Green Service Levels in Software Defined Networks

Bruno B. Rodrigues[1], **Marco A. T. Rojas**[1]
Viviane T. Nascimento[1], **Tereza C. M. B. Carvalho**[1] **and Catalin Meirosu**[2]

[1]Escola Politécnica – University of São Paulo (EP-USP)
São Paulo, Brazil

[2]Ericsson Research
Stockholm, Sweden

{brodrigues,matrojas,vianetn,carvalho}@larc.usp.br
catalin.meirosu@ericsson.com

Abstract. *The growing energy consumption has become a major concern for network service providers not only to reduce operational expenses, but also to minimize environmental problems related to the high energy expenditure. In this regard, assessing the energy consumed has become an important task to maximize energy efficiency, and thus, offer different green service levels for customers that desire to save energy. An SDN controller is presented that manages distinct green service levels based on power models to account for the consumed energy and energy saving capabilities. Specifically, we present three SLAs with distinct requirements in terms of energy consumption and models to account for power consumption for each network scenario. Our approach provides a fine-grained accounting of user power consumption in distinct network usage scenarios in order to increase the accuracy of GreenSLA management in next generation networks. The proposal is validated by emulating two use cases inspired on the Facebook topology.*

1. Introduction

Driven by the increasing number of users with broadband and mobile access as well as the availability of new services and experiences, the power demand of the infrastructure has become a major concern for Network Service Providers (NSPs). Services such as video streaming are driving the way network infrastructures are being designed, constantly imposing higher constraints on performance and on availability requirements. Fulfilling such requirements incurs not only in high CApital and OPerational Expenses (CAPEX and OPEX) but also leads to significant Green House Gases (GHG)[1] emissions. Ericsson presented an overview in a mobility report [Ericsson 2014], in which the number of ICT (Information and Communication Technologies) devices are estimated to increase from 6 billion in 2013 to 12.5 billion devices in 2020, being one of the main reasons of the increased carbon footprint by ICT. Figure 1 summarizes the scenario for ICT fixed and mobile networks.

Costs related to networking are among the most significant for NSPs to absorb [Ericsson 2014]. For fixed ICT networks the share of global GHG emission is estimated

[1]GHG: gasses capable of absorbing infrared radiation, trapping heat in the atmosphere and making the Earth warmer.

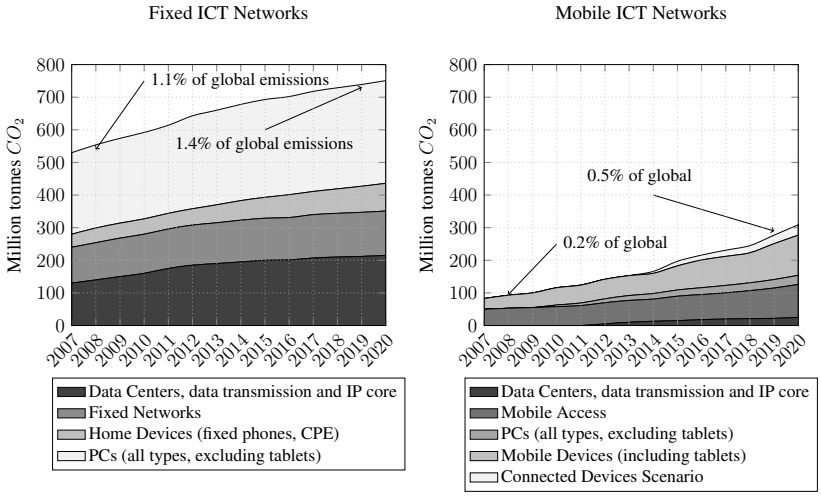

Figure 1. Global Emissions from ICT [Ericsson 2014].

to be 1.4% in 2020, and 0.5% for mobile networks, nearing 2% of the global carbon footprint. In this regard, telecom operators started to adopt environmentally responsible policies and deploy equipment that supports energy management features. A significant example comes from the Verizon 2013 Sustainability Report [Verizon 2013], is indicated that achieving energy efficiency has become critical to the ability to offer new capabilities and services, stating that 92% of the carbon emissions were due to power consumed to run their networks.

Besides, as organizations and governments around the world promote incentive programs to encourage the reduction of carbon emissions, an increasing number of customers are expected to require services tailored to specific energy conditions. However, instead of a one-size-fits-all class of service, in which energy savings and carbon credits are equally divided among customers, NSPs should offer off-the-shelf services in distinct categories of green services. In this regard, proposals were made to establish green services for data centers, ranging from green metrics (e.g., Power Usage Effectiveness - PUE and Data Center Efficiency - DCE) and services [Belady 2008, von Laszewski and Wang 2010, Le et al. 2010], to optimization frameworks [Hasan et al. 2014, Amokrane et al. 2015]. However, they do not address particularities of accounting and managing green services in a network infrastructure. As SDN facilitate the network management providing a software layer to control data plane nodes, this approach is used in this paper to implement the Green Plans.

We propose the GPController (Green Plans Controller), an SDN controller to manage the energy consumed at a user-level for network and compute resources. As main contributions, we outline the architecture and components that manage energy consumption in a fine-grained level, and power models to calculate energy consumption and savings for distinct network conditions. The proposal is evaluated based on the emulation of network and compute resources in Mininet, considering two use cases implemented on a topology inspired by the Facebook data center [Alexey Andreyev 2014]. The remainder of this work is organized as follows. Section 2 presents the related works highlighting its contribution. Section 3 presents the definition of green plans. Section 4 presents the architecture and implementation details. Section 5 presents the experimental evaluation

of Green Plans. Section 6 presents the final remarks and future works.

2. Related Work

A service is termed 'green' when deployed in computing systems designed to optimize energy efficiency and minimal environmental impact. Such services usually operate by relaxing traditional performance-based parameters for creating opportunities to save energy. In this context, Green Service Level Agreements (SLAs) are offered alongside regular SLAs, using eco-efficiency as a differentiating factor. Recently, several works have studied GreenSLAs ranging from the proposal of green metrics to optimization frameworks considering the availability of renewable energy. However, they do not address particularities on how to provide GreenSLAs for network infrastructures.

[von Laszewski and Wang 2010] introduced a framework including information about energy characteristics, focusing on energy metrics for green services (i.e., DCiE, PUE). Based on this, [Le et al. 2010] introduced a general, optimization-based framework and several request distribution policies enabling multi-data-center services to manage their brown energy[2] consumption and leveraged green energy, while respecting their SLAs. [Klingert et al. 2011] introduced the notion of GreenSLAs, identifying known hardware and software techniques to reduce energy consumption and to integrate green energy. In a case study, the authors compared three types of SLA: (i) a performance-based one that does not address energy consumption prioritizing performance and time; (ii) a relaxed SLA that requires key indicators to be within relaxed boundaries, and (iii) an energy-aware SLA that uses tight energy ranges for each job.

[Bunse et al. 2012] conducted a case study considering the SLA types proposed by Klingert et al. and providing an experimental evaluation of energy management in the context of web services and mobile clients. [Haque et al. 2013], proposed GreenSLAs in which a customer specifies a percentage of green energy to be used in the execution of data center workloads (e.g., x% of the job should run on green energy). Instead of a per job/application approach, [Hasan et al. 2014] proposed a specific time interval considering availability and price combination to buy green energy from the market.

[Hasan et al. 2015] continues to investigate the negotiation between green energy sources and cloud service providers, yet providing means to ensure that a data center can be proportionally green for the whole day, by exploiting available green energy sources and markets. [Amokrane et al. 2015] proposed a resource management framework, allowing providers to create a virtual infrastructure to provide resources (i.e., a set of virtual machines and virtual links with guaranteed bandwidth) across a geo-distributed infrastructure.

Most related GreenSLAs approaches consider the proposal of optimization frameworks within a triangle of energy, QoS, and cost to establish green services for data center resources. However, such approaches neglect particularities of networking services, and more specifically SDN networks, to propose differentiated green services. Inspired by the aforementioned approaches, we focus on enabling green services for SDN networks including a partial study on compute resources.

[2]Energy produced by polluting sources.

3. Green Plans

As governments and organizations promote incentives to reduce the carbon footprint, an increasing number of customers are expected to demand green products and services. For NSPs, this is not different. On the customer side, it is desirable to contract services in which the performance level can be adjusted according to daily requirements, defining time frames in which the performance level can be increased or decreased (e.g., increase the performance level during office hours, otherwise decrease outside office hours). On the NSP side, besides providing hardware and software capabilities to increase the energy efficiency, a system is required to manage the energy consumption for green customers. According to [Klingert et al. 2011], we defined three classes of service in Table 1.

Plan	**Network Resources**					**Compute Resources**	
	Bandwidth (Mbps)	**Delay (Ms)**	**Jitter (Ms)**	**Packet Loss (%)**	**Max. Energy (Watts)**	**vCPU (%)**	**Max. Energy (Watts)**
BP	B_1	D_1	J_1	PL_1	NE_1	C_1	CE_1
GP1	B_2	D_2	J_2	PL_2	NE_2	C_2	CE_2
GP2	B_3	D_3	J_3	PL_3	NE_3	C_3	CE_3

Table 1. Green Plans Parameters.

BP (Brown Plan) is a full-performance plan, in which customers have the best possible configuration in terms of performance. GP1 (Green Plan 1) represents a mid-term between performance and energy savings, and GP2 (Green Plan 2), an energy saving plan for customers desiring to save energy. Considering devices in which the energy consumed is proportional to the workload to be processed, we distinguished green plans based on provisioned $Bandwidth$ and $vCPU$. $Bandwidth$ considers a hierarchy $B_1 > B_2 > B_3$, meaning that BP customers have more bandwidth available than GP2 and so on. Since $Delay$, $Jitter$ and $Packet\ Loss$ relies on applications, they are selected in accordance with requirements imposed by applications. The $vCPU$ is the maximum compute workload for each host. Similarly to $Bandwidth$, the values consider $C_1 > C_2 > C_3$ to measure the energy consumed by servers.

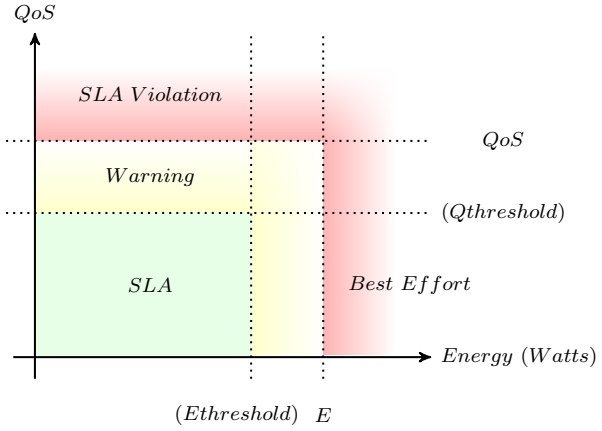

Figure 2. Thresholds for QoS and energy consumption.

Besides QoS parameters, we specified the amount of energy to be consumed by both networks (NE_i) and compute resources (CE_i). Therefore, customers desiring to save energy may set thresholds for the energy consumed. Thus, if the energy consumed reaches NE_i, or CE_i the customer enters a best effort mode until a new plan is contracted. Figure 2 illustrates the thresholds for both QoS and energy parameters. Based on QoS and energy parameters, alarms are set to define thresholds for both QoS ($QoS-k$) and energy ($E-k$). In the warning zone, policies can be employed to either renew the energy consumed for compute or network resources, or to adjust the QoS levels. Power capping is used when a customer reaches its NE_i or CE_i thresholds entering a best effort zone, in which its service levels are reduced. This occurs either by using traffic shaping capabilities or limiting the vCPU capacity.

IF eConsumed $> eThreshold$ **AND** eConsumed $\leq$ E **THEN**: Send alarm to customer

4. GPController Architecture

This section provides details of green services deployment in SDNs, presenting architectural and implementation solutions employed for GPController. Based on the ONF (Open Networking Foundation) SDN architecture, the GPController comprises network and compute resources structured in four abstraction planes: i) data, ii) control, iii) application, and iv) management. Figure 3 presents the architecture.

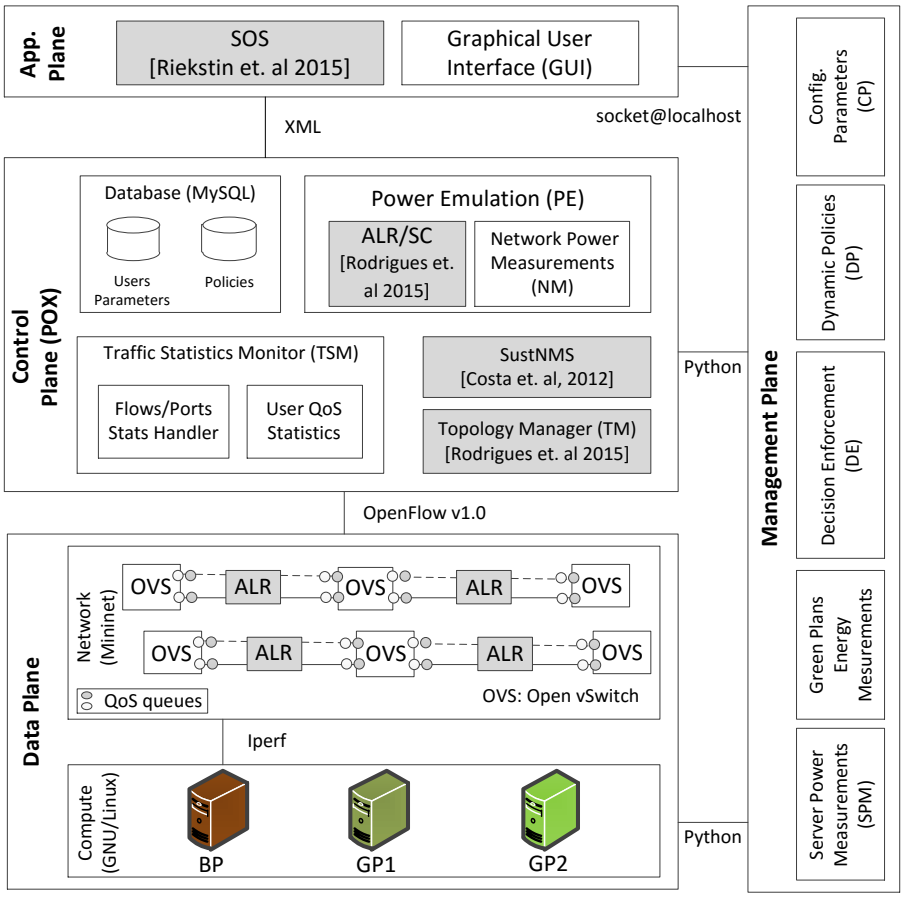

Figure 3. GPController Architecture

- **Application plane** comprising the SOS orchestration method and Graphical User Interface (GUI) elements, such as a topology viewer, statistics charts, to obtain user parameters;
- **Management plane**: interconnect the planes receiving information and taking decisions on which energy efficiency capabilities to employ and users requirements regarding energy consumption;
- **Control plane**: presents modules to obtain and to prepare data for the management layer, such as the Topology Manager (TM) that contains the network graph and controls flow tables; Traffic Statistics Monitor (TSM) to query nodes ports/flows statistics; Power Emulation (PE) to measure energy consumption; Database Manager to hold network and users logs; and the SustNMS capability to perform green traffic engineering; and
- **Data plane**: comprising Open vSwitch nodes configured with QoS queues to perform a traffic shaping capability, and servers based on virtual cores of the machine host. To generate network traffic we used the Iperf[3], and Cpuloadgen[4] to generate compute workload for each virtual core.

The GPController operates in two stages, configuration, and management. Configuration is related to parsing and deploying inputs from SOS and users. SOS performs a training stage before the GPController starts, to decide on the best combination of energy efficiency capabilities to be activated given a network utilization. As a result, it produces decision trees and policies describing environment and time conditions to change or to update decision trees. SOS parameters are parsed via XML (eXtensible Markup Language) files for configuring the Decision Enforcement (DE) and the Dynamic Policies (DP) modules. Users are able to define QoS and energy parameters using a GUI. Then, this information is sent via socket for configuring the databases with users policies and requirements.

After the deployment of SOS and users parameters, the management stage begins. It is composed of three steps: a) query data plane information regarding port/flow statistics, b) measurement of energy and QoS parameters, and c) enforcement of decisions. The Traffic Statistics Monitor (TSM) is responsible for querying node statistics regarding flow and ports usage. Compute resources are monitored by the Server Power Measurements (SPM) module at the management plane. It coordinates a workload generator (Cpuloadgen) capable of generating workload on each virtual core and comprises models to calculate the energy consumed by servers based on generated workload.

The PE module comprises models for each possible energy state in which nodes can be. Upon receiving bandwidth information for each active user, it calculates the energy consumed and saved by verifying current node's states (e.g., active, sleeping, or saving energy) for each node in the user path and enforcing the respective power model. QoS values are obtained by injecting probe packets in flows configured for each user at a regular time interval (detailed in subsection 4.1). Next, both QoS and energy information regarding each user, as well as an overall network are stored. As the final step in the management cycle, the DE (Decision Enforcement) query statistics to assess whether it is necessary to adjust the performance of the network or to compute resources. Two distinct

[3]https://iperf.fr/

[4]https://github.com/ptitiano/cpuloadgen

decisions are performed. The first aims to ensure users requirements regarding energy and QoS parameters. The second is based on a holistic network view, related to enforcing decisions given by SOS decision trees (i.e., given an overall network utilization decides whether to enforce energy efficiency capabilities).

4.1. Obtaining Data Plane Information

To match user packets and to account for network statistics, we used a MAC address flow instantiation. Based on prior knowledge of user's routes, two distinct rules were used to avoid the potential flood of rules in flow tables. One for edge nodes, specifying source and destination MAC address, and the other for interconnection nodes specifying destination MAC address. Furthermore, to proportionally calculate user's statistics accounting for the number of nodes that are shared among users is required. Thus, we account nodes in the path of active users, maintaining a dictionary of counters for each node. Once a user is inactive, counters of nodes in his/her path are decremented in the dictionary. Next, we provide details on how to obtain delay, jitter and packet loss in a per-user basis:

- **Delay**: given prior knowledge of user's path, whenever a workload is sent by an user probe packets are forwarded to the destination node from the source node with a timestamp as payload. Then, the switch-to-controller delay is estimated by determining its RTT (Round-Trip Time) injecting packets that are immediately returned to the controller, dividing the RTT by two to account for the bi-directionality of the given answer (Equation 1):

$$Delay = (t_{arrival} - t_{sent} - \frac{1}{2}(RTT_{src} + RTT_{dst})) \tag{1}$$

- **Jitter**: calculated as the average absolute value of the difference of consecutive delay samples. Given at least two consecutive delay samples (Equation 2), it is calculated as the average of absolute values of difference in two consecutive delay samples in a period of measuring time (Equation 3):

$$DelaySamples = [t_i, t_{i+1}, ..., t_k] \tag{2}$$

$$Jitter = abs\left(\sum_{t=1}^{T_k}(t_i - t_{i+1}) + (t_{i+1} - t_k)\right) \tag{3}$$

- **Packet loss**: estimated by polling flows statistics from source and destination nodes of each path. Uses delay probe-packet and control flags (src_flag and dst_flag) to detect when to subtract statistics from destination and source. When a probe packet is sent, src_flag is marked as true, and upon the packet's arrival, dst_flag also is marked as true. When both are true, the loss is calculated by subtracting the increase of the source switch packet counter from the increase of the packet counter of the destination switch. Then, both flags are set as false and another measuring round can be started.

To emulate compute workload on a per-user basis, a tool to generate workload on each core of the host machine was used (Cpuloadbench). Despite achieving a fine-grained generation of workload, a virtual core for each user has to be created, thus imposing a scalability restriction. Whenever a workload is sent, the SPM generates a random workload

between a fixed range (e.g., Brown Plan between 50-70% of CPU utilization) for each green plan. Then, a process is spawned to run the micro-benchmark. The CPU utilization of spawned processes is monitored and obtained the values are used in power models to calculate the energy consumed.

4.2. Green Plans Power Models

The component calculates the energy consumed and saved from users considering the state of nodes in the user path. Information on current workload and path are received from the TSM, and regarding servers consumption from the SPM. Then, the component checks the states of nodes in the user path, applying a specific power model. To obtain the energy consumed for network resources we consider the power models presented in [Rodrigues et al. 2015]. Compute resources consider a model in Equation 4 describing the energy consumed by a server including CPU, memory and disk resources [Beloglazov et al. 2012]:

$$P(u) = \underbrace{k}_{70\%} * \underbrace{P_{max}}_{250\ \mathrm{W}} + (1-k) * P_{max} * u \tag{4}$$

The $Pmax$ is the maximum power consumed when the server is fully utilized; k is the fraction of the power consumed by the idle server (i.e. 70%), and u is the CPU utilization. For our experiments, P_{max} is set to 250W according to [Beloglazov et al. 2012].

For network resources, the energy consumed is calculated by verifying states (i.e., active, sleeping, or applying an energy efficiency capability) of nodes in the user path and using the referred power model. However, such models do not calculate the energy consumption in a user granularity. Given that the energy consumed by the node chassis (comprising internal components such as CPU, RAM, fans) is not load proportional, we defined a model (Equation 5) in which the energy consumed by the node chassis ($P_{chassis}$) is distributed in accordance with the user workload and maximum link capacity. Thus, when distinct users are sharing a node in a certain measurement point in time, the energy consumed by the chassis is proportionally distributed among the users.

$$PP'_{on} = \underbrace{\left(\frac{P_{chassis} * W_{user}}{LinkCapacity}\right)}_{\text{Pchassis proportional energy distribution}} + \sum_{i=1}^{nPorts} (P_{port} * W_{user}) \tag{5}$$

$$PP'_{sleep} = \left(\frac{120}{NumGreenUsers}\right) \tag{6}$$

$$PP'_{ALR} = PP'_{on} - 15\% \tag{7}$$

$$PP'_{SC} = PP'_{sleep} * \left(\frac{tOn}{DutyCycle} - tOn\right) + PP'_{on} * tOn \tag{8}$$

Considering that several users can share the same nodes, their consumption can be obtained by splitting the fixed (representing the internal components such as CPU, memory, fans) consumption part among them. In this regard, Equation 5 is applied when a node is active, Equation 7 when ALR or SC is being applied. The energy consumed by users is measured as follows:

$$A_c\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{on}\,(W_{user}) \tag{9}$$

A_c: consumption from nodes powered on

$$B_c\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{(ALR\ or\ SC)}\,(W_{user}) \tag{10}$$

B_c: consumption from nodes enforcing ALR or SC

$$C_c\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{sleep} \tag{11}$$

C_c: consumption from nodes sleeping

$$D_c\ (W) = A + B + C \tag{12}$$

In A_c the energy consumed from active nodes is obtained. B_c calculates the number of Watts consumed from nodes applying either ALR or SC. C_c returns the Watts consumed for sleeping nodes. As SustNMS requires concentrating the traffic on a certain path while unused nodes are put to sleep, energy savings from affected users are obtained from nodes in sleep mode. In the last step, D_c performs the sum of the user's consumption. Savings per user is obtained by comparing their consumption with the maximum workload allowed in the network (W_{max}). Energy savings per user is measured as follows:

$$A_s\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{on}(W_{max}) - A_c \tag{13}$$

A_s: savings nodes powered on

$$B_s\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{on}(W_{max}) - B_c \tag{14}$$

B_s: savings nodes enforcing ALR or SC

$$C_s\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{on}(W_{max}) - C_c \tag{15}$$

C_s: Savings nodes sleeping

$$D_s\ (W) = A + B + C \tag{16}$$

D_s: Sum of savings

$$S\ (\%) = (D_s * 100)/D_s \tag{17}$$

The difference between the energy saving models and the energy consumption models is the consumption with the user workload subtracted from the consumption of the maximum workload used as a reference. To illustrate the operation of the module, Algorithm 1 presents the consumption measurements.

5. Experimental Evaluation

The goal of this evaluation is to assess the architecture and power models, as well as mechanisms to ensure a constraint regarding the amount of energy to be consumed. The GPController was deployed in a VM configured with 4 virtual cores and 4 GB RAM. The host machine is an Intel Core i5-3570 @ 3.40GHz with 8 GB RAM. The SDN network was emulated in Mininet and the GPController is based on the POX controller. Network traffic is generated through the Iperf, which is already available in Mininet. To create a separate workload for each virtual core, compute workload was emulated using Cpuloadbench. Thus, the emulation is restricted to 4 users, which is the number of virtual cores.

Algorithm 1: Algorithm to calculate the energy consumed and saved by users.

Input: active_hosts ⟵ list of active users
Input: sharedNodes ⟵ dictionary nodes shared by users
Output: Energy consumed (W) and savings (%) per user

```
1  begin
       /* Loop active users                                   */
2      for each user ∈ active_hosts:
           /* Loop nodes in the user path                     */
3          for each node ∈ user.path:
               /* Calculate the Energy Consumed               */
4              A_c ⟵ user.workload, node, sharedNodes
5              B_c ⟵ user.workload, node, sharedNodes
6              C_c ⟵ user.workload, node, sharedNodes
               /* Calculate the Energy Saved                  */
7              A_s ⟵ user.workload, node, sharedNodes
8              B_s ⟵ user.workload, node, sharedNodes
9              C_s ⟵ user.workload, node, sharedNodes
           /* Sum of the energy consumed                      */
10         D_c ⟵ A_c + B_c + C_c
           /* Sum of the energy saved                         */
11         D_s ⟵ A_s + B_s + C_s
           /* Percentage of the energy saved                  */
12         S ⟵ D_s * 100/D_s
           /* Store the result                                */
13         energyCS[user] ⟵ [D_c, S]
14     Return energyCS
```

5.1. Evaluation Settings

The topology is inspired by the Facebook data center fabric [Alexey Andreyev 2014]. To emulate ALR (which adjusts the link rate according to current load) we interconnected each pair of nodes using parallel links, which are configured with different rate limits. Standard links are configured to handle a maximum traffic of 30 Mbps, and ALR links 10 Mbps.

To send data across the network, four hosts were placed at node 21 and two sinks in nodes 1 and 27. Considering the Facebook scenario presented in [Alexey Andreyev 2014], two cases were considered for setting up flows. The first considers traffic going out of the data center (user to machine), and the second internal traffic (machine to machine). As more people connect to the Internet and new products and services are created, the user to machine traffic is large and ever increasing. However, machine to machine traffic is several orders of magnitude larger than that going out to the Internet (e.g., a simple scroll in a Facebook timeline require internal data that is spread internally within the data center servers). Furthermore, settings used in the evaluation are summarized in Table 2.

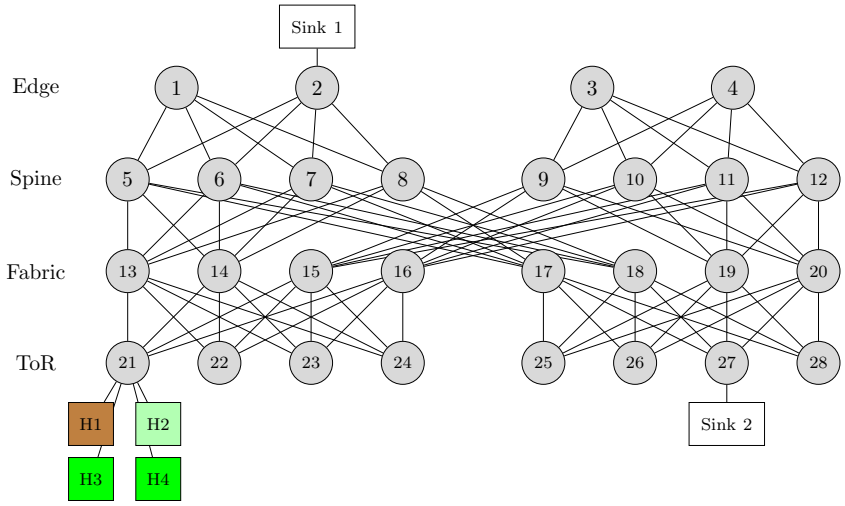

Figure 4. Topology of the evaluation system.

User	Case 1 Path	Case 2 Path	Bandwidth (Mbps)	vCPU (%)	Max. Energy C_E / N_E (Watts)
BP	[21-13-5-2]	[21-13-5-2]	15	70 - 100	1.5 / 14
GP1	[21-13-5-2]	[21-15-10-19-27]	9	30 - 40	1 / 10
GP2a	[21-13-5-2]	[21-15-10-19-27]	3	20 - 30	0.8 / 4
GP2b	[21-13-5-2]	[21-15-10-19-27]	3	20 - 30	0.8 / 4

Table 2. Evaluation settings for Cases 1 and 2.

Since the maximum link capacity was set to 30 Mbps, we divided the maximum reachable bandwidth for each user which was divided among users to provide a 100% link utilization in case four users share the same path. Thus, we considered 15 Mbps for a BP user, 9 Mbps for GP1, and 3 Mbps for GP2 (placing two GP users). The distribution of computational load in each vCPU allocated for users follows the same logic. For a BP user, the workload generated simulates an application that requires intensive processing, between 70 and 100% of the vCPU. However, the workload for GP1 and GP2 users is lower to avoid a computational overhead in the experiment. Thus, workload ranges between 30-40% for GP1 users, and 20-30% for GP2 users.

5.2. Evaluation Results

Figure 5 presents energy consumption and savings results in function of time for both use cases and employed energy efficiency capabilities. In Case 1, in which users share the same path [21-13-5-2], energy savings are distributed proportionally among green users considering nodes that were put in sleep mode. For instance, since the traffic engineering capability (SustNMS) performs a load balance using an alternate path to the predefined route [21-14-6-2] towards Sink 1, we forced users in the same route to evaluate QoS results with full link utilization. In this case, the BP user does not save energy once it uses the maximum bandwidth, however, savings from nodes in sleep mode are distributed among green users (GP1 and two GP2).

For Case 2, the BP user was configured with distinct routes from GP1 and GP2 users; thus individual decisions on energy efficiency capabilities were made due to the

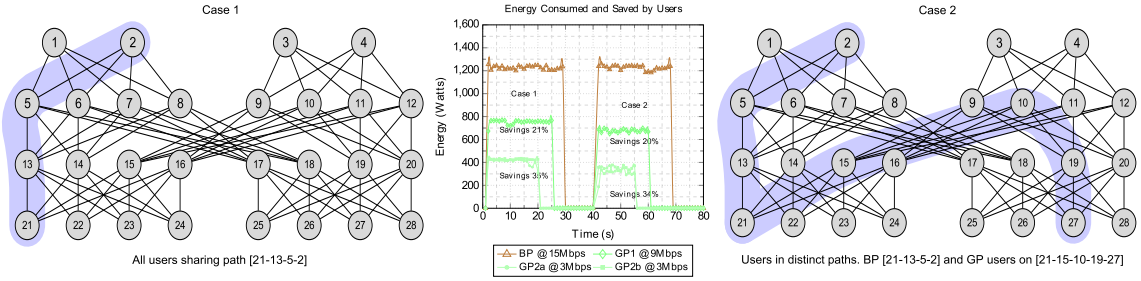

Figure 5. Per user energy consumption and savings and energy efficiency capabilities employed in both cases.

low utilization of nodes [15-10-19-27]. Despite GP2 users consuming a smaller amount of energy, they presented lower energy savings than in Case 1 because more nodes were active. Besides employing energy efficiency capabilities on such nodes, green traffic engineering is still more effective to save energy, aggregating traffic and putting unused nodes to sleep.

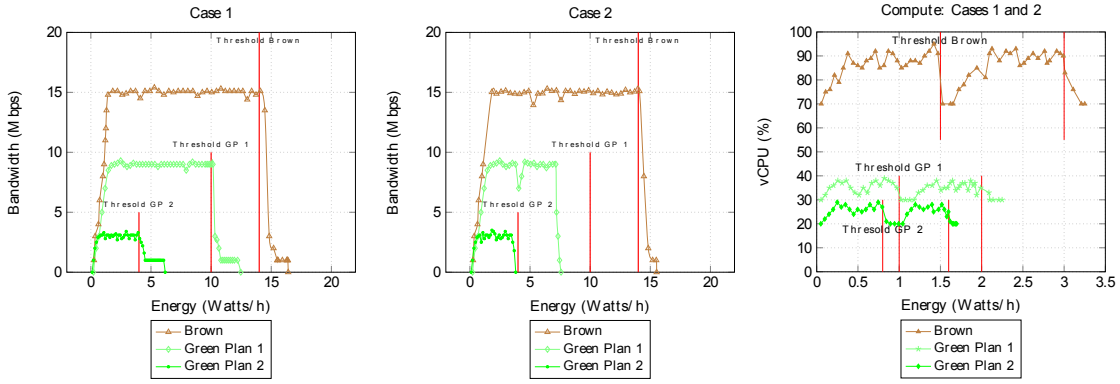

Figure 6. Max energy consumption thresholds

To evaluate energy threshold policies, random C_E and N_E values were defined. Figure 6 presents the energy consumed (W/h) per users in function of their workload, and aggregated to compute resources in a single run. When N_E is reached, user's traffic is forwarded onto a best effort QoS queue. Since the BP and GP1 users had the same bandwidth and routes for both cases, results on network energy thresholds were similar. For GP2 users, due the enforcement of SC+ALR capabilities on nodes [15-10-19-27] at the Case 2, the energy consumed did not reach the energy threshold. Conversely to network resources, we considered a single continue run for compute. Since the consumption relies on vCPU workload and the overall performance of the host machine, C_E thresholds were adjusted to force the GPController to restrict the vCPU usage. When C_E is reached, the workload generator component limits the workload on Cpuloadbench. Since we did not consider any energy saving feature for compute resources, thresholds for both cases were reached.

The quality of service effects of link utilization are presented in Figure 7. As in Case 1 users were configured to forward packets through the same path - ignoring the SustNMS load balance, their QoS statistics was bad contrasting with the Case 2. Thus, with 100% of link occupancy the segment between nodes 21 and 13 was saturated and

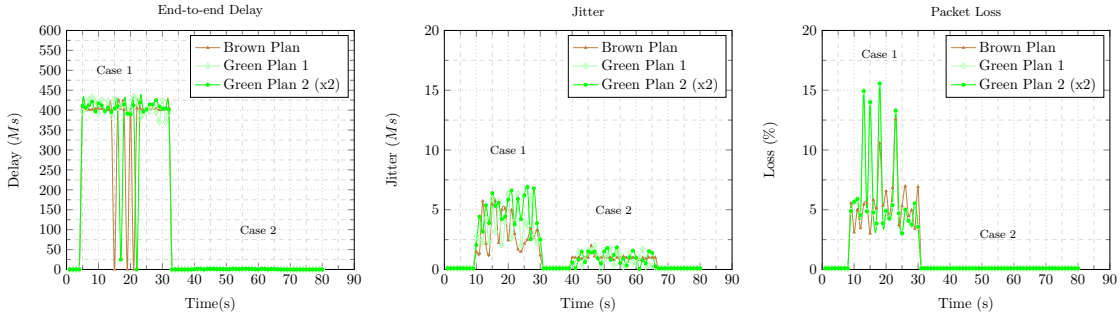

Figure 7. QoS results cases 1 and 2.

packets started to be queued and dropped. Since in Case 2 users were configured with distinct routes, links were not oversubscribed and QoS statistics was satisfactory for all users.

6. Final Considerations and Future Works

An SDN controller based on network energy efficiency capabilities was presented being a first step towards establishing green service levels for SDN networks. Thus, inspired on GreenSLAs works we presented an SDN architecture and power models to manage the usage of network and compute resources by users. Despite providing a closer view on a real deployment details, a validation by emulation imposes restrictions on scalability. However, it can provide a closer view of functioning details of the experiment, usually neglected by simulation. As future work, we consider combining emulation and simulation to scale the number of users at the different plans, as well as adjustments to the decision enforcement point to configure the network in case of QoS violations. For the Green Plans, we consider the deployment of policies in which users may define time frames to increase or to decrease performance levels by changing between green plans. Thus, allowing to adjust resources provisioning according to periods in which they are most required.

Acknowledgments

This work was supported by the Innovation Center, Ericsson Telecomunicações S.A., Brazil.

References

Alexey Andreyev (2014). Introducing data center fabric, the next-generation facebook data center network. Technical report, Facebook.

Amokrane, A., Langar, R., Faten Zhani, M., Boutaba, R., and Pujolle, G. (2015). Greenslater: On satisfying green slas in distributed clouds. *Network and Service Management, IEEE Transactions on*, PP(99):1–1.

Belady, C. (2008). Data center power efficiency metrics: Pue and dcie.

Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768.

Bunse, C., Klingert, S., and Schulze, T. (2012). Greenslas: Supporting energy-efficiency through contracts. *Energy Efficient Data Centers*, page 54.

Ericsson (2014). Ericsson Energy and Carbon Report - Including Results from the First-ever National Assessment of the Environmental Impact of ICT.

Haque, M. E., Le, K., Goiri, Í., Bianchini, R., and Nguyen, T. D. (2013). Providing green slas in high performance computing clouds. In *Green Computing Conference (IGCC), 2013 International*, pages 1–11. IEEE.

Hasan, M., Kouki, Y., Ledoux, T., and Pazat, J. (2014). Cloud energy broker: Towards sla-driven green energy planning for iaas providers. In *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS), 2014 IEEE Intl Conf on*, pages 248–255.

Hasan, S., Kouki, Y., Ledoux, T., and Pazat, J. (2015). Exploiting renewable sources : when green sla becomes a possible reality in cloud computing. *Cloud Computing, IEEE Transactions on*, PP(99):1–1.

Klingert, S., Schulze, T., and Bunse, C. (2011). Greenslas for the energy-efficient management of data centres. In *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking*, pages 21–30. ACM.

Le, K., Bilgir, O., Bianchini, R., Martonosi, M., and Nguyen, T. D. (2010). Managing the cost, energy consumption, and carbon footprint of internet services. In *ACM SIGMETRICS Performance Evaluation Review*, volume 38, pages 357–358. ACM.

Rodrigues, B., Riekstin, A., Januario, G., Nascimento, V., Carvalho, T., and Meirosu, C. (2015). GreenSDN: Bringing Energy Efficiency to an SDN Emulation Environment. In *Integrated Network and Service Management (IM), 2015 14th IFIP/IEEE Symposium on*.

Verizon (2013). 2013 Corporate Responsibility Supplement. Technical report, Verizon.

von Laszewski, G. and Wang, L. (2010). Greenit service level agreements. In *Grids and Service-Oriented Architectures for Service Level Agreements*, pages 77–88. Springer.