GreenSDN: Bringing Energy Efficiency to an SDN Emulation Environment

Bruno B. Rodrigues*, Ana C. Riekstin†,
Guilherme C. Januário†, Viviane T. Nascimento*
and Tereza C. M. B. Carvalho†
University of São Paulo
São Paulo, Brazil
Email: †{carolina.riekstin, gcjanuario, tereza.carvalho}@usp.br,
*{brodrigues, vianetn}@larc.usp.br

Catalin Meirosu
Ericsson Research
Stockholm, Sweden
Email: catalin.meirosu@ericsson.com

***Abstract*—A significant number of green, energy-saving network protocols have been invented in recent years in response to demand for reducing the amount of energy consumed by network infrastructure. In this paper, we report on the difficulties we encountered when building an SDN environment that could emulate energy saving protocols operating at different layers of the network. We propose solutions, based on the Mininet environment and the POX Openflow controller, that emulate the effects of three different energy saving protocols. Our approach is validated by comparing energy savings obtained by activating these protocols in an emulated network topology inspired by the Brazilian Research Network.**

I. Introduction

Datacenter and telecom service providers are struggling with the growing energy spending and associated growth of operational costs and greenhouse gases (GHG) emissions. The annual/yearly ICT energy consumption is forecasted to increase nearly 60% until 2020, reaching almost 1,100 TWh [1]. Networking is a considerable source of energy demand both in datacenters, where it accounts for 10 to 20% of the energy consumption, and operators, to whom networking accounted for more than 260 TWh in 2012 [2]. To reduce the power demand from networking, several energy efficiency capabilities (also called *green capabilities*) have been proposed, ranging from chip level capabilities to network management tools. However, environments to emulate these energy efficiency solutions have not been discussed as much as the green capabilities. Building an energy efficiency emulation environment using on IETF MIB models and SNMP as the management protocol was discussed in [3]. Nevertheless, that solution does not address challenges specific to Openflow-based Software Defined Networks (SDN), such as how to include autonomic green capabilities in the dataplane that are not aware of flow descriptors.

This paper proposes GreenSDN, an SDN emulation environment based on the Mininet and the POX controller, providing also the necessary extensions to emulate energy efficiency capabilities. To the best of authors knowledge it is the first work to provide an SDN environment easy to deploy and replicate in order to emulate green capabilities. The remainder of this work is organized as follows: Section II reviews the selected state of the art of green capabilities, as well as the Mininet network emulator and the monitoring extension. Section III details how we implemented the selected green capabilities and extended the POX controller. Section IV describes the system architecture that assembles together green capabilities emulated at dataplane and network levels. Section V presents results from the emulation environment, validating our method and technical choices. Finally, Section VI presents our conclusion and future works.

II. State of the Art

In this section we review the selected state of the art in energy efficiency capabilities, emulation tools and OpenFlow-based network monitoring. We describe the functioning and details about green capabilities and extensions required to build GreenSDN.

A. Energy Efficiency Capabilities

Several energy efficiency capabilities have been developed focusing on the increase of the energy efficiency of network infrastructures. These capabilities are usually categorized in accordance with the scope in which they are applied on the network infrastructure. For instance Adaptive Link Rate (ALR) [4] and Advanced Configuration and Power Interface (ACPI) [5] are examples of capabilities applied on specific chip-level components of network devices. Synchronized Coalescing (SC) [6] and IEEE 802.3az [7] work at the node-level. Other capabilities are employed at the network level, such as Green Traffic Engineering (GreenTE) [8], SustNMS [9] and ElasticTree [10]. Only ElasticTree was developed for SDNs, but we understood that the implementation is proprietary.

Considering the several capabilities available in the different network levels, we selected three green capabilities to be implemented in GreenSDN. Each of them is representative for one of the following network layers: ALR (transceiver chip-level - Layer 1), SC (node-level - Layer 2) and SustNMS (network-level - Layer 3). The literature contains good technical descriptions of ALR and SC to help us with the implementation, while SustNMS is a previous work performed in the research group so the source code was made available to us.

ALR (Transceiver chip-level - Layer 1) is an transceiver-level capability that copes with the underutilization and over-provisioning of Ethernet links by dynamically changing data

rates in response to traffic levels [4]. It is designed to modulate the capacity of network interfaces by scaling up or down existing Ethernet data rates. It consists of a mechanism and a policy. The mechanism determines how the data rate is changed through a link negotiation. The policy determines when to change the data rate, aiming to maximize the time spent in a low data rate and saving energy without packet losses [4]. For instance, if a low traffic level is detected, a low data rate should be used. For high traffic levels, a high link data rate is necessary.

SC (Node-level - Layer 2) is a capability based on the Low Power Idle (LPI) mode for Ethernet links (defined by IEEE 802.3az) [7]. LPI is a mode used for reducing the energy consumption of interfaces in a switch or router when no data is being transmitted. SC uses a mechanism to orchestrate LPI modes of all individual interfaces, coalescing the packets such that an entire switch may be put into LPI mode. The capability improves the efficiency of IEEE 802.3az by coalescing outgoing packets into bursts, making the number of transitions between LPI and active modes decrease [6].

SustNMS (Network-level - Layer 3) has the objective to allow operators to strike a balance between the assurance of QoS and green traffic engineering (TE) and ensures fast response to either failure or sudden traffic increase [9]. It operates through high-level policies, requiring a monitoring system to be aware of the bandwidth consumption, and the definition by a network operator of the set of paths to be used. SustNMS can be active in two different ways depending on the bandwidth usage, a sustainability mode (SustNMS-S) maximizing the number of nodes in sleep mode by concentrating traffic, and a performance mode (SustNMS-P), which routes the network prioritizing performance.

B. Mininet and POX Controller

Since the growth of virtualization in network infrastructures, there are efforts to emulate or simulate programmable networks in order to provide environments supporting realistic user traffic, at scale, and with interactive behavior (e.g. ns [11], Emulab [12], GENI [13], Mininet [14]). Among the available solutions to emulate a programmable OpenFlow network, Mininet combines the desirable features of simulators, testbeds and emulators, being considered the most popular and easier to use due to its capability to execute locally into a virtual machine, allowing also faster implementations [14]. It is readily available as open source and experiments replication is one of its main strengths. Mininet includes data plane switching functionality from Open vSwitch (OVS) [15]. However, for a research project, the Open vSwitch code is fairly complex and therefore difficult to modify. Instead, we opted for implementing the interface and node-level green capabilities emulation on the SDN controller, details are given in Section III.

While Mininet is the most popular SDN emulation tool, POX [16] is one of the most popular OpenFlow Controller. It is a python-based controller which has gained popularity due to its easiness of use and available documentation. Its open source code makes it easier to modify and add features.

C. OpenFlow Network Monitoring

A key requirement in order to satisfy QoS (Quality of Service), efficient TE (traffic engineering) and the measurement of energy consumption is accurate network monitoring [17]. Considering a controller running green capabilities, such requirement gains importance. Network-level capabilities such as SustNMS imposes a need to evaluate constantly throughput in order to adapt the network to a load proportional usage. For this, the OpenFlow protocol provides a good support providing the required information through the messages $OFPT\_FEATURES\_REQUEST$ and $OFPT\_FEATURES\_REPLY$. With these messages is possible to query counters about ports, flows and flow-tables [18]. For instance, the bandwidth of a switch is calculated as the sum of the received bytes (rx_bytes) counter of its ports, and by extension, we can measure the bandwidth of a path as an average of the rx_bytes of flows installed in switches of the path. Therefore, we can update SustNMS with the current workload in a switch or path.

III. GreenSDN Implementation Details

We developed GreenSDN considering the following process: implement a mechanism to collect traffic statistics; implement the power profiles taking into account traffic statistics to emulate switch energy consumption; and implement the energy efficiency capabilities.

A. Monitoring Traffic Statistics

To query nodes information regarding flows and ports usage we use the requests $OFPC\_FLOW\_STATS$ and $OFPC\_PORT\_STATS$ that are used as the body of the message $OFPT\_FEATURES\_REQUEST$. Each features request message is requested as a polling and is composed either by a flow or port stats request, generating in response a $OFPT\_FEATURES\_REQUEST$ message. The response contains flows or port statistics provided by the OpenFlow protocol being handled in separated methods. However, a straight polling of all nodes, although precise, has the potential to generate large amounts of control traffic and consequently to increase overall network energy consumption. We address this problem in a similar way as [17] and [19], by adapting our polling. Initially, to detect incoming workload we query the edge-nodes, which are connected to the hosts. Once a workload is detected, nodes through the used paths are also queried. The paths are obtained either from an initial proactive flow instantiation or from the SustNMS output to change a path, in case it is active.

B. Power Profiles Implementation

As stated in [3] in a sustainable-oriented environment, routers or switches must support different power profiles and provide energy consumption information [3]. Virtual switches such as OVS have no capabilities to provide power consumption information neither at the port level, nor at the overall process level. As the SDN controller has an inventory of all switches in the network, we defined power management application comprising power profiles as a way to parameterize energy consumption in accordance with the bandwidth utilization. Two types of power profiles were defined to simulate

real equipment: a load proportional, that is more energy efficient and desirable, and a linear, most common in legacy equipment, with a constant energy consumption independent of the workload. Load proportional power profiles have a fixed and a variable part as described in [20].

We used one of the load proportional power profiles from [3] for switches powered on and sleeping, as described in Equations (1) and (2). The power profile for powered on switches was combined with extra savings specific from the link rate reduction. In this case, we considered measurements made by [21], and reduced 15% when ALR is active as described in Equation (3). For SC, we combined the power profile with the time the SC is on or off as described by [6] in Equation (4). Despite considering a homogeneous network using the specific equations described herein, the environment could be easily adapted to use any other power profile or a combination of power profiles in different switches.

$$PP = 200 + \left(\frac{500}{30}\right) * workload \quad (1)$$

$$PP_{sleeping} = 120 \quad (2)$$

$$PP_{ALR} = 200 + \left(\frac{500}{30}\right) * workload - 15\% * ALR \quad (3)$$

$$PP_{SC} = PP_{sleeping} * tOff + PP * tOn \quad (4)$$

C. Implementing Green Capabilities

As mentioned in Section II-A, three capabilities were selected to validate GreenSDN: SustNMS, SC and ALR. This subsection provides details on how the capabilities were implemented in the GreenSDN environment.

1) ALR implementation strategy: The ALR capability uses a policy and a mechanism to adjust the link rate. As we use OVS and do not modify its behavior, and neither OVS or OpenFlow provide support for changing link rates as proposed originally by [4], we emulate ALR as detailed in Figure 1.

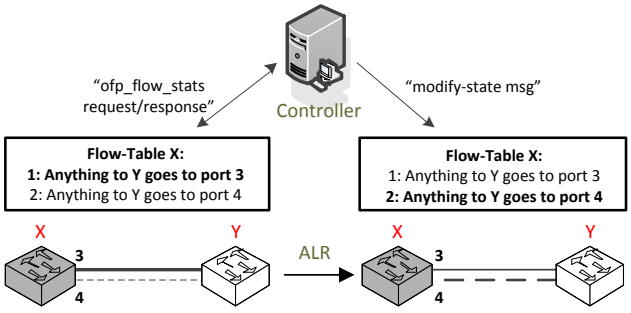

Fig. 1. ALR Emulation Scheme.

In Mininet, the data rate of each link is enforced by Linux Traffic Control, which has a number of schedulers to shape the traffic to a configured rate [22]. Based on this, we defined parallel links between each pair of nodes to emulate the ALR mechanism. A standard link, represented in the Figure 1 as the continuous line, was configured with a 30 Mbps rate limit, and a parallel link represented by the dotted line, configured with 10 Mbps. Only one of these links forwards traffic at a given point in time. The ALR policies were implemented inside the controller, which receives requests from applications to enable/disable ALR and determine the best moment when to reduce or increase the link rate. Algorithm 1 presents the implementation.

Algorithm 1: ALR Implementation.

```
If (ALR == ON) do:
    For each port ∈ switch do:
        If (port is not attached to host) do:
            out_port ← out_port + 1
            OFmsg ← odd_port_to_30Mbps
            sendOpenFlowMsg()
Else:
    For each port ∈ switch do:
        If (port is not attached to host) do:
            out_port ← out_port − 1
            OFmsg ← even_port_to_10Mbps
            sendOpenFlowMsg()
```

To make the traffic engineering easier, the standard path was configured to use the odd ports of the switch, while the alternative path was configured to use the even ports. Every time ALR is enabled, our application increments the out_port on all the rules associated with the switch. To disable ALR, the same process is executed, but decrementing out_port, so that the traffic is forwarded to the normal path.

2) SC implementation strategy: There is no way to implement SC without altering OVS functioning, since it does not natively support traffic bursts. An alternative would be to send packets from data level to the controller. The controller would simulate the buffer/queueing functionality, but this is not feasible since the controller cannot handle all the dataplane traffic. The approach we took was to emulate the energy-related effects of SC as an application on the controller based on information from the power profiles, as depicted in Figure 2.

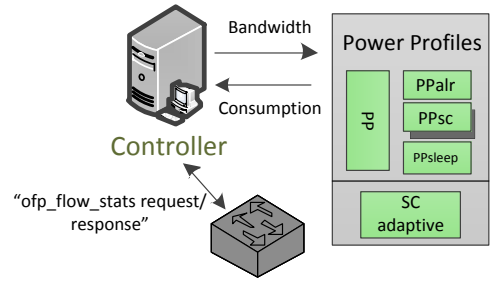

Fig. 2. SC Emulation Scheme.

The adaptive part of SC is implemented as described in the Algorithm 2. It has the objective to check if the gathered number of packets per second, is higher than a pre-determined buffer. If higher, the capability is disabled and the switch enters in a standard mode of operation in order to handle the workload without losing packets. While the number of packets per second still lower than the buffer, incoming packets are coalesced maximizing savings.

3) SustNMS implementation strategy: SustNMS demands the identification of predefined tunnels in order to be efficient.

Algorithm 2: SC Implementation.

$DutyCycle, buffer \leftarrow parameters$
For each $switch \in topology$ do:
  If $(SC == ON)$ do:
    $packets/second \leftarrow port\_stats$
    If $(packets/second > buffer)$ do:
      $SC \leftarrow OFF$
      $return$
    $Watts \leftarrow (tOn/DutyCycle - tOn)$
    $Losses \leftarrow packets/second - buffer * tOff$
  Else:
    calculate($WattsSC - ON$)
    $Losses \leftarrow 0$

The idea of the algorithm is to perform the green traffic engineering considering all flows being executed in a moment T, paths to be used, and the set of switches to be in sleep mode based on a set of predefined tunnels. The controller side of SustNMS identify the flows being executed using the OpenFlow messages, as described in Section II-B. As a previous work of our research group, the algorithm implemented in GreenSDN is similar to the one presented in [3]. Modifications were made to adjust the predefined set of tunnels and switches to be in sleep state in accordance to the topology used in the validation (Section V).

IV. Architecture

The system architecture is depicted in Figure 3. It is composed by the controller application and adjacent worker modules to support applications and the green capabilities. The controller uses the following modules:

- **Topology Manager**: deals with the network management operating switches and ports. It implements methods to manage flow-tables, installing or removing flows whenever SustNMS requests a new path, also defining switches and port states in order to measure energy consumption through the Power Manager;
- **QoS Services**: is responsible for the monitoring task, handling the response of flow and port status events. It updates the controller about active paths, providing bandwidth and losses;
- **Power Manager**: is used to implement the ALR emulation mechanism, the adaptive part of SC and the power profiles. In conjunction with the Topology Manager, it is able to modify flow-tables;
- **Graphical User Interface (GUI)**: gathers information from the controller and displays information about the topology, overall network consumption, packet loss and green capabilities applied;
- **Green Application Control**: implemented as a button, is used to enable or disable green capabilities. Also has part of its implementation in the Controller in order to apply the selected capability;
- **Controller**: the main component in the architecture, manages the exchange of information among modules, applications and capabilities.

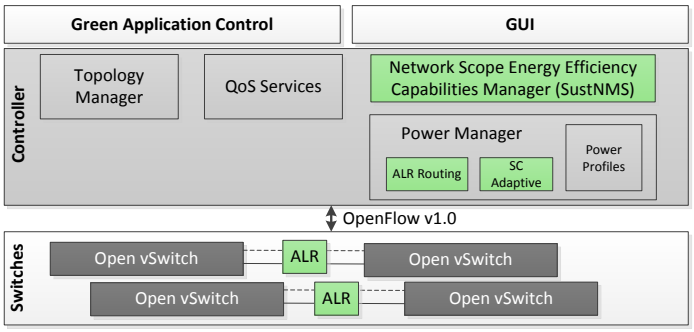

Fig. 3. Emulation Environment Architecture.

Algorithm 3 describes the Controller functionality in order to apply green capabilities that was requested by the Green Application Control, considering also a topology created beforehand. First, the controller creates instances of components to operate the network using their methods. Topology and flows are obtained via XML (eXtensible Markup Language), the same file used to create the topology in Mininet and check the edge-nodes. After setting instances and flow-tables, the controller start to monitor the network.

Algorithm 3: POX Controller Implementation.

$tm \leftarrow topoManager.createTM()$
$QoS \leftarrow qos.createQosServices()$
$pm \leftarrow PowerManager.createPM()$
$activeFlows \leftarrow readXML(\text{“}proactiveFlows.xml\text{”})$
$tm.l2mSpanningTree(\text{“}topo.xml\text{”})$
$tm.installFlows(activeFlows, edgeNodes)$
$QoS.netMonitor(edgeNodes, activeFlows)$
While $True$ do:
  $activeFlows \leftarrow QoS.checkActiveFlows()$
  For each $path \in activeFlows$ do:
    $mpbs, loss \leftarrow QoS.info(path)$
    $funct \leftarrow getCapabilityApp()$
    If $funct = SustNMS$ do:
      $flows[] \leftarrow SustNMS(path, mbps)$
      $activeFlows \leftarrow tm.setFlows(flows)$
    Else If $funct = ALR$ do:
      For each $switch \in path$ do:
        $pm.enableALR(switch)$
      End for
    Else If $funct = SC$ do:
      For each $switch \in path$ do:
        $pm.enableSC(switch)$
      End for
    Else If $funct = None$ do:
      $pm.disableSC(path)$
      $pm.disableALR(path)$
    End If
  End For
  $consumption, loss \leftarrow QoS.networkInfo()$
  $GUI \leftarrow consumption, loss$
End While

The *While* loop in Algorithm 3 represents the Controller side of the Green Application Control, receiving capabilities to be applied. In our case, the return of ”getCapabilityApp” is *None* case the network is already running efficiently, or a *string* to apply determined capability, otherwise each capability is

treated separately by the Controller, adjusting the network and updating the power profiles. The Power Manager component is used to emulate the nodes consumption due the OVS restriction, as described in Section III-B.

V. Validation

In this section we present the topology and an evaluation to check if the capabilities meet our expectations based on previous works on SustNMS [3], ALR [4] and SC [6]. Since capabilities configuration relies on specific requirements of each network, we aim to validate that GreenSDN is capable of emulate energy efficiency capabilities. We used as the host machine an Intel Core(TM) i5-3570 @ 3.40GHz with 8 GB RAM, and, aiming to use the processor at its full capacity, we disabled the processor low power mode (C-States) in the BIOS (Basic Input/Output System).

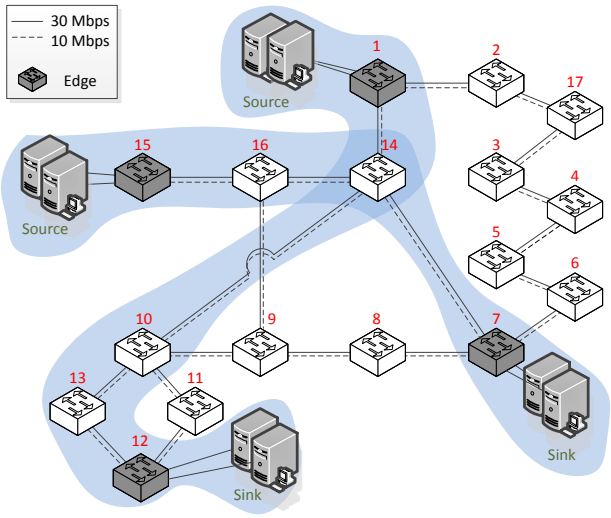

Fig. 4. Topology inspired by the RNP.

The topology in GreenSDN is inspired on the 10 Gigabyte RNP (*Rede Nacional de Ensino e Pesquisa*)[1] backbone. It was reproduced in GreenSDN using seventeen emulated nodes running OVS in kernel mode to switch packets across the interfaces, the Figure 4 present the topoly. Each pair of nodes is interconnected using parallel links, which are configured with different rate limits. In order to send data across the network, GreenSDN considers two main flows, from north to south and west to east, placing two *Sources* in north extremes and two *Sinks* in south extremes. The generation of traffic between the hosts is in charge of the Iperf tool, which is already available in Mininet.

Once paths are determined and considering that most of north-bound applications tend to impose excessive traffic control overhead, flows are installed proactively in flow-tables when the controller starts. By pre-defining flows and actions ahead of time in flow-tables, $Packet - In$ messages never occurs. As result every incoming packet is forwarded through a simple lookup in flow-table, minimizing the latency induced by $Packet - In$ messages. Reactive modifications in flow-tables are performed in accordance of applications requirements.

[1]The Brazilian National Education and Research Network

Considering this, a Control Green Application is implemented to simulate requests to apply green capabilities. It consists of two parts, a simple GUI with buttons to request the execution of capabilities, and the Controller side as observed in Algorithm 3, to apply the selected capability. Also, we kept the same configuration of power profiles and capabilities as a way to analyze the capabilities behavior based on a specific network configuration. Results of the 10 and 30 Mbps workloads are depicted in Figure 5.

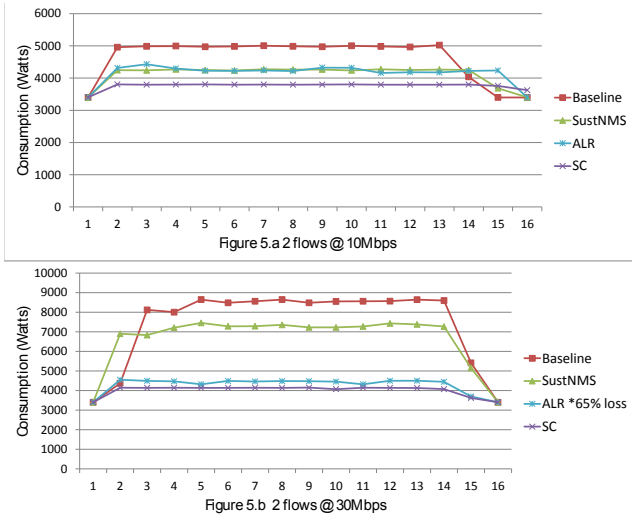

Fig. 5. Energy Savings Evaluation considering 2 Flows.

The baseline consumption represent the ordinary network operation, with all switches operating using the regular power profiles. The 10 Mbps evaluation, Figure 5.A, present savings of ALR and SustNMS were similar, with both presenting savings around 15% with a small difference. However, this scenario is different in accordance with another configuration, as observed in Figure 5.B.

Aware of links capacity, SustNMS modifies the routing for workloads bigger than 15 Mbps in order to avoid consequently losses by flows sharing switches, such as switch 14 of Figure 4. For workloads smaller than 15 Mbps, in our case the 10 Mbps evaluation, SustNMS maximized the savings concentrating traffic on the defined flows. Considering that ALR is intended for Ethernet speeds and we are working with 30 Mbps link capacity, it worked as expected. ALR presented savings in 10 Mbps and huge packet losses in 30 Mbps evaluation. The SC capability was the most aggressive functionality in terms of savings, in our evaluation it was configured with a *DutyCycle* of 50%, meaning that switches were 50% of time on and off, also the packet threshold (1000 packets/second) and buffer size (80 packets). With this configuration SC did not present losses in the evaluations. The GreenSDN GUI with the configured paths highlighted is depicted in Figure 6.

VI. Final Considerations

In this paper, we presented GreenSDN, an emulation environment for network energy efficiency capabilities. The system comprises three green capabilities and the necessary extensions to the controller in order to collect and provide the related network information, as well as apply the green

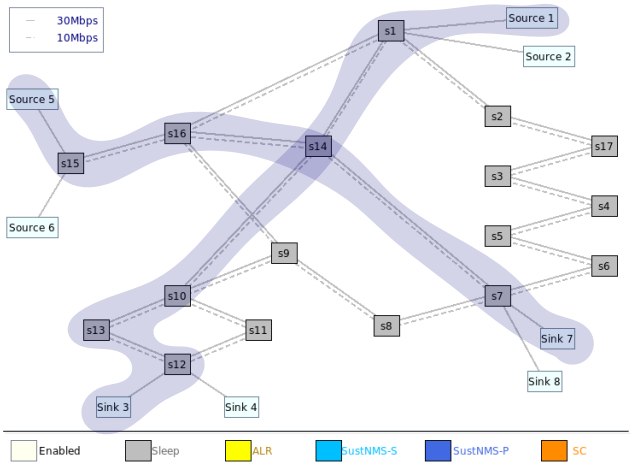

Fig. 6. RNP Topology in the GreenSDN GUI.

capabilities.SustNMS was the more difficult capability to be emulated, once it requires network awareness in terms of routes and current workloads, its complexity grows in accordance with the topology size. In contrast ALR and SC were simpler to be implemented through extensions in the controller emulating their mechanisms and policies. In this sense, the most important feature in GreenSDN is the QoS Services module. Accurate network information was the key to assure a good balance between QoS and energy efficiency, avoiding wrong routing decisions or inadvertently put a device to sleep.

The energy savings results presented in the validation section were in accordance with our expectations based on previous works with SustNMS, ALR and SC. GreenSDN was a first step towards an SDN environment designed to emulate different green capabilities. As future works, we believe that GreenSDN may lead to others interesting points of research, such as the orchestration of green capabilities, the development of new capabilities and protocols and a validation of the controller in a real network environment comparing with the evaluations from the emulation environment.

ACKNOWLEDGMENTS

This work was supported by the Innovation Center, Ericsson Telecomunicações S.A., Brazil. Additionally, we would like to thank Marco A. T. Rojas for the insightful comments during the development of this work.

REFERENCES

[1] Ericsson, "Ericsson Energy and Carbon Report - On the Impact of the Networked Society," Ericsson, Tech. Rep., July 2013. [Online]. Available: http://www.ericsson.com/res/docs/2013/ericsson-energy-and-carbon-report.pdf

[2] S. Lambert, W. V. Heddeghem, W. Vereecken, B. Lannoo, D. Colle, and M. Pickavet, "Worldwide electricity consumption of communication networks," *Opt. Express*, vol. 20, no. 26, pp. B513–B524, Dec 2012.

[3] G. C. Januario, C. H. Costa, M. C. Amaral, A. C. Riekstin, T. C. Carvalho, and C. Meirosu, "Evaluation of a policy-based network management system for energy-efficiency," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 596–602.

[4] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR)," *Computers, IEEE Transactions on*, vol. 57, no. 4, pp. 448–461, Apr 2008.

[5] R. Bolla, R. Bruschi, and A. Ranieri, "Performance and power consumption modeling for green cots software router," in *Proceedings of the First International Conference on COMmunication Systems And NETworks*, ser. COMSNETS'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 420–427. [Online]. Available: http://dl.acm.org/citation.cfm?id=1702135.1702189

[6] M. Mostowfi and K. Christensen, "Saving energy in LAN switches: New methods of packet coalescing for Energy Efficient Ethernet," in *IGCC'11*, 2011, pp. 1–8.

[7] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. Maestro, "IEEE 802.3az: the road to energy efficient ethernet," *Communications Magazine, IEEE*, vol. 48, no. 11, pp. 50–56, Nov 2010.

[8] M. Zhang, C. Yi, B. Liu, and B. Zhang, "GreenTE: Power-aware traffic engineering," in *Network Protocols (ICNP), 2010 18th IEEE International Conference on*. IEEE, 2010, pp. 21–30.

[9] C. H. Costa, M. C. Amaral, G. C. Januario, T. C. Carvalho, and C. Meirosu, "SustNMS: Towards service oriented policy-based network management for energy-efficiency," in *SustainIT'12*, 2012, pp. 1–5.

[10] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," in *NSDI'10*, 2010, pp. 17–17.

[11] NS, "The network simulator, url = http://www.isi.edu/nsnam/ns/."

[12] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," Boston, MA, Dec. 2002, pp. 255–270.

[13] GENI, "Global environment for network innovations, url = http://www.geni.net/."

[14] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 19:1–19:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466

[15] B. Pfaff, J. Pettit, and S. Shenker, "Extending networking into the virtualization layer."

[16] POX, "Python-based openflow controller, url = http://www.noxrepo.org/pox/about-pox/."

[17] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks," in *Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–8.

[18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[19] S. Chowdhury, M. Bari, R. Ahmed, and R. Boutaba, "Payless: A low cost network monitoring framework for software defined networks," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–9.

[20] M. Ricca, A. Francini, S. Fortune, and T. Klein, "An assessment of power-load proportionality in network systems," in *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*, Oct 2013, pp. 1–8.

[21] S. Ricciardi, D. Careglio, U. Fiore, F. Palmieri, G. Santos-Boada, and J. Solé-Pareta, "Analyzing local strategies for energy-efficient networking," in *Proceedings of the IFIP TC 6th International Conference on Networking*, ser. NETWORKING'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 291–300. [Online]. Available: http://dl.acm.org/citation.cfm?id=2039912.2039944

[22] L. Nussbaum and O. Richard, "A comparative study of network link emulators," in *Proceedings of the 2009 Spring Simulation Multiconference*, ser. SpringSim '09. San Diego, CA, USA: Society for Computer Simulation International, 2009, pp. 85:1–85:8. [Online]. Available: http://dl.acm.org/citation.cfm?id=1639809.1639898