BRUNO BASTOS RODRIGUES

GREENSDN: ENERGY EFFICIENCY IN SOFTWARE DEFINED NETWORKS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2016

BRUNO BASTOS RODRIGUES

GREENSDN: ENERGY EFFICIENCY IN SOFTWARE DEFINED NETWORKS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:
Engenharia de Computação

Orientador:
Profa. Dra. Tereza C. M. B. Carvalho

São Paulo
2016

FICHA CATALOGRÁFICA

Rodrigues, Bruno B

GreenSDN: Energy Efficiency in Software Defined Networks/ Bruno Bastos Rodrigues. São Paulo, 2016.

113 p.

Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

1. Redes de Computadores #1. 2. Redes Definidas por Software #2. 3. Eficiência Energética #3. 4. Gerência de Redes #4. 5. Testbeds #5 I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais (PCS). II. t.

Acknowledgments

First of all, I would like to thank God for being able to do what I love.

Thanks to my advisor, Prof. Dr. Tereza Cristina Melo de Brito Carvalho, for the opportunity to undertake the mastering program and all the support during the development of this work. Also, I would like to thank my previous supervisor, Prof. Dr. Charles Christian Miers, for his guidance and help during my Bachelor's and useful suggestions towards the mastering. To Dr. Catalin Meirosu for all his valuable tips and technical contributions during the whole master's period.

I would like to express my gratitude to everyone at LASSU (Laboratory of Sustainability), LARC (Laboratory of Computer Networks and Architecture) and Ericsson Research Sweden, that contributed with technical support during the development of this research. Thanks to FDTE (Foundation for the Technological Development of the Engineering) and Ericsson Brazil for the financial support of this research.

I am heartily thankful to my parents and brothers for their affection and support at all times. Further, to all my family and friends for their continuous encouragement and comprehension.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a Deus por ser capaz de fazer o que amo.

Obrigado à minha orientadora, Profa. Dr. Tereza Cristina Melo de Brito Carvalho, pela oportunidade de fazer o programa de mestrado e todo suporte oferecido durante o desenvolvimento deste trabalho. Também, gostaria de agradecer ao meu primeiro orientador, Prof. Dr. Charles Christian Miers, por sua orientação e ajuda durante o bacharelado e suas importantes sugestões para o mestrado. Ao Dr. Catalin Meirosu por todas suas valiosas dicas e contribuições técnicas durante o período de mestrado.

Gostaria de expressar minha gratidão à todos colegas do LASSU (Laboratório de Sustentabilidade), LARC (Laboratório de Arquitetura e Redes de Computadores) e Ericsson Research Suécia, que contribuíram com suporte técnico durante o desenvolvimento desta pesquisa. Obrigado à FDTE (Fundação para o Desenvolvimento Tecnológico de Engenharia) e Ericsson Brasil pelo suporte financeiro dado à esta pesquisa.

Agradeço de coração aos meus pais e irmãos por sua afeição e suporte durante todo o tempo. Além disto, para toda minha família e amigos pela compreensão e encorajamento contínuo.

ABSTRACT

A significant number of protocols and capabilities have been proposed in recent years in response to the demand for reducing the amount of energy consumed by the network infrastructure. Besides rising economical issues, there is a widespread sensitivity to ecological impacts since both energy costs and electrical demands are in a growing trend. In this scenario, the development and validation of energy saving strategies are a key point of making networks more efficient. However, there is a lack of experimental environments designed specifically to emulate and validate such energy efficiency solutions. This work proposes an environment not only supporting the development and discussion of energy-saving solutions but also management applications considering energy-saving primitives. For this purpose, the environment is built considering the implementation of energy efficiency capabilities that are representative of each network scope (interface, device, and network) in the Mininet environment taking as a basis the Software-defined Networking (SDN) paradigm. The proposed environment was evaluated with different experiments by comparing the energy savings obtained by activating these energy-efficiency capabilities.

Keywords: Computer Networks, Software-Defined Networking, Energy-Efficiency, Network Management, Testbed

RESUMO

Um significativo número de protocolos e funcionalidades foram propostos em resposta à crescente demanda de energia por infraestruturas de rede. Além de gerar problemas econômicos, existe uma preocupação quanto aos impactos ambientais uma vez que maior a demanda por eletricidade, maior o impacto ambiental para suprir esta demanda. Neste cenário, o desenvolvimento e validação de estratégias para economizar energia são um ponto chave para tornar infraestruturas de rede mais eficiente. No entanto, há uma falta de ambientes desenvolvidos especificamente para emular e validar soluções para eficiência energética. Com este propósito este trabalho propõe um ambiente capaz de suportar não apenas o desenvolvimento de soluções para tornar redes mais eficientes energeticamente, como também o desenvolvimento de aplicações de gerenciamento que baseiam-se em primitivas de economia de energia. Para este propósito, o ambiente é construído considerando a implementação de funcionalidades orientadas à eficiência energética que são representativas para cada escopo de rede (interface, dispositivo e rede) no ambiente de emulação Mininet tomando como base o paradigma de redes definidas por *software*. O ambiente proposto foi validade por meio de diferentes experimentos comparando a economia de energia obtida pela ativação destas funcionalidades.

Palavras-chave: Redes de Computadores, Redes Definidas por Software, Eficiência Energética, Gerência de Redes, Testbed

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS

ACPI	Advanced Configuration and Power Interface
ALR	Adaptive Link Rate
API	Application Programming Interface
CAPEX	CAPital EXpenses
CARPO	Correlation Aware Power Optimization
CLI	Command Line Interface
CM	Configuration Manager
CPU	Central Processing Unit
DB	DataBase
DC	Data Center
DE	Decision Enforcement
DVFS	Dynamic Voltage and Frequency Scaling
FIBRE	Future Internet BRazilian Environment for experimentation
GENI	Global Environment for Networking Innovations
GHG	Green House Gases
GPON	Gigabit-capable Passive Optical Networks
GUI	Graphical User Interface
GREEN TE	Green Traffic Engineering
LPI	Low-Power Idle

NCP	Network Configuration Parameters
NSP	Network Service Provider
MBPS	Megabits Per Second
MIB	Management Information Base
MIP	Mixed Integer Linear Programming
MS	MilliSecond
OVS	Open vSwitch
OPEX	OPerational EXpenses
OSPF	Open Shortest Path First
PBNM	Policy-Based Network Management
PE	Power Enforcement
PBM	Policy Based Management
QoS	Quality of Service
ICT	Information and Communication Technologies
IETF	Internet Engineering Task Force
IP	Internet Protocol
SDN	Software Defined Networks
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SOHO	Small Or Home-Office
SOS	Sustainability-Oriented System
SC	Syncronized Coalescing

TM	Topology Manager
USA	United States of America
WDM	Wavelength Division Multiplexing
XML	eXtensible Markup Language

CONTENTS

1 INTRODUCTION

Triggered by the increase of broadband and mobile access and a growing number of new services and experiences such as online gaming and video streaming, the rise in the power demand of data center infrastructures has become a critical issue for Network Service Providers (NSPs) (BOLLA *et al.*, 2011). Requirements imposed by many services are driving the way data centers are being designed, involving high-performance and high-availability constraints. Further, besides demanding power-hungry machines and supporting systems (e.g., cooling) to sustain their operation, they also rely upon redundant architectures to endure against peak loads and unexpected conditions.

Fulfill such requirements incurs not only in high CApital and OPerational Expenses (CAPEX and OPEX, respectively) but leads to significant Green House Gas (GHG)[1] emissions. Ericsson present this scenario in different mobility reports, in which the number of ICT (Information and Communication Technologies) devices is estimated to increase from 6 billion in 2013 to 12.5 billion devices in 2020, being one of the main reasons of the growing carbon footprint by ICT. Figure 1 presents the scenario for ICT fixed and mobile networks (ERICSSON b, 2014).

In 2013, Ericsson estimated the overall carbon footprint of the ICT sector (fixed, mobile) in about 1,1 million tons by 2020. The scenario presented in 2014

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

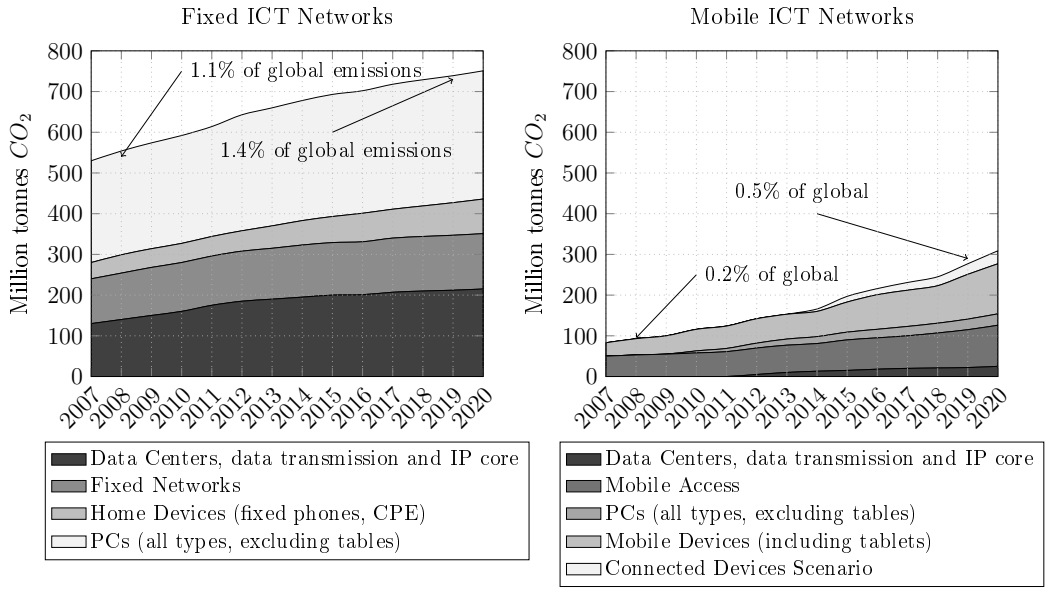


Figure 1: Global Emissions from ICT (ERICSSON b, 2014).

holds this estimation. For fixed ICT networks, it is estimated that the share of GHG emissions will be about 1.4% in 2020, and 0.5% for mobile networks, which also confirms the previous estimation (about 2%) of the ICT carbon footprint.

As one of the major categories in ICT, the data center power demand including servers, cooling and networking grew 7% from 2012 to 2013, reaching an annual/yearly electricity consumption of near 350 TWh (COOK *et al.*, 2014). This amount is predicted to increase 81% by 2020, reaching almost 630 TWh annually (COOK *et al.*, 2015). Although there is no consensus on how much the networking infrastructure contributes to the total data center power demand, studies indicated that the numbers vary between 4% to 33 %. Figure 2 present the view of the network consumption for many authors.

Even though there is no consensus on the network consumption within the data center, the average increase over the years is remarkable. The increase is leveraged mainly due to services such as Netflix, that accounts for 37% of all downstream Internet bandwidth in North America during peak periods, and at peak times when it consumes more bandwidth than YouTube, Amazon, and Hulu[2]

[2]A free video streaming website. http://www.hulu.com/

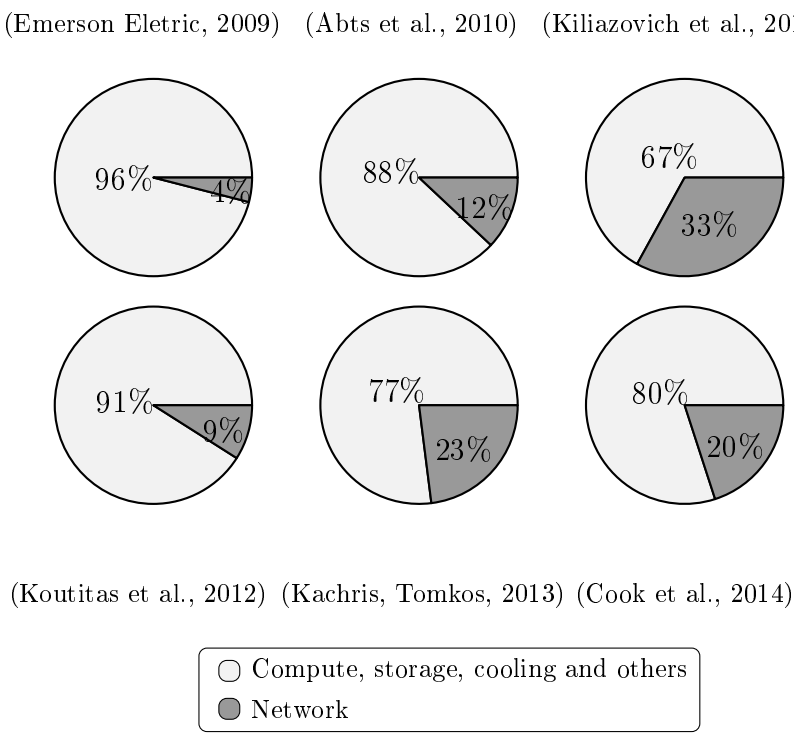


Figure 2: The network infrastructure share in the energy consumption of a data center.

combined (ADHIKARI *et al.*, 2012).

In response to the demand for reducing the amount of energy consumed by networks, several energy efficiency protocols and capabilities have been proposed. Proposals range from re-engineering of particular chip-level components to network traffic consolidation capabilities operating at the network level. However, most projects are designed to run on specific network conditions or devices, which may require an expensive or a large number of external instrumentation. In this scenario, there is a gap for testing the behavior of such capabilities without the deployment of large and expensive dedicated infrastructures. Testing is essential to prove concepts in conditions that approximate the real implementation through initial explorations based on modeling and simulation/emulation. Thus, facilitating the introduction of novel approaches or equipment into actual network deployment.

1.1 Motivation

With the high number of connected devices increasing power demand and carbon footprint, networking systems are being designed and dimensioned according to high-performance and high-availability requirements. On the NSP side, this implies in over-provisioning and redundancy of devices and links to endure against peak load periods. As a result, during times of low network traffic, the over-provisioned networks are also over-energy-consuming which create opportunities to employ energy optimization strategies, as illustrated in Figure 3.

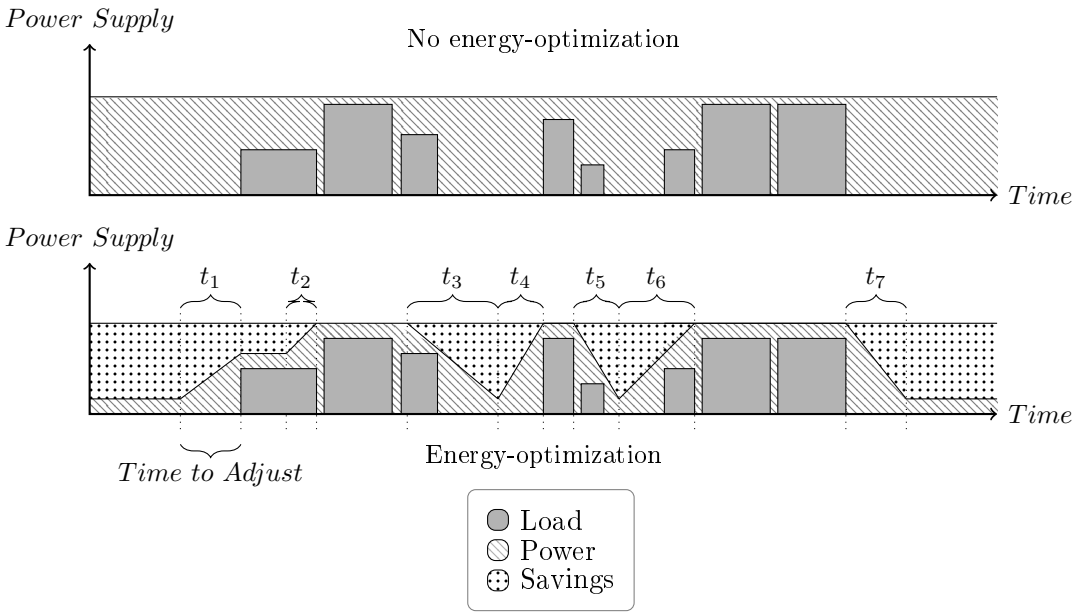


Figure 3: Energy-optimization vs no optimization

Non-optimized scenarios can be found in legacy network infrastructures, in which the network devices consume an amount of power that is independent of workload, and besides the over-provisioning strategies, it is a common approach to increase the overall electricity supply during peak times to prevent power outages. However, during periods of low network traffic, the adoption of energy optimization strategies presents opportunities to save energy.

A significant shift in networking research is required to introduce energy-

awareness in a controlled manner, without compromising the quality of service or reliability requirements. Adjust the network capacity to meet current load requires an adjustment time $(t_1, t_2, ..., t_6)$, to put devices into low energy states or wake up from a sleeping state. Thus, during unexpected traffic bursts, any delay to (re)configure the network may compromise the quality of service and reliability requirements. In this regard, Software Defined Networks (SDNs) change the way traditional systems are designed and managed (SEZER *et al.*, 2013), providing fast adjustments in response to sudden workload variations and better visibility (decoupling) and control (centralized management) to perform tasks such as network diagnosis and troubleshooting than in traditional networks.

Simulations can provide insight on how a particular algorithm would perform in distinct network conditions. However, a linear combination of emulation and implementation of major energy efficiency capabilities can provide a closer view of what may happen in a real scenario, facilitating the deployment of experimental features in real environments. While local network platforms allow a quick development and evaluation of network features and services, a distributed system platform can provide insight on how a particular feature would operate in a scalable and distributed scenario, such as the Internet.

For instance, OFELIA (SALSANO *et al.*, 2013), GENI (GENI,) and GreenStar Network (GSN, 2010) are examples of distributed network platforms that allows researchers to experiment features at scale. While the first two provide a general-purpose research infrastructure, the latter is designed to deliver cloud-based ICT services based entirely on renewable energy sources such as solar, wind and hydroelectricity. As examples of local network platforms, Mininet (LANTZ; HELLER; MCKEOWN, 2010) and EstiNet (WANG; CHOU; YANG, 2013a) can emulate a network infrastructure enabling rapid deployment and replication of experiments (Section 3 provide details on related works).

1.2 Objective

The objective of this work is to leverage green networking by providing a network platform comprising energy efficiency capabilities implemented at different network scopes, such as network, the system (node-level) and subsystem (chip-level/interfaces). Based in GreenSDN, researchers can develop management techniques and evaluate the impacts of energy efficiency capabilities on quality of service (QoS) requirements. Complementary, GreenSDN aims at providing a fast prototyping and troubleshooting environment to enable fast (re)configuration of network experiments.

1.3 Methodology

This work is based on basic and applied research in which the theoretical background related to the state-of-the-art in energy efficiency capabilities was obtained through qualitative research. The applied research involves the steps related to the design and development stages of GreenSDN applying techniques and concepts related to network and distributed systems. Furthermore, the author has taken part in correlated projects developed at Lassu (Laboratory of Sustainability). To summarize, the research methodology employed consists in the following steps:

1. **Literature Review and Analysis**: involves the analysis of the state of the art in energy efficiency capabilities and protocols, network emulation environments, and concepts related to software-defined networks and network monitoring.

2. **Architecture Proposal**: based on the network platform and the requirements imposed by energy efficiency capabilities, comprises the design of an

architecture able to fulfill our predefined objectives and leverage network management oriented to energy efficiency;

3. **Development**: once the architecture is designed, this stage involves development and troubleshooting of GreenSDN, considering that the development process reveals conditions not observed in the proposal;

4. **Experimental evaluation**: aims at evaluating if the proposed goals were met and report the benefits and related problems of GreenSDN.

1.4 Organization

The core chapters are highlighted in Figure 4. Chapter 2 describes and classifies approaches towards energy efficiency in wired networks and energy efficiency capabilities. Chapter 3 tackles topics related to the network platform choices and software-defined networking.

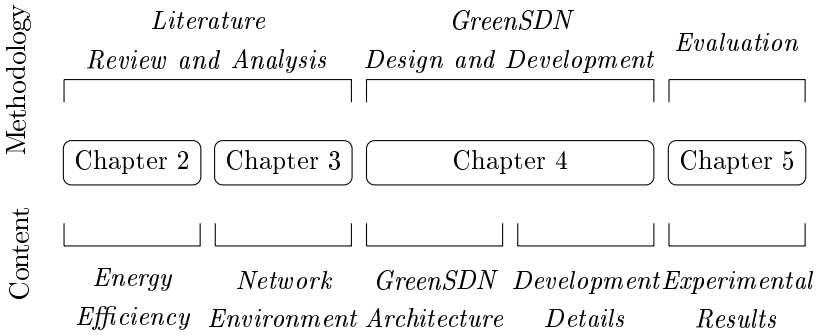


Figure 4: Organization and Methodology

Chapter 4 describes the proposed architecture and provide details on the development of selected capabilities and core components on the architecture. Chapter 5 presents the GreenSDN experimental validation. To conclude, Chapter 6 presents the concluding remarks and future works.

2 ENERGY EFFICIENCY IN WIRED NETWORKS

This chapter provides an overview of the different approaches to obtain energy efficiency in wired networks to understand how to classify the energy efficiency capabilities based on their characteristics and network scope (Section 2.1). Further, according to pre-defined criteria, we outline three representative to be developed in GreenSDN (Section 2.3) detailing its functioning. We summarize the chapter providing the final remarks.

2.1 Energy Efficiency Approaches

There are different approaches to managing a network focusing on energy efficiency. However, the largest part of undertaken approaches is founded on a few basic concepts, which are usually inspired by energy-saving mechanisms and power management criteria already available in computing systems (BOLLA *et al.*, 2011). Among the existing taxonomies to classify energy efficiency approaches, two main taxonomies are highlighted herein: one presented in Bolla *et al.* (2011), and the second in Bianzino *et al.* (2012). Bolla *et al.* (2011), presented a survey of existing approaches for energy efficiency in wired networks and Bianzino *et al.* (2012) developed an overview of green networking research providing a general view of current methods, comprising both network and compute resources.

Bolla *et al.* (2011) summarizes existing approaches into three general approaches, including techniques employed at each general approach. The objective is to classify procedures designed especially for wired networks. The taxonomy is presented in Figure 5.

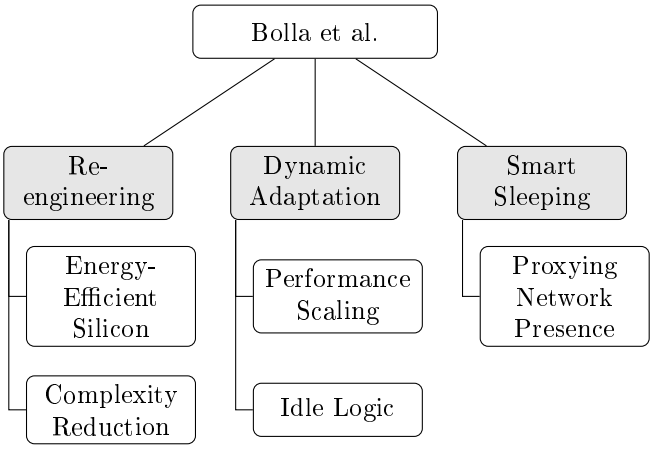


Figure 5: Taxonomy of Energy Efficiency Approaches for Wired Networks. Source: (BOLLA *et al.*, 2011)

- **Re-engineering**: this category aims at designing network elements that are more energy-efficient. Specially, it focuses on including new energy efficient silicon technology or reducing the complexity to execute embedded software. This approach is that one which can achieve higher energy savings, however, it is the most challenging regarding innovation;

- **Dynamic Adaptation**: comprises capabilities that can modulate the power capacity of internal components (e.g., packet-processing engines and network interfaces) to meet a load proportional usage. Most of the current approaches require a hardware interface, either to dynamically scale the performance or to enforce the idle logic. The idle logic allows reducing the energy consumption by shutting down, for a time frame, sub-components when no activities are performed.

- **Smart Sleeping**: it is similar to the idle logic. However, it enables devices or parts of them to turn off almost entirely, entering in a very low energy

consumption state. As a consequence, once most of its functionalities and applications are shut down, one cannot maintain network connectivity. In this regard, smart sleeping techniques consist of transferring the network presence to another host/device when entering in such deep sleep modes.

This taxonomy focuses on approaches that operate in a single node; not considering capabilities that require a broad network view, such as green traffic engineering methods. However, it is precise to describe the set of techniques that fall under one of the approaches. For instance, when it represents the difference between *idle logic* and *smart sleeping*, in which both present similar behavior, but one may achieve higher savings by shutting down more internal sub-components.

Bianzino *et al.* (2012) propose a more broad view than Bolla *et al.* (2011) in the sense of covering approaches beyond the network infrastructure. The main difference is the description of resource consolidation and virtualization strategies, which were not described by Bolla *et al.* (2011). In this regard, the authors use the term resource consolidation, which is broad enough to cover compute (e.g., nodes migration) and network approaches (e.g., green traffic engineering). The Figure 6 presents the proposed taxonomy.

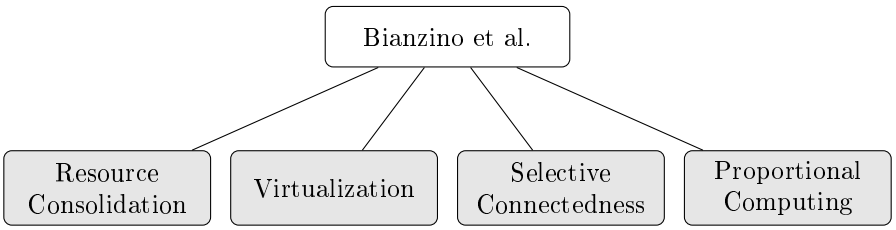


Figure 6: Taxonomy for Energy Efficiency Approaches. Source: (BIANZINO *et al.*, 2012)

- **Resource Consolidation**: regroups dimensioning strategies to reduce the overall network consumption. It aims at adapting the level of existing over-provisioning resources based on known behavior (e.g. traffic consolidation

based on traces of the network traffic), dimensioning resources to meet current traffic load;

- **Virtualization**: allows consolidating multiple services to operate on the same hardware. A typical example of virtualization consists in sharing servers in data centers, thus reducing hardware, energy, and cooling costs and improving energy management. In this regard, lightweight/container-based virtualization such as Docker[1] is a trend in hardware virtualization, removing layers of software and providing a more efficient usage of hardware resources;

- **Selective Connectedness**: it is similar to the smart sleeping technique, allowing the equipment to go into deep idle states for some time while proxying its network presence to maintain network connectivity;

- **Proportional Computing**: is based on the idea of a system consuming energy in proportion to its utilization. It was first proposed by (BARROSO; HÖLZLE, 2007), and can be applied to a system as a whole, network protocols, and devices.

The taxonomy proposed by Bianzino *et al.* (2012) cover approaches that are either for network or compute resources, being one of the most employed taxonomies in the current state-of-the-art. For instance, Bilal, Khan, Zomaya (2013) and Garg and Buyya (2012) use this taxonomy to describe their approaches in the context of green cloud computing.

On one side, Bolla *et al.* (2011) is precise to address energy efficiency techniques that are unique for the network infrastructure than Bianzino *et al.* (2012), that describes methods involving both network and compute resources. For instance, Bolla *et al.* (2011) describe in more detail techniques deployed at lower

[1] https://www.docker.com

network layers (physical and data link), presenting a relatively detailed level on re-engineering approaches, which is not addressed by Bianzino *et al.* (2012). However, Bolla *et al.* (2011) lacks of approaches to managing the network in a holistic view, i.e., approaches that administer a set of nodes to consolidate traffic into energy efficient routes given a particular network utilization.

2.2 Architectural Scope

Recognize energy efficiency approaches is important to understand how a given capability operates. Additionally, knowing the architectural scope enable us to understand where it works. In this regard, rather than just describing approaches for energy efficiency, Bianzino *et al.* (2012) presented a view that takes the network layer into account. It considers the TCP/IP protocol stack, in which the solutions can either be implemented as a single layer or require cross-layer interaction. Also, Bianzino *et al.* (2012) highlight the infrastructure scope to describe solutions that advocate a clean state redesign of the network architecture or incorporate resource consolidation approaches into their routing protocols.

Schlenk *et al.* (2013) presented a similar classification summarizing the capabilities by their architectural scope (i.e., describing where the approaches are deployed using a network architectural view). It takes into account the following ranges: network, the system (network elements) and subsystems (components of the elements). The architectural view, including the application and infrastructure scopes, is illustrated in Figure 7.

- **Application Scope**: includes research efforts that incorporate energy-awareness in the software design, e.g., coordination/orchestration of network scope capabilities;

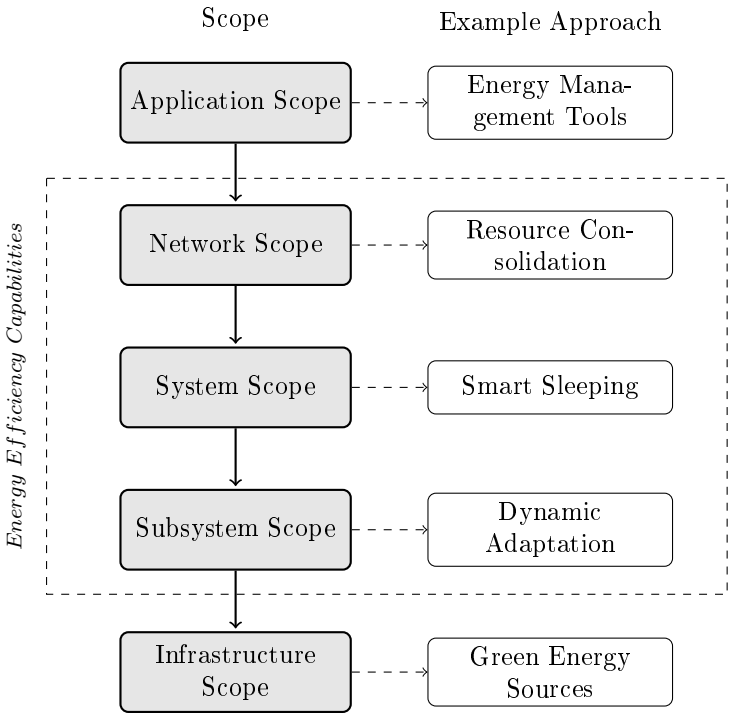


Figure 7: Architectural Scope of Energy Efficiency Capabilities

- **Network Scope**: is related to the management of nodes, such as energy-aware routing or green traffic engineering;
- **System Scope**: includes capabilities that coordinate network nodes, such as green traffic engineering;
- **Subsystem Scope**: modulates the performance of internal components, such as interfaces, and processing unit;
- **Infrastructure Scope**: includes renewable energy sources, air conditioning systems management, smart grids, and others;

Rather than just presenting energy efficiency approaches, Schlenk *et al.* (2013) aims at categorizing such approaches according to their architectural scope in the network infrastructure. For instance, it presents the network, system and subsystem scopes, in which most approaches are inserted (e.g., selective connectedness, proportional computing, power models estimation, re-engineering). Be-

sides, Bianzino *et al.* (2012) complement this view by including the application and infrastructure scopes, which may respectively include, network management approaches and the usage of green energy sources.

2.3 Energy Efficiency Capabilities

As network platforms may comprise several working modules (e.g., monitoring, topology manager) and there are many different capabilities per network scope we considered to select one of the following scopes to be implemented: network, system, and subsystem. Furthermore, three most important aspects were taken into account as criteria to select the capabilities: a) open source code, b) documentation or related works providing enough information to implement its logic, c) description of existing energy saving results. Table 1 presents an overview considering the following aspects: i) general approach and ii) technique is taken from the approaches' taxonomy; iii) the architectural scope and iv) a brief description.

Dynamic Voltage and Frequency Scaling (DVFS) (SEMERARO *et al.*, 2002) takes into account that the power consumption of an electronic circuit is proportional to its operating frequency and the square of the voltage. It consists of intentionally decreasing or increasing the performance of a processor by dynamically changing the frequency and the voltage. Following, the Adaptive Link Rating (ALR) deals with the underutilization of links dynamically modulating the capacity of network interfaces by scaling up or down existing Ethernet data rates. Similar as DVFS, it employs the performance scaling approach.

Low-Power Idle (LPI) (CHRISTENSEN *et al.*, 2010) is a capability that put network interfaces into a lower energy consumption state (subsystem scope) during periods of low link utilization. It allows for rapid transitions back to the

EE Capability	General Approach	Technique	Architectural Scope	Description
DVFS (SEMERARO *et al.*, 2002)	Proportional Computing	Performance Scaling	Subsystem	Dynamically change the clock the frequency according to load demand.
ALR (GUNARATNE *et al.*, 2008)	Proportional Computing	Performance Scaling	Subsystem	Adapt the link rate according to the traffic being handled by the interfaces.
LPI (CHRISTENSEN *et al.*, 2010)	Proportional Computing	Idle Logic	Subsystem	Put interfaces in a low energy consuming state during idle periods
ACPI (STEELE, 1998)	Proportional Computing	Performance Scaling	System	Based on DVFS, allow to configure power modes to meet current load.
SC (MOSTOWFI; CHRISTENSEN, 2011)	Proportional Computing	Smart Sleeping	System	Join packets to send data bursts and creating idle periods, then allowing links or nodes to sleep
GPON (TROJER *et al.*,)	Re-engineering	Energy Efficient Silicon	System	Optical passive network using point-to-multipoint technology Requires less equipment and has low maintenance cost
Energy-Aware Routing Green OSPF (CIANFRANI *et al.*, 2010)	Resource Consolidation	EE Routing	Network	Shares the shortest path trees among couple of routers coordinating routers to save energy during low traffic
CARPO (WANG *et al.*, 2012)	Resource Consolidation	EE Routing	Network	Unites correlated traffic based in heuristics to consolidates network traffic after unused equipments are turned off
Green Traffic Engineering (ZHANG *et al.*, 2010)	Resource Consolidation	EE Routing	Network	Performs Green TE to maximize link utilization and allowing unused links to sleep.
SustNMS (COSTA *et al.*, 2012)	Resource Consolidation	EE Routing	Network	Concentrates flows and put unused devices to sleep, according to the power models of the devices.
ElasticTree (HELLER *et al.*, 2010)	Resource Consolidation	EE Routing	Network	Manage a fat tree topology, concentrate traffic and put nodes to sleep, saving energy.

Table 1: Energy Efficiency Capabilities Classification

active state in case of high-performance data transmission. Advanced Configuration and Power Interface (ACPI) (STEELE, 1998) defines different states of energy that can be applied to systems during their operation. The most relevant ones are the C-states and P-states. C-states describes power consumption states which a CPU can be, for instance, C0 (operation state), C1 (halt), C3 (stop the clock); P-states describe the processor performance state representing different DVFS settings combinations.

Synchronized Coalescing (SC) (MOSTOWFI; CHRISTENSEN, 2011) is a system scope capability that is intended for low utilization periods in which is possible to put into sleep mode internal components of a device. It orchestrates LPI modes of all individual interfaces to coalesce incoming packets, creating short periods in which internal components (e.g., packet processor) can be put into sleep mode, and then coalesced packets are sent in bursts.

Gigabit-capable Passive Optical Networks (GPON) (TROJER *et al.*,) is a fiber network that only uses fiber and passive components like splitters and combiners rather than active components. It deploys the optical technique Wavelength Division Multiplexing (WDM) so a single fiber can be used for both downstream and upstream data, thus using less equipment than an Ethernet-based network.

The Energy-Aware OSPF (Open Shortest Path First) (CIANFRANI *et al.*, 2010) is a green traffic engineering capability based on OSPF-based mechanism that supports energy-aware traffic engineering solutions. It addresses the optimization problem based on the Multiple Commodity Flows (MCF) constraints with a weighted objective considering both energy consumption and network performance regarding maximum link utilization.

CARPO (Correlation Aware Power Optimization) (WANG *et al.*, 2012) is a network level capability that provides a scheme to consolidate traffic flows based on a correlation analysis of flows in a data center network. It proposes an

optimization algorithm that dynamically combines traffic flows into a small set of links and switches and then shuts down unused devices for higher energy savings. While Energy-Aware OSPF is based on OSPF to include power constraints to find energy-efficient shortest paths, CARPO performs a correlation between flows utilization focusing on minimizing the number of active links and devices for a data center network.

Similar to CARPO, GreenTE (Green Traffic Engineering) (ZHANG *et al.*, 2010) is a network level capability aiming to reduce the number of active devices and links in response to demand and performance constraints. The optimization problem is solved as a Mixed Integer Linear Programming (MIP) with the total network power saving as the objective to be maximized, being the performance requirements and network delay considered constraints this formulation. However, differently from CARPO, GreenTE provides a formal model that maximizes the number of links to be put into sleeping mode under the constraints of link utilization and path length, and additionally balances the network load.

SustNMS (COSTA *et al.*, 2012) and ElasticTree (HELLER *et al.*, 2010) are network level capabilities focusing on data center networks, similar to CARPO. SustNMS was designed by our research group at LASSU aiming to strike a balance between quality of service requirements and energy efficiency. It considers the manual input of routes by a network administrator, who sets a path based on user requirements. ElasticTree introduces energy proportionality in data center networks by turning off as many unneeded links and switches as possible. It consists of three logical modules - optimizer, routing, and power control. The optimizer's role is to find the minimum power network subset which satisfies current traffic conditions and outputs a set of active components to both the power control and routing modules.

Given that the literature contains good technical descriptions of ALR (Adap-

tive Link Rate) and SC (Synchronized Coalescing) to guide its implementation while SustNMS is a previous work performed by our research group, so the source code was available. Furthermore, capabilities and techniques that operate at the infrastructure level such as Smart Grids access, which aims to optimize the power supply in a data center, or the management of air flows between racks, are outside the scope of this work.

2.3.1 Subsystem scope: Adaptive Link Rate (ALR)

ALR is a capability that deals with the underutilization and over-provisioning of Ethernet links by dynamically changing data rates in response to traffic levels (GUNARATNE *et al.*, 2008). Figure 8 presents the ALR functioning.

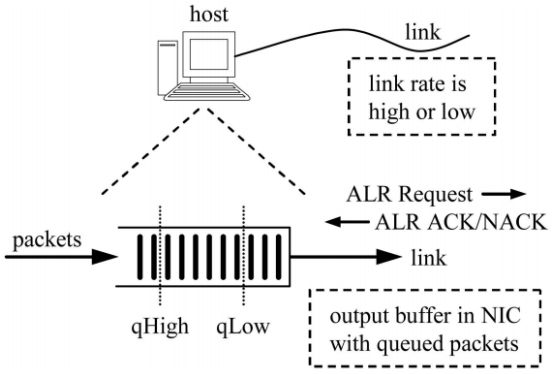


Figure 8: Adaptive Link Rate. Source: (GUNARATNE *et al.*, 2008).

It is designed to modulate the capacity of network interfaces scaling up or down existing Ethernet data rates (i.e., 10 Mbps, 100 Mbps, 1 Gbps). ALR consists of a mechanism and policy. The mechanism determines how the data rate changes through a link negotiation, and the policy determines when to change the data rate, aiming to maximize the time spent at a low data rate and saving energy without packet losses (GUNARATNE *et al.*, 2008).

The policy is based on a dual threshold policy in addition to counting the number of transmitted bytes (t_n) in time (t_{util}). If t_n is less than the defined threshold ($qLow$ - queue low and $qHigh$ - queue high in bytes), and then the link

rate switch process is invoked. When the queue length in an interface exceeds the $qHigh$ threshold, then it is requested to increase the link data rate. Conversely, when the queue length becomes lower than the $qLow$ threshold, a request is sent to reduce the link data rate. Thus, if a low traffic level is detected, a low data rate should be used. Otherwise, a high link data rate is necessary.

The authors performed experiments using distinct traffic patterns to observe policies and link negotiation behavior. Results presented that in average ALR can achieve power savings of about 8 to 20%, depending on link utilization t_{util}. In average, lower the t_{util}, the higher are the savings (around 5% of link utilization to reach 20% of energy savings). Therefore, whenever ALR is active, it is considered the use of the following power model:

$$\begin{aligned} AdaptivePower = Power_{chassis} + \ & Num_{linecards} * \ Power_{linecard} \\ & + \sum_{i=0}^{nPorts} port * \ workload \text{ - } \mathbf{15}\% \end{aligned} \tag{2.1}$$

According to Ricca *et al.* (2013), ALR can save up to 21% on the studied equipment. Furthermore, Ricciardi et al. (RICCIARDI *et al.*, 2011) studied the functionality and discovered that the energy spent after reducing the link rate depends on the native interface speed. The authors also state that half of the energy is due to the fixed part, and that, using ALR, the savings could reach 15%. ALR is interesting to use in scenarios in which the load is bigger since it spends much less time to wake up the interfaces (microseconds order of magnitude, while waking up a node from a sleep mode can take minutes). Additionally, the functioning of ALR is well documented and studied, e.g. surveys on ALR techniques (BILAL *et al.*, 2013) and (MAHADEVAN; BANERJEE; SHARMA, 2010).

2.3.2 System Scope: Synchronized Coalescing (SC)

SC is a capability intended for low utilization periods of network devices, such as Small Or Home-Office networks (SOHO). Although presenting a low energy consumption, the number of SOHO devices deployed around the world is so huge that their consumption becomes significant overall (CHRISTENSEN *et al.*, 2010). The SC objective is to prevent that links connected to a node forward data for a while, creating a time opportunity to turn off internal components. Figure 9 present its functioning.

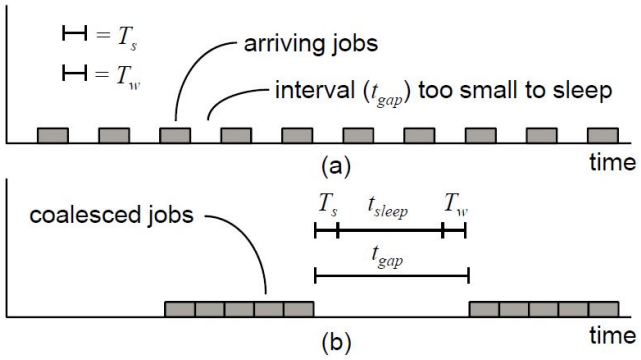


Figure 9: Arriving jobs (a) without, and (b) with coalescing. Source: (CHRISTENSEN *et al.*, 2010).

Example (a) in Figure 9 presents a case in which the packets arrival rate do no present a t_{gap} large enough to enforce LPI (Low Power Idle). LPI is a mode for Ethernet links (defined by IEEE 802.3az) used for reducing the energy consumption of interfaces in a switch or router when no data is transmitted (CHRISTENSEN *et al.*, 2010). In Figure 9 (b), arriving packets are coalesced creating t_{gap} large enough to enforce LPI. T_s and T_w are, respectively, times required to activate and deactivate SC.

SC uses a mechanism to orchestrate LPI modes of all individual interfaces, coalescing packets during the t_{gap} such that an entire switch may be put into lower consumption mode. The capability improves the efficiency of IEEE 802.3az by coalescing incoming packets and forwarding them into bursts, making the

number of transitions between LPI and active modes decrease (MOSTOWFI; CHRISTENSEN, 2011). Figure 10 details its operation sequence.

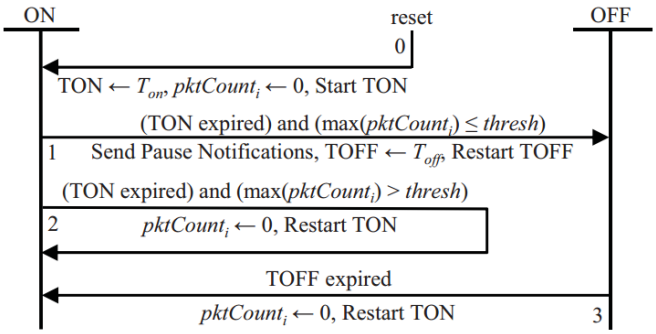


Figure 10: Synchronized Coalescing. Figure from (CHRISTENSEN *et al.*, 2010).

SC capability defines a DutyCycle, i.e., the total cycle time of SC considering the tOn (the time the node must stay in operation mode). The $tOff$ (time the capability is inactive) is given by: $tOff = \left(\frac{tOn}{DutyCycle} - tOn\right)$. While SC is active, incoming packets are buffered, and a packet counter ($pktCount$) is initialized with the tOn. When tOn expires, two cases may happen: 1) the maximum elements of $pktCount$ is greater or equal to $thresh$ (to compare to the maximum of all packets), in which Pause Notifications are sent on all links, $tOFF$ is reset, and starts to count down, and the switch enters into OFF state for another OFF period; or 2) the maximum elements of $pktCount$ is less than $thresh$, in which case: tOn is reset to its initial value and starts to count down, all elements of $pktCount$ are set to 0, and the switch remains in ON state for another ON period. Upon the expiration of $tOFF$, the tOn is set to its initial value and starts to count down, all elements of $pktCount$ are set to 0, the switch returns to ON state and the entire procedure repeats. The authors presented energy savings about 40% for SC, considering that the capability was intended for small or home office devices.

2.3.3 Network Scope: SustNMS

SustNMS was a prior work of our research group. It was designed as a network management system driven by policies which enable the enforcement of energy-efficiency according to high-level (business) decisions (COSTA *et al.*, 2012). It is based on IETF MIB models and SNMP as the management protocol, allowing operators to strike a balance between the assurance of QoS and green traffic engineering. The system integrates a real-time energy-efficiency assessment with an evaluation of network availability and performance. The architecture of SustNMS is depicted in Figure 11.

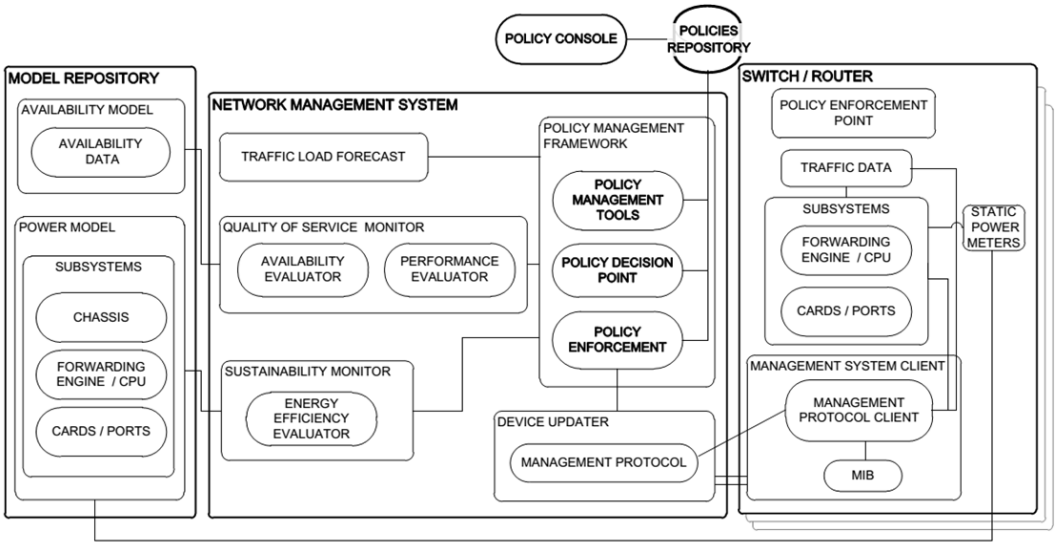


Figure 11: SustNMS Architecture. Source: (COSTA *et al.*, 2012).

It comprises three main components: a model repository that holds models for availability and power consumption for each device; the network management system that is the core model in the architecture; and the switch/router component. The network management system can be active in two ways, depending on the bandwidth usage: sustainability mode (SustNMS-S) which tries to maximize the number of sleep nodes concentrating traffic, and performance mode (SustNMS-P), which routes the network prioritizing performance.

2.4 Chapter Final Remarks

This chapter presented an overview of undertaken approaches towards energy efficiency and a characterization of energy efficiency capabilities based on their actuation scope in the network. Understand such ways to reduce energy expenditure is important to classify the energy efficiency capabilities based on their characteristics. For instance, if a given capability operates in a single node putting internal components to sleep in case of low utilization, it is a capability that implements the Smart Sleeping strategy. However, if a capability manages some nodes in the network being able to move workloads and putting nodes in a path to sleep, it would be a Resource Consolidation capability.

Another key aspect to understand how much energy a network equipment consume under the possible operating conditions is to model the nodes consumption through power profiles. From this abstraction is possible to design algorithms to perform traffic engineering based on energy requirements. For GreenSDN, two power models are taken into account. One of them considers nodes in which the power consumption is proportional to the workload (i.e., higher is the workload, the higher is the energy being consumed). The second one deploys a constant power profile representing devices in which the power consumed does not vary with the workload. Thus, it is possible to calculate the amount of energy being consumed by a given node at a given point in time. Further, based on network usage traces one may enforce energy efficiency capabilities in advance of an event, predicting a given behavior at a given point in time (e.g., during non-office hours).

Moreover, this section provided an overview of approaches towards energy efficiency introducing basic concepts techniques related to energy efficiency, and select energy efficiency capabilities that are representative of their network scope. The criteria to determine a representative capability was the availability of sup-

port and documentation, which enable a precise development of selected capabilities. As consequence, well-known capabilities such as ALR and SC provide enough information on their functioning and expected behavior. Moreover, SustNMS (a network-scope capability) was a previous research work led at LASSU, so the source code and documentation were available. To conclude, Table 2 outline the characteristics of the selected capabilities to be implemented by the GreenSDN.

Capability	**Scope**	**Objective**	**Restrictions**	**Achievable Savings**
ALR	Interfaces	Adjust the link rate on interfaces in order to meet current workload	Operate with existing Ethernet data rates 10 Mbps/100Mbps 1Gbps	Up to 15%
SC	Device	Coalesce packets to create idle periods in which the device can be put into sleep mode.	Intended for low bandwidth utilization being most common for SOHO devices	Up to 40%
SustNMS	Network	Aggregate traffic putting unused nodes into sleep mode	Requires manual input of alternative routes to strike a balance between QoS and energy efficiency	Relies on the network usage

Table 2: Capabilities of the GreenSDN.

Achievable energy savings by SustNMS relies on how the network is being used, i.e. during office hours in which the network usage is usually higher than non-office hours, energy savings may not be possible. However, during the night it is possible to put nodes to sleep and maintain a minimal graph to keep the network connected. Also, it is possible to combine capabilities of different approaches to increase energy savings during office hours and non-office hours. For instance, during office hours in which may not be possible to use green traffic engineering, adjusting the link rate in order to meet current workload may be the more efficient strategy.

3 NETWORK ENVIRONMENT

As the demand for network services increases, it is necessary not fulfill the current requirements, but also to anticipate and plan for satisfying future requirements and trends on network services. Nevertheless, the deployment of novel algorithms and protocols in real systems is difficult due to potential side-effects in critical services. Thus, decoupling experimental research from real deployment has become essential to avoid side-effects and leverage network research. In this regard, this Chapter reviews current network platforms to be used as the baseline for GreenSDN, concepts related to network management and Software-Defined Networking (SDN).

3.1 Network Platforms

For many years, experimental research platforms have been designed to study theoretical concepts at scale. Software-based simulations have always been considered an efficient approach to study physical systems (SIATERLIS; GARCIA; GENGE, 2013). However, they do not provide an accurate analysis of the diversity and complexity of the network protocol stack. In this regard, hardware-based emulation is considered a flexible and powerful approach to provide a closer view of how a particular capability would operate in a real deployment (PEDIADITAKIS; ROTSOS; MOORE, to be published).

As the widespread adoption of new technologies and services (i.e., video stre-

aming and online gaming) highlights limitations in current network infrastructures, it also puts similar efforts towards the development of experimental network platforms. As consequence, several works have explored the various properties of such systems under different requirements, including this work exploits energy efficiency aspects. As requirements to be addressed by a network platform, Pediatikis, Rotsos and Moore *et al.* (2014) and Holibaugh *et al.* (1988) identified the following key requirements:

- **Scalability**: is the ability to support and manage network experiments of growing size while still providing increased throughput and reduced response time. Thus, scalability is not defined as a fixed number, but as a function defined over minimum QoS (Quality of Service) levels associated with an overall throughput. However, based on Pediatikis, Rotsos and Moore (2014) the related works were classified taking into account the following metrics based on the network platforms category and their capacity to increase the overall throughput while maintaining QoS levels:
  - **High**: simulation-based network platforms based on mathematic models to deploy and evaluate experiments;
  - **Average**: emulation/simulation platforms implementing the network behavior on software and being still considered software-based experiments; and
  - **Low**: testbeds and real deployment network platforms.
- **Reproducibility**: is defined as the ability to export and replicate experimental scenarios and their results. It was considered the capacity to migrate the same experiment to a different environment and obtain the same results without major modifications (✓or ✗);

- **Usability**: corresponds to the ease of use, modify and deploy experiments, including Graphical User Interface (GUI) (if any) and documentation;
  - **Good**: has an intuitive GUI to setup and deploy experiments and, APIs accessed via CLI (Command Line Interface), in addition to provide support either through documentation and/or user community;
  - **Average**: does not provide GUI to setup and deploy experiments but contains APIs and support either from documentation or user community; and
  - **Bad**: does not provide GUI or documentation/community support.
- **Compatibility**: corresponds to a particular requirement to ensure that the environment is compliant with a given technology. In the case of GreenSDN, it is considered the support of OpenFlow and SDN (✓or ✗).
- **Availability**: since few platforms are outdated and do not have community support, we considered (✓or ✗); and
- **Hardware Requirements**: define the minimal hardware configuration to run experiments. As the scalability requirement, the minimal hardware requirement to run an experiment is closely related to the type of the network platform:
  - **High**: considers testbeds and real deployment network platforms.
  - **Average**: comprise platforms that are purely based on emulation; and
  - **Low**: contain platforms that are simulated and combines emulation and simulation.

Another requirement defined in (PEDIADITAKIS; ROTSOS; MOORE, to be published) is *fidelity*: as the ability of the experiment to replicate specific

system behavior with accuracy. However, the evaluation of this requirement demands a comparison between the emulation and the deployment in the real environment. Other requirements can be found in (HOLIBAUGH; PERRY; SUN, 1988), such as *extensibility* that is the ability to integrate new functions and tools; and *adaptability* that relates to the portability of the experiment, which can be understood as a reproducibility (the reason why it is not considered in the evaluation).

In addition, the network platform should be *open source or available* (if proprietary) and *straightforward to deploy/replicate* experiments. Thus, it has to enable quick adjustments in the experiment settings and, straightforward deployment or replication related to environments that run locally (e.g., a server or multiple VMs hosted on a single server). Based on the highlighted requirements, the Table 3 presents an evaluation of main network platforms in distinct categories.

Given the growth of network infrastructures virtualization in network infrastructures, there are efforts to emulate or simulate programmable networks to provide environments supporting realistic user traffic, at scale, and with interactive behavior. Table 3 present three categories of network platforms: testbeds, simulators, and emulators/simulators. Testbeds (Subsection 3.1.1) are examples of the global environment providing a broad range of network features through network slices. Simulators (Subsection 3.1.2) are software-based network experiments that allow an evaluation of protocols and involves modeling the underlying state of the target. Emulation (Subsection 3.1.3) still a software-based network experiment, however, it is the process of mimicking the hardware or software of a real network environment to test the performance of real applications over a virtual network.

Environment	Scalability	Reproducibility	Usability	Compatibility SDN Support	Availability	Hardware Requirements
Testbeds						
Emulab (WHITE *et al.*, 2002)	Low	✓	Good	✓	✓	Average
OFELIA (SALSANO *et al.*, 2013)	Average	n.a.	Good	✓	✓	High
Planetlab (ROSCOE, 2002)	Average	n.a.	Average	Custom OVS	✓	High
FIBRE (FIBRE, 2016)	Average	✓	Good	✓	✓	High
GENI (GENI,)	Average	n.a.	Good	✓	✓	High
AKARI (AKARI, 2007)	Low	n.a.	Average	✓	✓	High
FIRE (GAVRAS *et al.*, 2007)	Average	✓	Good	✓	✓	High
CANARIE (GSN, 2010)	Average	n/a	Good	✓	✓	High
Simulation						
ns2 (NS2, 2016)	High	✓	Low	✗	✓	Low
ns3 (NS3, 2016)	High	✓	Low	Partial	✓	Low
OMNeT++ (OMNET, 2016)	High	✓	Low	✗	✓	Low
FS-SDN (GUPTA; SOMMERS; BARFORD, 2013)	High	✓	Average	✓	✓	Low
Emulation/Simulation (Hybrid)						
Mininet (LANTZ; HELLER; MCKEOWN, 2010)	Average	✓	Good	✓	✓	Low
EstiNet (WANG; CHOU; YANG, 2013b)	High	✓	Good	✓	Proprietary	Low
ModelNet (VAHDAT *et al.*, 2002)	Average	✗	Bad	✗	Outdated	Average
DummyNet (CARBONE; RIZZO, 2010)	Average	✗	Average	✗	✓	Average
Selena (PEDIADITAKIS; ROTSOS; MOORE, to be published)	High	✓	Average	✓	✓	Low

Table 3: Comparison of well-known network experimentation platforms across different dimensions.

3.1.1 Testbeds

Emulab (WHITE *et al.*, 2002) is a management system for a network-rich cluster that provides a space and time shared hardware for studying networked and distributed systems, one of its goals is to transparently integrate a variety of experimental environments, including support to emulate/simulate OpenFlow and SDN environments. OFELIA (SALSANO *et al.*, 2013) is an initiative of the European Union 7th Framework Programme (FP7)[1] that provides a diverse OpenFlow-enabled infrastructure to allow Software Defined Networking (SDN) experimentation. It is currently composed of ten sub-testbeds (called islands), most of them in Europe and one in Brazil, which deploys SDN.

PlanetLab (ROSCOE, 2002) is a global research network established since 2002 that supports the development of new network services being composed by several compute nodes around the world. Currently, PlanetLab consists of 1353 nodes at 717 sites around the world. FIBRE (Future Internet Brazilian Environment for Experimentation) (FIBRE, 2016) is a federated research facility funded by the 2010 Brazil-EU (European Union) Coordinated Call in ICT to test new applications and network architecture models, being composed by 11 testbeds (also called islands or nodes) among USA (United States of America), Brazil and Europe. On Brazil side, the primary objective of FIBRE was to build a Future Internet Testbed federated with other worldwide Testbed initiatives.

Similar to Emulab and Planetlab, GENI (Global Environment for Networking Innovations) (GENI,) is a project founded by the USA that involves several nodes around the world to promote research on Future Internet topics and accelerate the transfer of this research results to the industry creating products and services. Other initiatives with similar research purposes are AKARI (AKARI, 2007) project in Japan, FIRE (GAVRAS *et al.*, 2007) for EU and CANARIE

[1] https://ec.europa.eu/research/fp7/

(GSN, 2010) for Canada.

As such platforms were conceived to leverage the Future Internet research, all testbeds mentioned above provide support to experiments based in the OpenFlow protocol and SDN. However, by using compute nodes and/or switches spread across different locations within the world or a country, the hardware requirements to execute an experiment are elevated and in most cases, the allocated resources are scheduled to perform during a pre-defined time slot. Also, due to hardware restrictions in the different nodes or islands, the reproducibility of the experiment may be limited. For instance, if an operation requires the latest version of a given network protocol or imposes a particular compute requirement, the experiment may not be easily reproduced in other islands.

3.1.2 Simulators

NS2 (NS2, 2016) and NS3 (NS3, 2016) are widely deployed network simulators based on discrete events used for the simulation of network protocols with different network topologies. NS2 was built in C++[2] language and provides the simulation interface through OTcl[3], an object-oriented dialect of Tcl. Like NS2, the NS3 is an open source, discrete event network simulator, but it supports parallel simulations and can be implemented in pure C++. By default, neither NS2 nor NS3 supports OpenFlow or SDN-based networking, however, NS3 can support OpenFlow relying on an external library OpenFlow switch library (OFSID)[4].

The motivation behind the development of OMNET++(OMNET, 2016) was to produce a powerful open source discrete event simulation tool that can be used by academic, educational and research-oriented commercial institutions for the simulation of computer networks, distributed and parallel systems. Unlike NS2

[2]http://www.cplusplus.com/doc/tutorial/
[3]https://en.wikipedia.org/wiki/OTcl
[4]https://www.nsnam.org/docs/release/3.13/models/html/openflow-switch.html

and NS3, OMNET++ is not only designed for network simulations. It can be used for modeling of multiprocessors, distributed hardware systems and performance evaluation of complex software systems (BILALB; OTHMANA *et al.*, 2013).

FS-SDN (GUPTA; SOMMERS; BARFORD, 2013) is a simulation tool for prototyping and evaluating new SDN-based applications. It is based on the FS[5] simulation platform that is a network flow record generator containing a discrete event simulation core to generate the flow records, which relies on existing TCP throughput models to drive the simulation. FS-SDN extends the underlying FS simulation engine by seamlessly incorporating the POX (POX, 2009) OpenFlow controller framework and API, and switch components that can be controlled and configured through the OpenFlow control protocol.

In general, network simulators are designed to test network protocols and applications at scale due to their modeling nature. For this reason, all network simulation platforms provide higher scalability levels in contrast to testbeds and emulation platforms. However, not all simulators provide full support to novel protocols such as OpenFlow, which may be a limiting factor to Future Internet research. Thus, to overcome this problem NS3 implemented a library that models a switch with OpenFlow capabilities, and initiatives such as FS-SDN are designed specifically for network simulation of SDN environments. To conclude, reproducibility is one of the main strength of simulation platforms in comparison with other platforms, and simulation allows to adjust settings quickly in the modeled nodes and reproduce experiments at scale.

3.1.3 Emulators/Simulators

Among the Emulators/Simulators, EstiNet (WANG; CHOU; YANG, 2013a) was the better choice of network platform. However, the solution is proprietary

[5] https://github.com/jsommers/fs

and not available for academic research. In this regard, Mininet (LANTZ; HELLER; MCKEOWN, 2010) 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 on a virtual machine, also allowing faster implementations. It is readily available as open source, and straightforward replication of experiments is one of its main strengths. Mininet includes data plane switching functionality from Open vSwitch (OVS) (PFAFF; PETTIT; SHENKER, 2009). However, the OVS code is relatively complex and, therefore, difficult to modify. Instead, in GreenSDN we opted for implementing the interface and node-level green capabilities emulation at the control plane.

The ModelNet (VAHDAT *et al.*, 2002) project established an approach in scalable emulation of Internet topologies, using edge hosts running unmodified applications, with nodes emulating virtual network topologies using DummyNet (CARBONE; RIZZO, 2010). ModelNet improved scalability by increasing hardware requirements and could only parallelize execution at the extent which a particular application and topology allow it. Despite being one of the pioneers in scalable network emulation, ModelNet is outdated and does not support SDN/OpenFlows experiments.

DummyNet (CARBONE; RIZZO, 2010) is a tool for bandwidth management and testing networking protocols implemented in FreeBSD[6] but portable to other protocol stacks. It works by intercepting packets in their way through the protocol stack and passing them through one or more pipes which simulate the effects of bandwidth limitations, propagation delays, bounded-size queues, packet losses, so on. Selena is a network emulation framework based in Xen[7] that offers reproducible experiments via an automation interface for the configuration of all experimental details. To emulate faster and larger networks, it adopts the tech-

[6] https://www.freebsd.org/
[7] http://www.xenproject.org/

nique of time-dilation and transparently slows down the passage of time for guest operating systems. Furthermore, it can emulate links by creating pairs of guest network interface devices bridged in Dom0[8].

By combining the advantages of emulation and simulation, it's possible to validate experimental models against real traffic loads, evaluate real applications against repeatable traffic derived from a rich variety of existing simulation models and scale to larger topologies. However, emulate real networking software imposes higher hardware requirements than simulation platforms and affect the experiment scalability, which is the case of ModelNet and DummyNet.

3.2 Software-Defined Networking (SDN)

As network infrastructures expand in response to a growing number of users and services, the management task increases in complexity. The heterogeneity of devices and technologies imposes a need for trained personnel to understand and deploy new and old features (VERMA, 2009). Fundamentally, network management is the process of monitoring and controlling network resources to ensure that it is operational and compliant with user requirements (SUBRAMANIAN, 1999). It is usually divided into three abstraction layers (GREENBERG *et al.*, 2005): i) data plane; ii) control plane; and iii) management plane. The data plane is responsible for forwarding packets based on local forwarding states; the control plane computes and coordinates forwarding states of the data plane, involving the coordination with the rest of the network; and the management plane visualizes and configures data provided by the control plane.

In the context of networks oriented to energy efficiency the management complexity grows. Implement and coordinate some existing features poses as a challenging task for network operators (RIEKSTIN *et al.*, 2015a). In traditional

[8]Privileged domain that starts first and manages the unprivileged domains.

networks, the control and data planes are combined into a single node and protocol. Once a forwarding policy is defined, the only way to make an adjustment to the policy is via changes to the devices configuration (SEZER *et al.*, 2013). However, this approach is restrictive to provide fast changes in response to sudden workload variations. In this regard, SDNs change the way traditional networks are designed and managed. Through the global awareness given by a centralized controller, SDNs can optimize the management of flows in response to events.

Feamster, Rexford and Zegura (2014) defines two key features of SDNs: i) it decouples the control plane (which decides how to handle the traffic) from the data plane (which forwards traffic according to decisions that the control plane makes); ii) an SDN consolidates the control plane, so that a single software control program controls multiple data plane elements. Such characteristics provide better visibility (decoupling) and control (centralized management) to perform tasks such as network diagnosis and troubleshooting.

In addition to SDNs, the OpenFlow protocol (MCKEOWN *et al.*, 2008) leverage network management by providing a programmable and standardized interface between data plane and control plane. Many vendors including, HP, NEC, NetGear, and IBM, produce OpenFlow-capable devices. Moreover, a standard interface between data and control planes avoids the implementation of several specific interfaces, therefore simplifying network operations while driving down hardware costs. Figure 12 presents a high-level SDN architecture.

Applications are programs that explicitly communicate and negotiate requirements with the control plane through one or more northbound interfaces, receiving updates about the desired network states. The controller is the central piece of an SDN architecture, translating and coordinating application requirements down to the data plane, by policies defined by SLAs (Service Level Agreements). Data plane nodes are devices that expose, through a southbound interface, the

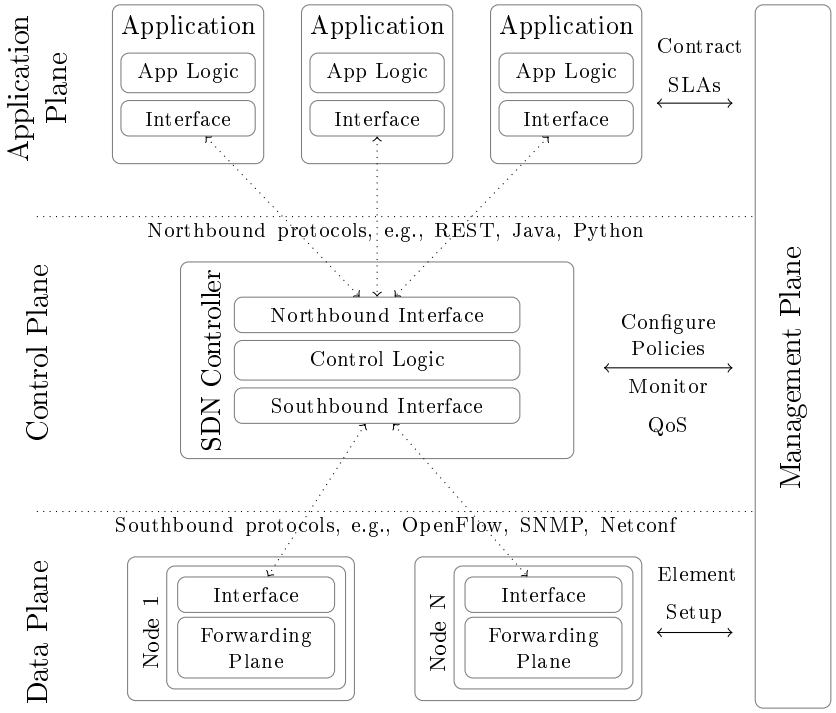


Figure 12: SDN Architecture. Source: (ONF, 2013).

control of its forwarding capacities and data processing capabilities. In traditional networks, forwarding capabilities are not managed directly by a controller. Usually, forwarding policies are sent to be updated in each separate device. Lastly, the management plane is responsible for controlling the relationship between client and provider, transforming high-level business requirements into low-level actions to be monitored and enforced at the data plane.

SDN introduces more programmability and flexibility to the control plane through a centralized management point, aware of the whole network. Thus, allowing the development of more sophisticated management techniques in an easier way compared to legacy networks. Thus, energy efficiency can be improved through high accuracy and flexibility of the data plane management in contrast with traditional networks. Accuracy can be achieved by eliminating device-by-device configurations, through a standard southbound interface, and flexibility through the logic implemented at the control plane, thus allowing single or multiple con-

trollers. Therefore, control decisions can be made based on a current and global view of the network state, rather than distributed and isolated as in traditional networks.

An SDN controller is a software platform where all the network control applications are deployed. SDN controllers. Controllers are the "brains"of an SDN environment, acting as the strategic control point in the network responsible for relaying information to the switches, routers, and other network devices via southbound APIs and the applications and business logic via northbound APIs.

Controllers in SDN have two operational modes, reactive and proactive. In the reactive approach, packets of each new flow coming to switch are forwarded to the controller to decide how to manage the flow. This method takes a considerable time while installing rules. The amount of latency can be affected by the resources of a controller, their performance, and the controller-switch distance. In the proactive approach, rules are already installed in the switches; therefore the numbers of packets that send to the controller are reduced. In this method, the performance becomes better and therefore the scalability. Evaluation of both approaches has been done in (FERNANDEZ, 2013), where a hybrid approach has been presented to gain the benefits of both reactive and proactive approaches.

Further, SDN controllers comprise a set of modules that can provide different network services, such as routing, topology management, host tracking and others. Barros *et al.* 2015 presented an evaluation of current SDN controllers outlining the key features of the described open source SDN controllers in Table 4:

The programming language used to build the controller platform is an important feature to be observed in the choice of the network controller as it directly influence metrics such as performance and learning curve. The controller performance is also determinant when choosing the correct platform for production

Characteristics	NOX (GUDE *et al.*, 2008)	POX (POX, 2009)	Ryu (Ryu, 2015)	Floodlight (Floodlight, 2015)	ODL (Linux Foundation, 2015)
Language	C++	Python	Python	Java	Java
Performance	High	Low	Low	High	High
Distributed	No	No	Yes	Yes	Yes
OpenFlow	1.0	1.0	1.0 - 1.4	1.0 - 1.3	1.0 - 1.3
Learning curves	Moderate	Easy	Moderate	Steep	Steep

Table 4: Comparison between SDN Controllers. Source: (BARROS *et al.*, 2015)

purposes, it can be affected by many factors, including the programming language, design patterns adopted and hardware compatibility. The learning curve is a fundamental metric to consider when starting a project, being important to measure the necessary experience to learn the SDN controller platform and build the necessary skills. Further, it directly influences the time to develop a project and also the availability of skilled developers.

3.3 Sustainability-Oriented System (SOS)

The SOS orchestration method is a previous work of our research group at LASSU. It has two primary objectives. First, it aims at coordinating different energy efficiency capabilities considering the possible combinations and conflicts among them, as well as the best option for a given bandwidth utilization and network characteristics (RIEKSTIN *et al.*, 2014). Secondarily, it examines the expression of business-level policies and its translation into device-level actions and configuration, increasing the automation level of the network management, rendering it less error prone and complex (RIEKSTIN *et al.*, 2015c).

SOS functioning consists of a few main steps. First, high-level policies given by a network administrator are refined down to network-level parameters. The refinement process takes into consideration the usage of table lookup techniques (RIEKSTIN *et al.*, 2015c). Following, the refined parameters are combined with distinct network conditions (e.g., time condition) to provide the best combination of capabilities to be enforced at a given point in time. In details, the refined

parameters are used as input to analytical solvers (one for each energy saving capability) that evaluate the best combination of capabilities for each network condition. As result, decision trees are generated. Then, the last step consists of configuring the decision trees and translated policies in GreenSDN. Figure 13 presents a high-level view of its functioning.

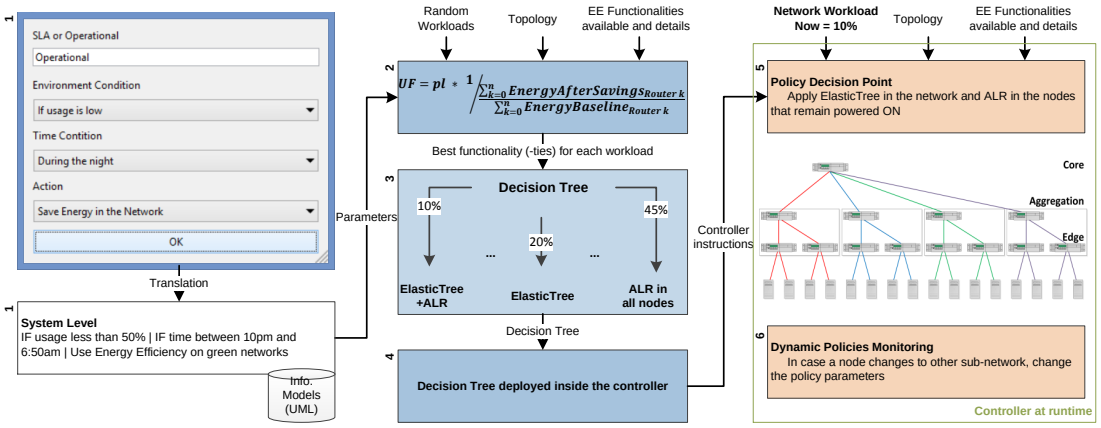


Figure 13: SOS orchestration method. Source: (RIEKSTIN *et al.*, 2014)

Illustrating the SOS functioning, a use case is presented in Figure 13. The numbers 1-6 represent the architecture module being used. An operational policy is translated, giving to the network level some information such as the environment conditions (it should save energy only if the network load is less than 50%), the period of the day the energy efficiency capabilities will take place (during the night), and the particular network or subnetwork in which they are going to be applied. It is assumed a fat tree topology with 1Gbps edges and aggregation nodes, and 10Gbps core nodes. The method takes as input power profiles considering the fixed and variable power spent and a set of random workload to train the method. These workloads are combined with a set of possible energy efficiency capabilities.

In the example, the possible capabilities considering the fat tree topology are ALR (Adaptive Link Rating), SC (Synchronized Coalescing), and ElasticTree (HELLER *et al.*, 2010). The possible combinations of the capabilities could lead

to conflicts during operation if one tries to put a node to sleep while the other is expecting this same node to be fully operational.

Considering a workload of 10%, a typical scenario during low-usage periods, ALR would reduce interfaces speed from 1Gbps to 100Mbps. Considering the expected savings, in our example, ALR would save approximately 20% of the original power demanded. SC, considering the experiments in Mostowfi and Christensen (2011) for the percentage of time the switch stays turned on as a function of load, the ON time for an amount of 10% is around 20%.

Executing ElasticTree in a 20% occupancy network scenario, the authors on Heller *et al.* (2010) reported 38% of energy savings (minimum spanning tree topology to ensure connectivity). This value would be similar for a 10% workload since the minimum spanning tree topology should be respected. The savings, in this case, are expected to be greater than with SC because the latter does not comprise a traffic engineering functionality while ElasticTree does. ElasticTree will relocate the traffic to allow more switches to sleep, thus saving more energy.

Besides, by coordinating ElasticTree energy saving capabilities with ALR, it is possible to potentially increase the savings ratio (reducing the link rate to 100Mbps). Therefore, the best option for this scenario is ElasticTree plus ALR. Dividing the possible savings with ALR for the whole network by the total number of nodes and then multiplying the result by the number of remained powered nodes after ElasticTree, the savings can reach 50% for a 10% load scenario and the given topology.

As SOS, the GreenSDN design takes into consideration the two modules: i) a decision point to coordinate capabilities based on one or more decision trees generated by SOS; and ii) a component that adjusts the current decision tree based on time and scenario changes (e.g., during the night use a decision tree DTB, if a node disconnects, change DTB).

3.4 Chapter Final Remarks

This chapter presented an overview of primary network platforms and concepts related to network management and SDN, considering the GreenSDN operation in conjunction with the SOS orchestration method. In the first part, it was presented briefly an overview of experimental network platforms to select a baseline environment to build GreenSDN. While Testbeds leverage large-scale experiments (e.g., Internet routing), emulators and simulators provide fast deployment and configuration of network experiments, which in alignment with our objectives. Thus, to enhance the development and configuration of the selected energy saving capabilities (and possible further management strategies), it was selected the Mininet network emulator with the POX controller.

Since there is an ongoing transition towards software-defined infrastructures, the second part outlines the relevance of combining SDNs and OpenFlow. While OpenFlow provides a standard/open interface between data and control planes, the SDN paradigm concentrate in a single management point a software abstraction that eases the network management and the development of energy saving capabilities at the control plane.

In the last part of this chapter we briefly presented the SOS orchestration method, which was built based on GreenSDN and published in (RIEKSTIN *et al.*, 2014; RIEKSTIN *et al.*, 2015c) and demonstrated in (RIEKSTIN *et al.*, 2015b). Based on the implementation of energy saving capabilities in GreenSDN, SOS coordinate the decision-making process on whether to enforce a single or a combination of capabilities given a network condition and a set of constraints provided by a network administrator.

The next Chapter presents the GreenSDN architecture based on the ONF (Open Network Foundation) standard architecture and the Mininet platform

using POX as a controller. Also, SOS is shown in the architecture as an application that coordinates the functioning of available energy efficiency capabilities.

4 GREENSDN

This chapter describes the technical details involved in the design and operation of GreenSDN considering the selected energy efficiency capabilities and worker modules that are required to the functioning of such capabilities. In this regard, we first introduce the architecture and its main components detailing how they relate to each other (Section 4.1) and then we present the full architecture and the development details in Section 4.2. Lastly, we summarize the chapter presenting the concluding remarks in Section 4.3.

4.1 Architecture

The architecture based on the SDN reference architecture (ONF, 2013) and encompassing the objectives and requisites is presented in Figure 14. It comprises four abstraction planes: i) data, ii) control, iii) application and iv) management:

- **i) data plane**: includes Open vSwitches (OVS) running in kernel mode to switch packets across the interfaces, and parallel links interconnecting each pair of nodes;
- **ii) control plane**: presents the worker modules of GreenSDN, such as the Topology Manager, QoS Services Monitoring, Power Emulation, Database Manager, and the SustNMS capability. Such modules, except SustNMS, are responsible for obtaining and preparing data for the management layer

(that effectively take decisions);

- **iii) application plane** comprises part of the SOS architecture and the Graphical User Interface (GUI) elements, such as the topology viewer, charts presenting the network utilization, and a screen to gather user parameters related to QoS and energy requirements; and
- **iv) management plane**: contains the modules responsible for managing the network, such as the Decision Enforcement.

Following, we present a high-level view of GreenSDN architecture and a workflow illustrating how components are related to each other.

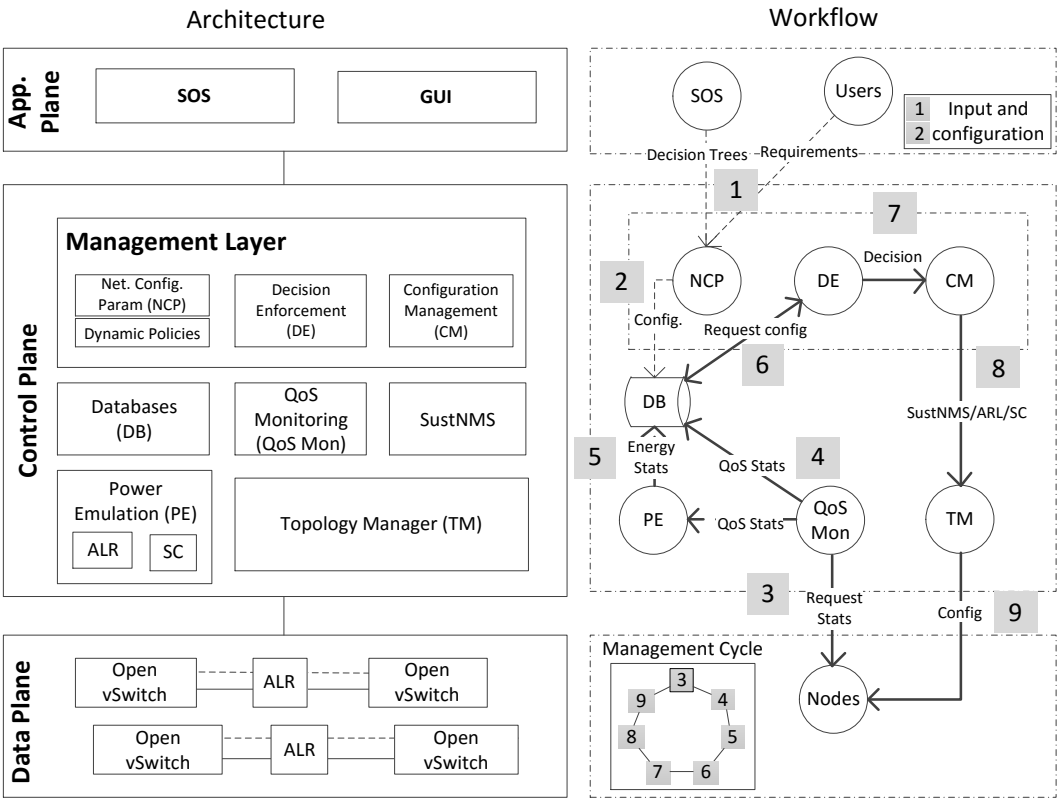


Figure 14: High-level view of the GreenSDN architecture and workflow.

The **first step** of the workflow requires input from SOS and user configuration parameters. Before the GreenSDN operation, SOS performs a training stage

to decide the best combination of energy efficiency capabilities to be applied. Then, SOS outputs a decision tree for each scenario given by a network administrator. Users may input QoS requirements and energy use to generate parameters to set usage policies in GreenSDN. In the **second step**, the NCP (Network Configuration Parameters) receives and parses parameters provided in the first step. Then, the NCP configures the parameters in databases to be queried by the DE (Decision Enforcement).

The **third step** is the beginning of the management cycle. In this step, the monitoring module not only detects when a workload is being sent on the network but also calculates QoS parameters (e.g. delay and jitter) injecting probe packets on the network. The **fourth step** is related to the organization and sending of the information to the PE (Power Enforcement) module and storage in the DB (Database).

In the **fifth step**, the PE receives usage statistics and calculates the energy consumed and saved by the network infrastructure and users. Further, the module stores the energy information in the DB (Database). After this, in the **sixth step** the DE (Decision Enforcement) collects usage statistics to assess whether it is necessary to adjust the network by configured policies. At this point, two different checks are performed. First is evaluated whether the user requirements are met, then, the defined actions in the decision tree given by SOS.

In the **seventh step** the CM (Configuration Management) carries out decisions determined by the DE. Although represented in the management plan, its operation is spread by features that produce changes in the network behavior, such as the energy efficiency features. In the **eighth** step, information related to changes be forwarded to the TM (Topology Manager). In the **ninth** and last step, the information is received by the TM and converted into corresponding actions through the OpenFlow. Then, the messages are forwarded to the nodes and ma-

nagement cycle is restarted. Figure 15 presents the full GreenSDN architecture. The full description is available in the Appendix A.

4.2 Full Architecture & Development Details

This section presents details of the main components of the GreenSDN architecture. First, it is described the configuration parameters of SOS and users (SLA settings) and how the system is configured, then as a key to enforce energy efficiency capabilities and QoS requirements, the QoS Service module is introduced. To calculate energy metrics is presented the Power Emulation module, that takes network statistics from the QoS Services and calculate the power usage. Lastly, it is described the implementation of the energy efficiency capabilities.

4.2.1 Configuration Parameters

This component is responsible for receiving and parsing user's requirements and the decision trees that are provided by the SOS orchestration method. SOS performs a training stage before the GreenSDN starts and requires a workload generator to create evaluate possible combination of capabilities for each scenario defined by an user. Also, it deploys the network topology graph and the business policies that are refined and deployed in the system.

In SOS, the refined policies determines when GreenSDN should switch a particular decision tree according to environment (e.g., if usage is high or low), time (e.g. during the nigh or during the day) or scenario (changes in the network graph) conditions. For instance, a network administrator may determine policies for each condition which are refined from a high-level/business policies into network commands. Then, a utility function is used to decide on the best combination of capabilities for each defined condition. As result, the SOS creates decision trees whereas each branch is equivalent to a particular workload and

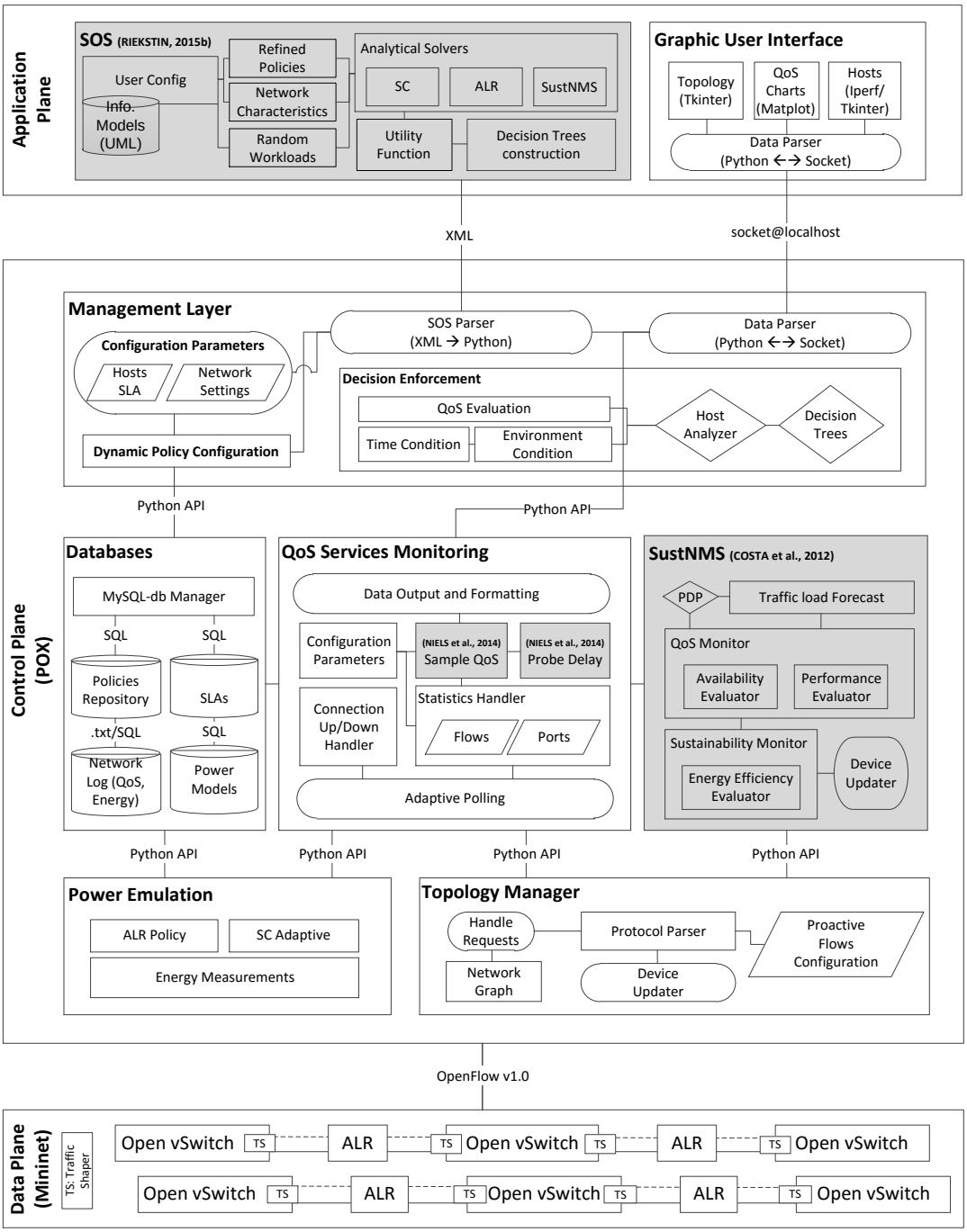


Figure 15: GreenSDN Architecture (gray related works used in GreenSDN).

each leaf, to a decision. Decision trees and refined policies are then deployed via XML (eXtensible Markup Language) inside GreenSDN.

User information describing acceptable QoS parameters and energy consumption thresholds are manually set into the database. QoS parameters are defined as: bandwidth (Mbps), delay (ms), Jitter (ms), packet loss (%). Among the QoS parameters, the bandwidth is the one who has a direct impact on the power consumption, meaning that, the higher the provisioned bandwidth on a set of nodes, higher is the probability of SLA violations. Energy parameter is determined as a maximum amount of Watts to be consumed. Table 5 presents the parameters.

SLA Type	**Bandwidth (Mbps)**	**Delay (ms)**	**Jitter (ms)**	**Packet Loss (%)**	**Max Energy (Watts)**
S1	B_1	D_1	J_1	J_1	W_1
S2	B_2	D_2	J_2	J_2	W_2
S3	B_3	D_3	J_3	J_3	W_3

Table 5: Table of Users Requirements.

Considering a network infrastructure comprising devices in which their consumption profiles are load proportional (i.e., their network devices present a load proportional energy consumption behavior), the maximum available bandwidth becomes a key parameter towards determining the overall power consumption. Therefore, we consider a hierarchy among the different plans, whereas $B_1 > B_2 > B_3$. As parameters such as Delay (D_i), Jitter (J_i) and Packet Loss (PL_i) relies on the user application, they are configured according to application requirements. The Maximum Energy (W), represents the amount of $Watts$ to be spent by the user. For instance, if a certain amount of energy is reached, the user can switch to more restricted SLA in terms of energy consumption, or renewal policies can be used to increase the number of Watts automatically to be spent. Figure 16 presents the thresholds for QoS parameters and energy consumption.

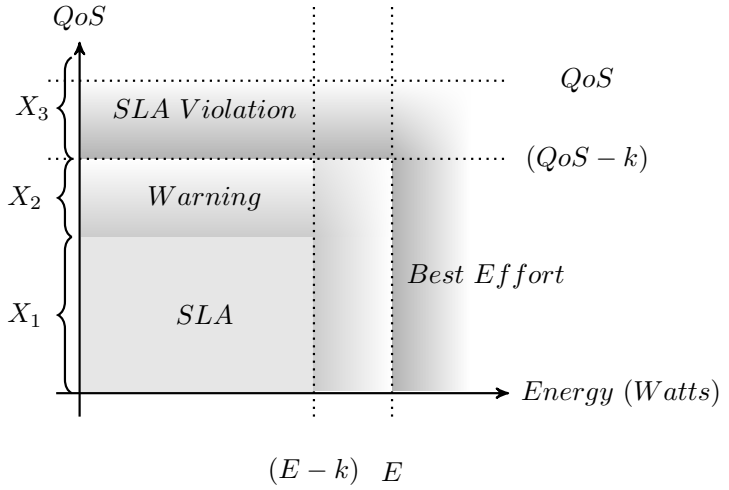


Figure 16: User's thresholds for QoS and Energy Consumption.

4.2.2 Topology Manager

The Topology Manager (TM) is a critical component in the GreenSDN functioning dealing with nodes and ports management, and installing or removing flows whenever a change is required by the control plane. It also defines switches and port states (e.g. standard operation mode, sleep, active ALR or SC) being an useful input to the Power Manager for calculating the energy being consumed in a certain point in time. Furthermore, the TM provides network information to the QoS Services component, such as hosts location and paths) and enforces topology changes that may be required by SustNMS. Also, it comprises two main functionalities: a) build a schematic description of the network modeling it as a graph, and b) install/modify forwarding rules for pre-defined users.

A schematic network view (network graph) is built by intercepting at the control-plane the LLDP (Link Layer Discovery Protocol)[1] messages triggered by a standard discovery component (provided by the POX controller). In addition to QoS user information, GreenSDN requires as input user network information such as the node and port in which the user is connected, and IP/MAC addresses.

[1]LDDP (Link Layer Discovery Protocol): messages to discover links between the OpenFlow switches.

Based on such information, forwarding rules are reactively installed whenever a workload is sent.

4.2.3 QoS Services

The QoS Services is responsible for collecting and providing network statistics. To query information from nodes regarding flows and ports usage, the message $OFPT\_FEATURES\_REQUEST$ is used to request the properties of $OFPC\_FLOW\_STATS$ and $OFPC\_PORT\_STATS$. These are messages of the OpenFlow protocol used to request, respectively, statistics of flows and ports of a particular node. However, there are several ways to request data plane information. In this regard, it is presented in subsection 4.2.3.1 a strategy to optimize the query for node's statistics. Further in subsection 4.2.3.2 is shown how per-user statistics are collected.

4.2.3.1 Dynamic Polling

A straight polling of all nodes (i.e., request for statistics on a fixed time interval) although precise, has the potential to generate significant amounts of control traffic and consequently to increase overall network energy consumption. It can be addressed in a similar way as Adrichem, Doerr, Kuipers (2014) and Chowdhury et al. (2014) , by adapting the polling time interval and the number of nodes in the "query-list". Initially, to detect incoming workload, only edge-nodes are queried at a fixed time interval. Based on Adrichem, Doerr, Kuipers (2014) , according to the throughput, the polling time interval can be decreased (down to a lower bound time) or increased (up to an upper bound). The result got based on the RNP topology (presented in the Experimental Evaluation Chapter 5) can be observed in Figure 17.

As the topology contains seventeen nodes and four border nodes (with hosts

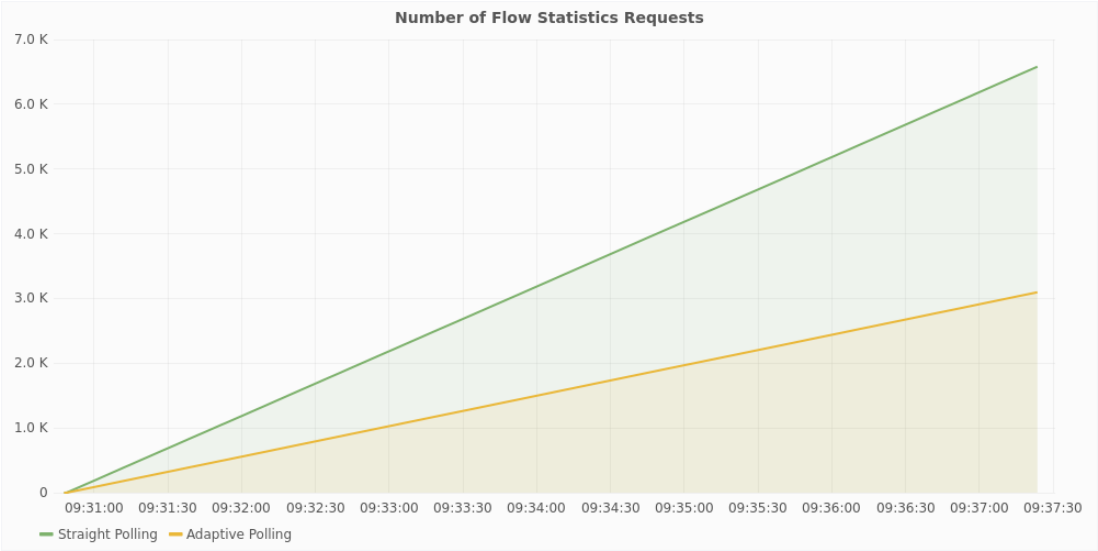


Figure 17: Adaptive polling vs straight polling.

connected), a simple measurement without workload presents the difference between the number of messages requested to the data plane using a straight polling and the adaptive polling. Once a workload is detected, nodes in the path are also queried. Paths are obtained either from the initial flow instantiation or from the SustNMS output, in case it is active. Also, based on historical usage of flows, it is possible to increase the expiration time from individual flows that are more frequently utilized to optimize the flow-tables (a problem known as finding Hierarchical Heavy Hitter flows, explored in (JOSE; YU; REXFORD, 2011)).

4.2.3.2 Collecting Per-User QoS Statistics

To precisely match user packets and account for network statistics a simple MAC-based flows instantiation is used. However, this has the potential to flood flow-tables since if there are N active users in the network, it is possible to have N^2 flows. Therefore, based on prior knowledge of user routes, it is possible to use two distinct rules to forward flows. One particular for edge nodes, specifying source and destination MAC address, i.e., if exists source and destination MAC addresses, then forward to an out_port), and other less specific for interconnec-

tion nodes (specifying only the target MAC).

Moreover, to proportionally account, the network statistics is required to count the number of shared nodes between the users. Once power models are represented by a fixed (CPU, memory, fans) and variable (interfaces) parts, it is required to 'split' the fixed power consumption parts among users sharing that node. By crossing the paths of active users, it is possible to account for shared nodes. Thus, to keep track of the common nodes (i.e., nodes being used by multiple active users), a dictionary of counters is maintained. Once a user is active, nodes in his/her path are incremented, otherwise decremented.

To obtain QoS parameters such as Delay and Jitter probe packets are injected using the OpenFlow capabilities. The delay is measured by calculating the difference between the packet's departure and arrival times, subtracting the estimated latency from the switch-to-controller delays. Whenever a workload is detected for each user, and given the prior knowledge of their active routes, probe packets with a timestamp in its payload are injected to the destination.

4.2.3.3 Delay

Upon the packet arrival, another time-stamp is used to compare with the 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. The total path delay is given by: $t_{delay} = (t_{arrival} - t_{sent} - \frac{1}{2}(RTT_{s1} + RTT_{s2}))$. Despite creating additional traffic (OpenFlow 1.0 does not allow to tag packets), injecting probe packets is the most certain strategy to infer the path delay in OpenFlow 1.0 without external instrumentation. Figure 18 illustrates the delay measurement strategy.

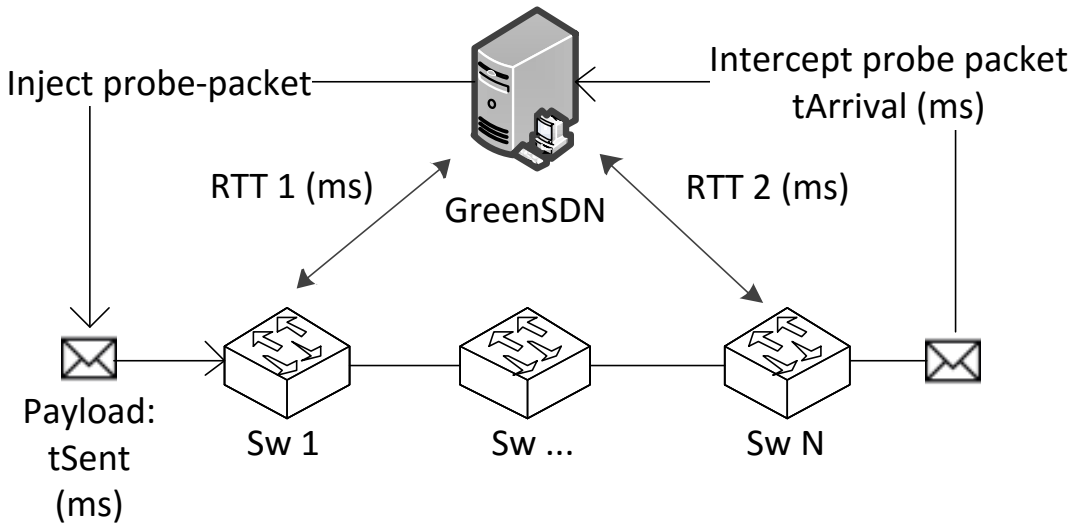


Figure 18: Delay measurement with OpenFlow 1.0.

4.2.3.4 Jitter

Jitter is obtained by calculating the absolute value of the difference between consecutive samples of delay. Given at least two delay samples $delay = [t_i, t_{i+1}, ..., t_k]$, jitter is calculated as the average of absolute values of $\sum_{t=1}^{T_k}(t_i - t_{i+1}) + (t_{i+1} - t_k)$ in a period of measuring time.

4.2.3.5 Packet Loss

Per-flow packet loss is estimated by polling flow statistics from the source and destination node of each path. It follows the delay probe-packet using control flags to detect when to subtract polling statistics from destination and source. When the packet is sent, a flag is marked. Upon the arrival at its destination, another flag is marked, and when both are marked the packet loss is calculated. It is done by subtracting the increase of the source switch packet counter from the increase of the packet counter of the destination switch.

4.2.4 Power Emulation

Virtual switches such as OVS have no capabilities to provide power consumption information neither at the port level nor the overall process level. As the SDN controller has an inventory of all switches in the network, the environment applies a power management application (Power Emulation) using power models as a way to parameterize energy consumption by the network utilization. Two types of power profiles were defined (in Chapter 2) to simulate real equipment: a load proportional (*AdaptivePower*), that is more energy efficient and desirable, and a constant (*StaticPower*), most common in legacy equipment, with a constant energy consumption independent of the workload.

Based on measurements and values described in (JANUARIO *et al.*, 2013; RICCA *et al.*, 2013; RICCIARDI *et al.*, 2011) the functions described in Equations (4.1) and (4.2) were considered for powered and sleeping nodes, respectively. The $Power_{chassis}$ energy consumption when the node is active is 200 Watts, whereas when internal components are powered off it is 120 Watts. Energy consumption from interfaces is given by the fraction $\left(\frac{500}{30}\right)$ Watts varying according to the workload.

$$PP_{on} = 200 + \left(\frac{500}{30}\right) * workload \tag{4.1}$$

$$PP_{sleeping} = 120 \tag{4.2}$$

The power model for active nodes is combined with ALR savings (15%), as presented in Equation (4.3). For SC, the power model with the time the SC is *on* or off as described by (MOSTOWFI; CHRISTENSEN, 2011) in Equation (4.4).

$$PP_{ALR} = PP_{on} - 15\% \tag{4.3}$$

$$PP_{SC} = PP_{sleeping} * \left(\frac{tOn}{DutyCycle} - tOn\right) + PP_{on} * tOn \tag{4.4}$$

4.2.5 Per User Energy Measurement

The energy measurements module calculates the energy consumption and savings from users considering node's states to apply a particular power model, and the user workload. Information on current workload and path are received as input from the QoS Services module. Then, the component checks the states of nodes in the user path, applying a particular power model. Energy consumption and savings are calculated based on the following models:

$$PP'_{on} = \left(\frac{200}{NumUsers}\right) + \left(\frac{500}{30}\right) * UserWorkload \tag{4.5}$$

$$PP'_{sleep} = \left(\frac{120}{NumUsers}\right) \tag{4.6}$$

$$PP'_{ALR} = PP'_{on} - 15\% \tag{4.7}$$

$$PP'_{SC} = PP'_{sleep} * \left(\frac{tOn}{DutyCycle} - tOn\right) + PP'_{on} * tOn \tag{4.8}$$

Considering that several users can share same nodes, their consumption can be obtained by splitting the fixed[2] consumption part among users sharing the node. In this regard, Equation 4.5 is applied when a node is active, Equation 4.7 when ALR or SC is being used. To measure the **energy consumed** by a user, the following models are considered:

[2]Representing the internal components such as the forwarding engine.

$$A_c\ (W) = \sum_{i=1}^{\mathrm{N}_{switches}} PP'_{on}\,(W_{user}) \tag{4.9}$$

A_c: consumption from nodes powered on

$$B_c\ (W) = \sum_{i=1}^{\mathrm{N}_{switches}} PP'_{(ALR\ or\ SC)}\,(W_{user}) \tag{4.10}$$

B_c: consumption from nodes enforcing ALR or SC

$$C_c\ (W) = \sum_{i=1}^{\mathrm{N}_{switches}} PP'_{sleep} \tag{4.11}$$

C_c: consumption from nodes sleeping

$$D_c\ (W) = A + B + C \tag{4.12}$$

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 achieved by comparing their consumption with the maximum workload allowed in the network (W_{max}). Energy savings per user is measured as follows:

$$A_s\ (W) = \sum_{i=1}^{\mathrm{N}_{switches}} PP'_{on}(W_{max}) - A_c \tag{4.13}$$

A_s: savings nodes powered on

$$B_s\ (W) = \sum_{i=1}^{\mathrm{N}_{switches}} PP'_{on}(W_{max}) - B_c \tag{4.14}$$

B_s: savings nodes enforcing ALR or SC

$$C_s\ (W) = \sum_{i=1}^{\mathrm{N}_{switches}} PP'_{on}(W_{max}) - C_c \tag{4.15}$$

C_s: Savings nodes sleeping

$$D_s\ (W) = A + B + C \tag{4.16}$$

D_s: Sum of savings

$$S\ (\%) = (D_s * 100)/D_s \tag{4.17}$$

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, the Algorithm 1 presents the consumption measurements.

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

Input: active_hosts ⟵ dictionary of active users
Input: sharedNodes ⟵ dictionary of shared nodes
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
```

4.2.6 Energy Efficiency Capabilities

This Section presents implementation details of the energy efficiency capabilities. ALR is presented in Subsection 4.2.6.1, SC in Subsection 4.2.6.2, and SustNMS in Subsection 4.2.6.3.

4.2.6.1 Subsystem Scope: Adaptive Link Rate

The ALR capability uses a policy and a mechanism to adjust the link rate. The policy decides when to change the link rate and the mechanism effectively switch the rate. As originally proposed by the author's, mechanisms and policies

should be inherent to the network device. However, as we use standard OVS (Open vSwitch) nodes, and OVS does not provide native support for changing link rates, a different strategy has to be implemented to emulate the link rating capability.

To emulate the link rating functionality, we defined parallel links configured with different maximum rates between each pair of nodes, as represented in Figure 19.

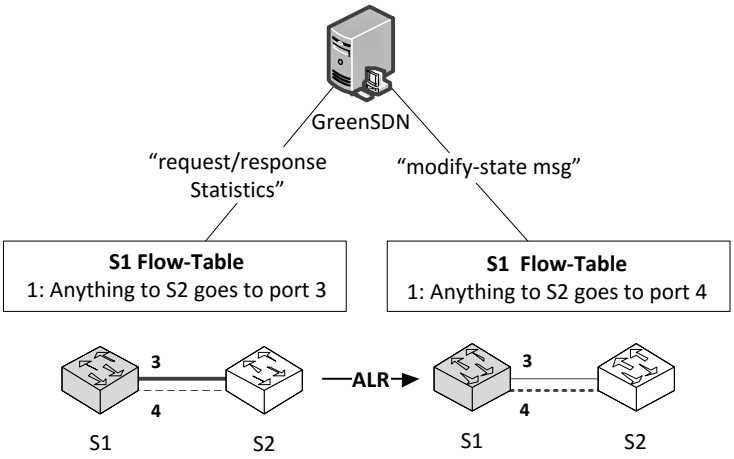


Figure 19: ALR Emulation Scheme. Parallel links with different forwarding rate interconnecting each pair of nodes.

The standard link (i.e., ALR deactivated) represented by the continuous line was configured with a 30 Mbps maximum rate limit and a parallel link represented by the dotted line set with 10 Mbps maximum speed. Only one of these links forwards traffic at a given point in time. Algorithm 11 presents the mechanism implementation that switches the links.

The ALR dual threshold policy is implemented inside the Decision Enforcement, which receives requests to enable/disable ALR and determine the best moment when to reduce or increase the link rate. Furthermore, the standard path was configured to use the odd ports of the node, while the alternative path was set to use the even ports. Every time ALR is enabled, it is incremented the

Algorithm 2: Mechanism to activate or deactivate ALR in a certain port

```
   Input: target_node, target_port ⟵Node and port in which ALR will
          be activated or deactivated
   Input: ALR_command ⟵activate or deactivate
   Output: OpenFlow message to activate or deactivate ALR
1  begin
2      for each node ∈ active_nodes:
           /* Search in the set of active nodes the target          */
3          if node == target_node:
4              for each port ∈ node:
                   /* Search the targeted port within the node      */
5                  if node.port == target_port and
                   port is not attached to host:
                       /* Activate (forward to an even port) or deactivate
                          (forward to an odd port) ALR              */
6                      if ALR_command == True and node.port is odd:
7                          even_port ← port + 1
8                          msg = OpenFlowMsg ← even_port
9                      elif node.port is even:
10                         odd_port ← port − 1
11                         msg = OpenFlowMsg ← odd_port
```

out_port on all the rules associated with the node. The same process is executed to disable ALR, but decrementing *out_port*, so that the traffic is forwarded to the normal path.

4.2.6.2 System Scope: Synchronized Coalescing

The Synchronized Coalescing is a system scope capability demanding the coordination of all the subsystem scope capabilities in a node. As originally proposed, SC coordinates the Low Power Idle (LPI) modes on all interfaces of a node when it is active and coalesce incoming packets to put in sleep mode internal components of a node, when a duty cycle is reached coalesced packets are sent in bursts. However, there is no way to implement LPI without altering the Open vSwitch functioning, since it does not natively support some capabilities required by SC, such as the LPI and traffic bursts.

An alternative for traffic bursts support would be to use the control plane to intercept packets when SC is active via Packet-In messages and resend them in bursts, emulating the buffer/queuing functionality, but it is not feasible since the controller cannot handle all the data plane traffic and this would introduce enormous latency in the network. The adopted approach was to simulate the energy-related effects of SC as an application on the controller based on information from the power models, as depicted in Figure 20.

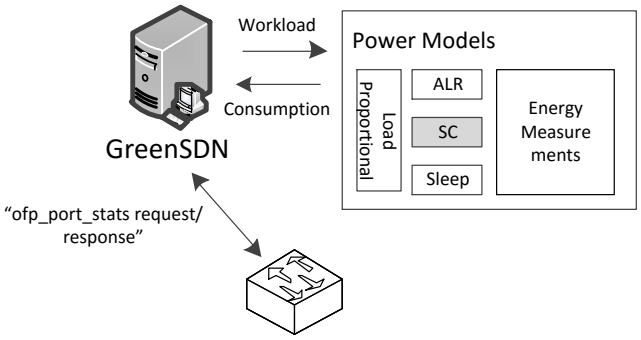


Figure 20: Synchronized Coalescing emulated through power models.

Therefore, when the workload in a node is below a certain threshold, its energy consumption is calculated through a power model. The adaptive part of SC is implemented as described in Algorithm 13. It has as objective to check if the number of packets per second, is higher than a threshold. If so, SC is disabled,

and the node operates in the standard mode to handle the incoming workload without losing packets. While the number of packets per second are still lower than the buffer capacity, incoming packets are coalesced to maximize savings.

Algorithm 3: Synchronized Coalescing simulation through power models.

```
   Input: set of active nodes, port_statistics
   Input: tOn ⟵ duration of the period with the device active (ms)
   Input: DutyCycle ⟵ percentage of cycle time the device must be active
          tOff ⟵ (tOn / DutyCycle) - tOn
   Output: Nodes with SC on Consumption in Watts
1  begin
2      node ⟵ event.connection.dpid
       /* Traverse the list of active nodes to check wheter enable or
          disable SC                                                  */
3      for node in active_nodes:
4          port_list ⟵ port_statistics[node]
           /* Traverse the list of ports statistics to accumulate the rx
              workload                                                */
5          for rx_workload in port_list:
6              packets/second ⟵ packets/second + rx_workload
           /* Threshold verification                                  */
7          if packets/second >= qHigh:
8              SC[node] ⟵ OFF
9          elif packets/second <= qLow:
10             SC[node] ⟵ ON
11             WattsON ← power consumption node ON
12             WattsSLEEP ← power consumption node SLEEP
               /* DutyCycle, 50% on and 50% off                       */
13             Energy ← WattsON * tOn + WattsSLEEP * tOff
```

4.2.6.3 Network Scope: SustNMS

SustNMS demands the predefined routes to be efficient. The idea of the algorithm is to perform green traffic engineering considering all flows being executed at a given moment T, the set of alternate routes to be used, and the set of switches to be in sleep mode based on a set of predefined routes. In the context of GreenSDN, only the forwarding capabilities of SustNMS are required. Therefore, its architecture was simplified. For instance, the Switch/Router component, previously designed for SNMP/MIB-based devices, is not necessary once SDNs decouple the forwarding logic from devices. Figure 21 presents the SustNMS architecture within the GreenSDN scope (details in Chapter 4 - GreenSDN Design).

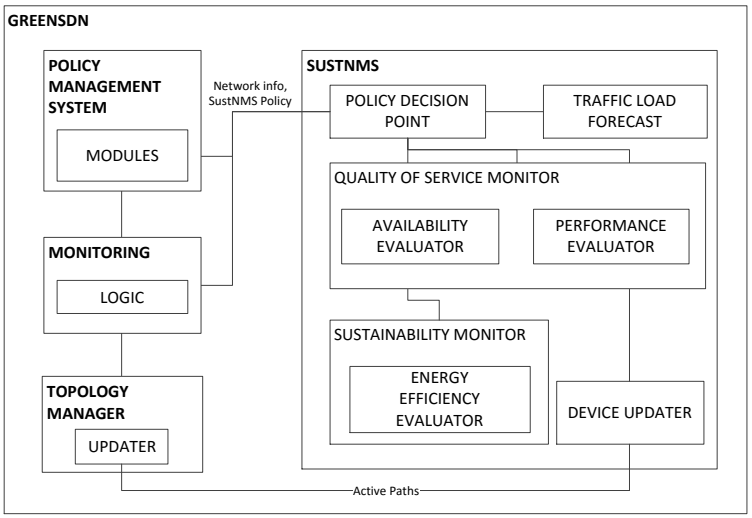


Figure 21: SustNMS architecture adjusted to GreenSDN.

In this architecture, SustNMS operates as a component that receives network information and returns a set of paths to be used. It receives as input predefined routes from GreenSDN defining main and alternative routes. The functioning of the QoS monitor is preserved. However, it receives network information (e.g. current workload on predefined routes) directly from the GreenSDN monitoring component. Moreover, the input contains high-level policies to be enforced by SustNMS by the network state. The SustNMS device updater, responsible for

setting active paths through SNMP commands, was altered to return the set of current paths instead of enforcing the routing decision.

4.3 Chapter Final Remarks

This chapter presented the GreenSDN architecture comprising the development details of its core modules (i.e., monitoring, topology management) and the energy saving capabilities. The architecture was inspired by the standard ONF architecture aiming to provide a separation between the control and management planes to leverage the development of independent applications and management strategies based on energy efficiency capabilities. As example, the SOS orchestration method was deployed on top of the GreenSDN environment.

The energy efficiency capabilities were designed to operate independently upon an "enforcement"decision by the Decision Enforcement module[3]. It receives network statistics and analyzes which capability to enforce based on predefined rules. In our current deployment, SOS was in charge of the decision about when (the best moment) and how (one or a combination) to enforce the energy efficiency capabilities given a set of policies.

The Adaptive Link Rate (ALR) capability (subsystem scope) was built using a combination of emulation and simulation. The link rating functionality was emulated using parallel links (configured with different Ethernet rates) interconnecting each pair of nodes, and the mechanism to effectively change the rating) was simulated using forwarding policies to route traffic through an individual link depending on current workload. To decide when to change the ALR mechanism, a dual threshold policy (upper and lower thresholds) policy was declared in the Decision Enforcement module, or, it is activated when required by the SOS orchestration method.

[3]The first version of GreenSDN was published by the author in Rodrigues *et al.* 2015

Syncronized Coalescing, was implemented using a simulation strategy once it requires the implementation of functionalities not natively supported by Open vSwitches (OVS), such as packet coalescing, traffic bursts and the Low Power Idle (LPI) capability. In this regard, energy savings from SC were simulated in the control plane using power models. The network scope capability, SustNMS, was adjusted from its original SNMP-based architecture to operate in GreenSDN. However, it still maintains its main functionalities to find the best route given a set of predefined tunnels and a routing policy (performance or sustainability).

The components to collect network statistics and calculate energy consumption based on a fine-grained measurement of statistics. Energy consumed and saved is calculated using power models describing the behavior of the network nodes under different circumstances, such as active, sleep mode and enforcing an energy saving capability. Then, based on mathematical models it was calculated the energy being consumed and saved by each user based on the network state.

5 EXPERIMENTAL EVALUATION

This chapter presents an experimental evaluation of the GreenSDN development. The first Section 5.1 describes the environment configuration, including the characteristics of the physical machine, the topology, and configured flows. Section 5.2 presents results on the evaluation of energy efficiency capabilities being executed without the SOS orchestration. Then, in Section 5.3 are given results of energy efficiency capabilities being orchestrated by SOS. An per user evaluation of energy consumption and savings is performed at Section 5.4. Lastly, concluding remarks are presented in Section 5.5.

5.1 Testing Environment

The host machine used was an Intel Core i5-3570 @ 3.40GHz with 8 GB RAM. For deploying the processor in its full capacity, the processor's low power mode (C-States) was disabled. The SDN network was emulated in Mininet, and the GreenSDN was based on the POX controller running OpenFlow 1.0. Network traffic is generated through the Iperf, already available in Mininet. The topology implemented was inspired on the 10 Gigabyte RNP (*Rede Nacional de Ensino e Pesquisa*)[1] backbone. Figure 22 presents the topology. Each pair of nodes was interconnected using parallel links, which were configured with different rate limits. Standard links were set to handle a maximum traffic of 30 Mbps, and

[1]The Brazilian National Research and Education Network

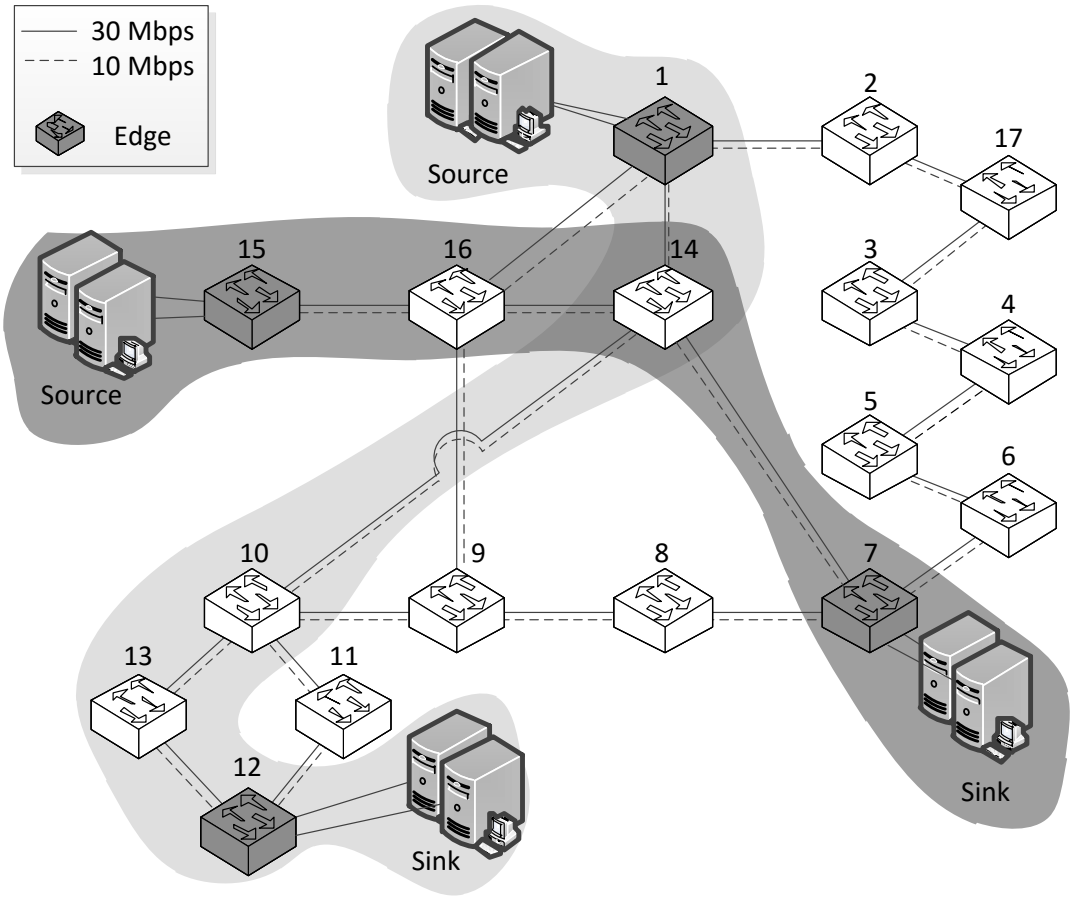


Figure 22: Topology inspired by the RNP. Figure from (RODRIGUES *et al.*, 2015).

ALR links at 10 Mbps. To send data across the network, GreenSDN considered two main flows. From North to South and West to East, placing two *Sources* in North extremes and two *Sinks* in southern extremes. The generation of traffic between the hosts was in charge of the Iperf tool, which is already available in Mininet. In the experiments, a load proportional Power Profile (PP) was used for all nodes based on (JANUARIO *et al.*, 2013) as described previously (Equations 4.1, 4.2, 4.3 and 4.4).

5.2 Energy Efficiency Capabilities

This Section presents the analytical evaluation of thresholds for the ALR (Subsection 5.2.1) and SC capabilities (Subsection 5.2.2), followed by emulated

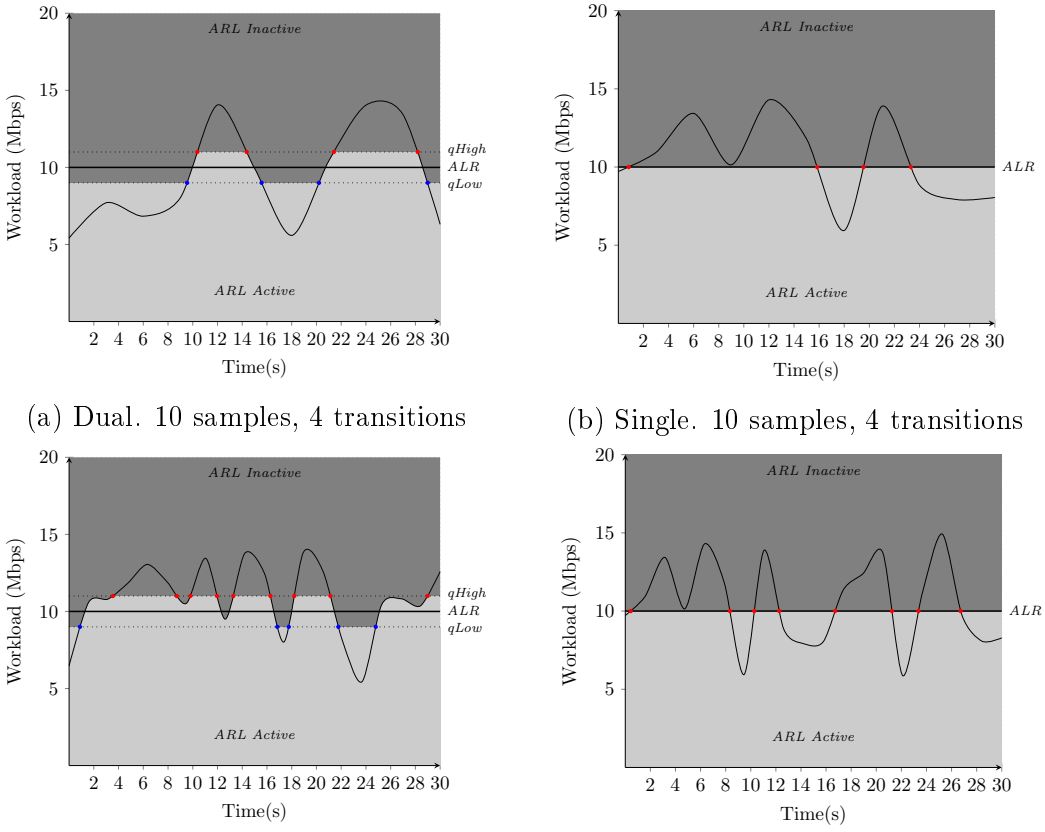
evaluation of all three capabilities operating in GreenSDN (Subsection 5.2.3).

5.2.1 ALR Threshold Evaluation

To evaluate ALR policies it was considered an analytical solver to activate/deactivate ALR based on the dual policies and the single policy. As ALR is configured to operate with current Ethernet rates, the main threshold was configured with 10 Mbps rate and two auxiliary thresholds $qHigh$ and $qLow$ respectively configured with 11 and 9 Mbps. Thus, when the incoming workload is below $qLow$ the ALR is activated, and it is deactivated if and only if the workload exceeds $qHigh$, likewise the same occurs when the workload is above $qHigh$. The single threshold policy is a simple threshold in which ALR can be activated or deactivated. We evaluate both policies to contrast the difference between both implementations.

To generate workload it was considered the function $f(x) = Ax + C$ in which x is a random number between -1 and 1 and two constants A and C. Therefore, to increase the variability of the random number this number was multiplied by a constant A; and to maintain this variability near the ALR threshold, a constant C was added that is equivalent to the ALR threshold. Furthermore, to evaluate the latency of activating/deactivating ALR it was considered the values presented in Schlenk *et al.* (2013), which states that for subsystem level capabilities the latency for adjusting internal components of a device ranges from $10\mu s$ to $20ms$. As in GreenSDN the mechanism involves the control and data planes, it was considered the higher value, $20ms$ to change the link rate. Figure 24 and Figure 24 presents the experiment considering the following values $A = 5$ and $C = 10$ to generate workload.

The experiment considered four different workloads varying (increasing) the number of samples, in which each run was configured to 30 seconds. By increasing



(a) Dual. 10 samples, 4 transitions

(b) Single. 10 samples, 4 transitions

(c) Dual. 20 samples, 5 transitions

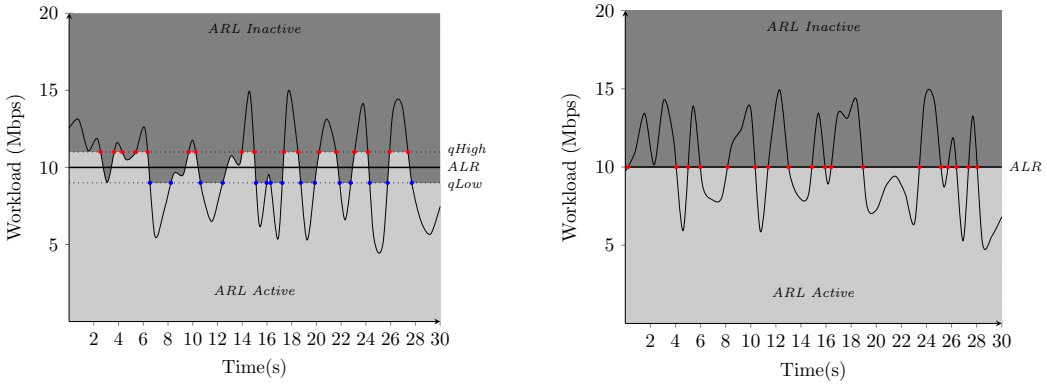
(d) Single. 20 samples, 8 transitions

Figure 23: ALR analytical dual versus single threshold evaluation using 10 and 20 samples

the number of samples per second, it is expected to highlight the relevance of the dual threshold policy against the single threshold policy. In the scenarios a) and b) of Figure 24, were generated ten samples (calls to the defined function) during 30 seconds of the experiment, which was equivalent to 0.33 workload samples per second. As a result, the experiment a) presented low workload variation performing only four transitions of the ALR mechanism for both policies, and thus a good scenario using single threshold policy (scenario b) due to the low workload variation.

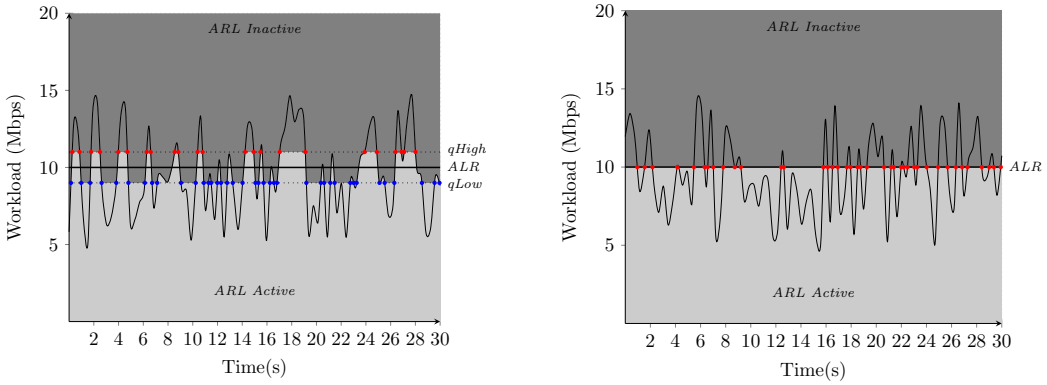
The number of transitions represents the amount of times in which the ALR capability is activated or deactivated.

In the scenario b) (Figure 24) the number of samples was duplicated (0.66 samples per second), and the number of ALR transitions was similar to the sce-



(a) Dual.40 samples, 12 transitions

(b) Single. 40 samples, 18 transitions



(c) Dual. 80 samples, 22 transitions

(d) Single. 80 samples, 40 transitions

Figure 24: ALR analytical dual versus single threshold evaluation using 40 and 80 samples

nario a) due to the randomness of the generated samples, and the number of changes of the single policy threshold (5 transitions) was slightly higher than the dual threshold (8 transitions). However, in the scenario a) depicted in Figure 24 this difference increases, not as significantly as in the scenario c). With more than two samples per second (2.66 samples) the difference between the dual and single threshold policies was almost duplicated (22 dual thresholds and 40 single thresholds). Table 6 presents a summary of the impacts on delay due to the ALR mechanism considering $20ms$ delay to adjust the link rating (SCHLENK *et al.*, 2013).

Despite being important to characterize the generated workload, the number of samples per second does not have a direct relation to the number of ALR transitions. This because the randomly generated workload may often remain

Scenario	Samples	Samples/ Second	Latency Samples/ Second	Policy	Num. Transitions	Aggregated Latency
A	10	0.33	$20ms$	Dual	4	$80ms$
				Single	4	$80ms$
B	20	0.66	$20ms$	Dual	5	$100ms$
				Single	8	$160ms$
C	40	1.33	$40ms$	Dual	12	$240ms$
				Single	18	$360ms$
D	80	2.66	$60ms$	Dual	22	$440ms$
				Single	40	$800ms$

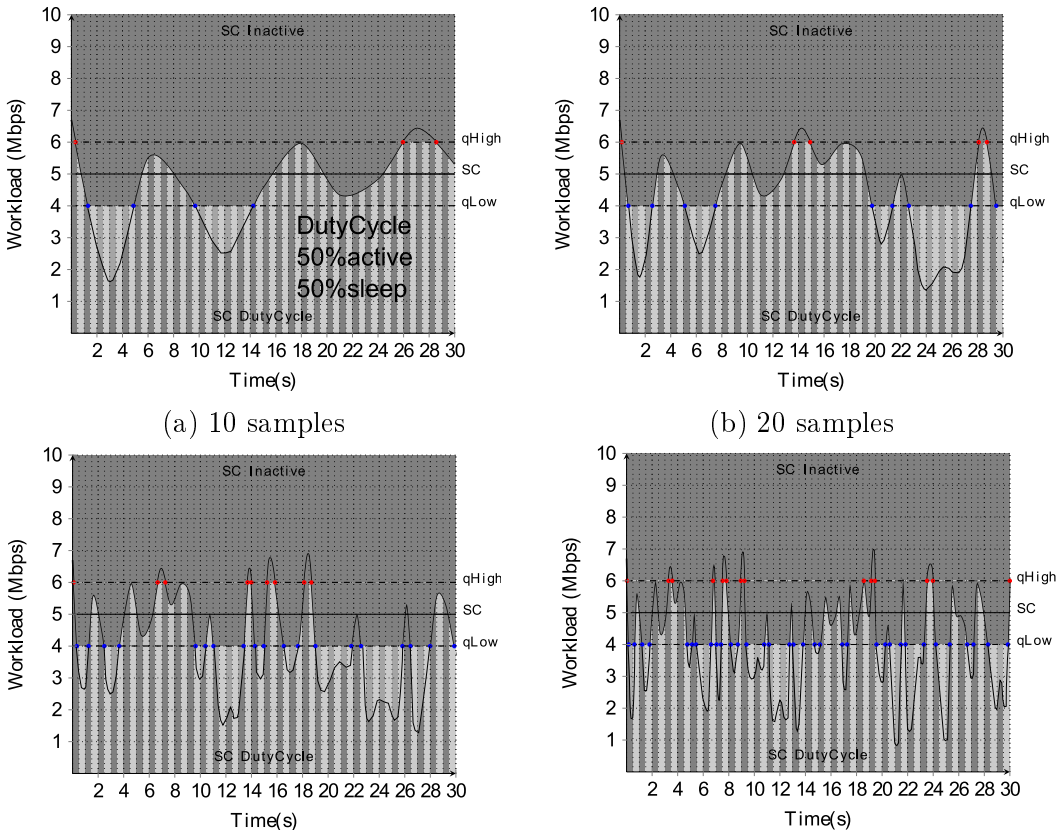
Table 6: Estimated impact of ALR transitions on latency.

below or above a certain threshold for some samples in sequence, as it can be observed in Figure 24 scenarios a) and c). However, as it was seen in the Table 6, the number of transitions has a direct impact on the latency to adjust the link rating. For the analytical evaluation the reference value per change of $20ms$ was considered, however in GreenSDN this value might be significantly higher due the simulation of the ALR policies in the control plane, and the emulation of the ALR mechanism in the data plane (with parallel links). Switching the link rate implies in adding the monitoring latency and the effectively switching the forwarding rule in the data plane, which according to Schlenk *et al.* (2013) may increase the latency to the level of network capabilities (ranging from $400ms$ to $1s$).

5.2.2 SC Threshold Evaluation

Similar to ALR, an analytical solver was adopted to activate/deactivate SC including the dual threshold approach. The experiment considered a function $F(x) = Ax+C$ with x as a random number between -1 and 1. SC dual thresholds were defined as 5 Mbps for the main SC threshold, and 6 Mbps for $qHigh$ and 4 Mbps for $qLow$. Therefore, A and C constants were defined as $A = 3$ and $C = 4$. Figure 25 presents a SC evaluation.

As the dual threshold has proven to be a more efficient strategy than the



(a) 10 samples

(b) 20 samples

(c) 40 samples

(d) 80 samples

Figure 25: SC threshold evaluation using workload F(x) = 3x+4 in which x is a random number between -1 and 1.

single threshold, the same strategy was implemented in SC. Furthermore, the main difference between the ALR and SC experiment is how energy consumption is calculated when SC is active with the DutyCycle. As SC was simulated instead of emulated, its functioning in GreenSDN, becomes simpler.

5.2.3 Individual Evaluation of Capabilities in GreenSDN

Results, presented in (RODRIGUES *et al.*, 2015), showed the energy consumption for two runs with different workloads to three capabilities and the baseline consumption (without savings). Table 7 presents the settings used in the experiments.

Since SC and ALR are capabilities intended to operate with lower workloads

Run	Time (s)	Workload (Mbps)	Path
1	90	10	[15-16-14-7] [1-14-10-13-12]
2	90	30	[15-16-14-7] [1-14-10-13-12]
SC Configuration			
tOn(ms)	11 (MOSTOWFI; CHRISTENSEN, 2011)		
DutyCycle(%)	50		

Table 7: Settings of the energy efficiency capabilities experiment.

(equal or less than 10 Mbps), the first run was configured to send two flows at 10 Mbps. The second run, sending 30 Mbps, was chosen to verify the SustNMS behavior. Given that the two pre-configured flows share node 14, and to avoid losses for workloads higher than 15 Mbps, SustNMS should adjust the network paths. Figure 26 shows that in the experiment with 10 Mbps savings of ALR and SustNMS were similar, in which both presented savings around 15% with a small difference. The baseline consumption represents the ordinary network operation, with all switches operating using the regular power profiles. However, for the 30 Mbps evaluation, as observed in Figure 27, the results were different.

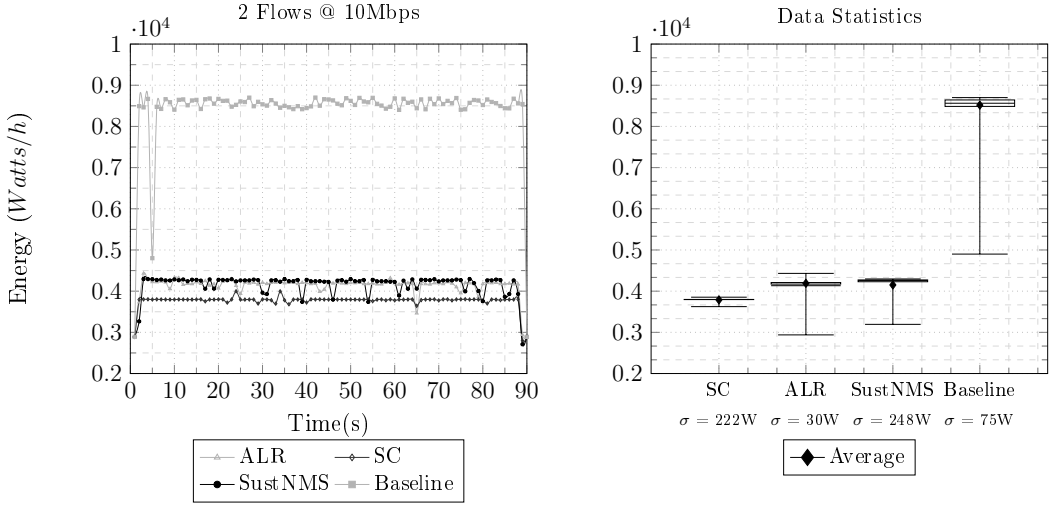


Figure 26: Energy consumed by energy efficiency capabilities and a baseline scenario (active nodes in standard mode of operation). Two flows sending 10 Mbps.

Aware of links capacity, SustNMS modified the routing for workloads gre-

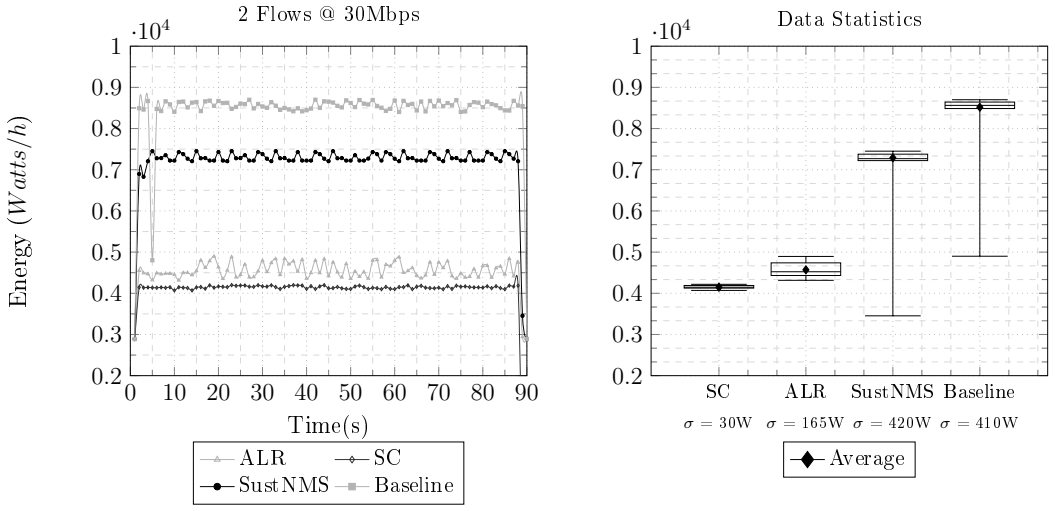


Figure 27: Energy consumed by energy efficiency capabilities and a baseline scenario. Two flows sending 30 Mbps.

ater than 15 Mbps to avoid losses due to flows sharing switches, such as node 14 of Figure 22. For workloads smaller than 15 Mbps, the 10 Mbps evaluation, SustNMS maximized the savings concentrating traffic on the defined flows. Considering that ALR is intended for Ethernet speeds, and working with 30 Mbps link capacity, the experiment operated as expected. ALR presented savings in 10 Mbps and massive packet losses in 30 Mbps evaluation (65%). The SC capability was the most aggressive functionality regarding savings; in the evaluation it was configured with a *DutyCycle* of 50%, meaning that nodes were 50% of the time on and 50% off, also the packet threshold (1000 packets/second) and buffer capacity configured with a maximum of 80 packets.

5.3 SOS Orchestration of Energy Efficiency Capabilities

Since SOS results were published in (RIEKSTIN *et al.*, 2015c) and demonstrated with GreenSDN in (RIEKSTIN *et al.*, 2015b), this Section does not detail the functioning of the SOS orchestration. Figure 28 consolidates the energy consumption results for two flows sending workloads of 10 Mbps and 20 Mbps during

90 seconds.

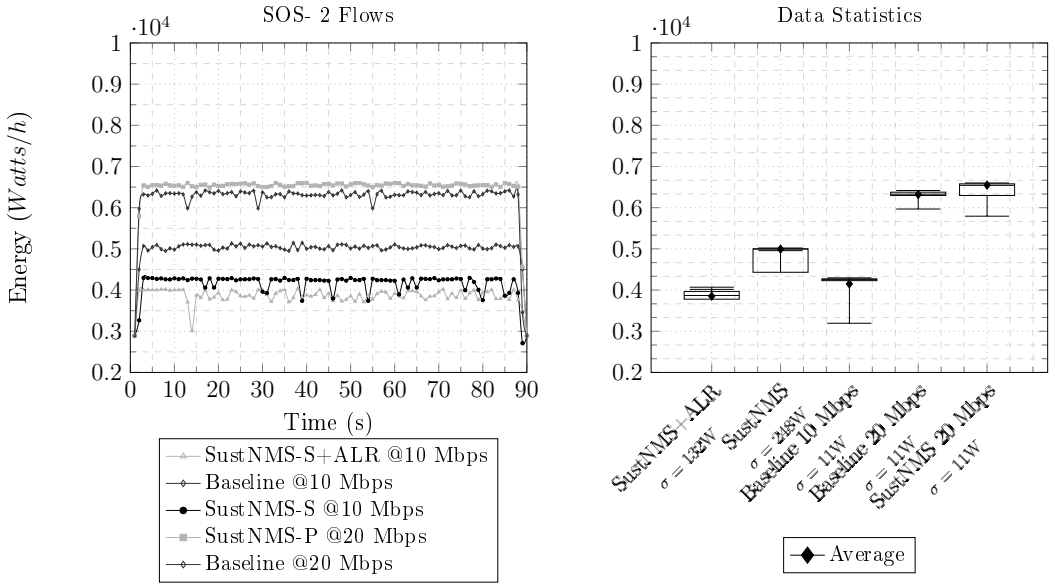


Figure 28: Energy consumption of capabilities orchestrated by SOS.

Since the SOS decision trees determine a combination of capabilities to be enforced in response to a given workload, energy consumption results were compared with the same workloads without SOS. In 10 Mbps run the combination selected by SOS was SustNMS-S[2] and ALR. While SustNMS-S concentrated flow and deactivated unused nodes, ALR adjusted the link rate in flows being used. The difference in enforcing ALR in conjunction with SustNMS can be observed by contrasting a baseline against SustNMS-S without ALR. Figure 29 presents both SustNMS-S and SustNMS-P operation.

For workloads higher than 15 Mbps the SOS choose the SustNMS-P to avoid losses. In this regard, the energy consumption of the 20 Mbps run was compared to a baseline energy consumption. SustNMS-P and the baseline present a similar energy consumption. However, the difference was that SustNMS-P still deactivates unused paths, as observed in SustNMS-P in Figure 29 (gray nodes were in sleep mode).

[2]SustNMS could be activated in two different ways. SustNMS-S, which concentrates flows, and a SustNMS-P, which spread flows through distinct paths to avoid losses.

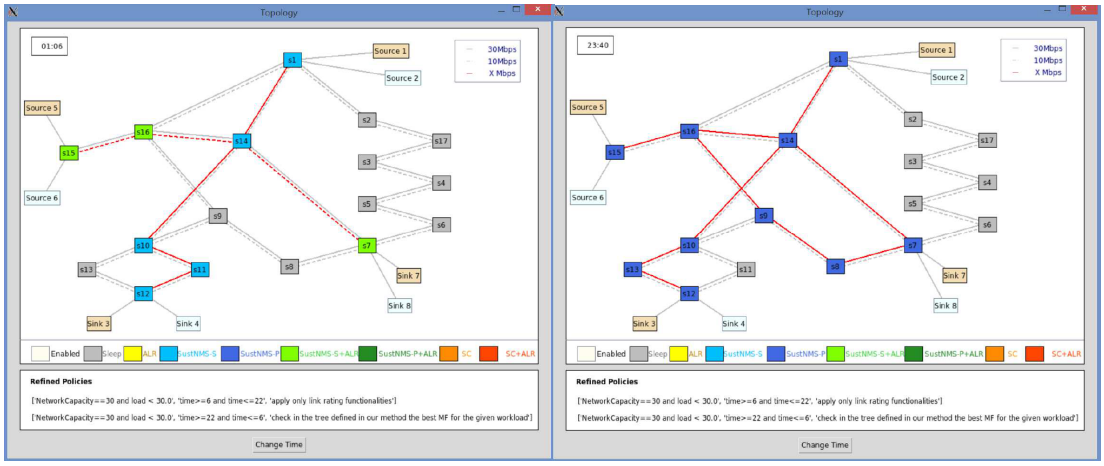


Figure 29: GreenSDN Topology Viewer. SustNMS-S + ALR (left) and SustNMS-P (right).

5.4 Per User Energy Consumption and Savings

Settings to perform user measurements are presented in Table 8. The experiment considered four users based on three SLAs, one user for full performance, one for an intermediary (midterm between performance and economy), and two users based on the economic SLA (greener SLA).

Links were configured to handle a maximum of 30 Mbps load, the maximum reachable workload for each SLA was divided among users to provide a 100% of link utilization in case of four users sharing the same path. Considering a 100% of link utilization is possible to observe the effects caused by the QoS levels, as well as decisions enforced by SOS in two scenarios. Figure 30 presents the scenarios.

User	Workload (Mbps)	Scenario A Path	Scenario B Path	Time (s)
Performance	15	[15-16-14-7]	[1-14-7]	30
Intermediary	9	[15-16-14-7]	[15-16-14-7]	30
Economy A	3	[15-16-14-7]	[15-16-14-10-13-12]	30
Economy B	3	[15-16-14-7]	[15-16-14-10-13-12]	30

Table 8: Settings to evaluate the energy consumed and saved by users.

Besides presenting the scenarios for the experiment, Figure 30 shows the capabilities selected by SOS. In Scenario A (Figure 30 left), the users forward

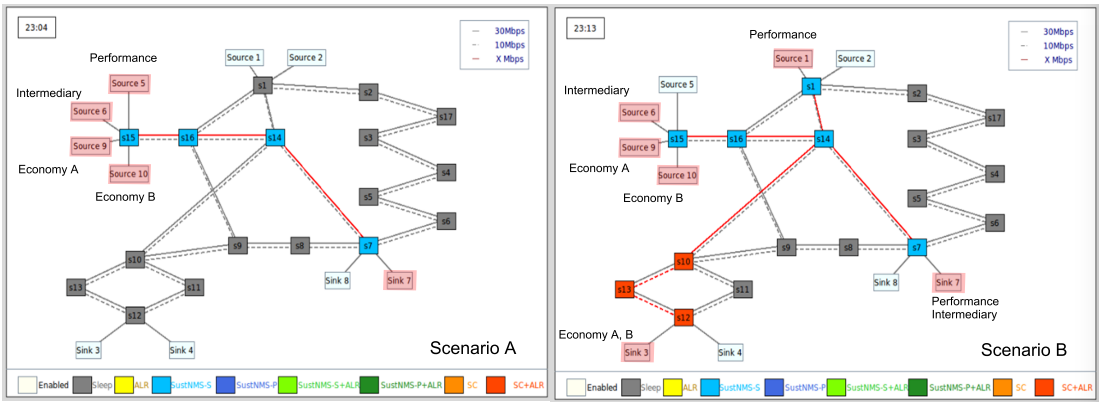


Figure 30: Scenarios A and B, and capabilities selected by SOS. Scenario A (left) with users sharing the same path. Scenario B (right) users in distinct paths.

data using the path [15-16-14-7] to Sink 7. As the aggregated workload on the path varied around 30 Mbps, SustNMS-S is the best decision to optimize the network. In Scenario A, it is interesting to observe the side effects on QoS values due to using 100% of the link capacity.

In Scenario B (Figure 30 right) it is interesting to evaluate the SOS behavior when users forward data through distinct paths. In case of the Performance user data from Source 1 was forwarded to Sink 7 using the path [1-14-7]. For an intermediary user it was configured with the same path as in Scenario B. Finally, for Economy data was forwarded through [15-16-14-10-13-12] to Sink 3. Results for both scenarios are depicted in Figure 31.

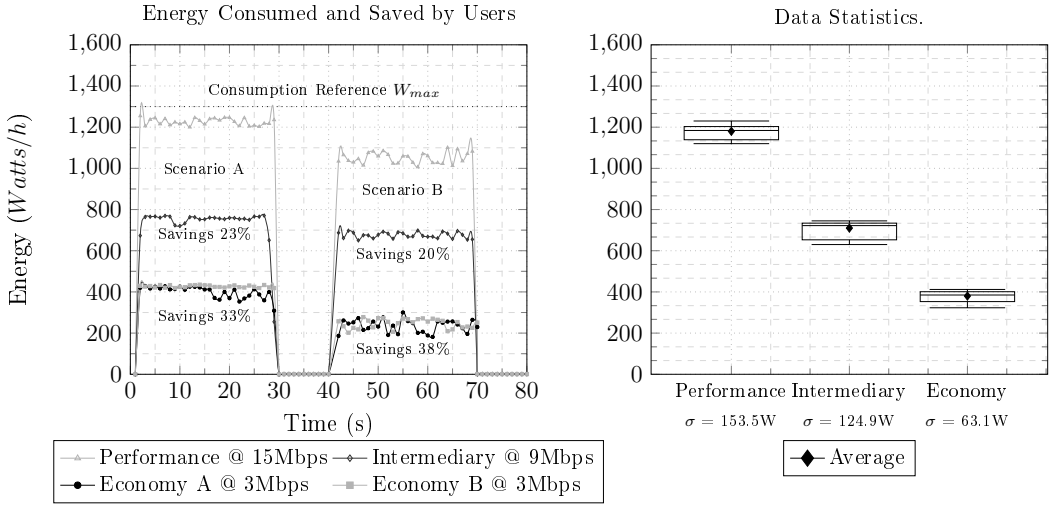


Figure 31: Energy consumed by users in scenarios A and B.

Since the aggregated traffic did not allow to apply neither ALR nor SC, SustNMS-S was enforced by SOS to concentrate the traffic and put unused nodes into sleep mode. In this sense, the energy consumed for all users in Scenario A was proportional to their network usage. Energy savings were calculated taking into account the nodes in sleep mode (savings were distributed among users), and a comparison with the energy consumed by a reference user configured the maximum bandwidth allowed (W_{max}).

In Scenario B, as distinct routes were used, individual decisions were enforced. Even though here the user's consumption was lower than in Scenario A, the overall network consumption was higher because more nodes were active. For instance, despite consuming more energy, the Intermediary user had lower savings than in Scenario A. As more nodes were activated, the lower was the distribution of savings from deactivated nodes. However, this was different for the Economy users because of the enforcement of SC+ALR in nodes 10, 13 and 12. Figure 32 presents the QoS statistics for both scenarios.

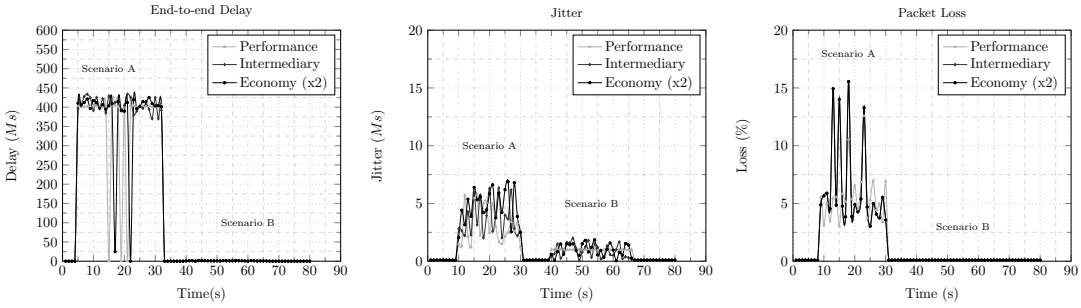


Figure 32: QoS Statistics for Scenarios A and B.

Side effects of link utilization and the queue buffer configured in 1000 packets size are presented in Figure 32. As users were configured with the same path, their statistics were similar to those of Scenario A. When a node is unable to forward packets immediately, and it starts to queue them. When a queue is saturated, packets are dropped. Furthermore, the higher was the queue utilization, higher was the delay to forward packets. This is the situation presented in Sce-

nario A with all hosts configured with the same path and nearing 100% of link utilization. However, the opposite is observed for Scenario B, in which different routes were configured for the different users, and thus the nodes/links were not oversubscribed.

5.5 Chapter Final Remarks

This chapter presented an evaluation of the main aspects of GreenSDN, such as the energy efficiency capabilities. The first part presented an evaluation of threshold approaches (or policies) for ALR and SC to reduce errors when enforcing a capability due to workload variations. Two policies were evaluated: single and dual threshold. While the single threshold policy presents a simpler implementation through a basic if-then-else code, it was not as precise as the dual threshold when it comes to environments with highly varying workloads. For both ALR and SC, the number of capabilities activation/deactivation was significantly lower with the dual rather threshold as the number of samples increased during the experiment, which, induces a lower adjustment latency in the network. Thus, the dual threshold technique was adopted to control the ALR and SC activation/deactivation.

During the capabilities evaluation presented in Subsection 5.2.3, the dual threshold policy was not effective due the constant workloads generated by the use of Iperf. In this regard, either the dual or the single threshold policies would present the same performance as there was no workload variation. However, the evaluation was effective to show how the selected capabilities performed under the different workloads, for instance, during the 10 Mbps evaluation the SC capability presented the higher energy savings, but due to its number of activation/deactivation (DutyCycle configured in 50% on/off), it was not the best capability for workloads higher than 5 Mbps.

Through analytical solvers, the SOS method discover in a similar approach whether a single or a combination of capabilities are more energy efficient in a certain network condition without compromising the quality of service. Results of combining SOS and GreenSDN were presented in Subsection 5.3 and published in (RIEKSTIN *et al.*, 2015c), proved that SOS is more efficient than applying separately the capabilities.

As a way to provide different levels of green services based on the energy saving capabilities and the SOS orchestration method, the GreenSDN was architecture was modified to calculate the energy consumed and saved per user in the network. Subsection 5.4 presented the an evaluation comprising four users with different requirements in terms of energy consumption, using the RNP-based topology. In addition, such changes in the GreenSDN architecture were presented in (RODRIGUES *et al.*, 2016). In this sense, the tests considered two different cases to evaluate energy consumption and savings of these users providing useful information for orchestration components to organize the network more efficiently.

6 CONCLUDING REMARKS

To handle a massive amount of data, network infrastructures have been designed considering high-performance and high-availability requirements. In these network infrastructures, the inadvertent use of energy saving mechanisms may compromise performance and availability parameters and strike a balance between efficiency and QoS provided by these networks. Given that it is critical to have experimental network platforms, which enable and ease network innovation allowing researchers to design and evaluate novel approaches regarding sustainability, and more specifically energy efficiency guaranteeing precise and reliable network adjustments without compromising loss of QoS. Based in this context, GreenSDN was proposed.

GreenSDN has as goal to leverage green networking providing a network platform comprising energy efficiency capabilities as a baseline towards the development of applications and management strategies. To reach these goals, this work was developed taking as basis different steps of a methodology, whose main contributions are listed below:

1. **Literature Review and Analysis**: this step involved the elaboration of Chapters 2 and 3 of this thesis providing the theoretical basis for its Design and Development stage.

    - **Chapter 2**: presented an overview of approaches to obtain energy efficiency in network infrastructures and a table summarizing different

energy efficiency capabilities based on such approaches. The main contribution of this chapter is the categorization of energy efficiency capabilities according to their approaches and network scope, which is important to understand how and where a particular capability operates in the network. Therefore, this categorization serves as a guide to understanding approaches to save energy in networks.

- **Chapter 3**: using the same approach of Chapter 2, it introduced network platforms that could be used as the basis for GreenSDN development and concepts related to network management and SDN. Therefore, its main contribution is the evaluation of network platforms according to criteria established in the literature, such as hardware requirements and scalability. Thus, it is a useful input towards evaluating a target network platform according to the desired characteristics. For instance, to deploy GreenSDN we considered an network emulation platform that is open source and provides fast deployment of network experiments.

2. **Design and Development**: the main step in the methodology was the technical solution to fulfill our goal regarding the creation of a testbed for experimenting energy efficiency network capabilities, as it was described in Chapter 4.

    - **Chapter 4**: introduced the GreenSDN architecture and its main components detailing how each component relate to each other. The architecture was based on the ONF (Open Network Foundation) SDN aiming to provide a separation between the control and management planes to leverage the development of independent applications and management strategies based on energy efficiency capabilities. The main contribution of this Chapter was the architecture and its techni-

cal description.

3. **Evaluation**: the experimental evaluation of GreenSDN was described in Chapter 5.

    - **Chapter 5**: presented an evaluation simulating thresholding techniques for activation/deactivation of ALR and SC in which the dual thresholding method was the most efficient for high workload variation; an individual evaluation of capabilities in GreenSDN using the Iperf traffic generator in which it was possible to observe the behavior of each capability under a particular workload; an assessment of GreenSDN being orchestrated by SOS; and the per-user evaluation in order to verify the models and monitoring components to collect per-user statistics. In addition to the Design and Development stage, this Chapter contributes with the evaluation of the architecture and the energy efficiency capabilities.

In this regard, GreenSDN was able to fulfill its primary goal providing the basis for the development and enforcement of the SOS orchestration method and Policy-Based Network Management strategies. As result, many publications were possible (as presented in Section 6.1). However, GreenSDN also requires some improvements to automate basic functionalities and introduces many research directions (as presented in Section 6.2).

Furthermore, this thesis is the result of the author collaboration in the projects Sustainability-Oriented System based on Dynamic Policies with Automated Policy Refinement (SOS) and Energy Efficiency to Clouds (E2C) at the Laboratory of Sustainability (LASSU). Both projects were developed in collaboration with Ericsson Telecomunicações S.A., Brazil, and Ericsson Research Sweden.

6.1 Publications

As contributions directly related with this thesis, it can be mentioned:

- **2014 - Short Paper/Published** - Riekstin, A.C.; Januario, G.C.; **Rodrigues, B.B.**; Nascimento, V.T.; Pirlea, M. R.; Carvalho, T.C.M.B.; Meirosu, C.. **Orchestration of Energy Efficiency Functionalities for a Sustainable Network Management**, in *Network Computing and Applications (NCA), 2014 IEEE 13th International Symposium on*, vol., no., pp.157-161, 21-23 Aug. 2014.

- **2015 - Experience Paper/Published - Rodrigues, B.B.**; Riekstin, A.C.; Januario, G.C.; Nascimento, V.T.; Carvalho, T.C.M.B.; Meirosu, C. **GreenSDN: Bringing energy efficiency to an SDN emulation environment**, in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium* on , vol., no., pp.948-953, 11-15 May 2015.

- **2015 - Demo/Published** - Riekstin, A.C.; **Rodrigues, B.B.**; Januario, G.C.; Nascimento, V.T.; Carvalho, T.C.M.B.; Meirosu, C., **A demonstration of energy efficiency capabilities orchestration in networks**, in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium* on , vol., no., pp.1149-1150, 11-15 May 2015.

- **2015 - Journal/Published** - Riekstin, A. C., Januario, G., **Rodrigues, B.B.**, Nascimento, V., Carvalho, T., Meirosu, C. (2015). **A Survey of Policy Refinement Methods as a Support for Sustainable Networks**, in *Communications Surveys and Tutorials, IEEE*, PP, n.99, p.1-1, 2015.

- **2015 - Journal/Published** - Riekstin, A. C., Januario, G. C., **Rodrigues, B. B**., Nascimento, V. T., Carvalho, T. C., Meirosu, C. (2015). **Orchestration of energy efficiency capabilities in networks**. *Journal of*

Network and Computer Applications, pp, n.99, p.1-1, 2015.

- **2015 - Poster/Published - Rodrigues, B.B.**, Miers, C. C., Carvalho, T.C.M.B. **GreenSDN: an Emulation Environment Towards the Development of Network Energy Efficiency Capabilities**. IV Workshop de Pos-Graduação da Area de Concentração Engenharia de Computação (IV WPG-EC). October, 2015.

- **2016 - Paper/Published - Rodrigues, B. B.**, Rojas, M. A. T., Nascimento, V. T., Carvalho, T. C., Meirosu, C. **Green Service Levels in Software Defined Networks**. Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2016). June, 2016.

- **2016 - Journal/Accepted** - Riekstin, A. C., **Rodrigues, B. B.**, Progetti, Claudia, Nascimento, V. T., Carvalho, T. C., Meirosu, C. **Sustainability Information Model for Energy Efficiency Policies**. *IEEE Communications Magazine (COMMAG), Green Communications and Computing Network Series.* To appear.

In addition, the following items were submitted:

- **2015 - Patent/Submitted** - Meirosu, C., **Rodrigues, B. B.**, Carvalho, T. C. M. B, Nascimento, V. T., Riekstin, A. C. **Virtual Software-Defined Power**. PCT/IB2015/055362. Submitted in July, 2015.

- **2016 - Patent/Submitted** - Meirosu, C., **Rodrigues, B. B.**, Carvalho, T. C. M. B, Rojas, M. A. T. **Power Manager and Method Performed thereby for Managing Power of a Datacentre**. PCT/SE2016/050686. Submitted in July, 2016.

- **2016 - Patent/Submitted** - Meirosu, C., **Rodrigues, B. B.**, Carvalho,

T. C. M. B, Rojas, M. A. T., Pereira, R. M, Sousa, R. M. **Network Prediction Driven DVFS**. PCT/SE2016/050721. Submitted in July, 2016.

As result of the author undergrad final work under the advisor of Professor Dr. Charles C. Miers, the book chapter entitled "Security Analysis for Cloud Computing Solutions"was published during the master's period, but not being directly related to this thesis subject:

- **2014 - Book Chapter/Published** - Miers, C. C.; Koslovski, G. P.; Simplicio, M.; Carvalho, T.C.M.B.; Redigolo, F. F.; **Rodrigues, B. B.**; Barros, B. M., Gonzalez, N. M.; Rojas, M. A. T.; Iwaya, L. H. **Analise de Segurança para Soluções de Computação em Nuvem**. Em: Joni da Silva Fraga, Frank Siqueira, Carlos Alberto Maziero. (Org.). Minicursos / XXXII Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos (SBRC 2014). 1ed.Porto Alegre/RS: Sociedade Brasileira de Computacao (SBC), 2014, v. 1, p. 194-243.

6.2 Future Works

Despite providing a baseline environment towards the mitigation of energy efficiency aspects, GreenSDN requires improvements on its basic functionalities, such as forwarding and monitoring. Its GUI the use of forecasting algorithms. Some potential future works worthy to be mentioned are:

- **Forwarding and Monitoring**: although providing forwarding and monitoring capabilities, GreenSDN still requires further improvements towards the automation of these core modules, i.e., providing dynamic forwarding and monitoring capabilities regardless of external topology information. Currently, forwarding capabilities in GreenSDN requires hosts information

(e.g., as node and port to which the host is connected to) for reactively configuring flows, and by extension, the monitoring module relies on this information to configure border nodes and monitored paths. Besides, the development of automated forwarding and monitoring capabilities are not straightforward. In this regard, it is let as future work the improvement of such capabilities in GreenSDN. To provide forwarding and monitoring capabilities regardless of external information about hosts, a spanning tree algorithm is required. An evaluation in GreenSDN was conducted using a topology inspired on the Facebook (Alexey Andreyev, 2014)[1] data center topology. However, during the experiments the standard spanning tree algorithm provided on the POX library did not work as expected. Thus, it is let as future work the improvement of spanning tree algorithm provided on the POX library, possibly replacing its Floyd-Warshall-based packet forwarding by an implementation based on Kruskall or Prim algorithms.

- **GUI Improvements**: Also, GreenSDN require improvements on its GUI, to display the network topology and its active capability(ies) and the monitoring information. Currently, the GUI implemented on GreenSDN is a based on Tkinter[2], a standard Python library to build GUIs. However, it does not provide enough tools in its library to address the dynamicity presented in GreenSDN. In this regard, it is let as future work the development of a web-based GUI interface for displaying the network topology.

- **Prediction Engine**: based on information collected by the monitoring and power components, algorithms to forecast energy consumption can be used to anticipate a network behavior. Instead of reactively adjusting the network, proactive configuration of energy efficiency capabilities can be done to increase power efficiency.

[1] The topology was implemented by the author in (RODRIGUES *et al.*, 2016)
[2] https://wiki.python.org/moin/TkInter

Furthermore, this thesis provides a baseline environment towards the research and development of novel energy management strategies, such as the SOS, and the further investigation of trade-offs considering the energy efficiency aspect, and QoS topics is very important.

REFERENCES

ADHIKARI, V. *et al.* Unreeling netflix: Understanding and improving multi-cdn movie delivery. In: *INFOCOM, 2012 Proceedings IEEE*. [S.l.: s.n.], 2012. p. 1620–1628. ISSN 0743-166X.

ADRICHEM, N. L. M. van; DOERR, C.; KUIPERS, F. A. Opennetmon: Network monitoring in openflow software-defined networks. In: IEEE. *Network Operations and Management Symposium (NOMS)*. [S.l.], 2014. p. 1–8.

AKARI. New generation network architecture. Citeseer, 2007.

Alexey Andreyev. *Introducing Data Center Fabric, the Next-Generation Facebook Data Center Network*. [S.l.], 2014. Disponível em: <https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/>.

BARROS, B. *et al.* Applying software-defined networks to cloud computing. In: *Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos*. [S.l.: s.n.], 2015. p. 1–54.

BARROSO, L. A.; HÖLZLE, U. The case for energy-proportional computing. *Computer*, IEEE, n. 12, p. 33–37, 2007.

BIANZINO, A. *et al.* A survey of green networking research. *Communications Surveys Tutorials, IEEE*, v. 14, n. 1, p. 3–20, First 2012. ISSN 1553-877X.

BILAL, K. *et al.* A survey on green communications using adaptive link rate. *Cluster Computing*, v. 16, n. 3, p. 575–589, 2013.

BILAL, K.; KHAN, S. U.; ZOMAYA, A. Y. Green data center networks: Challenges and opportunities. In: IEEE. *Frontiers of Information Technology (FIT), 2013 11th International Conference on*. [S.l.], 2013. p. 229–234.

BILALB, S. M.; OTHMANA, M. *et al.* A performance comparison of network simulators for wireless networks. *arXiv preprint arXiv:1307.4129*, 2013.

BOLLA, R. *et al.* Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures. *Communications Surveys Tutorials, IEEE*, v. 13, n. 2, p. 223–244, Second 2011. ISSN 1553-877X.

CARBONE, M.; RIZZO, L. Dummynet revisited. *ACM SIGCOMM Computer Communication Review*, ACM, v. 40, n. 2, p. 12–20, 2010.

CHOWDHURY, S. *et al.* Payless: A low cost network monitoring framework for software defined networks. In: *Network Operations and Management Symposium (NOMS), 2014 IEEE*. [S.l.: s.n.], 2014. p. 1–9.

CHRISTENSEN, K. *et al.* IEEE 802.3az: the road to energy efficient ethernet. *Communications Magazine, IEEE*, v. 48, n. 11, p. 50–56, Nov 2010. ISSN 0163-6804.

CIANFRANI, A. *et al.* An energy saving routing algorithm for a green ospf protocol. In: IEEE. *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*. [S.l.], 2010. p. 1–5.

COOK, G. *et al. Clicking Clean: How Companies are Creating the Green Internet*. [S.l.], 2014. Disponível em: <http://www.greenpeace.org/usa/Global/usa/planet3/PDFs/clickingclean.pdf>.

COOK, G. *et al. Clicking Clean: A Guide to Building the Green Internet*. [S.l.], 2015. Disponível em: <http://www.greenpeace.org/usa/wp-content/uploads/legacy/Global/usa/planet3/PDFs/2015ClickingClean.pdf>.

COSTA, C. H. *et al.* SustNMS: Towards service oriented policy-based network management for energy-efficiency. In: *SustainIT'12*. [S.l.: s.n.], 2012. p. 1–5.

ERICSSON b. Ericsson Energy and Carbon Report - Including Results from the First-ever National Assessment of the Environmental Impact of ICT. In: . [s.n.], 2014. Disponível em: <http://www.ericsson.com/res/docs/2014/ericsson-energy-and-carbon-report.pdf>.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to sdn: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 44, n. 2, p. 87–98, 2014.

FERNANDEZ, M. P. Comparing openflow controller paradigms scalability: Reactive and proactive. In: *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*. [S.l.: s.n.], 2013. p. 1009–1016. ISSN 1550-445X.

FIBRE. Future internet brazilian environment for experimentation. 2016. Disponível em: <https://fibre.org.br/>.

Floodlight. *Floodlight OpenFlow Controller*. 2015. Disponível em: <http://www.projectfloodlight.org/floodlight/>.

GARG, S. K.; BUYYA, R. Green cloud computing and environmental sustainability.

GAVRAS, A. *et al.* Future internet research and experimentation: the fire initiative. *ACM SIGCOMM Computer Communication Review*, ACM, v. 37, n. 3, p. 89–92, 2007.

GENI. Global environment for network innovations. Disponível em: <http://www.geni.net/>.

GREENBERG, A. *et al.* A clean slate 4d approach to network control and management. *ACM SIGCOMM Computer Communication Review*, ACM, v. 35, n. 5, p. 41–54, 2005.

GSN. Greenstar network. 2010. Disponível em: <http://www.greenstarnetwork.com/>.

GUDE, N. *et al.* Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 3, p. 105–110, 2008.

GUNARATNE, C. *et al.* Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR). *Computers, IEEE Transactions on*, v. 57, n. 4, p. 448–461, Apr 2008.

GUPTA, M.; SOMMERS, J.; BARFORD, P. Fast, accurate simulation for sdn prototyping. In: ACM. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. [S.l.], 2013. p. 31–36.

HELLER, B. *et al.* Elastictree: Saving energy in data center networks. In: *NSDI'10*. [S.l.: s.n.], 2010. p. 17–17.

HOLIBAUGH, R. R.; PERRY, J. M.; SUN, L. Phase i testbed description: Requirements and selection guidelines. 1988.

JANUARIO, G. C. *et al.* Evaluation of a policy-based network management system for energy-efficiency. In: IEEE. *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. [S.l.], 2013. p. 596–602.

JOSE, L.; YU, M.; REXFORD, J. Online measurement of large traffic aggregates on commodity switches. In: *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*. Berkeley, CA, USA: USENIX Association, 2011. (Hot-ICE'11), p. 13–13. Disponível em: <http://dl.acm.org/citation.cfm?id=1972422.1972439>.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: Rapid prototyping for software-defined networks. In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. New York, NY, USA: ACM, 2010. (Hotnets-IX), p. 19:1–19:6. ISBN 978-1-4503-0409-2. Disponível em: <http://doi.acm.org/10.1145/1868447.1868466>.

Linux Foundation. *OpenDaylight, a Linux Foundation Collaborative Project*. 2015. Disponível em: <http://www.opendaylight.org/>.

MAHADEVAN, P.; BANERJEE, S.; SHARMA, P. Energy proportionality of an enterprise network. In: ACM. *Proceedings of the first ACM SIGCOMM workshop on Green networking*. [S.l.], 2010. p. 53–60.

MCKEOWN, N. *et al.* Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1355734.1355746>.

MOSTOWFI, M.; CHRISTENSEN, K. Saving energy in LAN switches: New methods of packet coalescing for Energy Efficient Ethernet. In: *IGCC'11*. [S.l.: s.n.], 2011. p. 1–8.

NS2. The network simulator - ns2. 2016. Disponível em: <http://www.isi.edu/nsnam/ns/>.

NS3. The network simulator - ns3. 2016. Disponível em: <https://www.nsnam.org/>.

OMNET. Omnet discrete event simulator. 2016. Disponível em: <https://omnetpp.org/>.

ONF (Ed.). *Open Network Foundation: SDN Architecture Overview*. [S.l.], 2013.

PEDIADITAKIS, D.; ROTSOS, C.; MOORE, A. W. Faithful reproduction of network experiments. In: *ANCS'14*. [S.l.: s.n.], to be published.

PFAFF, B.; PETTIT, J.; SHENKER, S. Extending networking into the virtualization layer. In: . [S.l.: s.n.], 2009.

POX. Python-based openflow controller. 2009. Disponível em: <http://www.noxrepo.org/pox/about-pox/>.

RICCA, M. *et al.* An assessment of power-load proportionality in network systems. In: *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*. [S.l.: s.n.], 2013. p. 1–8.

RICCIARDI, S. *et al.* Analyzing local strategies for energy-efficient networking. In: *Proceedings of the IFIP TC 6th International Conference on Networking*. Berlin, Heidelberg: Springer-Verlag, 2011. (NETWORKING'11), p. 291–300. ISBN 978-3-642-23040-0. Disponível em: <http://dl.acm.org/citation.cfm?id=2039912.2039944>.

RIEKSTIN, A. *et al.* Orchestration of energy efficiency functionalities for a sustainable network management. In: *Network Computing and Applications (NCA), 2014 IEEE 13th International Symposium on*. [S.l.: s.n.], 2014. p. 157–161.

RIEKSTIN, A. *et al.* A survey of policy refinement methods as a support for sustainable networks. *Communications Surveys Tutorials, IEEE*, PP, n. 99, p. 1–1, 2015. ISSN 1553-877X.

RIEKSTIN, A. *et al.* A demonstration of energy efficiency capabilities orchestration in networks. In: *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. [S.l.: s.n.], 2015. p. 1149–1150.

RIEKSTIN, A. C. *et al.* Orchestration of energy efficiency capabilities in networks. *Journal of Network and Computer Applications*, p. –, 2015. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804515001435>.

RODRIGUES, B. *et al.* Greensdn: Bringing energy efficiency to an sdn emulation environment. In: *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. [S.l.: s.n.], 2015. p. 948–953.

RODRIGUES, B. *et al.* Green Service Levels in Software Defined Networks. In: . [S.l.: s.n.], 2016. p. 1–14.

ROSCOE, T. (Ed.). *PlanetLab Phase 0: Technical Specification*. [S.l.], 2002.

Ryu. *Component-based Software-defined Networking Framework*. 2015. Disponível em: <http://osrg.github.io/ryu/>.

SALSANO, S. *et al.* Information centric networking over sdn and openflow: Architectural aspects and experiments on the ofelia testbed. *Comput. Netw.*, Elsevier North-Holland, Inc., New York, NY, USA, v. 57, n. 16, p. 3207–3221, nov. 2013. ISSN 1389-1286. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2013.07.031>.

SCHLENK, R. *et al.* Taxonomy of dynamic power saving techniques in fixed broadband networks. In: *Photonic Networks, 14. 2013 ITG Symposium. Proceedings*. [S.l.: s.n.], 2013. p. 1–8.

SEMERARO, G. *et al.* Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In: *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*. [S.l.: s.n.], 2002. p. 29–40. ISSN 1530-0897.

SEZER, S. *et al.* Are we ready for sdn? implementation challenges for software-defined networks. *Communications Magazine, IEEE*, v. 51, n. 7, p. 36–43, July 2013. ISSN 0163-6804.

SIATERLIS, C.; GARCIA, A.; GENGE, B. On the use of emulab testbeds for scientifically rigorous experiments. *Communications Surveys Tutorials, IEEE*, v. 15, n. 2, p. 929–942, Second 2013. ISSN 1553-877X.

STEELE, J. Acpi thermal sensing and control in the pc. In: *Wescon/98*. [S.l.: s.n.], 1998. p. 169–182. ISSN 1095-791X.

SUBRAMANIAN, M. Network management: An introduction to principles and practice. Addison-Wesley Longman Publishing Co., Inc., 1999.

TROJER, E. *et al.* Current and next-generation pons: A technical overview of present and future pon technology.

VAHDAT, A. *et al.* Scalability and accuracy in a large-scale network emulator. *ACM SIGOPS Operating Systems Review*, ACM, v. 36, n. SI, p. 271–284, 2002.

VERMA, D. C. *Principles of computer systems and network management*. [S.l.]: Springer, 2009.

WANG, S.-Y.; CHOU, C.-L.; YANG, C.-M. Estinet openflow network simulator and emulator. *Communications Magazine, IEEE*, IEEE, v. 51, n. 9, p. 110–117, 2013.

WANG, S.-Y.; CHOU, C.-L.; YANG, C.-M. Estinet openflow network simulator and emulator. *Communications Magazine, IEEE*, v. 51, n. 9, p. 110–117, September 2013. ISSN 0163-6804.

WANG, X. *et al.* A survey of green mobile networks: Opportunities and challenges. *Mobile Networks and Applications*, Springer US, v. 17, n. 1, p. 4–20, 2012. ISSN 1383-469X. Disponível em: <http://dx.doi.org/10.1007/s11036-011-0316-4>.

WHITE, B. *et al.* An integrated experimental environment for distributed systems and networks. In: USENIXASSOC. *OSDI02*. Boston, MA, 2002. p. 255–270.

ZHANG, M. *et al.* GreenTE: Power-aware traffic engineering. In: IEEE. *Network Protocols (ICNP), 2010 18th IEEE International Conference on*. [S.l.], 2010. p. 21–30.

A THE GREENSDN ARCHITECTURE

- **Application Plane**

  Graphical User Interface (GUI): exchange data with the control plane to display topology information, network status, and obtain user requirements.

  * **Topology**: present the network topology, current energy efficiency capabilities and state of links;
  * **QoS Charts**: build energy and QoS charts based on the Matplot library.
  * **Hosts Param.**: interface to select QoS and energy requirements (e.g., an amount of Watts to be spent) and send workloads.
  * **Data Parser**: communication interface with the control plane. It send user requirements and receive data regarding network usage both from users and the network.

- **Management Plane**

  Data Parser: implement a socket to receive and parse information from the Hosts GUI.

  Configuration Parameters: receive and parse settings both from SOS and users.

  * **Hosts SLA**: update QoS and energy consumption requirements,

as well as users network information (i.e., the node and interface the user is connected); and

* **Network Settings**: receive network settings from SOS (policies, decision trees), a proactive flow instantiation, and users policies and requirements.

SOS Parser: parse XMLs describing environment condition, time and actions to be enforced by network policies.

Dynamic Policy Configuration: update policies when conditions or requirements are modified. Conditions are due to time or scenario (i.e., a node is disconnected) changes and requirements when new goals are configured both for users and overall network.

Decision Enforcement: verify network or user conditions to select decisions for meeting user SLA requirements or high-level network goals.

* **Time and Environment Condition**: as SOS produces two decision trees (day and night periods), the management layer implements a clock that verify when it is required to switch decision trees. Scenario may require for changes in the active tree, adjusting to the number of active nodes;
* **QoS Evaluation**: implement thresholds to avoid the enforcement of erroneous decisions. This component may implement a dual threshold (based on ALR thresholds) to enforce decisions.;
* **Hosts**: check for SLA violations; and
* **Decision Trees**: given a network condition (time and network workload) select a rule in the decision tree to enable or disable energy efficiency capabilities.

- **Control Plane**

Topology Manager: deals with the management of nodes and ports. It implements methods to install or remove flows a change is required in the network, it also define switches and port states (e.g. standard, sleep, ALR, SC) in order to measure energy consumption through the Power Manager.

* **Network Graph**: present the network topology and details about nodes (i.e., number of ports, energy states, and others);
* **Requests**: handle requests from other components for configuring paths;
* **Device Updater**: forward messages to the data plane;
* **Protocol Parser**: given a set of nodes in a data structure (i.e., a dictionary), the component built correspondent OpenFlow messages; and
* **Proactive Flows Configuration**: flows are instantiated based on manual input from a network administrator. It receives the initial setup of flows and instantiates

QoS Services Monitoring: responsible for the monitoring task, handling the response of flow and port status events.

* **Network Configuration**: take as input the network graph, set of border nodes, and initial polling frequency;
* **QoS Sampling**: calculate QoS parameters on nodes that are not being queried;
* **Read Stats Handler**: receive statistics from flows and ports and calculate required metrics (e.g. current workload);
* **Connection Up/Down Handler**: adjust the network graph when an connection up or down is triggered;
* **Data Output and Formatting**: forward network statistics the

Data Parser (to be plotted) and the Network Log database through the MySQL-db Manager; and

  - **Adaptive Polling**: based on the nodes throughput and set of active paths, it is able to adjust the polling frequency and set of nodes to be queried.

Power Manager: component used to implement ALR policies, the adaptive part of SC and calculate power consumption either from the network or hosts.

  - **ALR Policy**: implement the policy to effectively adjust the link rate;
  - **SC Adaptive**: emulate the adaptive part of SC (dutycycle); and
  - **Per-user Energy Measurements**: infers the overall energy consumption and in a per-user basis given the nodes utilization.

Databases: store information regarding policies, users QoS and energy requirements, power models, and logging of network information both overal and in a per-user basis (energy consumption and QoS parameters).

  - **MySQL-db Manager**: framework to manage tables and plain text documents;
  - **Policies Repository**: keep information regarding network policies (given from SOS) and user energy policies;
  - **SLAs**: maintain users requirements regarding energy consumption (amount of Watts to be spent) and QoS requirements;
  - **Network Log**: logging of energy consumption and QoS parameters both for network and users; and
  - **Power Models**: store the nodes energy profiles.

- **Data Plane**

Open vSwitch: standard OVS nodes running in kernel mode (i.e., nodes are mapped as a process in a process-based virtualization (LANTZ; HELLER; MCKEOWN, 2010)).

Traffic Shaper: since OpenFlow 1.0 is not able to fully configure queues on OVS (only enqueue flows and set minimum rate queues), it was used an external configuration (based on Linux Hierarchical Token Bucket[1]) to emulate a traffic shape functionality for users exceeding a determined amount of Watts.

ALR: parallel links interconnecting each pair of nodes to emulate the ALR mechanism.

[1]The Linux Documentation Project: http://tldp.org/HOWTO/Traffic-Control-HOWTO