is driving the evolution of the Future Internet is certainly the global increase of network traffic: in May 2013 [1], Cisco estimated that global IP traffic had increased more than fourfold in the previous 5 years, and would increase threefold over the next 5 years. Moreover, Internet video was estimated to account for over 50% of consumer Internet traffic by 2012. Since video traffic has strict Quality of Service (QoS) requirements, the service level must be sufficient to support such services.

The estimated increase in IP traffic has also drawn attention to the power consumed by the Information and Communications Technology (ICT) sector. The Global e-Sustainability Initiative (GeSI) [2] estimated an overall network energy requirement of about 21.4 TWh in 2010 for European Telcos, with a figure of 35.8 TWh in 2020 if no Green Network Technologies (GNTs) will be adopted. Considering that energy costs are raising quickly all over the world, the IP traffic increase trend would soon be unbearable if current network architectures and technologies were maintained.

Hence, re-thinking and re-designing the Internet becomes inevitable to keep the pace of any technological improvements. For this reason, many current and upcoming research efforts are emphasizing the weaknesses of the Internet infrastructure identifying the main issues and proposing novel solutions to guarantee a viable evolution of the Future Internet.

The most common criticisms regard the lack of flexibility of the currently deployed solutions: in fact, network infrastructures have not changed much throughout the years, being still based on the TCP/IP paradigm, and hence not providing an efficient degree of integration between network architectures and services. As a result, this architectural ossification has in many cases delayed the introduction of new technologies. In order to support novel solutions for the Future Internet development, programmability will play a key role.

In order to address the increasing need for flexibility of the Future Internet, Software Defined Networking (SDN) has recently emerged as a ground-breaking technology for realizing flexible network nodes able to customize their behavior according to the network and service needs. Although it represents a recent solution, SDN is rapidly evolving and gaining attention from both researchers and manufacturers. As stated in [3], SDN will be a $3.52 billion global market by 2018 thanks to the growth of cloud services and the need for mobility. Among all existing realizations of the SDN paradigm, OpenFlow (OF) [4] is probably the most popular one, reaching a point where it is often considered a synonym for SDN.

A simple schematization of an OF network can be seen in Figure 1. In an OF network, the Controller plays a very important role: it guarantees communication among hosts using a standard language (namely OF protocol), which implements a set of primitives to manage the flow table of each node in the network and to obtain a complete knowledge of the node itself.

We think that this architecture, which presents a hierarchical structure and a standard communication channel, would be the perfect support for introducing energy-aware

978-0-9836283-9-2

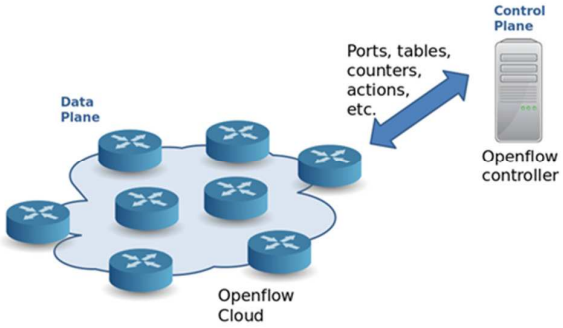

Figure 1. OpenFlow in its most frequent deployment.

network control policies. Such "green" enhancements would allow the allocation of traffic flows and the management of the internal power states of each single node at the same time: with this strategy, network consumption would be minimized and the correct bandwidth allocation for each service would be guaranteed. To the best of our knowledge, there is no technique which makes the controller aware of the power consumption of the internal hardware components of the switches: we candidate the Green Abstraction Layer (GAL), defined in [5], to this purpose. The GAL represents an internal communication interface/middleware to exchange power management data among packet processing elements, and apply control strategies in a simplified way. The GAL has been officially standardized in March 2014 as ETSI Standard 203-237 [6].

In this respect, the purpose of this paper is to integrate the GAL primitives inside the OF protocol to manage the network power configuration along with the traffic flows: the expected result is the dynamic and optimal configuration of the path between a source and a destination from the energy point of view. In order to show a quantitative evaluation of the gain achieved with this integration, we propose in the following an algorithm to solve this "*in-network consolidation*" problem.

The paper is organized as follows. Section II describes the main architectural features of today's network processors, while Section III presents the green capabilities introduced in OpenFlow. The in-network consolidation algorithm and the test results are in Sections IV and V, respectively. Finally, conclusions are drawn in Section VI.

II. Network Processor Architecture

Next-generation network processors need to move forward from the typical proprietary architectures to being more flexible and support the capabilities needed for developing a Future Internet network. As mentioned in the Introduction, such processors must be easily programmable and customizable, which is usually obtained through the adoption of a central intelligence and a number of function-specific hardware (HW) components with some level of "energy-awareness".

Today, one of the most popular trends consists in using multi-core architectures for network processors. The main reason is that multi-cores allow increasing network performance without using too high clock frequencies, which typically cause heat dissipation and data synchronization issues. In this respect, most solutions are derived from PC-based software routers (SRs) or Systems on a Chip (SoC) platforms. The former have represented a diffused trend in research for a long time. Their advantages are ease of programmability (especially in the presence of the Linux Operating System) and low costs. In addition, PC platforms already embed power management capabilities, usually controlled through the Advanced Configuration and Power Interface (ACPI) standard. The main criticism that has been moved to SRs in the past concerns their performance level, which has not been comparable to commercial routers for a long time. However, the introduction of specific HW able to improve the system efficiency, such as Network Interface Cards (NICs) that can guarantee some packet pre-processing, has boosted their potentials. SOC platforms represent a better performing and, of course, more expensive solution for the design of programmable network processors. They are characterized by a number of dedicated HW components to perform various kinds of operations outside the central cores, and also include various forms of power management capabilities.

Network processors, either PC- or SoC-based, present very similar architectures when it comes to the packet elaboration. Proprietary architectures typically work in the same way, but of course they do not provide so many degrees of freedom as programmable/configurable devices. A simple example of this architecture can be seen in Figure 2.

Packets are received by one or more NICs, which might provide some pre-processing operations, like packet classification, or such operations might be performed inside the system but in a similar manner.

In detail, packet classification is usually performed by a HW parser, which extracts pre-selected header fields from the incoming packets and matches the so composed key against the entries of a flow table. The efficiency of this building block is crucial to guarantee high-performing packet processing. For this reason, the memories used for these tables have become more and more complex to provide fast matching capabilities.

RAM memories, which represent the most diffused support for flow tables, are known to take a memory location as input and return the value contained at that address. In contrast, Content Addressable Memories (CAM) [7] work exactly in the opposite way, as they perform an exact match of data and return the address at which they are contained. This characteristic makes CAM a very interesting technology for building flow tables. However, since only exact matches can be performed on a CAM, it is really suitable only for flow matching, but not for prefix matching. A possible solution

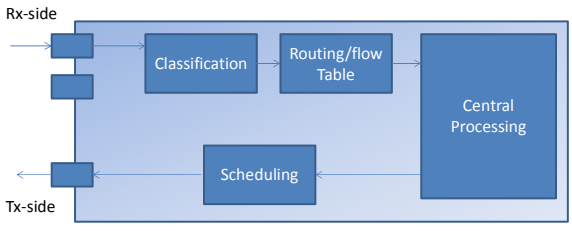

Figure 2. Network processor architecture.

could be that of "expanding" each prefix into all possible flows but that would quickly exhaust the memory space.

A more flexible type of CAM is represented by Ternary CAM (TCAM) [8]. Ternary means that, in addition to the binary "0" and "1" digits, there is also an "X" symbol representing a wildcard choice. The inclusion of wildcards allows non-exact matching, hence the possibility of prefix matching. The main drawback of TCAMs is represented by their high power consumptions: in fact, this parallel structure takes a lot of energy to power the match line, while wasting most of the energy on non-matches. The possibility of decomposing a single TCAM into logical sub-tables would allow independently managing their power configurations, for example by turning off the power supplied to the empty memory portions.

In Linux-based platforms, forwarding is usually performed at kernel level [9]. Although this strategy allows covering all functionalities needed in a forwarding engine, and the parallelization of the forwarding process among cores allows strongly improving performance, some recent solutions based on moving forwarding from kernel level to user-space level have provided massive performance improvements, at the price of a reduced number of network functionalities. Since the framework runs in user-space, the high overhead associated with kernel operations and with copying data between kernel and user space is removed. Moreover, significant time is also saved with further improvements, like disabling interrupts generated by packet I/O. In this respect, one of the most promising solutions is represented by the Intel Data Plane Development Kit (DPDK) [10].

At the transmission (Tx) side, simpler systems do not perform any further operations. If supported, a scheduling discipline allows processing traffic flows respecting their priority level, which means a higher bandwidth percentage is assigned to traffic with stricter QoS requirements.

The architecture of a network processor, with the "cohabitation" of separated reception (Rx) and Tx processing chains, maps perfectly with the OF protocol operations. Moreover, the logical distinction between data- and control-plane typical of such devices clearly corresponds to the centralized structure of OF. Finally, the way such centralization can be exploited for a more hierarchical purpose will be clarified in the next section.

III. The OpenFlow Protocol Extension towards Green Capabilities

OpenFlow specifies an interface, called OF Channel, used to connect each switch to a controller. The implementation of the channel can be vendor specific as long as the messages are compliant with the OF Protocol. Such messages can travel from the Controller to the network node, when information is required, or vice versa. In this respect, our aim would be to exploit the OF Channel characteristics to integrate its set of messages with some primitives concerning the power state of the network devices inside the network, like available power configurations and relative behavior (e.g., power consumed by the device in that particular state), or current power state. As mentioned in the Introduction, these communication primitives are already specified by the Green Abstraction Layer.

The GAL allows exchanging power management data and applying control strategies in a simplified way. Its application is driven by two main capabilities. On one hand, it hides the implementation details of energy-saving approaches. On the other hand, it offers a standard interface to guarantee interactions between heterogeneous green capabilities and HW technologies, as well as between control and monitoring frameworks. The power management settings are driven by the Local Control Policies (LCPs), control processes developed to optimize the router configuration to achieve the desired trade-off between energy consumption and network performance according to the incoming traffic load. The GAL has a hierarchical structure, where multiple instances are used for managing the exchange of information between the LCPs of different levels and between LCPs and hardware components.

By means of the GAL, control applications are allowed to get information on how many power management settings are available at the data-plane, and on the potential effect of using such settings. The other way around, control applications are capable of setting a certain power management configuration to the device data-plane. It is worth noting that, in order to provide the information needed by control applications to reduce the energy consumption while meeting QoS constraints, the GAL must explicitly represent the impact on network performance when different power management settings are applied.

An OF switch can be represented, from a network point of view, by a set of ports (Physical, Logical and Reserved), a set of actions that can be performed on traffic flows (the list of actions of a switch depends on the specific hardware) and one or more flow tables. Given that the GAL considers each device as a set of logical entities, an OpenFlow version of the GAL can consider as logical entities the ports, the actions and the tables of all the OF switches. Since there is no constraint about how those resources have to be implemented inside the device, manufacturers can use their own technologies and solutions and thanks to the GAL each resource can be defined in terms of power consumption with respect to the offered load. In this way, all the entities of the network can be completely described, from the energy perspective, by a set of Energy Aware States (EASs), each of them characterized by a set of parameters including a maximum supported data rate and the corresponding power consumption.

The extension of the OpenFlow protocol to make the network efficient using the GAL requires both the definition of new OpenFlow messages and the design of a new module inside the latter.

As far as the new message definition is concerned, the network nodes have to send the description of their own logical resources and related EAS to the controller. So, we have defined a set of new messages to implement the main GAL primitives: *discovery*, *provisioning* and *monitoring*.

The *discovery* primitive allows the Controller to get information about the power consumption of each entity. The related packet from the network node to the Controller contains

the complete description of the EAS: an identifier, the power gain with respect to the maximum power consumption and the throughput in terms of packets per second and bits per second. In order to completely support the GAL, the message also contains wake-up time, sleep time and transition power.

The *provisioning* primitive allows the Controller to set the state of a network node. The related packet from the Controller to the network node contains the logical resource identifier and the EAS identifier.

The *monitor state* and *monitor consumption* primitives allow the Controller to get information about the current EAS set on a logical entity and the current power consumption of the entity, respectively. The related packets from the switch to the Controller contain the identifier of the entity, along with the identifier of the current state in the first case, and the power consumption of the entity in the second case.

In addition, we propose a new Controller module called Energy Optimizer: it is a possible demonstration of how the controller can exploit the information provided by the GAL to reduce the power consumption of the network: the Controller gathers information about the EASs, and the Energy Optimizer module calculates a nearly optimal configuration to process a given traffic demand consuming the minimum energy.

IV. In-Network Consolidation

In the previous sections the advantages that can be obtained through the introduction of power management capabilities inside the OF protocol, in terms of energy efficiency and resources management, have already been roughly sketched. Now we want to move a step forward and provide an example to clarify how our study would help overcoming scalability issues and promote energy efficiency at the same time. To do this, we can start by describing the network management as a consolidation problem.

The term consolidation describes the optimal allocation of functionalities/resources to guarantee a certain outcome. The concept of consolidation usually refers to the Virtual Machines (VMs) allocation in a variable number of physical servers inside a datacenter. Since we aim at extending the same concept to the optimal allocation of network resources, in the following the tackled problem will be called in-network consolidation.

A. The In-Network Consolidation Algorithm

This section presents an algorithm to calculate the solution of the "in-network consolidation" problem. We consider a network, as represented in Figure 3, composed by R routers connected by L links. Each router, aside from forwarding, can also perform A possible off-loading operations on traffic, or can be turned off when it is not required, like the bottom left one in the figure.

The traffic demand is represented by a set of F flows, with the generic flow f defined by:

- a source s_f and a destination d_f
- a load L_f
- a subset of the available A actions required by the flow.

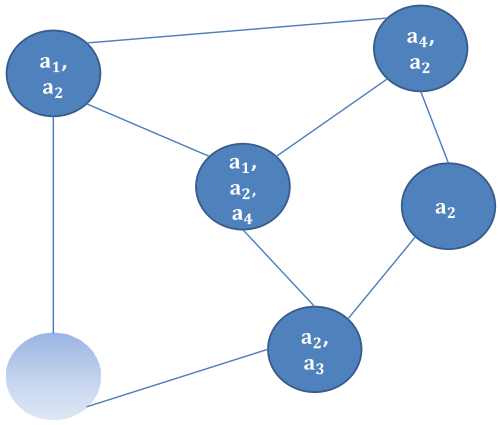

Figure 3. Representation of a network with sleeping capabilities.

Here, we formally define a problem that takes the topology, the logical resource and the traffic demands as input, and finds the flows and actions configuration that satisfies the traffic demand and minimizes the power consumption of the entire network. Table I contains the description of the variables and parameters that are used for the problem formulation.

TABLE I. Notations Defiintion

u_{lf}	binary variable that equals 1 if the link l is crossed by the flow f
v_{rfa}	binary variable that equals 1 if the router r crossed by the flow f implements the action a
x_l	binary variable that equals 1 if the link l is active
y_{ra}	binary variable that equals 1 if the router r enables the action a
z_r	binary variable that equals 1 if the router r is active
R	total number of available routers
L	total number of available links
A	total number of available actions
F	total number of flows
T_r	activation cost of the router r
U_l	activation cost of the link l
V_{ra}	activation cost of the action a on the router r
L_f	load of the flow f
M_l	capacity of the link l
w_{fa}	binary constant that equals 1 if the flow f requires the action a
h_{rl}	binary constants that equal 1 if the link l enters the router r
k_{rl}	binary constants that equal 1 if the link l exits the router r

$$\min_{u,v,x,y,z} \left(\sum_{r=1}^{R}\sum_{l=1}^{L}\sum_{a=1}^{A} T_r + U_l x_l + V_{ra} y_{ra}\right) \quad (1)$$

$$y_{ra} \geq \frac{1}{F}\sum_{f=1}^{F} v_{rfa}\,, \quad \forall r, a \quad (2)$$

$$\sum_{r=1}^{R} v_{rfa} = w_{fa}, \forall f, a \quad (3)$$

$$\sum_{f=1}^{F} L_f u_{lf} \leq M_l, \forall l \quad (4)$$

$$\sum_{l=1}^{L} h_{lr} u_{lf} - \sum_{l=1}^{L} k_{lr} u_{lf} = 0, r \neq s_f, d_f, \ , \forall f \quad (5)$$

$$\sum_{l=1}^{L} h_{lr} u_{lf} - \sum_{l=1}^{L} k_{lr} u_{lf} = 1, r = \ s_f, \ , \forall f \quad (6)$$

$$\sum_{l=1}^{L} h_{lr} u_{lf} - \sum_{l=1}^{L} k_{lr} u_{lf} = -1, r = \ d_f, \ , \forall f \quad (7)$$

$$z_r \geq \sum_{l=1}^{L} (h_{lr} + k_{lr)} \, x_l, \ \forall r \quad (8)$$

$$v_{rfa} \leq \sum_{l=1}^{L} (h_{lr} + k_{lr)} \, u_{lf}, \forall r, f, a \quad (9)$$

$$x_l \geq \frac{1}{F} \sum_{f=1}^{F} u_{lf}, \forall l \quad (10)$$

$$u_{lf}, v_{rfa}, x_l, y_{ra}, z_r \in [0,1] \quad (11)$$

Equation 1 represents the network cost and is composed by the sum of all routers, links and enabled actions. Equation 2 states that the action *a* has been enabled on the router *r* for all the flows requiring it, while Equation 3 guarantees that, for a given flow *f*, each one of its actions is performed only once, and Equation 4 allows a flow to pass through a link only if it has enough available capacity to handle it. Equations 5-7 represent the flow conservation constraints. According to Equation 8, if a router is off, so must be all links crossing it, while Eq. 9 allows an action to be implemented on a router only if there is a flow crossing it and requiring that action. Equation 10 makes sure that a switched off link is not crossed by any flow. Finally, Equation 11 constraints the values that can be assumed by the problem variables.

V. Test Results

The problem presented in the previous section has been applied to the simple topology shown in Figure 4. This simple network is composed of five routers connected by twelve mono-directional links. Four actions are made available on the network and can be activated according to the demands. In this first case, the cost is 5 for each link and for each action, and equals 20 for each router. The capacity of each link corresponds to 1.

The traffic demand that has been chosen for this example consists of three flows, which have the characteristics reported in Table II:

TABLE II. Notations Definition

Flow	Load	Source	Dest	Actions
f_1	0.1	1	5	1, 3, 4
f_2	0.1	2	3	2, 3
f_3	0.1	1	3	2, 3, 4

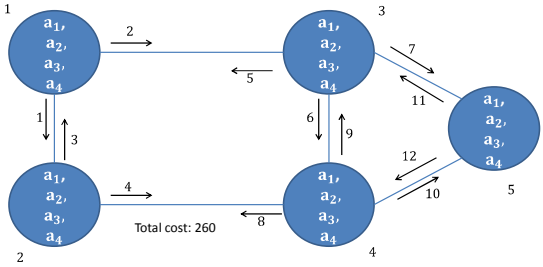

Figure 4. Topology of the network used for testing the optimization problem.

Figure 5 shows the flows and actions allocation obtained through the algorithm presented in the previous section. The optimization strategy allows managing all flows using only 3 of the available 12 links, and also switching off Router 4. Moreover, each action is enabled only once on the network and serves all flows requiring it. As a result, the total cost of the network is around 56% lower in the optimized configuration.

In the next example, presented in Figure 6, we suppose to increase the load of f_3 from 0.1 to 1. This new parameter brings to a stress situation in the network configuration: in fact, the load of f_3 now corresponds to the link capacity, hence no other flow can be transmitted along with f_3. For this reason, the flows allocation requires the activation of the sleeping router and of

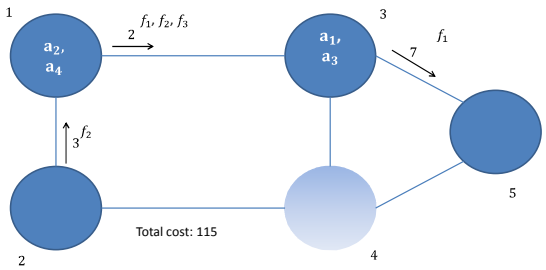

Figure 5. Optimized flows and actions allocation.

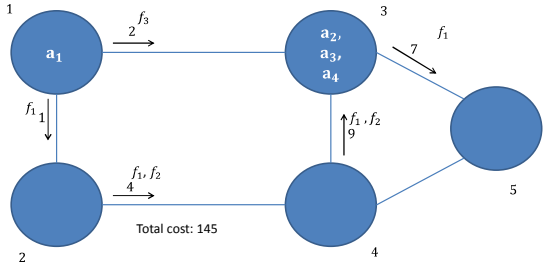

Figure 6. Flows and actions allocation in the presence of an increased flow load.

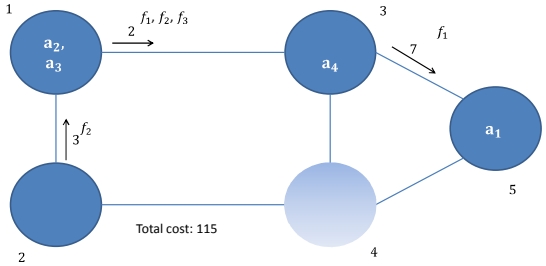

Figure 7. Flows and actions allocation in presence of an increased action cost.

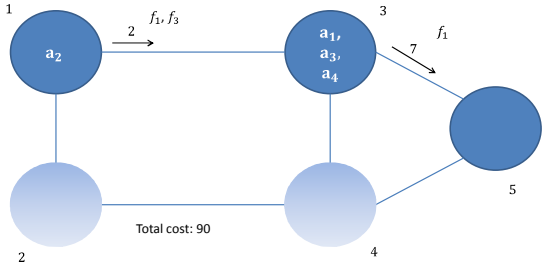

Figure 8. Flows and actions allocation according to a new traffic demand.

two more links, which brings to a cost increase but still guarantees the service availability for all flows.

Figure 7 represents a situation in which the activation of Action 1 on Router 3 has a higher cost. In this case, it is possible to activate this action on another router (Router 5) and maintain the network cost constant.

Finally, in the example depicted in Figure 8, f_2 has been removed from the traffic demand. This allows switching off a link and another router to increase the overall energy saving.

VI. Conclusions

The evolution of the Future Internet in terms of number of customers and offered services will be feasible only if modern network architectures and devices will be more programmable, to be able to follow new paradigms and protocols. In this respect, Software Defined Networking (SDN), especially in the OpenFlow declination, has recently emerged as a ground-breaking technology for realizing flexible network nodes able to customize their behavior according to the network and service needs. This paper introduces an extension of the OpenFlow protocol to include and deliver power management primitives among the network nodes. For this purpose, we have decided to use the Green Abstraction Layer (GAL), recently standardized by ETSI. In detail, we have integrated the GAL primitives inside the OpenFlow (OF) protocol to make the OF Controller aware of the power consumption of the internal hardware components of each network node. In this way, as shown in the test results, it is possible to allocate resources according to the incoming traffic characteristics while reducing the overall network energy consumption.

The proposed in-network consolidation algorithm aims to clarify how our study would help overcoming scalability issues and promote energy efficiency at the same time. Future improvements on this study will focus on a heuristic approach to further improve the algorithm efficiency in larger networks.

Acknowledgment

This work was supported by the ECONET integrated project, funded by the European Commission under the 7th Framework Programme (FP7), theme ICT call 5, grant agreement no. 258454, and by the GreenNet project, funded by the Italian Ministry of University and Education under the program FIRB “Futuro in Ricerca”.

References

[1] Cisco, “Cisco Visual Networking Index: Forecast and Methodology, 2012-2017,” White paper, May 2013, URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html

[2] Global e-Sustainability Initiative (GeSI), SMARTer2020: The Role of ICT in Driving a Sustainable Future, Report. http://gesi.org/SMARTer2020.

[3] http://community.comsoc.org/blogs/alanweissberger/sdn-be-352-billion-market-2018-cyans-blue-orbit-carrier-based-sdn-and-nfv-appl

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, “OpenFlow: Enabling Innovation in Campus Networks”, ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69-74, March 2008.

[5] 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, 2013.

[6] ETSI Standard 203 237, “Green Abstraction Layer (GAL): Power management capabilities of the future energy telecommunication fixed network nodes,” version 1.1.1, March 2013, http://www.etsi.org/deliver/etsi_es/203200_203299/203237/01.01.01_60/es_203237v010101p.pdf.

[7] K. Pagiamtzis, A. Sheikholeslami, “Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey,” IEEE Journal of Solid-State Circuits, vol. 41, no. 3, pp. 712–727, 2006.

[8] V.C. Ravikumar, R. N. Mahapatra. ”TCAM Architecture for IP Lookup Using Prefix Properties”, IEEE Micro, vol. 24, no. 2, pp. 6069, March/April 2004.

[9] K. Wehrle, F. Phlke, H. Ritter, D. Mller, M. Bechler, ”The Linux Networking Architecture: Design and Implementation of Network Protocols in the Linux Kernel”, Pearson Prentice Hall, Upper Saddle River, NJ , USA, 2004.

[10] The Intel “Data-Plane Development Kit,” URL:http://www.dpdk.org.