ANA CAROLINA RIEKSTIN

ORCHESTRATION OF ENERGY EFFICIENCY CAPABILITIES FOR A SUSTAINABLE NETWORK MANAGEMENT

São Paulo
2015

ANA CAROLINA RIEKSTIN

ORCHESTRATION OF ENERGY EFFICIENCY CAPABILITIES FOR A SUSTAINABLE NETWORK MANAGEMENT

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia Elétrica.

São Paulo
2015

ANA CAROLINA RIEKSTIN

ORCHESTRATION OF ENERGY EFFICIENCY CAPABILITIES FOR A SUSTAINABLE NETWORK MANAGEMENT

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia Elétrica.

Área de Concentração:
Sistemas Digitais

Orientador: Profa. Dra. Tereza Cristina Melo de Brito Carvalho
Co-orientador: Dr. Catalin Meirosu

São Paulo
2015

Catalogação-na-publicação

Riekstin, Ana Carolina
Orchestration of Energy Efficiency Capabilities for a Sustainable Network Management / A. C. Riekstin -- versão corr. -- São Paulo, 2015.
149 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Redes de computadores 2.Gerenciamento de redes 3.Sustentabilidade 4.Eficiência energética 5.Refinamento de políticas I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Para Carlos Eduardo e Caco

ACKNOWLEDGEMENTS

First of all, I would like to thank God for this achievement.

Thanks to my advisor, Prof. Dr. Tereza Cristina Melo de Brito Carvalho, for the opportunity to develop this work and for all the support from the beginning of the research to the end, with her reviews and comments.

I would also like to thank my co-advisor, Dr. Catalin Meirosu (Ericsson Research, Sweden), for the valuable technical contributions throughout the research and the careful correction of the thesis in the final stage.

Thanks to Ericsson Brasil and to the Fundação para o Desenvolvimento Tecnológico da Engenharia (FDTE) for the financial support.

And thanks to the Centro de Tecnologia da Informação de São Paulo (CeTI-SP) from the Superintendência de Tecnologia da Informação of University of São Paulo (USP) for the information about the USP traffic profiles used in this thesis.

I should not forget to thank my colleagues at LASSU-PCS-EPUSP (Laboratory on Sustainability of the Computer Engineering and Digital Systems Department of the Polytechnic School of USP) for all the journey of discovery and development shared in the last years.

I would also like to thank the friends from Univesp (Virtual University of the State of São Paulo), for the support in the final stage of this work.

To my parents and brothers, to the relatives and long-standing friends, my gratitude for the support and understanding in so many absences motivated by this great objective. Special thanks to my brother, Fernando Vilas Boas Riekstin, who was near in different moments of this work.

Finally, my deepest gratitude to my husband, Carlos Eduardo de Almeida Ladeira, always by my side, supporting and sharing every minute of the development of this work.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a Deus por esta conquista.

Obrigada à minha orientadora, Profa. Dra. Tereza Cristina Melo de Brito Carvalho, pela oportunidade de desenvolver este trabalho de doutorado e por todo o suporte, desde o início da pesquisa, até o final, com suas revisões e comentários.

Gostaria de agradecer também ao meu co-orientador, Dr. Catalin Meirosu (Ericsson Research, Suécia), pelas preciosas contribuições técnicas durante toda a pesquisa e pela correção minuciosa da tese na etapa final.

Obrigada à Ericsson Brasil e à Fundação para o Desenvolvimento Tecnológico da Engenharia (FDTE) pelo apoio financeiro.

E obrigada ao Centro de Tecnologia da Informação de São Paulo (CeTI-SP) da Superintendência de Tecnologia da Informação da Universidade de São Paulo (USP) pelas informações de tráfego da USP utilizadas nesta tese.

Não poderia deixar de agradecer também aos meus colegas do LASSU-PCS-EPUSP (Laboratório de Sustentabilidade do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da USP) por toda a jornada de descoberta e desenvolvimento compartilhada nos últimos anos.

Também gostaria de agradecer aos amigos da Univesp (Universidade Virtual do Estado de São Paulo), pelo suporte na reta final deste doutorado.

Aos meus pais e aos meus irmãos, aos familiares próximos e aos amigos de longa data, também o meu muito obrigada pelo suporte e pela compreensão em tantas ausências motivadas por este objetivo maior. Um agradecimento especial ao meu irmão, Fernando Vilas Boas Riekstin, que esteve próximo em diversos momentos deste trabalho.

Por fim, meu agradecimento mais profundo vai para meu marido, Carlos Eduardo de Almeida Ladeira, sempre do meu lado, apoiando e compartilhando cada minuto do desenvolvimento deste trabalho.

Para ser grande, se inteiro: nada

Teu exagera ou exclui.

Se todo em cada coisa. Poe quanto és

No mínimo que fazes.

Assim em cada lago a lua toda

Brilha, porque alta vive.

Fernando Pessoa (Ricardo Reis)

ABSTRACT

The energy demand for operating Information and Communication Technology (ICT) systems has been growing, implying in high operational costs and consequent increase of carbon emissions. Both in datacenters and telecom infrastructures, the networks represent a significant amount of energy spending. Given that, there is an increased demand for energy efficiency solutions, and several capabilities to save energy have been proposed. However, it is very difficult to orchestrate such energy efficiency capabilities, i.e., coordinate or combine them in the same network, ensuring a conflict-free operation and choosing the best one for a given scenario, ensuring that a capability not suited to the current bandwidth utilization will not be applied and lead to congestion or packet loss. Also, there is no way in the literature to do this taking business directives into account. In this regard, a method able to orchestrate different energy efficiency capabilities is proposed considering the possible combinations and conflicts among them, as well as the best option for a given bandwidth utilization and network characteristics. In the proposed method, the business policies specified in a high-level interface are refined down to the network level in order to bring high-level directives into the operation, and a Utility Function is used to combine energy efficiency and performance requirements. A Decision Tree able to determine what to do in each scenario is deployed in a Software Defined Network environment. The proposed method was validated with different experiments, testing the Utility Function, checking the extra savings when combining several capabilities, the decision tree interpolation and dynamicity aspects. The orchestration proved to be valid to solve the problem of finding the best combination for a given scenario, achieving additional savings due to the combination, besides ensuring a conflict-free operation.

Keywords: Policy-Based Network Management; Sustainability-Oriented Policies; Policy Refinement; Sustainability; Energy Efficiency; Energy Efficiency Capabilities; Capabilities Orchestration.

RESUMO

A demanda de energia para operar os Sistemas de Tecnologia da Informação e Comunicação (TIC) tem crescido, implicando em altos custos operacionais e consequente aumento de emissão de carbono. Tanto em *datacenters*, quanto nas infraestruturas de telecomunicações, as redes têm uma contribuição significativa nos gastos de energia. Isto leva, como consequência, a uma crescente demanda por soluções de eficiência energética, e diversas funcionalidades para economizar energia têm sido propostas. No entanto, é muito difícil orquestrar tais funcionalidades, ou seja, coordená-las ou combiná-las na mesma rede, garantindo uma operação sem conflitos e escolhendo a melhor funcionalidade para um determinado cenário, assegurando que uma funcionalidade não adequada para a atual taxa de utilização da rede não será aplicada, levando-se a situações de congestionamento ou perda de pacotes. Também não há uma forma na literatura de fazer esta escolha a partir de diretivas de negócio. Neste âmbito, um método capaz de orquestrar diferentes funcionalidades de eficiência energética é proposto, considerando as possíveis combinações e conflitos entre elas, bem como a melhor opção para uma dada carga de trabalho e características da rede. No método proposto, as políticas de negócios são refinadas até o nível de rede de modo a trazer as diretivas de negócios para dentro da operação da rede, e uma Função de Utilidade é usada para combinar requisitos de eficiência energética e desempenho. Uma Árvore de Decisão capaz de determinar o que fazer em cada cenário é implementada em um ambiente de Redes Definidas por Software. O método proposto foi validado com diferentes experimentos, testando-se a Função de Utilidade, checando a economia adicional de energia ao combinar funcionalidades, a interpolação da Árvore dá Decisão e aspectos de dinamicidade. A orquestração mostrou-se válida para resolver o problema de encontrar a melhor combinação de funcionalidades para um determinado cenário, obtendo economias adicionais de energia devido à combinação de funcionalidades, além de garantir uma operação sem conflitos.

Palavras-chave: Gerenciamento de Redes Baseado em Políticas; Políticas Orientadas à Sustentabilidade; Refinamento de Políticas; Sustentabilidade; Eficiência Energética; Funcionalidades de Eficiência Energética; Orquestração de Funcionalidades.

STATEMENT OF CONTRIBUTION

This thesis is the result of the author participation in the project “Sustainability Oriented System based on Dynamic Policies with Automated Policy Refinement” (SOS) at the Laboratory on Sustainability (LASSU) of the Department of Computing and Digital Systems Engineering (PCS), Escola Politécnica, University of São Paulo, in partnership with Ericsson Telecomunicações S.A. and Ericsson Research.

Many of the ideas developed in this thesis are the result of group discussions with Professor Tereza C. M. B. Carvalho, Dr. Catalin Meirosu, Guilherme C. Januário, Marcelo C. Amaral, Bruno B. Rodrigues, Viviane T. Nascimento and other colleagues that did internships in the laboratory.

The development of this thesis started taking as basis the results of a previous project named “SustNMS”, developed by Professor Tereza C. M. B. Carvalho, Dr. Catalin Meirosu, Dr. Carlos H. A. Costa, Marcelo C. Amaral, and Guilherme C. Januário. SustNMS is a Sustainability Oriented Policy-Based Network Management system that enforces energy-efficiency procedures considering the trade-off among energy savings, performance, and reliability of the network. This thesis uses SustNMS as one of the energy efficiency capabilities that can be applied to a network in conjunction with other capabilities. SustNMS is detailed in Chapter 2. The proposed method also used the Sustainability-Oriented Information Models developed by the group, explained in Chapter 4.

The proposed method orchestrates energy efficiency capabilities (a) selecting the best option considering the network scenario; (b) combining capabilities in order to increase the energy savings; and (c) ensuring a conflict-free operation, that is, ensuring that such capabilities will not conflict with each other, negating the savings or leading to failures, also ensuring that the best capability for a given situation is selected.

The original contributions of this work are the definition of sustainability-oriented policies; the identification of the requirements that a policy refinement process should address in order to tackle energy efficiency; and the method to orchestrate energy efficiency capabilities. The method was validated using an energy efficiency emulation environment developed within the scope of the project.

FIGURES LIST

TABLES LIST

TERMINOLOGY

ACPI	*Advanced Configuration and Power Interface*
ALR	*Adaptive Link Rate*
CBR	*Case-Based Reasoning*
CeTI-SP	*Centro de Tecnologia da Informação de São Paulo*
CIM	*Core Information Model*
CIM-SPL	*Common Information Model Simplified Policy Language*
CLI	*Command Line Interface*
CNF	*Conjunctive Normal Form*
COPS	*Common Open Policy Service*
DTMF	*Distributed Management Task Force*
DNF	*Disjunctive Normal Form*
EAS	*Energy-Aware State*
ECR	*Energy Consumption Rating*
EMAN	*Energy Management*
EPI	*Energy Proportionality Index*
ETSI	*European Telecommunications Standards Institute*
FDT	*Final Decision Tree*
FDTI	*Final Decision Tree with Interpolation*
GAL	*Green Abstraction Layer*

GHG	*GreenHouse Gases*
GreenTE	*Green Traffic Engineering*
ICT	*Information and Communication Technologies*
ISP	*Internet Service Provider*
IETF	*Internet Engineering Task Force*
KPAT	*KAoS Policy Administration Tool*
LCP	*Local Control Policy*
LDAP	*Lightweight Directory Access Protocol*
LPI	*Low Power Idle*
MPLS	*Multiprotocol Label Switching*
MVM	*Mininet Virtual Machine*
NCP	*Network Control Policy*
NECR	*Network Energy Consumption Rating*
NETCONF	*NETwork CONFiguration, protocolo for configuring network elements*
NFV	*Network Functions Virtualization*
NP	*Non-deterministic Polynomial*
NREN	*National Research and Education Network*
OPEX	*Operational Expenditures*
OSPF	*Open Shortest Path First*
PBM	*Policy-Based Management*
PBNM	*Policy-Based Network Management*
PCIM	*Policy Core Information Model*

PCIMe	*Policy Core Information Model Extensions*
PDL	*Policy Description Language*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
PI	*Power Consumption at Idle Mode*
PM	*Power Consumption at Maximum Workload*
PMS	*Policy Management System*
PP	*Power Profile*
PTA	*Preliminary Tree A*
PTB	*Preliminary Tree B*
PUE	*Power Usage Effectiveness*
QoS	*Quality of Service*
QPIM	*QoS Policy Information Model*
RBAC	*Role-Based Access Control*
RNP	*Rede Nacional de Ensino e Pesquisa*
SC	*Synchronized Coalescing*
SDN	*Software Defined Networks*
SLA	*Service-Level Agreements*
SLS	*Service Level Specifications*
SNMP	*Simple Network Management Protocol*
SOS	*Sustainability-Oriented System*
SustNMS	*Sustainability-Oriented Network Management System*

TE	*Traffic Engineering*
UF	*Utility Function*
UML	*Unified Modeling Language*
USP	*University of São Paulo*
XML	*eXtensible Markup Language*
XACML	*eXtensible Access Control Markup Language*

CONTENTS

1 INTRODUCTION

The energy demand for operating Information and Communication Technologies (ICT) systems is growing. The ICT total electricity consumption is forecasted to increase almost 60% until 2020, reaching approximately 1,100 TWh (ERICSSON, 2013). Only in the U.S., ICT facilities are responsible for 120TWh of energy consumption annually, corresponding to 3% of all U.S. expenses. The country is the second in energy consumption, close to the first, China, and four times higher than the third one, Japan (COOK et al., 2014). Besides incurring in high operational costs, this significant energy consumption leads to GreenHouse Gases (GHG) emissions. According to GeSI (The Global e-Sustainability Initiative), ICT is responsible in average for 2% of the carbon emissions worldwide, and this amount is expected to grow to 2.3% in 2020[1](GeSI, 2012). In absolute terms, this is a significant amount that must be addressed.

In the ICT sector, there are two major fields to discuss when it comes to numbers of energy consumption: datacenter service providers and telecom operators. The datacenter energy demand is the fastest growing part (including servers, networking, cooling), aligned with the massive building of infrastructure for both public and private cloud computing. From 2012 to 2013, its power demand increased 7%, reaching 40GW (near 350TWh), and it is now expected to grow 81% by 2020 (COOK et al., 2014). However, how much the network actually consumes is not consensus:

[1]These figures include usage (operation) and embodied (production) emissions, but, according to (MALMODIN et al., 2010), usage leads to more carbon emissions. The carbon emissions are determined considering the direct and indirect operational emissions, besides the electricity purchase. To determine the amount of CO_2 equivalent (CO_2e) correspondent to the spent energy, an emission factor in $kgCO_2e/kWh$ is used. Considering the global electricity emission factor of $0.6kgCO_2e/kWh$, the 2020 ICT energy consumption results in $0.66GtCO_2e$ (and the 2.3% in (GeSI, 2012) is equivalent to $1.27GtCO_2$).

4% in (EMERSON-ELECTRIC, 2009), 12% in (ABTS et al., 2010), “one third” in (KLIAZOVICH; BOUVRY; KHAN, 2010), 9% in (KOUTITAS; TASSIULAS; VLAHAVAS, 2012), 23% in (KACHRIS; TOMKOS, 2013), 22% in 2011, projected to 24% in 2020 in (COOK et al., 2014). Figure 1 summarizes the different findings.

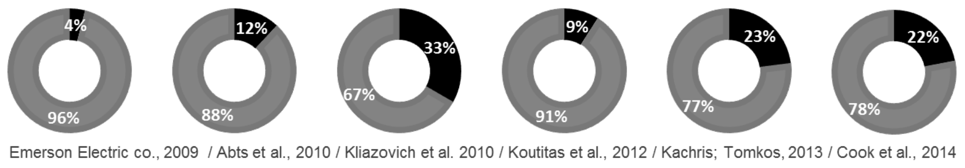


Figure 1: Network power consumption in datacenters according to different authors

Assuming the networking accounts for 10 to 20% of energy consumption in datacenters considering the aforementioned references, current efforts on servers and cooling energy consumption reduction, responsible for roughly 70% of datacenters consumption, will make the share of networking much higher, having the potential to raise this figure up to about 50% (ABTS et al., 2010).

For operators, the network infrastructure scenario is even more challenging. According to an Ericsson report (ERICSSON, 2013), energy costs are among the most significant ones that network operators have to absorb. Verizon, for instance, indicates that the electricity to run its networks surpassed 92% of their carbon emissions in 2013 (VERIZON, 2013). According to Lambert et al. (LAMBERT et al., 2012), the consumption of telecom operators’ networks (only the networks) was 260TWh in 2012. Moreover, their energy consumption growth rate is higher than the world’s growth rate. Bolla et al. (BOLLA et al., 2011) assert that barely relying on novel low-consuming silicon technology is not enough to cope with such a demand, and sheds light upon novel management paradigms. They suggest adapting the network requirements and management to take advantage of the periods when the traffic load is not as high as in rush hours, a typical situation.

Given the operational expenditure to account for energy consumption that networks impose on the telecom operators and Internet Service Providers (ISP) and the consequent GHG emissions of network infrastructure, there is an increased demand for a network energy efficient operation. Besides the efforts on using different energy sources, such as (RIEKSTIN et al., 2013) and (NGUYEN et al., 2013), several capabilities and protocols have been proposed to cope with energy efficiency in networks. Some examples of capabilities that act at different scopes and that have implementation details available are:

- Adaptive Link Rate (ALR) (GUNARATNE et al., 2008);
- Synchronized Coalescing (SC) (MOSTOWFI; CHRISTENSEN, 2011);
- Energy-aware routing (a green OSPF - Open Shorthest Path First) (CIANFRANI et al., 2012);
- Green Traffic Engineering (GreenTE) (ZHANG et al., 2010);
- ElasticTree (HELLER et al., 2010);
- Sustainability-Oriented Network Management System (SustNMS) (COSTA et al., 2012); and
- Green Paths (VELDT et al., 2014).

However, to the best of the author's knowledge, there is no proposal on how to orchestrate energy efficiency capabilities, i.e., coordinate and combine many of such capabilities in the same network, or choose the best capability for a given scenario (e.g. in a particular load situation, SustNMS will save more energy than ALR), ensuring that a capability not suited to the current bandwidth utilization will not be applied and lead to congestion or packet loss. Also, there are no solutions in the literature focused on orchestrating capabilities using business policies.

The expression of business-level policies and its subsequent translation to device-level actions and configuration increases the automation level of the network management, turning it less error prone. This can be achieved through Policy-Based Network Management (PBNM). PBNM uses policies to manage systems (STRASSNER, 2003), providing a centralized solution to reduce the complexity of the management task. Policies can be expressed in different abstraction levels, and the translation between them is called Policy Refinement.

To manage energy efficiency capabilities or provide green services to customers, one can use sustainability-oriented policies. Through these policies, it is possible to offer green Service-Level Agreements (green SLAs) to customers interested in saving energy and reducing emissions, and to manage the network in an energy efficient manner using operational policies. However, how to refine and use high-level business sustainability-oriented policies is not evident either in legacy networks or in more modern infrastructures based on Software Defined Networks (SDN). Several requirements must be addressed in this regard.

1.1 Objectives

Within this context, the objective of this work is to present a method[2], implemented in the Sustainability-Oriented System (SOS), which orchestrates energy efficiency capabilities using sustainability-oriented policies.

1.2 Organization

This work is structured as follows. Chapter 2 describes the related work regarding energy efficiency capabilities, policy-based network management and the different policies abstraction levels, besides bringing these concepts to the sustainability domain. A discussion on the motivation and requirements for a complete refinement method and the analysis of the existing methods in light of these requirements are

[2]For simplicity, “SOS method” will be the term used to refer to the method applied in the Sustainability-Oriented System (SOS).

presented in Chapter 3. In Chapter 4, a method to address the related requirements is developed. Chapter 5 presents the implementation details of the method. Chapter 6 covers the experiments and the method validation. Final considerations and future work are presented in Chapter 7.

2 RELATED WORK

There are many proposals to improve the energy efficiency in a network. Such capabilities can act locally, inside a node and its components, or have a global view of the whole network. In this Chapter, different energy efficiency capabilities are presented, including their scope of actuation in Section 2.1. Such capabilities are expected to act in a network that already has other capabilities being applied, such as the Quality of Service (QoS) ones. However, managing a system with different capabilities, each of which having distinct purposes, is not an easy task.

One solution to deal with such a complex task is Policy-Based Network Management (PBNM), which uses policies to manage systems, presented in Section 2.2. Policies can be related to QoS, access control or, more recently, to sustainability. All these types of policies can have different levels of abstraction, and the translation between them is called Policy Refinement. The existing approaches for Policy Refinement are presented in Section 2.3. The Chapter is concluded with an analysis of the open challenges that will be addressed later on this thesis.

2.1 Energy Efficiency Capabilities

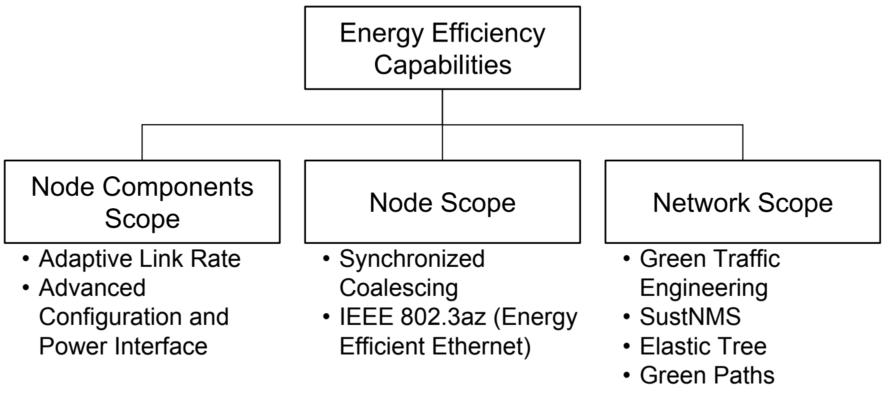
There are many different solutions focusing on energy efficiency in networks, ranging from local chip-level enablers, for more power-efficient nodes, to routing protocols. They are applied in different devices, such as routers and switches. Most of the current approaches use capabilities and protocols that, in general, are based on standard techniques or mechanisms that are already partially available in computing systems.

2.1.1 Energy Efficiency Capabilities Categories

Bolla et al. (BOLLA et al., 2011) classified the existing solutions to reduce energy consumption in communication networks into three categories: (1) re-engineering, which addresses the design and materials used in networking equipment; (2) dynamic adaptation, which deals with adapting the network according to traffic or service requirements; and (3) sleeping/standby, which puts to sleep unused devices or parts of the device. Some capabilities could be applied at the network level, thus requiring knowledge of the entire network, or at the device level, requiring only local knowledge.

Other classification of techniques and mechanisms can also be found in (BILAL; KHAN; ZOMAYA, 2013), (BIANZINO et al., 2012), and (GARG; BUYYA, 2012): traffic/resource consolidation, selective connectedness, proportional computing, and virtualization. Traffic/resource consolidation creates opportunities to save energy based on the network behavior (workloads), adapting the network in order to change the state of unused equipment. Selective connectedness refers to the dynamic adaptation of devices. It allows parts of the device to go idle for some time, as transparently as possible, moving network-related traffic processing from high-consumption main board CPUs to low-power devices or external proxy devices. This technique is also referred to as interface proxying in (BOLLA et al., 2011) and (BIANZINO et al., 2012). Proportional computing was first introduced by (BARROSO; HOLZLE, 2007) and refers to the idea of the system consuming energy in proportion to its utilization, including techniques of link rate adaption. Virtualization allows more than one service to operate on the same piece of hardware, thus improving the equipment utilization.

Schlenk et al. (SCHLENK et al., 2013) proposed a taxonomy that, rather than just describing the capability (rate adaption, sleeping, and energy-aware routing), takes the scope of the capability into account: **Sub-system** (node components, memory), **System** (network nodes), and **Network**, as illustrated in Figure 2. As this taxonomy also takes into account the scope of actuation, we further detail the capabilities following this classification.



Energy Efficiency Capabilities
Node Components Scope
Node Scope
Network Scope
• Adaptive Link Rate
• Advanced Configuration and Power Interface
• Synchronized Coalescing
• IEEE 802.3az (Energy Efficient Ethernet)
• Green Traffic Engineering
• SustNMS
• Elastic Tree
• Green Paths

Figure 2: Energy Efficiency Capabilities Classification

2.1.2 Energy Efficiency Capabilities Details

There are many proposals related to the **sub-system scope**, mainly because of the influence of personal computers and battery energy solutions. Adaptive Link Rate (ALR) is among the most cited; it allows reducing or increasing the link rate between two interfaces in accordance with the traffic. It is intended to use existing Ethernet data rates (GUNARATNE et al., 2008). Another capability in the **sub-system scope** is ACPI (Advanced Configuration and Power Interface). It comprises rate adaption (P-States) and sleeping capabilities (C-States) (BOLLA; BRUSCHI; RANIERI, 2009).

In the **system scope**, one can cite Synchronized Coalescing (SC) (MOSTOWFI; CHRISTENSEN, 2011). The approach aims to create more idle periods during which it is possible to put not only interfaces, but also other components of the device in a low power consumption state. While the device is in this state, incoming packets are buffered until a threshold is exceeded. The device then wakes up and forwards the buffered packets. One can also cite the standard IEEE 802.3az, known as Energy Efficient Ethernet, that defines mechanisms to put the device in idle mode when there is no data to be sent, in a way that allows it to wake up quickly when a new packet arrives (CHRISTENSEN et al., 2010).

Control gets more complicated in the **network scope** because techniques of traffic consolidation require network-wide coordination in order to cooperate with system scope capabilities. If each device performs energy optimizations in an independent way, the overall network energy consumption could be greater than in a cooperation scenario, but the end-to-end QoS requirements might not be fulfilled (BOLLA et al., 2014).

In this scope, Cianfrani et al. (CIANFRANI et al., 2012) proposed a modification of current link-state routing protocols, enabling some network links to power off during low traffic periods, a type of "Green" Open Shortest Path First (OSPF). Green Traffic Engineering (GreenTE) is used to free some links by moving their traffic onto other links (ZHANG et al., 2010).

The authors of SustNMS, a sustainability-oriented Policy-Based Network Management (PBNM) system (COSTA et al., 2012), designed it based on Green TE (Traffic Engineering) and network-level policies to analyze trade-offs between energy efficiency, performance, and reliability. This system extends the PBNM architecture defined by IETF (Internet Engineering Task Force) (WATERS et al., 1999), including three modules: a sustainability monitor, a QoS (Quality of Service) monitor, and a model repository (including power profiles and availability models to be used by the system). With the information from these modules, it performs Green TE and puts unused devices to sleep.

In SustNMS, to calculate the Watts/bits ratio used to select the paths in which the traffic will be concentrated, freeing others to enter in sleep mode, the authors used power profiles. Power profiles represent how many Watts are dissipated in an equipment according to the load it is handling. There are two types of power profiles: one that does not vary significantly with the load (more common in legacy devices), and one that linearly scales with load (an expected behavior in more recent and green devices) (JANUARIO et al., 2013). The authors suggested four power profiles that represent these two types of equipment (l_{max} refers to the maximum load of the equipment):

$$PP_1 = \begin{cases} 120W & \text{if standby state} \\ 200 + l(500/l_{max})W & \text{otherwise} \end{cases} \tag{2.1}$$

$$PP_2 = \begin{cases} 170W & \text{if standby state} \\ 200 + l(1000/l_{max})W & \text{otherwise} \end{cases} \tag{2.2}$$

$$PP_3 = \begin{cases} 250W & \text{if standby state} \\ 1000W & \text{otherwise} \end{cases} \tag{2.3}$$

$$PP_4 = \begin{cases} 220W & \text{if standby state} \\ 300 + l(5000/l_{max})W & \text{otherwise} \end{cases} \tag{2.4}$$

For instance, the amount of Watts dissipated by a device with the first power profile while sleeping is always 120W. While operating, the amount should be calculated considering the current load. If the load is 10Mbps and the maximum load the equipment can handle is 50Mbps, the device will dissipate $200 + 10 * (500/50)$, that is, 300W.

As another example of **network scope** capability, there is the Elastic Tree (HELLER et al., 2010), a network-wide power manager targeted at data centers fat-tree topologies, which dynamically adjusts the set of active network elements (links and switches) in accordance with the traffic load, putting unused devices to sleep.

Going further in the question of energy savings, Van der Veldt et al. (VELDT et al., 2014) proposed the Green Paths, a capability able to choose the paths on the network based on carbon emissions. The authors propose to calculate the GreenHouse Gases emissions per path in the network considering the average emission factor in each location (in CO_2/kWh). To calculate this for all paths, the authors compute the emissions for each simple path using the energy spent in such path multiplied by the average emission factor of the location. To find all possible simple paths, the authors propose to have them all precomputed or use some type of heuristics. In all studied cases, using load proportional equipment reduces the carbon footprint, and using the Green Paths approach reduces the footprint even further. Table 1 summarizes the described energy efficiency capabilities, their goals, and scope.

Table 1: Examples of Energy Efficiency Capabilities

Capability	**Goal**	**Scope**
ACPI (BOLLA; BRUSCHI; DAVOLI, 2009)	Change clock level or power level according to load demand (voltage/frequency scaling and sleep states)	Sub-system
Adaptive Link Rate (GUNARATNE et al., 2008)	Reduce energy by reducing the link rate (e.g., from 1Gbps to 100Mbps), according to the traffic being handled by the interfaces	Sub-system
Switch Coalescing (MOSTOWFI; CHRISTENSEN, 2011)	Join packets to send data bursts and create more idle periods, then allowing nodes to sleep	System
IEEE 802.3az (Energy Efficient Ethernet) (CHRISTENSEN et al., 2010)	Put the device in idle mode when there is no data to be sent in a way that allows it to wake up quickly when a new packet arrives	System
Energy-Aware Routing (Green OSPF) (CIANFRANI et al., 2012)	Coordinate routing to enable links to be put to sleep	Network
Green Traffic Engineering (ZHANG et al., 2010)	Perform green traffic engineering, moving traffic to maximize link utilization and allowing unused links to sleep	Network
SustNMS (COSTA et al., 2012)	Concentrate flows and put unused devices to sleep, according to the power models of the devices	Network
Elastic Tree (HELLER et al., 2010)	Manage a fat tree topology, concentrate traffic and put some nodes to sleep, saving energy	Network
Green Paths (VELDT et al., 2014)	Choose the paths to use according to the carbon emissions, calculating the emissions using the average emission factor of each location.	Network

These capabilities have different parameters that can be configured according to the necessity. As examples, it can be mentioned: in SC, the *DutyCycle* (the amount of time the equipment must remain in sleep mode), *tOn* (the time the equipment must stay in operation mode), and *threshold* (the number of packets that determine if the node should continue operating or sleeping), and, in ElasticTree, the *Safety Margin.* The *Safety Margin* “is the amount of capacity reserved at every link by the optimizer” (HELLER et al., 2010). It is planned to accommodate processing overheads, traffic bursts, and load increases. The higher the safety margin gets, the higher the reserved capacity at the links and the possible obtained performance.

2.1.3 Energy Efficiency Capabilities Isolated Operation Issue

The energy management capabilities outlined in this Section were designed to operate autonomously and independently from each other. When more than one capability is present in a network node or the whole network, there is a significant potential for conflicts among these capabilities. Such conflicts could reduce or negate energy savings, or even lead to undesired behavior, such as repeatedly turning on or putting a node to sleep. Figure 3 presents an example of what could happen if conflicting capabilities are acting in the same network/equipment.

In Figure 3, it is shown the case in which SustNMS and SC conflict while operating on the same nodes, since both end up trying to put nodes to sleep. Such scenario does not consider any type of hierarchy between these two capabilities. In the given example, SC is operating and decides to put two nodes, R2 and R3, to sleep. All the links connected to the nodes are stopped from sending traffic for a determined period of time (time to remain off, tOff). The network interfaces enter in Low Power Idle (LPI) mode and some of the node components can be turned off.

At this time, the tOff (time off) counter is started. SustNMS monitors the whole network and may understand this as a connectivity issue. This happens because SustNMS, as a centralized network capability, ensures the presence of all nodes in a network, and both nodes sleeping, R2 and R3, out of the SustNMS control, means there is no path available in the network. If that happens, it may decide to wake both nodes up, negating the SC savings.

After some time, SustNMS decides to concentrate the traffic in one of the two paths and to put one of the two nodes, R3, to sleep, in order to save energy. However, the tOff counter can be expired at this time, and SC puts that node, R3, back to work (and starts the tOn counter, the time powered on), negating the savings again.

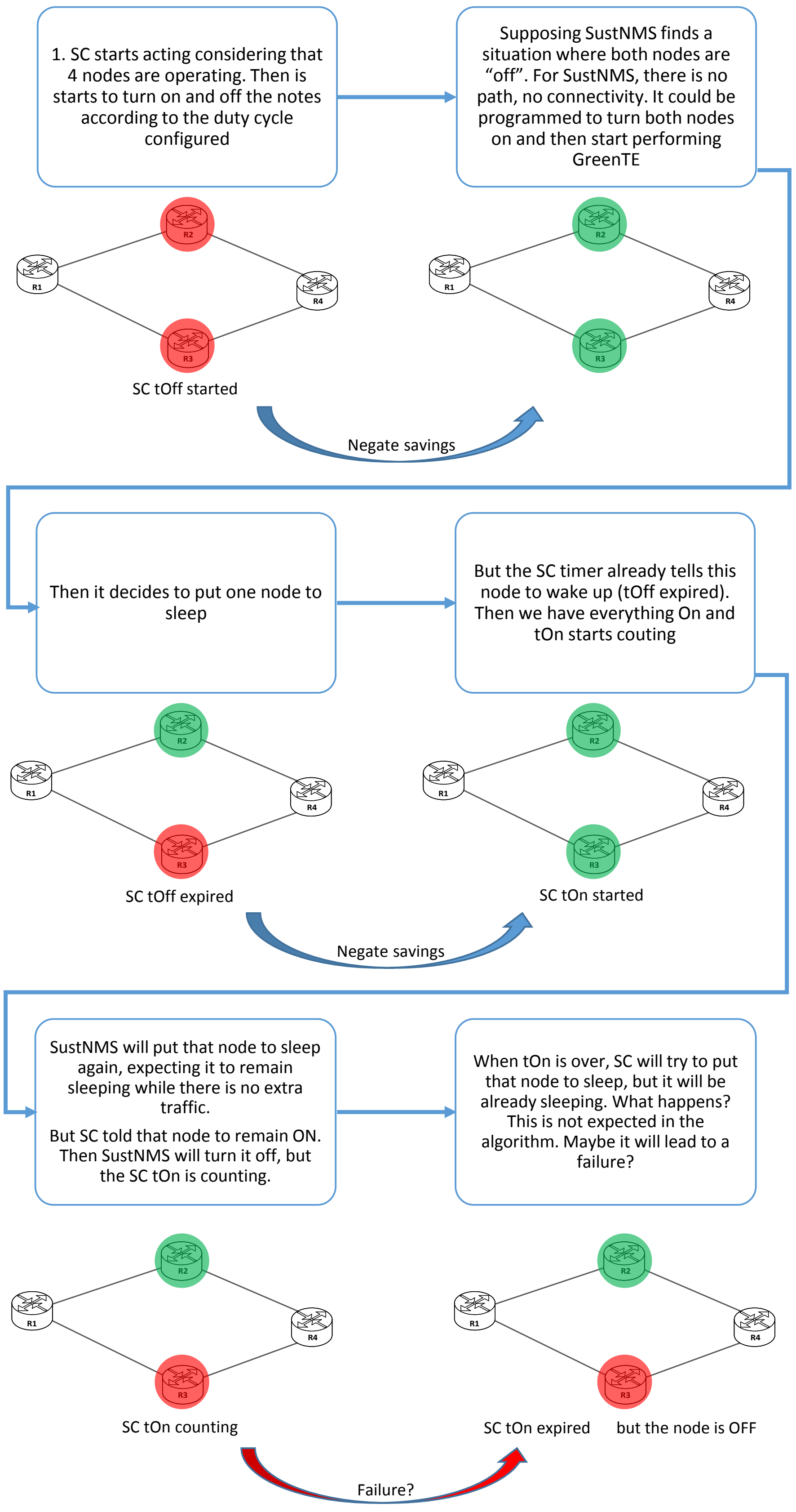

1. SC starts acting considering that 4 nodes are operating. Then is starts to turn on and off the notes according to the duty cycle configured
Supposing SustNMS finds a situation where both nodes are "off". For SustNMS, there is no path, no connectivity. It could be programmed to turn both nodes on and then start performing GreenTE
R1
R2
R3
R4
SC tOff started
Negate savings
Then it decides to put one node to sleep
But the SC timer already tells this node to wake up (tOff expired). Then we have everything On and tOn starts couting
SC tOff expired
SC tOn started
Negate savings
SustNMS will put that node to sleep again, expecting it to remain sleeping while there is no extra traffic.
But SC told that node to remain ON. Then SustNMS will turn it off, but the SC tOn is counting.
When tOn is over, SC will try to put that node to sleep, but it will be already sleeping. What happens? This is not expected in the algorithm. Maybe it will lead to a failure?
SC tOn counting
SC tOn expired
but the node is OFF
Failure?

Figure 3: Example of a conflicting situation between two energy efficiency capabilities

SustNMS puts that node, R3, to sleep again, but the SC tOn is counting. The packet count, responsible for ensuring that the node will not sleep under high loads, will be smaller than the defined threshold because no packets will be being transmitted. The SC counters (packet count, timer) are no longer valid and there is no way to indicate this to SC. When tOn expires, SC tries to put it to sleep, but it is already sleeping.

2.2 Policy-Based Network Management and Sustainability

Managing a system with different capabilities, both for energy efficiency, or QoS, or access control, is a complex task. Policy-Based Network Management (PBNM) uses policies to manage systems, providing a centralized solution to reduce the complexity of the management task, besides making it more efficient and less error prone.

Moffett and Sloman (MOFFETT; SLOMAN, 1993) define a policy as “a plan of an organization to achieve its objectives”. Policies in ICT define the desired behavior of systems and their components. The RFC 3198 (WESTERINEN, 2001) states that a policy can be defined from two perspectives: as a goal to guide decisions, executed within a particular context; or as a set of rules to manage and control access to network resources, as defined in the RFC 3060 (MOORE, 2003). Strassner (STRASSNER, 2003) defines a policy as a set of rules or goals used to manage or control access to a set of Information and Communication Technology (ICT) resources and services.

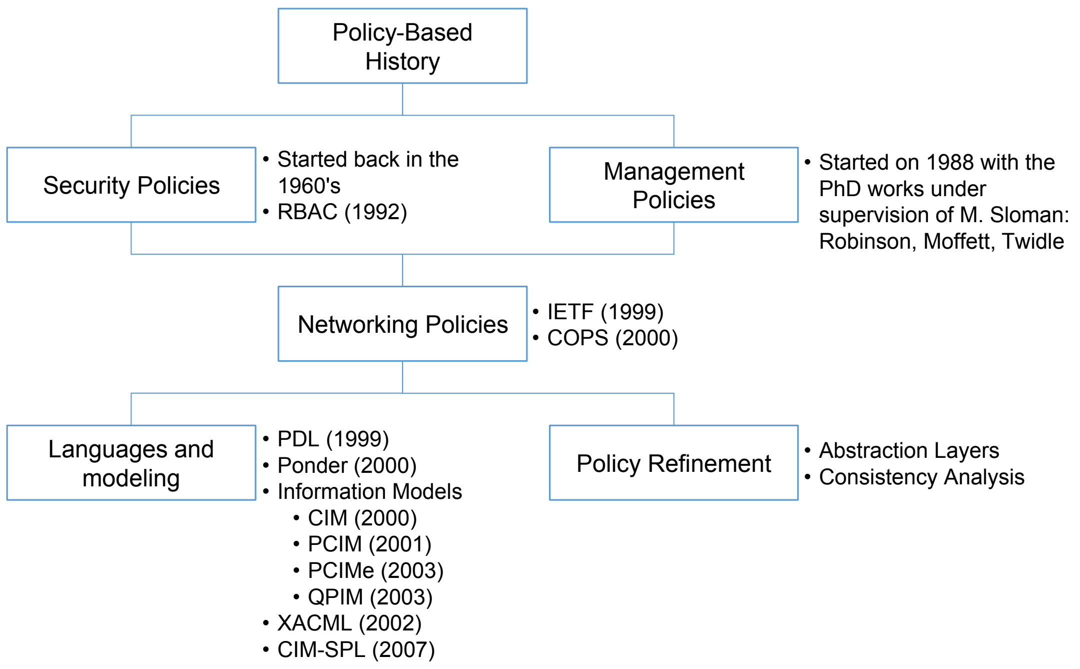
Most authors divide policies in two types: one to state what must be done in the system, given a particular condition, called **obligation** or **management policy** (e.g. if there is a failure on an equipment, put the standby equipment to work); and another to define what is allowed or not, restricting the system access, called **authorization** or **access control policy** (e.g. students should not have access to other students grades in the management system) (SLOMAN, 1994) (STRASSNER, 2003). **Authorization** policies are expected to be less dynamic than **obligation** policies, since the latter are triggered by events, but are dormant until the event occurs, while **authorization** policies are acting all the time (SLOMAN, 1994). Wies (WIES, 1995) extended this classification including information regarding lifetime, geographical scope, organizational structure, type of service, targets, and management capabilities to which the policy applies.

Considering the challenges and the complexity of managing large distributed systems, especially sustainability-oriented ones, the management of systems driven by policies is being used to decrease the management complexity. This is called Policy-Based Management (PBM) and, in case of networks, Policy-Based Network Management (PBNM). Boutaba and Aib (BOUTABA; AIB, 2007) presented an extensive work on the historical perspective of Policy-Based Management systems until 2007. Figure 4 shows the topics and some examples.

Besides presenting the different types of policies, the authors show the history of works related to networking policies, such as the framework proposed by the Internet Engineering Task Force (IETF). This framework represents the essential components that a policy system must consider: a management console, a human friendly interface for policies specification; a policy repository; a Policy Decision Point (PDP), which controls the system and decides the actions that are going to be enforced; and some Policy Enforcement Points (PEP), which will apply the decision taken by the PDP (WATERS et al., 1999). Figure 5 shows the IETF framework. IETF also proposed the Common Open Policy Service (COPS) protocol to exchange information between the PDP and the PEP (DURHAM et al., 2000).

Policies are created, modified and stored by the Policy Management System (PMS); searched and retrieved by the PDP; and enforced by the PEP. This architecture presents the essential components that a PBNM system must comprise, in a device and vendor-independent, interoperable and scalable manner (WATERS et al., 1999).

More recently, with the development of cloud infrastructures, policies have been



RBAC: Role-Based Access Control
IETF: Internet Engineering Task Force
COPS: Common Open Policy Service
PDL: Policy Description Language
CIM: Core Information Model
PCIM: Policy Core Information Model
PCIMe: Policy Core Information Model Extensions
QPIM: QoS Policy Information Model
XACML: eXtensible Access Control Markup Language
CIM-SPL: Common Information Model Simplified Policy Language

Figure 4: Policy-Based topics (BOUTABA; AIB, 2007)

gaining more importance. These infrastructures are getting bigger and the need for automation is gaining importance, and so does the policy-based management in such environment. In this environment, there are different inputs driving policies, such as regulations, privacy concerns, application requirements, and business rules (BALLAND; HINRICHS, 2014).

For OpenStack (OPENSTACK, n.d.b) environments, some of the recent efforts are in Congress, a "project to provide policy as a service across any collection of cloud services in order to offer governance and compliance for dynamic infrastructures" (OPENSTACK, n.d.a). Congress ensures that applications managed by the orchestration module (in the case of OpenStack, the Heat module) are consistent with business policies across different resources, such as compute, storage, and

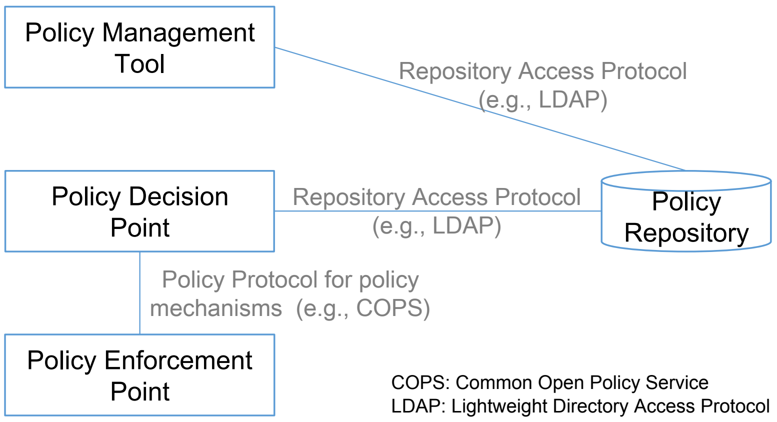

Policy Management Tool
Repository Access Protocol (e.g., LDAP)
Policy Decision Point
Repository Access Protocol (e.g., LDAP)
Policy Repository
Policy Protocol for policy mechanisms (e.g., COPS)
Policy Enforcement Point
COPS: Common Open Policy Service
LDAP: Lightweight Directory Access Protocol

Figure 5: IETF Policy Architecture

network. It works in conjunction with isolated policy engines, such as the Neutron[1] for networks in OpenStack environments, being a single point of entry for administrators to define policies that are later distributed for the enforcement points (OPENSTACK, n.d.a).

Lopez et al. (LOPEZ; KRISHNAN; FIGUEIRA, 2015) proposed a policy architecture to NFV (Network Functions Virtualization) services requirements, also applicable to general cloud infrastructures. Global Policies are defined and enforced by a Global Policy Engine. The Local Policies are enforced by their specific subsystems. A publication/subscription bus is necessary for subsystems to subscribe to global policy updates. Figure 6 represents the proposed policy architecture and framework for NFV.

Several proposals have been developed to represent policies. The Policy Description Language (PDL) (LOBO; BHATIA; NAQVI, 1999), developed in 1999 at Bell-Labs, is an event-based language . It has a simple syntax that represents event-condition-action rules, only supporting obligation rules (DAMIANOU, 2002). Ponder (DAMIANOU et al., 2001) is the most widely-used policy language (BRADSHAW; USZOK; MONTANARI, 2014). The language name became associated with a complete toolkit to specify, analyze and enforce the policies. The last version, Ponder2,

[1]Neutron is responsible for providing networking as a service in OpenStack cloud environments

is a language for security and management policies, based on the object-oriented paradigm (TWIDLE et al., 2008).

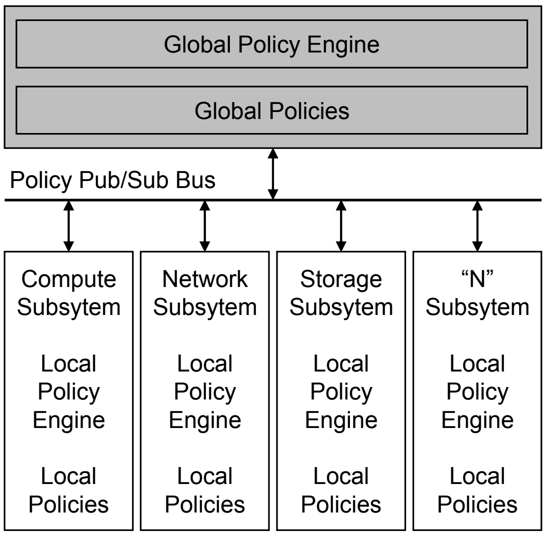


Figure 6: Policy Architecture and Framework for NFV (LOPEZ; KRISHNAN; FIGUEIRA, 2015)

IETF proposes to employ an object-oriented information model to represent policies. The Policy Core Information Model (PCIM) (later extended to PCIMe (MOORE, 2003)), defines policies rules and their different parts in a vendor independent manner, supporting the definition of several levels of abstraction. It was based on the IETF/DTMF (Distributed Management Task Force) Core Information Model (CIM), a conceptual framework for the schema of the managed environment (DMTF, n.d.). QoS (Quality of Service) Policy Information Model (QPIM) (SNIR et al., 2003) specializes PCIM to deal with QoS management. In QPIM and PCIMe, all components of a policy are represented as a class. Representing policies based on information models has as advantage that the classes can be mapped to structures specifications, such as XML. These structures can then be represented in the policy repository.

In OpenStack environments, Datalog is the policy language used by Congress. It is a formal language based on first order logic. The grammar is the following (OPENSTACK, n.d.a):

```
<policy> ::= <rule>*

<rule> ::= <atom> COLONMINUS <literal> (COMMA <literal>)*

<literal> ::= <atom>

<literal> ::= NOT <atom>

<atom> ::= TABLENAME LPAREN <term> (COMMA <term>)* RPAREN

<term> ::= INTEGER | FLOAT | STRING | VARIABLE
```

2.2.1 Policy Abstraction Levels and Policy Refinement

A PBNM solution must offer a level of abstraction to the network administrator, expressing policies in a high-level language rather than as sets of configuration parameters or commands specific to particular types of network devices (HU; FU, 2008).

A policy can be seen within a hierarchy of different abstraction levels, according to the way it is expressed, communicated, or automated (MAULLO; CALO, 1993). There may be several layers of policies, starting at high-level business policies, down to rules for implementation in network devices (SLOMAN, 1994). The number of layers may be arbitrary and application specific. Koch et al. (KOCH; KRELL; KRAEMER, 1996) state that the minimum number of layers is **three** in order to bridge the gap between the highest and the lowest level. Strassner (STRASSNER, 2003) proposes the *Policy Continuum*, composed by five levels of abstraction: Business, System, Network, Device, and Instance Levels.

Table 2 summarizes different levels of policies proposed by different authors using three layers as the basis for classification: high-level, in which the business goals are expressed; intermediate-level, in which the policies are more formally structured; low-level, in which there are procedures and implementation mechanisms. The views vary from high-level device-independent to low-level device-specific, thus enabling different constituencies to detail policies with a proper terminology (MEER et al., 2006).

Table 2: Policy Levels Proposed by Different Authors

Policy Type	Maullo and Callo (MAULLO; CALO, 1993)	Sloman (SLOMAN, 1994)	Wies (WIES, 1995)	Koch (KOCH; KRELL; KRAEMER, 1996)	Strassner (STRASSNER, 2003)
High Level	Societal (principles)	Goals (actions or operations that have to be interpreted by humans or refined by an expert, application dependent systems)	Corporate or high-level (derived from corporate goals)	Requirements enterprise viewpoint)	Business (SLA, processes, guides and goals)
	Directional (goals, such as organizational and corporate goals)				
	Organizational (practices)		Task oriented policies (related to process management)	Goal (information viewpoint)	System (details the business, including metrics, device and technology independent policies
	Functional (targets, policy maps to more precise methods like configuration specifications or workload targets)				
Interm. Level	Process (guidelines, in some structured language)	Rules (actions or operations that can be executed by automated tools)	Functional (define the use of management functions)	Operational (computational viewpoint)	Network (structures, languages, device independent, technology specific policies)
					Device (device and technology specific)
Low level	Procedure (rules, encoded procedures that are executables)	Mechanism information, rules for implementation	Low level (operate at the level of managed objects)		Instance (device specific commands)

Perry and Bauer (PERRY; BAUER, 2004) acknowledge the existence of two types of policies: those derived from Service Level Agreements (SLA) with the clients, and those that are internal, called “operational policies”, created to ensure the environment operation and to avoid violation of the SLAs. They can be derived from SLAs as well, being more likely defined by the provider.

Policy refinement is the process of translating a policy described in a high-level of abstraction (business rules, operator language) into a device-specific corresponding configuration (BANDARA; LUPU; RUSSO, 2003) (STRASSNER, 2003). Bandara et al. (BANDARA; LUPU; RUSSO, 2003) provide a more formal definition of policy refinement:

> If there exists a set of policies $P_{set} = p1, p2, ..pn$, such that the enforcement of a combination of these policies results in a system behaving in an identical manner to a system that is enforcing some base policy P_{base}, it can be said that P_{set} is a refinement of P_{base}. The set of policies $P_{set} = p1, p2, ..pn$ is referred to as the refined policy set.

Verma (VERMA, 2000) stated that, with the increasing technical complexity, a largely automated policy-based network management would simplify the administration process. However, Bandara (BANDARA; LUPU; RUSSO, 2003) identified a trade-off between automation and generality of an approach, showing that, the more automated the refinement is, the more domain-specific it is, as illustrated in Figure 7. This indicates that an automated refinement solution for the sustainability domain can be very specific.

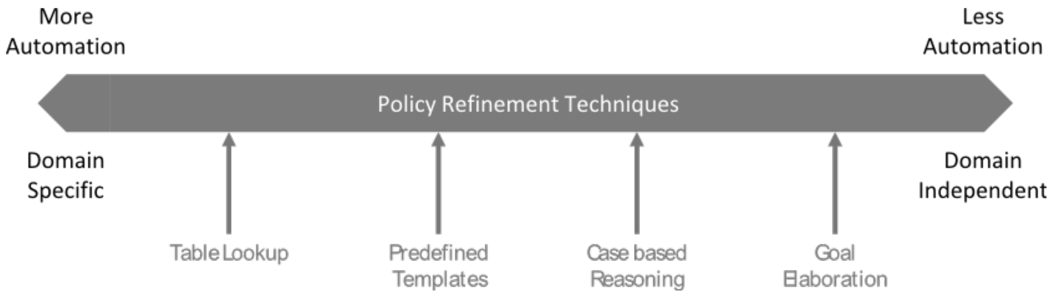


Figure 7: Trade-off between automation and generality (BANDARA; LUPU; RUSSO, 2003)

According to (CRAVEN et al., 2011), the automation of policy refinement, besides promising significant benefits, has few concrete approaches emerging. Lobo et al. (LOBO et al., 2011) stated that the policy refinement should be as automated as possible because the manual process is error prone and very dependent on a specialist, thus potentially incurring in high costs.

Figure 8 shows the pros and cons of the general existing approaches in the vision of (VERMA, 2000).

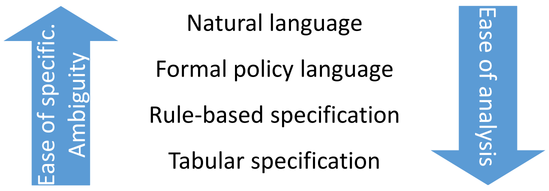


Figure 8: Analysis of the existing approaches to defining policies (VERMA, 2000)

2.2.2 Sustainability-Oriented Policies

Policies have been commonly used to specify Quality of Service (QoS) and access control rules (LYMBEROPOULOS; LUPU; SLOMAN, 2003). Now, with the new requirement of making networks more energy efficient, policies should encompass sustainability (CARVALHO et al., 2012).

According to this thesis author, a sustainability-oriented policy can be defined as follows:

A sustainability-oriented policy is a policy that manages energy efficiency features in the network.

Relaxing QoS requirements in ICT systems may enable opportunities to achieve more energy savings (KLINGERT; SCHULZE; BUNSE, 2011). According to the authors, a “Green SLA” is a type of SLA (Service Level Agreement) that offers an extended scope of energy optimization by relaxing the existing performance parameters, introducing new energy performance parameters, and providing incentives to the customers in exchange for a specified performance degradation of the services.

2.2.3 Sustainability-Oriented Policy Refinement Example

To complete the definition of sustainability-oriented policies, the definitions and an example for each abstraction level defined in (CARVALHO et al., 2012) are presented below. Carvalho et al. (CARVALHO et al., 2012) propose a methodological approach for sustainability-oriented policies refinement that uses the Policy Continuum levels defined in (STRASSNER, 2003) as basis.

A **Business Level** policy in the Policy Continuum expresses business goals or Service Level Agreements[2] (SLAs) parameters, defined in the Service Level Specifications (SLSs), the technical specifications deriving from the SLAs. Goals can be established inside a company, as the operational policies described in (PERRY; BAUER, 2004), to accomplish sustainability objectives, such as to reduce energy consumption or greenhouse gases emissions. Or they can be established based on SLAs. For sustainability, an interesting approach is to allow a relaxation of traditional SLAs as proposed by (KLINGERT; SCHULZE; BUNSE, 2011). Furthermore, the business level policies also need to encompass the description of objectives in terms of energy efficiency.

The **System Level** in the Policy Continuum describes the operation of a policy in a device and technology independent fashion, without using networking specific terminology, stating how the business level policy can be operationalized. At this level, it is expected to specify the metrics to accomplish the SLAs and the goals are detailed. Table 3 lists sustainability metrics examples.

A **Network Level** policy in the Policy Continuum is device independent, but technology-specific. It is commonly expressed in an Event-Condition-Action form. Sustainability-oriented policies at the network level should comprise obligation

[2]An SLA is a contract between a service provider and its customers to formally define expectations and obligations in their business relationship, which generally concentrate to specify QoS parameters (LYMBEROPOULOS; LUPU; SLOMAN, 2003)

Table 3: Examples of Sustainability Metrics in the Policy Continuum

	Metric	**Details**
Business	Energy related OPEX	Operational expenditures with energy
System	Watts-hour per energy source	Amount of energy used per different energy source
	Power Usage Effectiveness (PUE)	Total Facility Power divided by IT Equipment Power (WANG et al., 2012)
Network	Network Energy Consumption Rating (NECR)	Watts/bits of a set of equipment (MANRAL, 2010)
	CO_2 emissions	Given the energy consumption of the network and energy source (WBCSD; WRI, 2005)
Device and Instance	Energy Consumption Rating (ECR)	Energy Consumption divided by the Effective System Capacity (WANG et al., 2012)
	EPI (Energy Proportionality Index)	(Power Consumption at Maximum Workload (PM) - Power Consumption at Idle Mode (PI)) divided by PM multiplied by 100% (WANG et al., 2012)

policies, e.g., to put the routers in path X to sleep if the workload is smaller than 20%, and authorization policies, e.g., to allow a network function to put the router to sleep. This level faces a greater degree of dynamicity than the system level. For instance, suppose a virtual router that was migrated to a new locality. Although such an operation is transparent to the system level, the network level must take into account some particularities of the new locality, such as another energy source or another underlying topology.

The **Device Level** in the Policy Continuum is device and technology specific, implying that a policy at this level is described with respect to protocols and features directly supported by a network node. The role of the devices is also relevant at this level. An administrator can thus create a green role for devices that take part in a green solution.

The last level of the Policy Continuum is the Instance Level. The instance level policies express the machine-readable commands (e.g. NETCONF, SNMP, OpenFlow, or CLI commands) for each device. This level is tightly related to node and vendor-specific characteristics and to particular software releases. Figure 9 summarizes all the levels for sustainability-oriented policies.

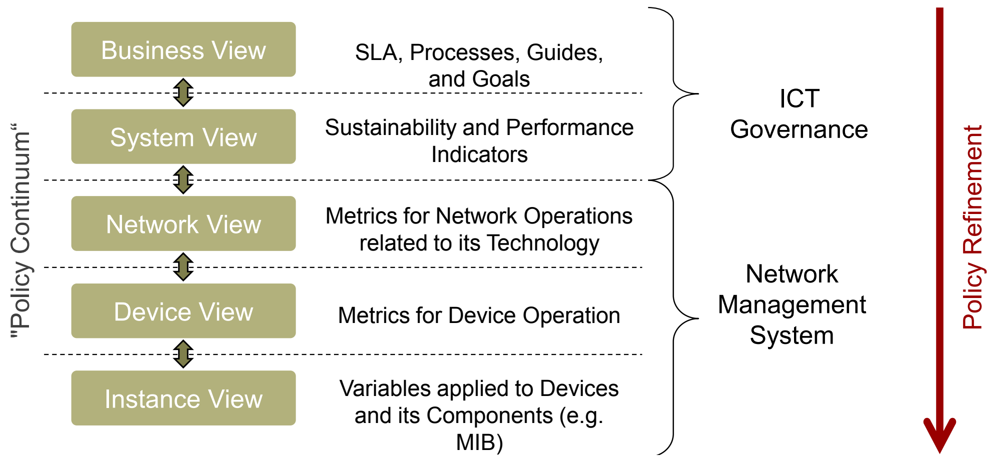

Business View
SLA, Processes, Guides, and Goals
System View
Sustainability and Performance Indicators
Network View
Metrics for Network Operations related to its Technology
Device View
Metrics for Device Operation
Instance View
Variables applied to Devices and its Components (e.g. MIB)
"Policy Continuum"
ICT Governance
Network Management System
Policy Refinement

Figure 9: Sustainability-Oriented Policies Levels according to the Policy Continuum (CARVALHO et al., 2012)

According to (CRAVEN et al., 2011), the automation of policy refinement promised important benefits, but very few concrete approaches have emerged since then. According to (RUBIO-LOYOLA, 2008) and (CRAVEN et al., 2011), we are still far away from a generic solution that covers the gaps between SLAs and high-level goals definition. The refinement problem remains as a worthy long-term goal, and the existing solutions are practical only in simple scenarios (BRADSHAW; USZOK; MONTANARI, 2014).

2.3 Policy Refinement Methods

Figure 10 groups the existing methods for Policy Refinement according to their approaches and evolution.

2.3.1 Rule-Based Approaches

The approach of Verma (VERMA, 2000) is domain specific and one of the most automated proposals. It uses table lookup techniques for network QoS management. It defines only two levels of policies: one at the business level, another at a technology-specific level, between which the refinement method is automated. The author

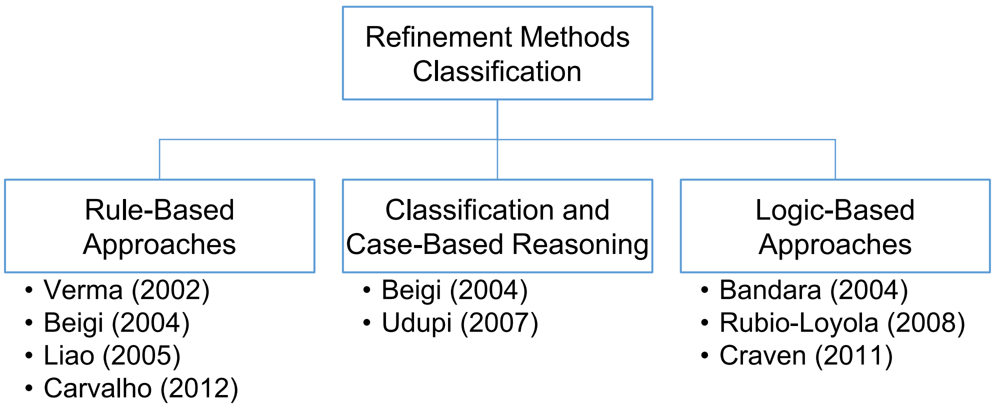


Figure 10: Refinement Methods Classification

proposes to use tables to relate users, applications, and devices to classes of service. The method performs table lookups to build the relationships during the refinement, thus depending on the correctness of the table contents. This drawback is compensated by the easiness of analyzing contradictions and coverage of such a rule-based notation.

Verma proposes a module able to determine the network topology, users, and applications, in conjunction with the capabilities available. The result validation is certified by the table lookup approach, responsible for validating the information and for performing various types of parameters checks. Regarding policy analysis, the author mentions coverage verification (if there is any type of request that is not covered by the policies) and proposes algorithms to be used to check policy conflicts and unreachable policies. To detect conflicts, the author suggests using topological spaces. Any potential conflict detected is solved through the attribution of priorities to the conflicting policies. Verma also checks feasibility; that is if the refinement target can be achieved. This can be determined by using queue models to predict if the policy target is going to be achieved.

Verma models the period of the day in the policies, but does not model scenario changes. As the method is only able to handle QoS policies, it does not support the representation of sustainability-oriented policies and the orchestration of green capabilities. Beigi et al. (BEIGI; CALO; VERMA, 2004) classify this approach as part of the Static Transformation category of their tripartite classification.

Liao and Gao (LIAO; GAO, 2005) extend the translation of Verma (VERMA, 2000) by proposing a non domain specific approach based on recipes, i.e., policy templates. The recipes define all possible refinement alternatives for each business level policy, which are branches describing possible steps based on high-level policies. The policy refinement engine automatically refines policies by choosing the refinement template, based on the conditions of the templates. The refinement process starts with the policy refinement engine receiving a tagged abstract policy and recipes. If there is a match between the tags and a policy in the repository, the refinement engine produces a concrete policy (i.e., ready to be applied) or an enforceable policy (i.e., to be used by an agent).

Carvalho et al. (CARVALHO et al., 2012) propose a methodological approach for sustainability-oriented policies refinement, from the business level down to the instance level, with grounds on the rule-based methods and based on the Policy Continuum described in (STRASSNER, 2003). The process of policy refinement starts at the business level, where high-level policies are translated down to the system level by incorporating sustainability and performance indicators. The system level policy is operationalized at the network level using the Ponder2 framework (TWIDLE et al., 2008). The policies described in Ponder2 must be interpretable by the devices in the network. The device level uses a protocol, such as SNMP (Simple Network Management Protocol), to refine the policy to the instance level. The instance level policy then applies actions and provides information to the upper levels. This approach can be seen as a first step towards an automated policy refinement for sustainability-oriented policies, defining a methodology that could be used to develop an automated approach to the policy refinement problem.

2.3.2 Classification-Based Refinement and Case-Based Reasoning

Classification-Based and Case-Based Reasoning (CBR) approaches usually have just two abstraction levels, including the implementation level. Nonetheless, they represent important techniques in the policy refinement field. Beigi et al. (BEIGI; CALO; VERMA, 2004) detail three approaches to perform refinement: static rules, table lookup, and case-based reasoning application, which is more detailed. The method deploys the knowledge learned from the past system behavior for predicting its present and future behavior. The system maintains a database of previous cases, in which each case is a combination of business objectives and configuration parameters corresponding to those objectives. When a new configuration is needed, the system tries to find the closest matching case in the database or an interpolation between a set of matching cases to determine the appropriated configuration. However, the authors affirm that the effectiveness of the approach depends on having a rich enough set of cases to be consulted in the database.

According to Boutaba and Aib (BOUTABA; AIB, 2007), the approach has a number of weaknesses, such as the difficulty to populate the case database, and the possibility of false acceptance due to " generalizations made based on wrongly constructed sets of cases". The main advantage would be that the system " becomes increasingly effective as its case database grows in size". Beigi et al. (BEIGI; CALO; VERMA, 2004) state that a policy transformation module must be used in conjunction with the technique, so that the policy can be translated and, at the same time, take advantage of the CBR, using and learning from the past to predict the system future behavior.

The approach of Udupi et al. (UDUPI; SAHAI; SINGHAL, 2007) statistically classifies relevant low-level system attributes through static rules and decision trees. This is performed in order to generate policies maintaining the relevant attributes for system consistency. To execute this, the approach counts on four main phases:

- Test and Development: in this step, the method creates a specific system configuration based on a given high-level policies and execute a workload. This workload is a manual input of data around the target high-level policies to generate specific results in order to perform the classification phase;
- Classification: in this phase, a classification is performed on the data set collected before, based on a classification algorithm to generate a decision tree. The decision tree is used to verify if a path satisfies or not the high-level policies;
- Policy Derivation and Refinement: this phase derives the paths generated in the classification phase into policies. The refinement strategy applied at this level uses the distribution statistics of the attributes on these true paths;
- Allowed and Restricted Ranges: the allowed ranges parameters are derived from all the refined policies, by unifying operations over the individual allowed ranges.

The test and development phase relies on a manual input of workloads and static rules and the main goal is to classify important parameters. Besides, the authors state that their method was developed for performance related goals, but the method could be extended to comprise sustainability-oriented goals. The approach presents a useful method for system monitoring and health checking when it is deployed and running. The work of Beigi et al. could be used in conjunction with other modules to fulfill sustainability-oriented requirements, in addition to providing classification and learning features.

2.3.3 Logic-Based Approaches

Bandara et al. (BANDARA; LUPU; RUSSO, 2003) proposed a method based on Event Calculus as the formalism base, a useful way to specify event-driven systems. The first step is to translate abstract goals into operationalized goals, relying on the KAOS methodology (DARIMONT; LAMSWEERDE, 1996). The second step takes these goals and maps them to specific modules. The method demands the system description with domain specific information: objects and respective domains; available management operations and which objects they affect; and policy rules. This modeling can be done using UML (Unified Modeling Language). The relationship between the goals and the system description is called a Strategy. After describing the system and goals, the next step is to perform abductive reasoning[3], which allows to derive the facts that must be true for goals to be achieved considering the system description.

Rubio Loyola et al. (RUBIO-LOYOLA et al., 2006), instead of using abductive reasoning, applied model checking. The authors presented a policy refinement framework, applicable to any domain, grounded in goal-elaboration methodologies and analysis of reactive systems (a system that responds to external events). The approach uses Linear Temporal Logic[4] to define relationships between goals, requiring one expert to define the goals, another to select which goal to use in each case, and an automated policy encoding to translate the defined goals into Ponder2 expressions (TWIDLE et al., 2008). The process of deriving low-level actions from high-level policies is done using a formal approach and model checking, instead of abductive reasoning.

In both cases, the goals were modeled in a formal manner, what can help, for instance, in verifying if the refined policies meet the high-level policies requirements. Charalambides et al. (CHARALAMBIDES et al., 2006) show how such technique can be used to detect conflicts that emerge at run-time, besides presenting a proposal for specifying policies to automate conflict resolution. A set of rules with logical

[3]Abductive Reasoning “is concerned with generating hypotheses about the observations or with reasoning to the best explanation” (SCHVANEVELDT; COHEN, 2010).

[4]Linear Temporal Logic “is an infinite sequence of states where each point in time has a unique successor, based on a linear-time perspective” (http://www.cs.colostate.edu/ france/CS614/Slides/Ch5-Summary.pdf)

predicates detects and signals conditions where conflicts may occur. To solve the identified conflicts, the authors proposed attributing different priorities to the policies. However, such approach may not solve all conflicts, and some of them may require human intervention for resolution. Application-specific conflicts are even harder to handle since they can depend on the system current state. In this case, the network administrator can predefine policies that provide a resolution if a conflict occurs.

Craven et al. (CRAVEN et al., 2011) propose an automated policy refinement method based on four stages: policy decomposition, operationalization, re-refinement and deployment. The inputs of this approach are:

- The initial business level policy, defined in a structured natural language;
- The domain description, which is a UML model containing a representation of the structure of the classes, kinds of possible associations, possible operations on instances of the classes and an instance repository, which records the objects existing in the domain and the relations between them;
- Obligation or authorization policies that are decomposed and operationalized; and
- Decomposition rules representing how actions and objects described at high level relate to those at a lower level.

The authors use a variant of Event Calculus[5] to describe the state of the system and to express conditions under which a policy applies. The refinement process interleaves two stages: decomposition and operationalization. The decomposition stage receives the decomposition rules given by the input and matches the operationalized policies with object classes. The operationalization stage uses the domain description and the high-level policies to provide information about how an action can be implemented.

[5]Event calculus “is a formal language for representing and reasoning about dynamic systems” (BANDARA; LUPU; RUSSO, 2003).

At this point, by comparison of the instance repository with the conditions in a policy, the resources to which actions should be applied are selected. Then the policies are tested to assert whether they are expressed in terms that a Policy Enforcement Point (PEP) can understand. Finally, the decomposition stage is performed again, if necessary; this is a stage of the re-refinement, intended to guarantee the accuracy of the refined instance level policies.

The decomposition rules relate actions to components. In a dynamic scenario, new decomposition rules must be defined. The authors say that, in such cases, re-refinement can be a way of automating the necessary adjustments, but do not give further details. The authors also state that a powerful policy analysis component is essential (CRAVEN et al., 2011).

The logic-based approaches can be adapted to interpret sustainability metrics, rules, or actions, and the domain description could encompass them. However, to address the orchestration of green capabilities, they would require the implementation of a completely new module as an extension.

2.3.4 Other Related Initiatives

KAoS (USZOK et al., 2008) is a framework that uses ontologies to define policies and the relationships among their parameters. Agrawal et al. (AGRAWAL et al., 2005) define KAoS as “a collection of componentized policy and domain management services originally designed for governing software agent behavior, and then adapted to grid computing”. A policy is specified in the KPAT (KAoS Policy Administration Tool) module. The policies are then translated using pre-defined ontologies to a format that can be monitored and enforced. The method is also able to detect conflicts, but the decision on what to do after detecting a conflict relies on the network administrator. The method is applicable to any domain, since the domain is ontologically modeled in the system. Automated policy refinement is mentioned by the authors, but not further detailed (USZOK et al., 2011) (BRADSHAW et al., 2013).

Recently, as Software Defined Networks (SDN) have gained increasing importance, some approaches to policy-based network management have been proposed for such environment, such as Procera (VOELLMY et al., 2013) and Monsanto et al. (MONSANTO et al., 2013).

The authors of Procera (VOELLMY et al., 2013) propose a controller architecture and a control language intended to offer more expressiveness in SDN domains, providing means for network operators to express policies in an easier way. It is based on principles of functional reactive programming, which consist of continuous time-varying behavior and event-driven reactivity (WAN; TAHA; HUDAK, 2002). Additionally, Procera responds to events from sources other than OpenFlow, such as events triggered by user authentication or bandwidth usage. The approach is intended to network operators, and the high-level policies must be manually translated down to the network level.

The work of Monsanto et al. (MONSANTO et al., 2013) presents another policy-based approach for SDN environments. They propose a framework on top of a POX controller[6] and use the syntax of the Pyretic language to allow high-level definition of policies. Pyretic is a platform embedded in Python language that embodies concepts such as packet-forwarding policy, network conditions monitoring, and dynamic policy to respond network events, enabling network operators to create sophisticated SDN applications. The Pyretic language extends the Frenetic project (FOSTER et al., 2013), a collaborative effort between researchers to develop a language for SDN applications. The method also proposes parameterized policies, similarly to what was proposed in the patent US7617304 (DEVARAKONDA; HERGER, 2009). This allows updating the parameters of the policies whenever a scenario changes. The Pyretic language provides several features in order to support network management, such as QoS support with

[6]POX is a platform for developing and prototyping network control software using Python

rate limiting and prioritization, which is useful to sustainable purposes. However, like Procera, the highest level in this solution is focused on network operators, not suiting for business level.

2.4 Chapter Final Remarks

Several capabilities have been proposed to provide energy efficiency in networks, acting inside the components of a node or considering the whole node or network. They are proven to achieve significant savings when applied in an isolated way. However, there is no proposal on how to orchestrate them, i.e., coordinate or combine such capabilities in order to operate in a conflict-free and more efficient manner, choosing the best capability (or combination of capabilities) for each network scenario of utilization. PBNM can help in performing such complex task, but there is no automated way to translate sustainability-oriented policies down to the network, neither considering orchestration.

This work proposes a method to overcome these challenges. Some of the energy efficiency capabilities presented in this Chapter will be emulated and used to validate the implementation of the orchestration method. The method uses sustainability-oriented policies and policy refinement concepts to bring business directives into the network operation. The next Chapter presents the motivation and elicits the requirements a method for Policy Refinement should fulfill in order to overcome the identified challenges.

3 REQUIREMENTS AND MOTIVATION FOR A SUSTAINABILITY-ORIENTED POLICIES REFINEMENT AND ORCHESTRATION METHOD

In this Chapter, the motivation for a sustainability-oriented policies refinement method is presented in Section 3.1 and the requirements for such method are discussed and specified, including the orchestration of energy efficiency capabilities. As shown in Figure 11, after investigating and studying the energy efficiency capabilities, PBNM (Policy-Based Network Management), the different policy levels and sustainability-oriented policies, the existing refinement methods were evaluated. The requirements and the evaluation of the existing refinement methods are also available in (RIEKSTIN et al., 2015b).

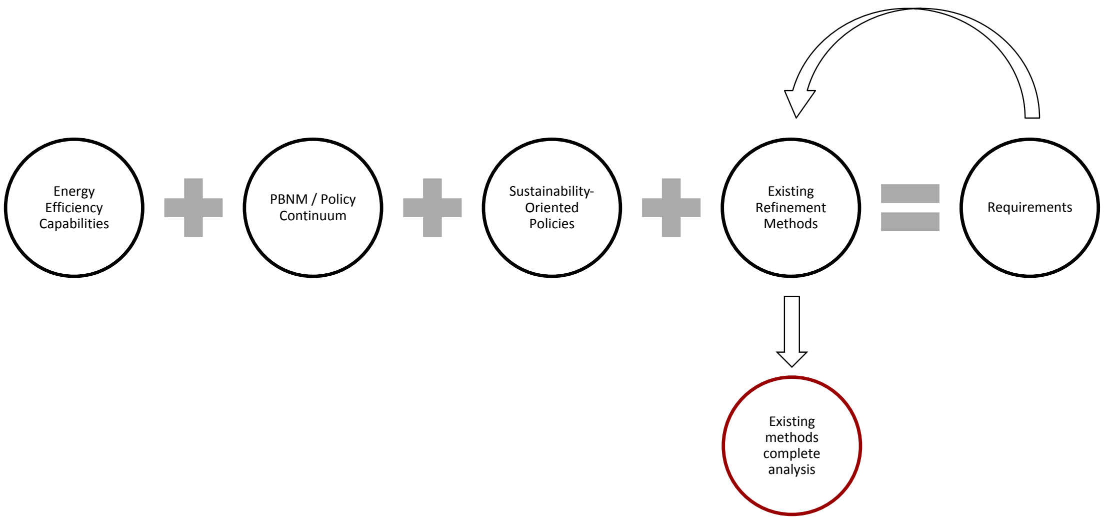

Energy Efficiency Capabilities
PBNM / Policy Continuum
Sustainability-Oriented Policies
Existing Refinement Methods
Requirements
Existing methods complete analysis

Figure 11: Method to identify the requirements

This evaluation enabled the identification of the requirements that should be fulfilled by a complete refinement method. The same methods were after re-evaluated in light of the identified requirements to check their completeness in this scenario. This analysis is presented in Section 3.3.

3.1 Motivation for a Sustainability-Oriented Policies Refinement Method

In order to understand the applicability of a sustainability-oriented policies refinement method in a production scenario, some traffic profiles are discussed here. One important characteristic to observe is the opportunity to take advantage of recurrent idle periods, such as during the night. The Brazilian NREN (National Research and Education Network), RNP, for instance, presents interesting diurnal-nocturnal patterns in some links which can foster the use of energy efficiency capabilities, as shown in Figures 12 and 13.

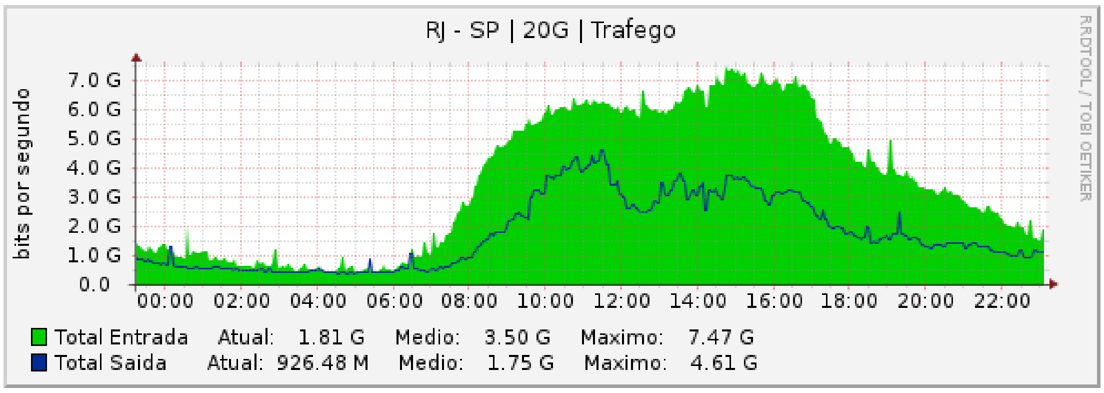


Figure 12: Brazilian NREN link usage example between two different states, São Paulo and Rio de Janeiro, during March, 16th 2015 (RNP - Rede Nacional de Ensino e Pesquisa, 2015)

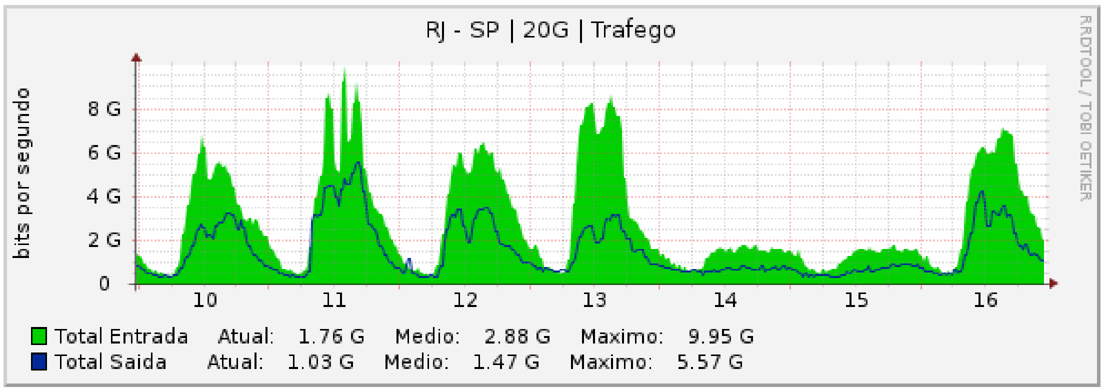


Figure 13: Brazilian NREN link usage example between two different states, São Paulo and Rio de Janeiro, from March 10th to March 16th, 2015 (RNP - Rede Nacional de Ensino e Pesquisa, 2015)

In this case, and in many other links in the RNP network, a sustainability-oriented policies refinement method could take advantage of the usage pattern to save energy. During the low usage periods, and even in the medium usage periods, the method should be able to orchestrate the existing capabilities and select which ones are better for each particular bandwidth utilization.

Another example of such an opportunity to save energy during the night is illustrated in Figure 14. This is a traffic profile from one link of the University of São Paulo campus network. It also shows a diurnal-nocturnal pattern in which the refinement method could orchestrate energy efficiency capabilities and save energy.

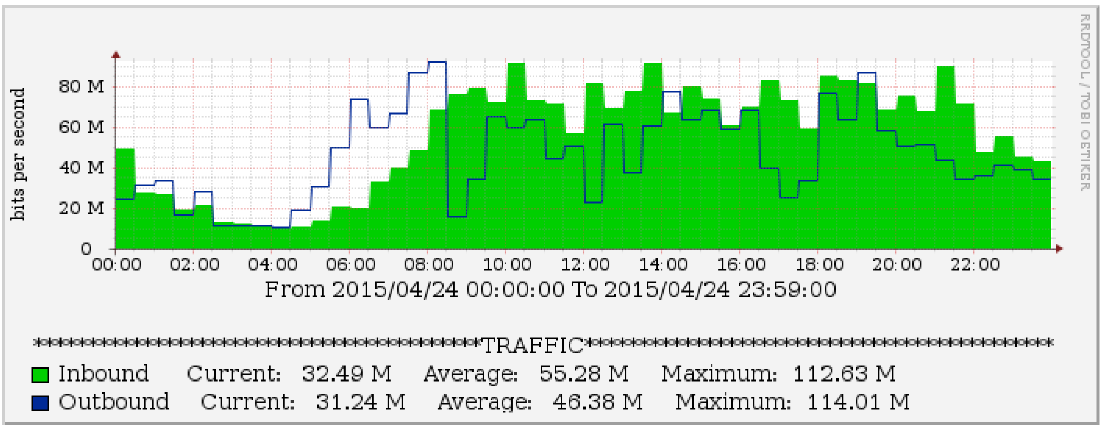


Figure 14: Example of link usage of the University of São Paulo campus network during April 24th, 2015, measured by Centro de Tecnologia da Informação de São Paulo (CeTI-SP) from the Superintendência de Tecnologia da Informação of USP

To understand in details which energy efficiency capabilities could be applied in this example, a capture file was evaluated. Figure 15 shows the traffic profile in a time interval during the night, summarized using Wireshark IO Graph tool. It is possible to see that, for instance, ALR could be applied some times, when the traffic goes below 10Mbps. Using *pcaputils* and *dpkt*, python modules which help parsing and analyzing packet captures, the SC applicability was evaluated. From 4a.m. to 5:10a.m., SC with a tOn of 11ms, a DutyCycle of 10%, and a threshold of 1000 packets could reduce the time the equipment is turned on to near 11% of the time[1].

It is important to note that not all links show very different diurnal-nocturnal

[1]SustNMS and other network scope capabilities applicability depend on the network topology.

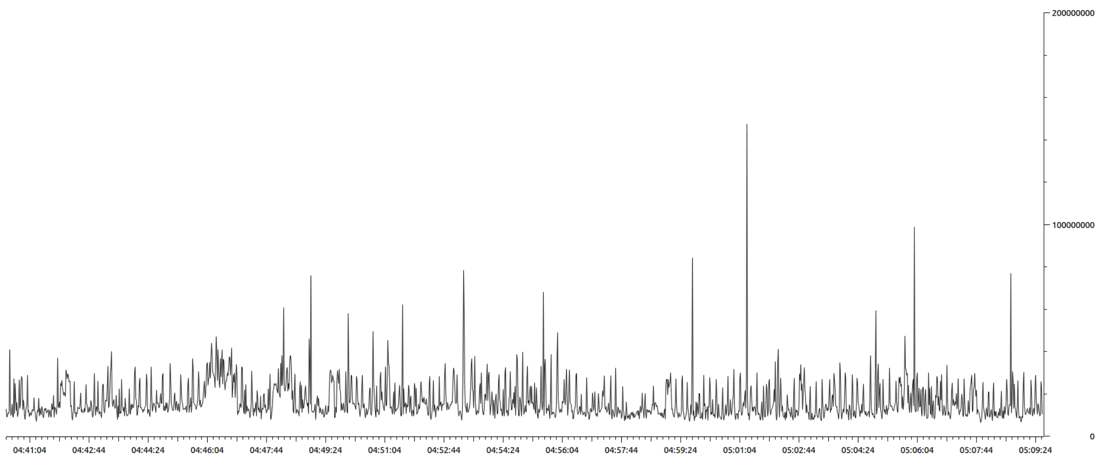


Figure 15: Sample traffic profile from the University of São Paulo campus network, measured by Centro de Tecnologia da Informação de São Paulo (CeTI-SP) from the Superintendência de Tecnologia da Informação of USP (in bits/second)

patterns. The profiles in Figure 16 and Figure 17, for instance, are from (CHO, 2008), summarized using Wireshark IO Graph tool. They represent profiles that do not vary a lot during the day. Other intervals from this source are very similar in terms of traffic variation.

In this case, while the equipment might support different energy efficiency capabilities, enabling them would not bring significant savings and could even degrade the service levels most of the time.

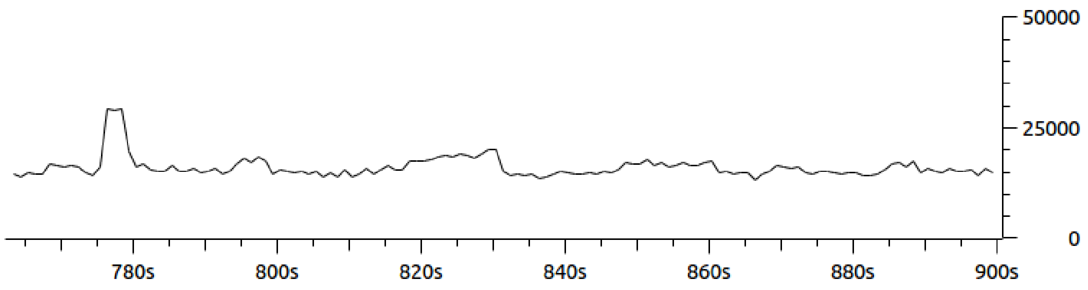


Figure 16: DatCat WIDE-TRANSIT 150 Megabit Ethernet Trace - 19h15 (CHO, 2008)

With these examples, it was possible to note that the traffic profiles may vary both at macro scale (day-night) and micro scales (as illustrated in the cap traces). A sustainability-oriented policies refinement method can account for such variations and orchestrate energy capabilities in networks, selecting the best options to enforce for each bandwidth usage level.

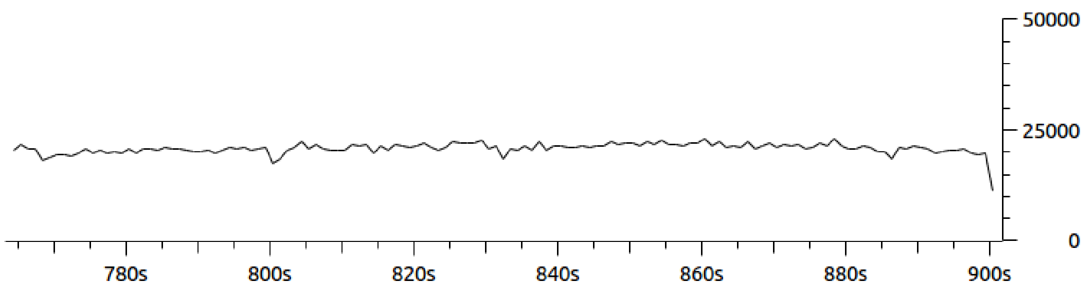


Figure 17: DatCat WIDE-TRANSIT 150 Megabit Ethernet Trace - 00h15 (CHO, 2008)

3.2 Requirements for a Sustainability-Oriented Policies Refinement Method

After describing the related work and the motivation for a sustainability-oriented policies refinement method, in this section we exploit the requirements a refinement method should fulfill in order to support the refinement of policies in general, including the support to sustainability-oriented policies.

As requirement (i), the refinement process should comprise **translation** steps (automated or human guided) (MOFFETT; SLOMAN, 1993). A policy refinement approach considered fully automated must refine the policies from the highest level down to the lowest level of abstraction, where the actions are applied. This translation could interpret the semantics of policies or could be limited to a syntactical transformation, which, albeit more limited, could meet the refinement requirements (WIES, 1995).

As requirement (ii), the transformation process should take into account the **resources** presented in the network (WIES, 1995) (VERMA, 2000), including the capabilities available (such as QoS and energy efficiency capabilities).

The requirement (iii) is the **verification** of whether the refined policies meet the requirements of the original policy (MOFFETT; SLOMAN, 1993). This relates to coverage analysis, which verifies if the refined policies cover all the high-level objectives. Examples of coverage analysis are to check if every member of the high-level policy is addressed by the lower-level policies, or if at least one authorization policy is applied to each object under management. Coverage analysis completeness is hard to achieve (SLOMAN, 1994). Coverage is part of a broader study area called Policy Analysis, which deals with coverage gaps, policy comparison, behavioral simulation, and conflicts.

A conflict may occur when one policy interferes in the behavior of another policy, denying its action or putting the managed objects in undesired states. This happens when there is an overlap between subjects or targets. The task of **detecting and solving conflicts**, the requirement (iv), is extremely difficult and is generally related to authorization policies. For other types of policy, conflict detection demands application knowledge or some human intervention (SLOMAN, 1994) (CHARALAMBIDES et al., 2006). Verma (VERMA, 2000) stated that, if a policy is represented without any constraints, the policy conflict detection can be shown to be NP-complete (Non-deterministic Polynomial). Therefore, some constraints are required when representing policies. Considering these constraints, the author proposed a solution using topological spaces to detect conflicts. The proposed constrain is to define the conditions separated from the action part. Each of these two parts can have multiple independent terms. Each term can be seen as an independent axis in a hyper dimensional space, and each rule defines a region in this space. Each region can be associated with a dependent term, identified by the rule. If any point in space has multiple dependent terms, there is a potential conflict, such as the policies C and D in Figure 18.

A simpler solution to detect conflicts is to search for overlaps in each pair of policies. Such solution has a running time of $O(dimensions * number\_of\_policies^2)$ (VERMA, 2000). The author suggests combining these and other simplification techniques to improve the running time of detecting conflicts. And, to solve the detected conflicts, the most common way is to make the administrator choose which

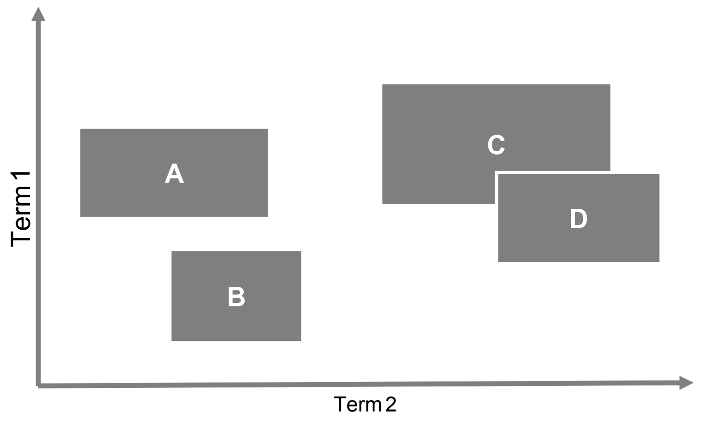


Figure 18: Topological Spaces example (VERMA, 2000)

policy has more priority, an approach also used by Kagal (KAGAL, 2002). The conflicts can be solved during translation, starting at the highest level, or in a lower level. Solving conflicts during translation results in a more complex process and may cause problems in dynamic domains (WIES, 1995).

Conflicts detection and resolution is a requirement of any type of policy. However, there can be more conflicts in the network environments in which sustainability-oriented policies are applied. It can be due to their antagonic characteristic (the trade-off between saving energy and providing services with maximum performance), whereas the sustainability-oriented policies work to reduce energy consumption, the usual QoS policy tries to maximize performance and not necessarily enables energy savings. Relaxing QoS requirements may enable opportunities to achieve more energy savings.

Policy refinement methods should also deal with **dynamicity**, the requirement (v). That is, to apply policies in different time slots or be able to determine what to do when the scenario changes (for instance, when a node migrates to another network). To deal with temporal dynamicity, Sloman (SLOMAN, 1994) cites policy constraints, which are predicates referring to global attributes such as time or action parameters. They can define allowed values in management operations, or define preconditions for the policies. Wies (WIES, 1995) suggests treating scenario changes like the initial refinement. This is because changing a target or an action, for instance, may require a complete new refinement. The same author complements the idea by stating that a policy should be able to emit notifications when any of its parts changes, so the necessary actions can be performed. Monsanto et al. (MONSANTO et al., 2013) propose to use parameterized policies. The parameters of the policies can be updated whenever a scenario changes.

Considering sustainability-oriented policies, dynamicity gains importance, given the attempts to save energy that take advantage of time periods in which the bandwidth utilization is low, as the example illustrated in Figure 19, in which it is possible to see that there is a significant difference between the usage rates during the day and the night. Or the attempts to save energy that lead to moving, for instance, a virtual node[2] to a more energy efficient location, or to a location with a different energy matrix. This would imply changing the parameters of the policies of this virtual node, which now may be in a location with different probe rate, power consumption, or even green capabilities.

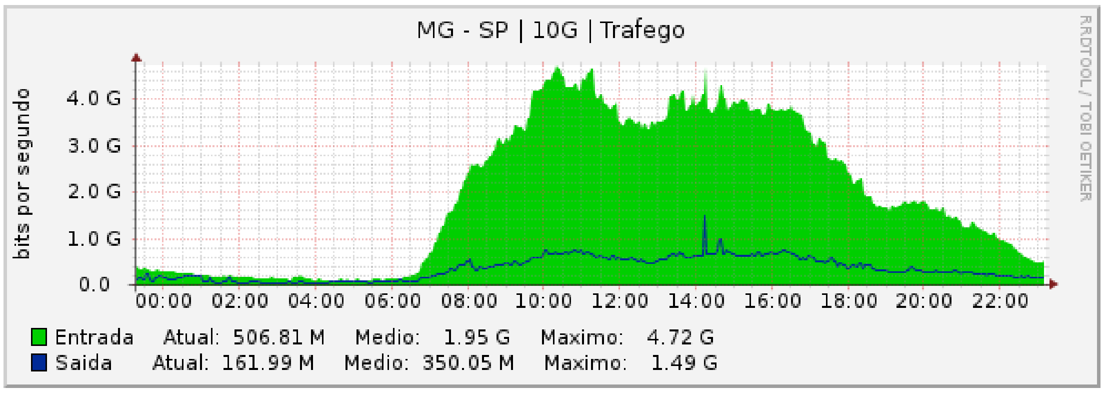


Figure 19: Brazilian NREN link usage example between two different states, São Paulo and Minas Gerais, during March, 16th 2015 (RNP - Rede Nacional de Ensino e Pesquisa, 2015)

As the next requirement, (vi), the method should be able to **represent policies** in order to keep context, coherence, and integrity of the network under determined

[2]A virtual node is a network node that runs on a virtual machine, with its own interfaces and IP addresses, as a real network node.

conditions (MAULLO; CALO, 1993). This implies that, in order to handle sustainability issues, besides supporting traditional obligation and authorization policies, the method should be flexible to accept sustainability metrics, energy efficiency rules, and interface with new features, such as changing the equipment chip operating frequency to save energy or put a router to sleep. Without modeling, for instance, the sleeping action, the system will not be able to enforce the action during the operation. Or, without modeling a new type of variable, such as the Watts/bits ratio, the system will not be able to monitor this value and take the necessary decisions.

Maullo and Callo (MAULLO; CALO, 1993) suggested using object-oriented modeling for such system representation. An information model using object-oriented diagrams, such as UML, could be developed to represent sustainability-oriented policies and all the components and relationships among them.

Another requirement, (vii), is the **orchestration** of green capabilities, i.e., the method should be able to (a) choose the best capability considering the network situation, as indicated in the motivation Section in 3.1, (b) combine different capabilities in order to increase the energy efficiency, and (c) avoid combining conflicting capabilities, such as previously illustrated in Figure 3.

Table 4 summarizes the sustainability-oriented policies refinement requirements.

Table 4: Requirements Summary

Requirement	Summary
(i) Translation	Refine down high-level policies, considering the different abstraction levels
(ii) Resources	Take into account the resources in the underlying network, including the capabilities available
(iii) Verification/ Coverage	Verify if the refined policies fulfill the requirements of the original, high-level policy
(iv) Conflicts	Detect and solve conflicts among policies
(v) Dynamicity	Deal with dynamicity of time and scenario changing
(vi) Sustainability representation	Represent sustainability-oriented and other types of policies, including metrics and specific actions
(vii) Capabilities orchestration	Orchestrate (coordinate and combine) energy efficiency capabilities to save more energy and ensure a conflict-free operation

These requirements model the desired behavior of a refinement method, which, besides supporting the refinement of different types of policies, also enables energy efficiency capabilities orchestration, a mandatory requirement considering the current efforts on saving energy. The requirements and the trade-off between generality and automation presented in Section 2.1 show that it is not an easy task to refine sustainability-oriented policies through the different policy abstraction levels. For example, it involves the translation of sustainability-related parameters and metrics from high to low-level, the construction of new network rules, and the binding of network rules with specific green technologies.

In a large distributed system, the management of policies in general, not only of sustainable ones, is a very complex and error-prone task, usually requiring experienced professionals. The proposed policy refinement method should be conceived fulfilling the discussed requirements and minimizing the human dependency and intervention in the network management, thus decreasing the probability of errors and the costs involved in retaining trained professionals specialized in low-level tasks.

The next Section evaluates different existing approaches for policy refinement, with respect to the discussed requirements. The evaluation focuses on the automation level of policy refinement approaches and on how easily they can be adapted to refine sustainability-oriented policies.

3.3 Analysis of the existing refinement methods in light of the requirements

Table 5 summarizes the analyzed methods categories with respect to the attendance of the seven requirements. In addition to these requirements, it was included in this table a row concerning “Automation versus Generality” to compare the methods also considering this aspect. None of the evaluated methods complies with every requirement of a refinement method for sustainability-oriented policy.

Table 5: Refinement Methods Categories Comparison

	Rule-Based Approaches	**Classification and CBR**	**Logic-Based Approaches**
Automation x Generality	The most automated	Somehow automated, not complete refinement	Not domain specific, less automated, demands modeling
Translation	√	Partial: only one level up the implementation	√
Resources	√	×	√
Verification	√	×	√
Conflicts	√	×	√
Dynamicity	Partial: only time	Partial: can handle time as a parameter	Partial: can handle time, but dynamic scenario is not detailed
Sustainability representation	Partial: Carvalho et al. do, others should support information models	Partial: could model sustainability parameters	Partial: could model sustainability as an input to the method
Capabilities orchestration	×	×	×

The Rule-Based Approaches category addresses the automated translation (requirement i), resources required (requirement ii), verification and coverage analysis (requirement iii), and conflicts detection and resolution (requirement iv) requirements. The dynamicity requirement (requirement v) is addressed partially, since time conditions can be modeled, but the dynamic scenario cannot. Regarding sustainability representation (requirement vi), Carvalho (CARVALHO et al., 2012) supports it, and the others could be adapted to accept sustainability information models. To conclude, regarding the seventh requirement, orchestration, the approaches that fall in this category do not address it. The proposed approaches deal with only one management capability, not being able to handle different elements and capabilities.

The Classification-Based Refinement and Case-Based Reasoning (CBR) category addresses partially the automated translation(requirement i), since it deals with only one level up to the implementation. The methods do not cover the resources required (requirement ii), verification and coverage analysis (requirement iii), and conflicts detection and resolution (requirement iv) either. The dynamicity requirement

(requirement v) is addressed partially, since time conditions can be modeled as a parameter, but there is nothing related to the dynamic scenario. Regarding sustainability representation (requirement vi), the methods could model sustainability parameters. To conclude, regarding orchestration (requirement vii), the category does not deal with capabilities, but with parameters and metrics.

The Logic-Based approaches addresses the automated translation (requirement i), resources required (requirement ii), verification and coverage analysis (requirement iii), and conflicts detection and resolution (requirement iv) requirements after the domain is modeled. The dynamicity time (requirement v) is addressed, but the dynamic scenario is not. Regarding sustainability representation (requirement vi), the domain modeling could comprise sustainability aspects, so that the methods could handle such parameters. Like the others, this category does not address the orchestration requirement (requirement vii). The proposed methods detail only one management capability being applied. It is not possible to use the evaluated methods for orchestration of various capabilities at the same time.

3.4 Chapter Final Remarks

A desirable policy refinement method should be fully automated for a network management to support sustainability and QoS. From the survey literature, it is possible to conclude that the more restricted an application domain is, the greater are their chances of developing a fully automated solution for refinement. To fully satisfy the sustainability requirements, the policy refinement method should thus be domain-specific. A generic approach demands much effort from an expert to provide application specific information. Thus, in addition to being automated, in order to be effective a method should be specific and extensible.

Among the evaluated methods, it can be verified that the most automated approaches are those related to a rule-based refinement, which are more domain-specific. The logic-based methods are more generic, but demand a significant effort on modeling the system, for example with UML. The approaches generally support policy translation (requirement i). Rule-based and logic-based are the most complete categories and allow fulfilling the requirements of resources discovery (requirement ii) and policy analysis, i.e., verification (requirement iii) and conflict detection (requirement iv). The temporal dynamicity requirement is usually fulfilled, but scenario dynamicity needs more effort in order to be fully automated (requirement v). Energy efficiency policies may demand more of this requirement considering their more dynamic behavior.

Regarding the sustainability requirements specific to representation (requirement vi), on the one hand, the methods could handle metrics, events, and actions related to energy efficiency after some adaptation. On the other hand, for the orchestration scenario (requirement vii), a whole new module is required. There is no method able to coordinate energy efficiency capabilities considering conflicts and determining which single capability or group thereof is the best option for a given network scenario.

Kephart (KEPHART, 2005), and more recently (BRADSHAW; USZOK; MONTANARI, 2014), proposed that utility function policies, which can be seen as extensions of traditional goal policies, are key for the future of PBM. Utility Functions may be interesting because they combine values for different parameters, expressing an optimization objective. However, they would only be useful if associated with interfaces and algorithms in order to be more user-friendly (KEPHART, 2005).

The next Chapter proposes a method that fulfills the requirements for a sustainability-oriented policies refinement, including energy efficiency capabilities orchestration.

4 THE SUSTAINABILITY-ORIENTED SYSTEM: SUSTAINABILITY-ORIENTED POLICIES REFINEMENT CONSIDERING CAPABILITIES ORCHESTRATION

Taking as basis the specified requirements for a sustainability-oriented policies refinement method specified in Chapter 3, the proposed Sustainability-Oriented System (SOS) is presented in detail in this Chapter. The method was outlined in (RIEKSTIN et al., 2014) and detailed in (RIEKSTIN et al., 2015a). Among all the specified requirements, this work focuses on those that change more with sustainability-oriented policies:

(i) Translation of high-level policies into enforceable policies;

(v) Addressing policies dynamicity;

(vi) Description of sustainability-oriented policies in a standardized fashion; and

(vii) Orchestration[1] of energy efficiency capabilities.

The requirement (ii), determination of the resources that are required for the policy execution, is cited, but not detailed. The requirements (iii) validation and verification if the refined policy is in accordance with the high-level policy; and (iv), policy conflicts detection and resolution, can be supported by the translation approach, but their adaption to this domain is left as a future work. Both need to be adapted to comprise sustainability-oriented policies, but the solutions themselves should be similar, such as bound checks for validation (VERMA, 2000), Petri Nets for policies verification (KAHLOUL et al., 2010) (AKRAM; PASCAL; THIERRY, 2011), or topological

[1]In cloud environments, there is a standard orchestration module that performs resource allocation (e.g., OpenStack Heat), different from the orchestration as defined in this thesis.

spaces for policy conflicts (VERMA, 2000). To fulfill the prioritized requirements, the sustainability-oriented method proposed herein:

- Addresses requirement (i) adopting a Table Lookup approach, based on Verma (VERMA, 2000), and deploying a Sustainability-Oriented Information Model to represent the policies. This approach was chosen because, albeit domain specific, it is the most automated. The translation steps could interpret the semantics of policies or could be limited to a syntactical transformation which, albeit more limited, could meet the requirements;
- Defines a Utility Function to support the selection of green capabilities in conjunction with capabilities meta-data to support conflict management among capabilities;
- Addresses requirement (v) by incorporating time conditions in the policies description and refinement of policies.

4.1 Overview of the Proposed Method

The SOS method uses Table Lookup for the high-level policies translation, in conjunction with a Utility Function. The Utility Function supports the orchestration of energy efficiency capabilities by selecting the one (or a combination of them) that maximizes the benefits considering energy efficiency and quality of service. The translation is supported by the Sustainability-Oriented Information Models to represent policies, which also support policies dynamicity. Figure 20 presents the SOS method summary, highlighting in red the supporting parts and, in blue, with dashed lines, the main contributions of this thesis, the orchestration part with the Utility Function.

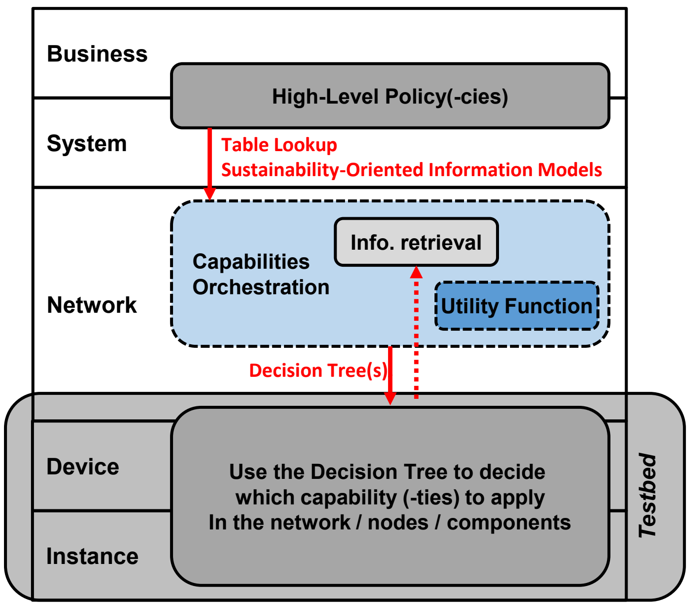

Business
High-Level Policy(-cies)
System
Table Lookup
Sustainability-Oriented Information Models
Network
Capabilities Orchestration
Info. retrieval
Utility Function
Decision Tree(s)
Device
Instance
Use the Decision Tree to decide which capability (-ties) to apply In the network / nodes / components
Testbed

Figure 20: SOS method Summary highlighting the main contributions of this thesis

4.2 Sustainability Information Models

In order to support the refinement process and to address requirement (vi), Sustainability Information Models specified using UML (Unified Modeling Language) were developed by our research, comprising what is needed for a policy to express sustainability issues in each Policy Continuum level. This thesis' author contributed in the information models design and development, which are described in (NASCIMENTO et al., 2015). This section includes only the information necessary for understanding how the models were employed in the orchestration.

The information model considers the definitions and relationships between different managed objects. Other classes were included in order to accommodate the table lookup method. According to (DAVY; JENNINGS; STRASSNER, 2008), extracting application specific information directly from the information models has some advantages, such as easy modification when needed, policy reuse in different application domains by deploying a different information model, and standardized information representation. Damianou (DAMIANOU, 2002) states that representing policies using information models has an advantage: the classes can be mapped to structure specifications, such as XML. These structures can then be implemented in the policies repository.

To model the policies at each level of the Policy Continuum, the Policy Core Information Model Extensions (PCIMe) RFC (MOORE, 2003) was used as basis, and extended to comprise sustainability-oriented aspects. PCIMe was built on top of PCIM (MOORE, 2001), and two main changes were introduced: the inclusion of new elements, extending PCIM to areas that it did not previously cover; and the update of deprecated elements, such as policy rule priorities, replaced by priorities tied to associations that refer to policy rules. PCIM, in turn, was built on top of the Common Information Model (CIM). It describes an object-oriented information model to represent the information of policies.

PCIM is not bound to a particular implementation; therefore, it can be used to exchange information in a variety of ways. The model structure comprises two types of objects:

- **Structural classes**, which define ways of representing and controlling policy information; and
- **Associative classes**, which indicate how the class instances are related.

In PCIM, a Policy (class "Policy") is defined by a set of rules (class "Policy Rules"), grouped by the "Policy Group" class. Each rule is composed of a set of conditions (class "Policy Condition") and a set of actions (class "Policy Action"). The rules can also comprise time conditions (class "Policy Time Period Condition"). Conditions can be grouped in two ways, represented by the attribute "PolicyRule.ConditionListType":

- **Disjunctive Normal Form (DNF)**: the conditions inside the same group are represented by the "AND" operation, and the groups by the "OR" operation;

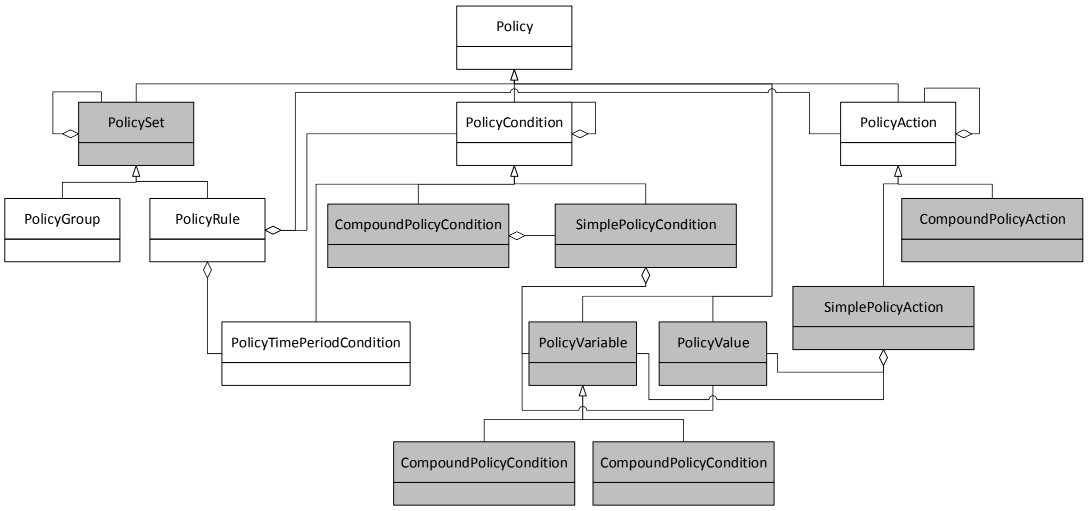


Figure 21: PCIM and PCIMe general structure (BELLER; JAMHOUR; PELLENZ, 2004)

- **Conjunctive Normal Form (CNF)**: the conditions inside the same group are represented by the “OR” operation and the groups by the “AND” operation.

The actions can be executed in a specific order using the attribute “PolicyRule.SequencedAction”. Variables and Values are used to build conditions following the structure “(<variable> MATCH <value>)” (RFC3460). Figure 21 depicts the general structure of PCIM (white) and the extensions proposed by PCIMe (gray).

Specializing the PCIM, the Quality of Service Policy Information Model (QPIM) proposes new classes to describe QoS actions (SNIR et al., 2003). To specialize PCIM to sustainability-oriented policies, new classes were created to describe sustainability actions. Besides sustainability-oriented representation, additional classes must be created to put the information models in practice.

The classes are generic enough to represent any type of policies, but the implementation requires some classes specialization, as did (BELLER; JAMHOUR; PELLENZ, 2004) for QoS. In order to support such information, the information model should also model the elements the administrator manipulates to compose the business rules (User, SLA, Class of Service, Application, Servers), and the information needed to configure the devices. For instance, Beller et al. (BELLER; JAMHOUR; PELLENZ, 2004), in their PCIMe QoS specialization information model, describe some extra classes to support the following semantics: “User (or group of users), accessing Applications (or groups of applications) in a Server (or group of servers), at a given time, receives a determined Service Level”.

Considering the sustainability-oriented policies requirements, the different Policy Continuum levels, the PCIM/PCIMe characteristics, how QoS can be handled by QPIM and the extension proposed by (BELLER; JAMHOUR; PELLENZ, 2004), and how the SLA semantics was supported by (BELLER; JAMHOUR; PELLENZ, 2004), three information models were proposed: Business/System, Network, and Device Levels for sustainability-oriented policies representation. The Instance Level Information Model is particular for each technology or vendor, and therefore was not detailed. The lower the level, the more detailed is the information model.

At this moment, the Business and System information models are the same - they completely represent both levels with the same classes. They are still considered as two different layers to ensure the generality necessary for the near future systems. The Network Level Information Model is directly influenced by QPIM and includes technology-specific, device-independent information. The Device Level Information Model is the most detailed and considers device specific information, including the variables that are expected to be managed to put the green capabilities to work.

The **GreenPolicyRule** acts as a continuation of the PolicyRule and is used to determine the conditions and actions for sustainability-oriented policies. The SimplePolicyCondition is extended by the **GreenPolicyCondition**. The latter is extended by the **EnvironmentCondition**, which relates to the traffic being handled by the network, and by the **EnergySourceCondition**, which relates to the change of energy source triggered by the user. The GreenPolicyCondition is also extended by the **TopologyCondition**, which represents the different topologies for the network. The

Variables are extended by the **GreenPolicyImplicitVariable** class, which comprises, for instance, the Watts/bits relation. The SimplePolicyAction is extended by the **GreenPlanAction** in the Business/System Levels Information Model, which determines different Green Plans for the users. This is the part that changes more in the Network Level Information Model and in the Device Level Information Model.

The Network Level Information Model details the Business and System classes, giving the policy a more technical format. In the Action part, the Sleeping and Rating actions appear, detailing the technology each policy uses to put the higher-level policies into practice. The Device Level details the Conditions and the device-specific Actions, naming the variables each green capability must handle. It models, in a generic way, the possible energy efficiency Actions in a device: sleeping and rating. The SimplePolicyAction is extended by the **GreenPolicyAction**, which determines different actions related to energy efficiency capabilities: **PolicySleepingAction** and **PolicyRatingAction**. These classes can then be extended to comprise information for the capabilities available in the network. Figure 22 illustrates the device level, the most detailed information model, in which the devices technologies are shown.

The information modeling exercise done for PCIM can also be done in the cloud computing context with the OpenStack Congress, for instance, so that the framework is able to model energy efficiency capabilities using specific constructs. The Sustainability Information Models for cloud computing environments must take into account, besides the business level policies, the capabilities, the different infrastructures (compute, storage, network), and consider virtual and physical resources, as proposed by (BARACHI et al., 2013).

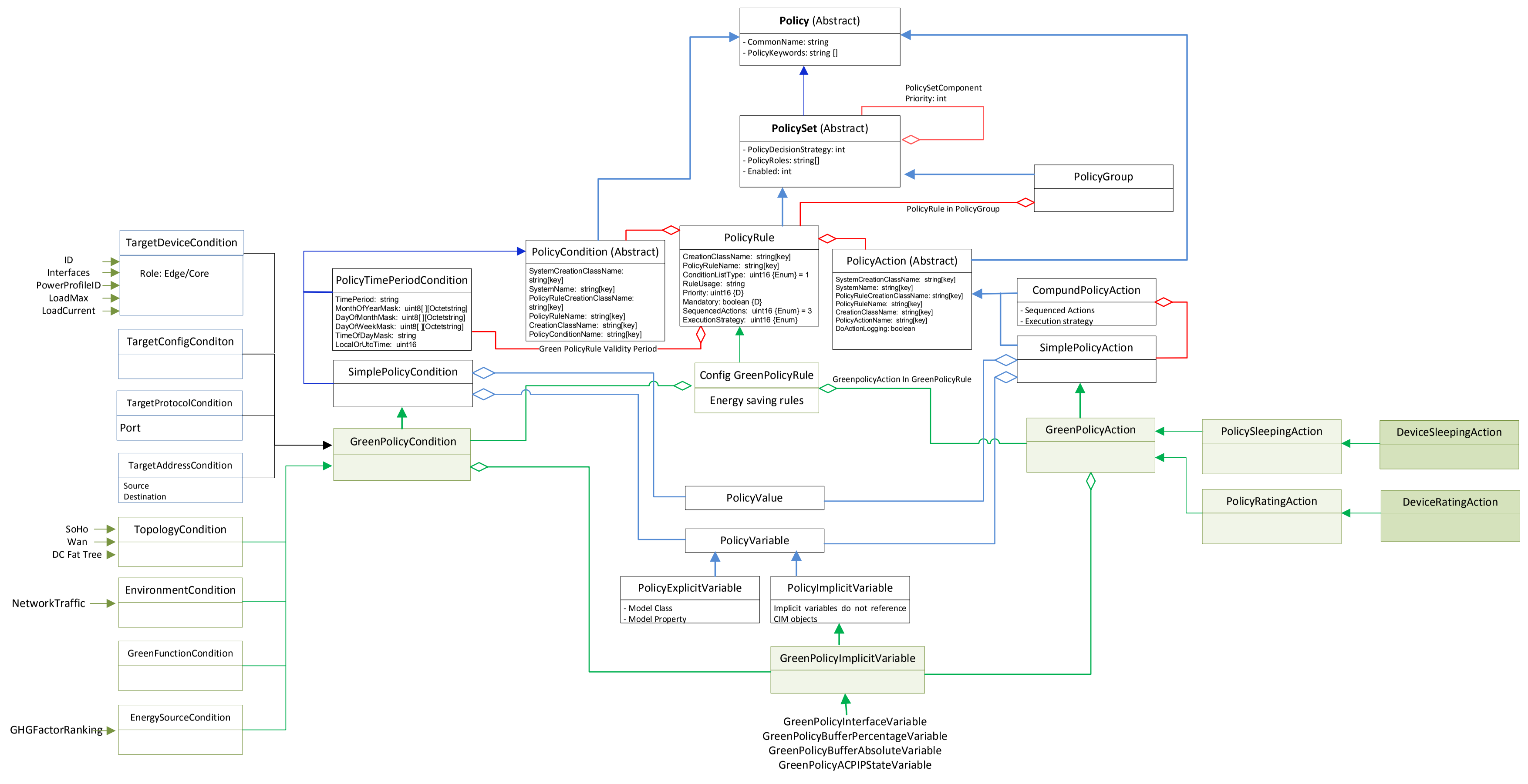


Figure 22: Device Level Information Model (NASCIMENTO et al., 2015)

4.3 Proposed Policy Refinement Approach

In order to bring business directives into the energy efficient network operation, this work proposes to deploy Table Lookup as the policy refinement approach (addressing requirement i), in conjunction with the Sustainability-Oriented Information Models presented in Section 4.2 to represent the policies (addressing requirement vi).

As described in Chapter 2, Table Lookup is a rule-based approach. The approach is a good option for domain-specific automated policy refinement. Besides the correctness relying on who defines the tables, the approach makes it easier to analyze contradictions and coverage. IETF has also chosen a tabular specification in conjunction with an LDAP (Lightweight Directory Access Protocol) database. KAoS also uses what the authors call an "efficient lookup format" (IHMC, n.d.).

The policy management tool must provide the mappings between the tables that define high-level policies and those ones that define low-level policies. With these rules defined for a specific domain, the translation can be executed in a generic manner. To represent the tables, XML (eXtensible Markup Language) (different descriptions for the different policy abstraction levels) can be used. Such translation would not interpret the semantics of policies, being closer to a syntactical transformation. Such approach is more limited, but could meet the requirements. In this work, such translation addresses the requirements, being simpler and more automated, bringing the business goals to the network in which the energy efficiency capabilities should be orchestrated.

To conclude this Section, let us present a refinement example. In order to use Table Lookup to refine policies, the tables need to be pre-defined. Considering the Information Models presented in Section 4.2, the tables to be defined are related to Environment Condition, Time Condition and Action.

The **Environment Condition** is the variable related to the high or low use of the capacity of the network structure: the input will determine the traffic load conditions for the use of the energy capabilities. Table 6 illustrates the data that can be defined as environment conditions. Each row represents one possible environment condition, while the column represents the translated policy from the Business Level to the System Level in the Policy Continuum. Table 7 represents the same process, but from the System Level to the Network Level in the Policy Continuum, already considering the network maximum capacity.

Table 6: Environment Condition Information - Business to System Level

Environment Condition - Business Level	**Environment Condition - System Level**
"If usage is low"	20%
"If usage is high"	80%
"In any condition"	-

Table 7: Environment Condition Information - System to Network Level

Environment Condition - System Level	**Environment Condition - Network Level**
20%	NetworkCapacity = 1Gbps and Load < 200Mbps
80%	NetworkCapacity = 1Gbps and Load < 800Mbps
-	NetworkCapacity = 1Gbps and Load ANY

The **Time Condition** is the input that will provide the data regarding the period of the day when the energy capabilities can be applied to the network infrastructure. Table 8 illustrates the data that can be defined as time conditions. Each row represents one possible time condition, while the column represents the translated policy from the Business Level to the System Level in the Policy Continuum. Table 9 represents the same process, but from the System Level to the Network Level in the Policy Continuum.

The last input is **Action.** It determines the energy efficiency behavior that must be applied in the network infrastructure or if the performance of the network is more

Table 8: Time Period Condition Information - Business to System Level

Time Condition - Business Level	**Time Condition - System Level**
During the night	22h < x < 6h
During the morning	6h < x < 12h
During the afternoon	12h < x < 18h
During the evening	18h < x < 22h
During the day	6h < x < 22h

Table 9: Time Period Condition Information - System to Network Level

Time Condition - System Level	**Time Condition - Network Level**
22h < x < 6h	Start: 22h / End: 6h
6h < x < 12h	Start: 6h / End: 12h
12h < x < 18h	Start: 12h / End: 18h
18h < x < 22h	Start: 18h / End: 22h
6h < x < 22h	Start: 6h / End: 22h

important than the reduction of the electrical expenditure at a given point of time. Table 10 illustrates the data that can be defined as actions. Each row represents one possible action, while the column represents the translated policy from the Business Level to the System Level in the Policy Continuum. Table 11 represents the same process, but from the System Level to the Network Level in the Policy Continuum.

Table 10: Action Information - Business to System Level

Environment Condition - Business Level	**Environment Condition - System Level**
“save energy”	“use energy efficiency in the network”
“prioritize performance”	“provide maximum QoS”
“save energy without penalizing performance”	“use energy efficiency without reducing QoS”

Each one of the business policies received is compared with the content of the respective table. The identified translated information is selected from the table, and this translated data is saved on a repository for future use.

To help the understanding of how the Table Lookup is employed by the method, an example of information defined in the Interface and its respective translation is provided below.

Table 11: Action Information - System to Network Level

Environment Condition - System Level	**Environment Condition - Network Level**
“use energy efficiency in the network”	“check in the decision tree the best capability for the given bandwidth utilization, all capabilities available are allowed”
“provide maximum QoS”	“do not apply any energy efficiency functionality”
“use energy efficiency without reducing QoS”	“apply only link rating capabilities”

An example of business policies defined in the interface is:

Environment Condition: “if usage is low”;

Time Condition: “during the night”;

Action: “save energy”.

The module opens the Environment Condition - Business to System Level table, searches for the “if usage is low” input, and then selects the percentage of the load for the condition. The module then opens the Environment Condition - System to Network Level table and performs the same operation to translate the percentage to Mbps information.

“if usage is low” 20% NetworkCapacity = 1Gbps and Load < 200Mbps

The module opens the Time Period Condition table, searches for the “during the night” input, and then selects the time (start and end time) that the method can be employed.

“during the night” 22h < x < 6h starting time: 22h, ending time: 6h

The module opens the “Action” - Business to System table, searches for the “save energy” input, and then selects the action required for the action. The module performs the same to translate from the System to the Network Level of the Policy Continuum,

“save energy” “use energy efficiency in the network” “check in the decision tree

the best capability for the given bandwidth utilization, all capabilities available are allowed”

The result is the metrics definition in a repository, already translated, and ready for the use by the orchestration part of the method:

[’NetworkCapacity==1000 and load < 200.0’, ’time>=22 and time<=6’, check in the decision tree the best capability for the given bandwidth utilization, all capabilities available are allowed ’]

4.4 Utility Function

In Economics, a utility function represents the user satisfaction level with respect to some goal. In networks, satisfaction relates to throughput, availability, and power or energy cost. The utility varies depending on external conditions; for example, when the energy is limited, a utility that decreases its power transmission is appropriate.

Jung et al. (JUNG et al., 2008) proposed using utility functions to determine the best configuration in a server environment. The purpose was to define how to allocate resources maximizing the utility provided considering SLAs, resources, and workloads. Ozel and Uysal-Biyikoglu (OZEL; UYSAL-BIYIKOGLU, 2013) stated that, for wireless networks, *Bits successfully sent per Joule* has been a well-known Utility Function that combines throughput and cost, encouraging energy efficiency:

$$UF = \frac{\sum_{k=0}^{n} MpbsSent \ - MpbsLost_{Router\ k}}{\sum_{k=0}^{n} EnergyAfterSavings_{Router\ k}} \tag{4.1}$$

The Utility Function does not generate solutions, only grades pre-defined possible solutions. Once the possible combinations of capabilities and bandwidth utilization are defined, the combination that gives more benefits considering the objectives need to be chosen. The Utility Function helps to choose the best option.

To propose a Utility Function (UF) to be used by the SOS method in order to maximize savings, but at the same time, assure a minimum level of service, the following criteria were taken into consideration:

- The UF must compare the energy spent when there is no energy efficiency capability being applied (a baseline value) with the energy spent after applying energy efficiency capabilities (the Wh value after savings);
- The UF must combine energy savings with performance parameters, in order to assure a minimum level of service. In this work, the performance is represented by the packets lost when applying energy efficiency capabilities;
- The UF must also ensure that the scenario "everything turned off" will not be selected, since it is the scenario with 100% of savings, but with no service being provided.

The following UF was defined:

$$UF = pl * \frac{1}{\frac{\sum_{k=0}^{n} EnergyAfterSavings_{Router\ k}}{\sum_{k=0}^{n} EnergyBaseline_{Router\ k}}} \tag{4.2}$$

The utility function proposed in this thesis combines energy savings and QoS, assuring a minimum level of service while trying to save energy, for an n-router topology. The term *pl* refers to packet loss. The amount of losses could be grouped, for example, in six categories, as described in Table 12. The more packets lost, the lower is the value of the utility function, and, if more than 0.5% of the packets are lost, the utility function value is zero. The categories can be changed according to the network services provided. The higher the grade a capability achieves with the utility function, the better is the combination of this capability (-ties) with the given bandwidth utilization and topology.

The graph depicted in Figure 23 illustrates the UF solution space. The UF needs to

Table 12: Example of values of packet loss and the respective “pl” values

Packet Loss	**“pl”**
0% <= x < 0.1%	1
0.1% <= x < 0.2%	0.9
0.2% <= x < 0.3%	0.8
0.3% <= x < 0.4%	0.7
0.4% <= x < 0.5%	0.6
x >= 0.5%	0

grade the goodness of a solution within this area. Ideally, the solution is close to zero losses and standby power. Therefore, the UF needs to give worse grades to solutions that increase the energy. A simple way of doing this is to consider $1/Energy$. At the same time, it needs to give better grades the lower the packet loss is. So an approach where the multiplication is done with a $1/PacketLoss$ factor could be taken. However, such a simple solution would make it difficult to distinguish between loss values associated to different service classes. Hence, the *pl* factor was defined according to the table in which arbitrary values were considered representing six service classes. The details about how the UF is used in the SOS method are presented in the next section.

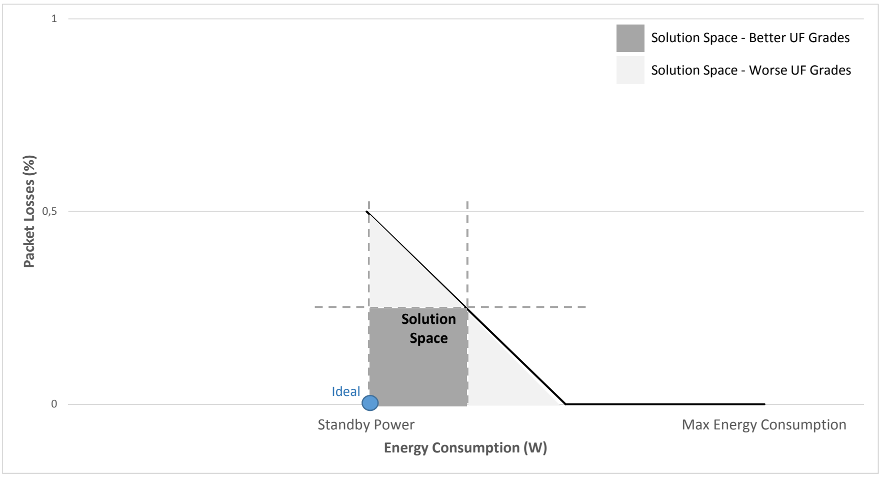


Figure 23: Utility Function Solution Space

More sophisticated mechanisms could be used to decide which option to select, like game theory tools. Such tools are able to predict the results from complex interactions and have been used to solve problems in communications networks. Further information can be found in (MACKENZIE; WICKER, 2001) and (SRIVASTAVA et al., 2005). A more detailed study on this possibility can be conducted in the future.

4.5 Orchestration of Energy Efficiency Capabilities with Policies Refinement

The SOS method orchestrates different energy management capabilities of the network infrastructure considering business directives. The method sequence diagram is presented in Figures 24 and 25. The objects represented in this diagram are the Policy Console, in which the users input the policies; the Policy Repository, where the policies are stored; the Policy Information Model, used to define the sustainability-oriented policies; the Policy Refinement, which uses the information models in conjunction with a Table Lookup approach to translate the high-level policies; and the Network Evaluator, responsible for the orchestration.

This module takes information from the Network Topology; the Managed Functionalities available in the studied network; the Power Profiles, stored in the Model Repository; the Workload Generator, which will generate a set of random workloads to be used in the Decision Trees construction; and the Network Configuration Generator, which defined the possible capabilities combinations. In the next paragraphs, each step is explained. They were grouped to facilitate the understanding.

The first step is the definition of the business policies by the network operator, based on the Business Level Information Model. This policy is translated as described on Section 4.2 (Group 0 in Figure 24). The translated policy allows the identification of parameters necessary for the network to be configured (Group 1 in Figure 24).

The next action is the determination of which capability or combination of capabilities will be employed in the network structure. The definition is performed

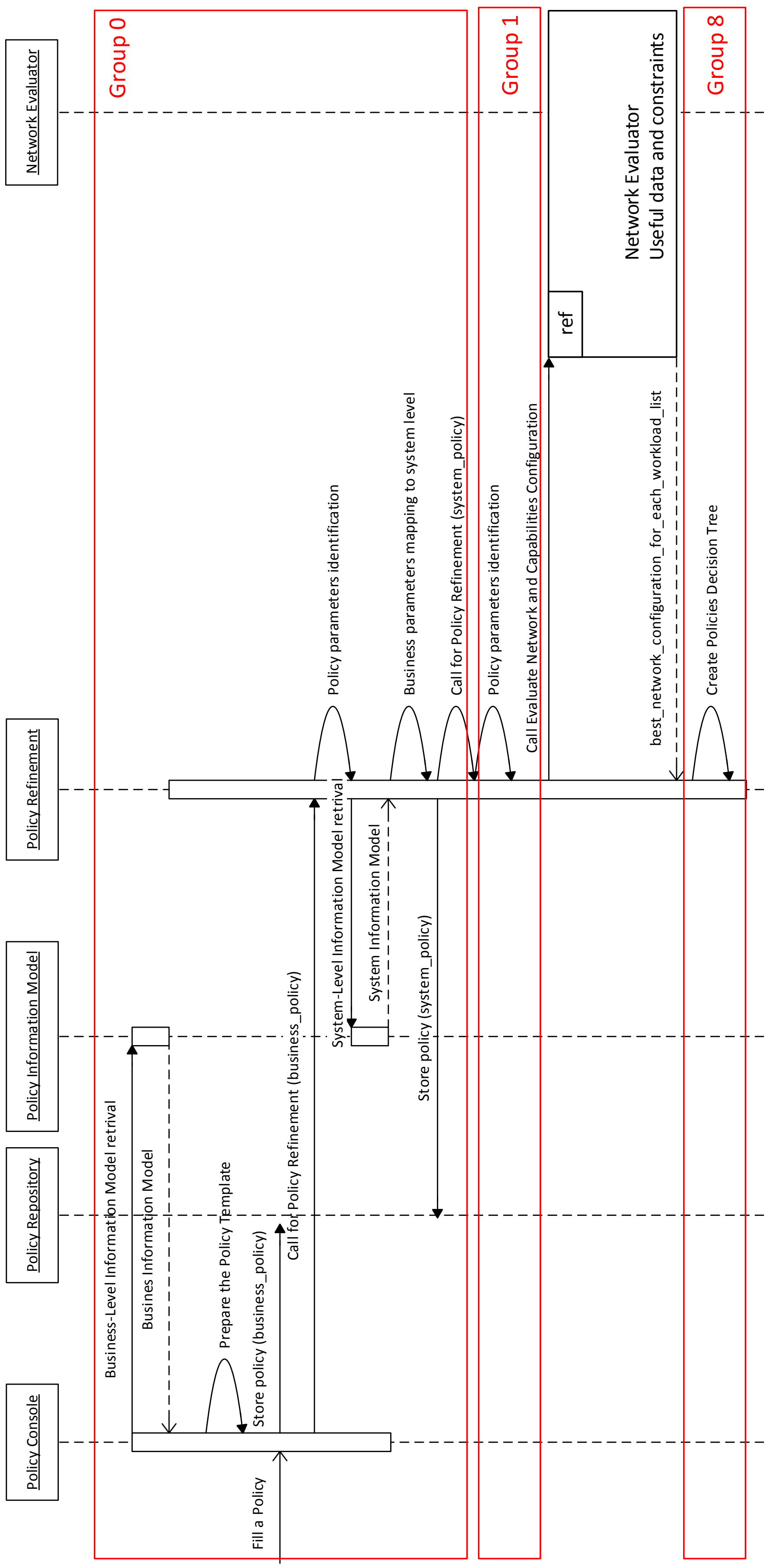

Policy Console
Policy Repository
Policy Information Model
Policy Refinement
Network Evaluator
Group 0
Fill a Policy
Business-Level Information Model retrival
Busines Information Model
Prepare the Policy Template
Store policy (business_policy)
Call for Policy Refinement (business_policy)
Policy parameters identification
System-Level Information Model retrival
System Information Model
Business parameters mapping to system level
Store policy (system_policy)
Call for Policy Refinement (system_policy)
Group 1
Policy parameters identification
Call Evaluate Network and Capabilities Configuration
ref
Network Evaluator
Useful data and constraints
best_network_configuration_for_each_workload_list
Group 8
Create Policies Decision Tree

Figure 24: SOS Sequence Diagram

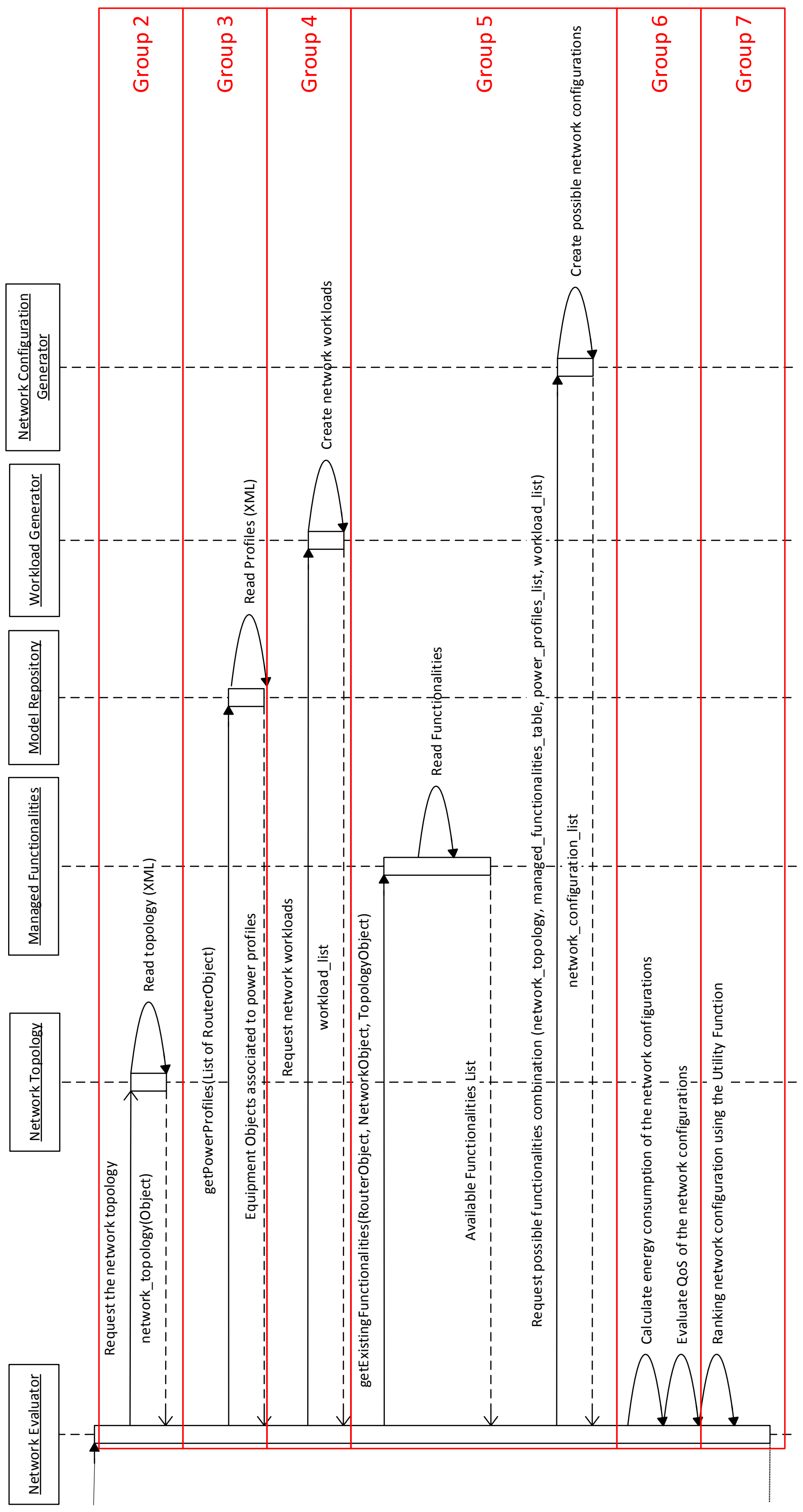

Network Evaluator
Network Topology
Managed Functionalities
Model Repository
Workload Generator
Network Configuration Generator
Group 2
Group 3
Group 4
Group 5
Group 6
Group 7
Request the network topology
network_topology(Object)
Read topology (XML)
getPowerProfiles(List of RouterObject)
Equipment Objects associated to power profiles
Read Profiles (XML)
Request network workloads
workload_list
Create network workloads
getExistingFunctionalities(RouterObject, NetworkObject, TopologyObject)
Available Functionalities List
Read Functionalities
Request possible functionalities combination (network_topology, managed_functionalities_table, power_profiles_list, workload_list)
network_configuration_list
Create possible network configurations
Calculate energy consumption of the network configurations
Evaluate QoS of the network configurations
Ranking network configuration using the Utility Function

Figure 25: SOS Sequence Diagram - Network Evaluator Details

Table 13: Energy Efficiency Capabilities information as input to avoid conflicts between them

	Capability		Scope		Topology		
	Rating	Sleeping	Device	Network	SoHo/LAN	WAN	Fat Tree
ALR	√		√		√	√	√
SC		√	√		√	√	√
SustNMS		√		√	√	√	
ElasticTree		√		√			√

based on information about the network topology (Group 2 in Figure 25), the equipment power profiles (Group 3 in Figure 25), the bandwidth utilization values (Group 4 in Figure 25) and the available capabilities (Group 5 in Figure 25).

The first step is to build a preliminary tree ("Preliminary Tree A", or PTA) in which the leaves correspond to a set of random bandwidth utilization values, as depicted in Figure 26.

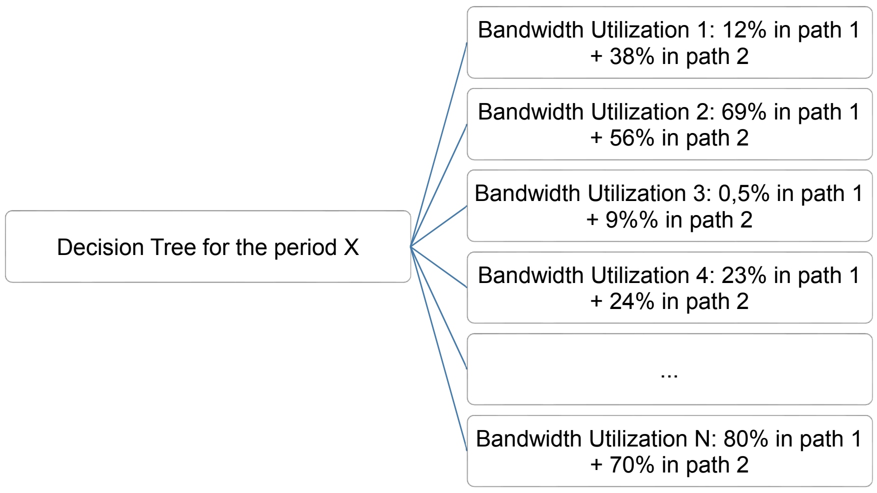

Decision Tree for the period X
Bandwidth Utilization 1: 12% in path 1 + 38% in path 2
Bandwidth Utilization 2: 69% in path 1 + 56% in path 2
Bandwidth Utilization 3: 0,5% in path 1 + 9%% in path 2
Bandwidth Utilization 4: 23% in path 1 + 24% in path 2
...
Bandwidth Utilization N: 80% in path 1 + 70% in path 2

Figure 26: Preliminary Tree A (PTA) Example, supposing two paths available

For each bandwidth utilization, the method tests the allowed capabilities combination to be able to give the best answer for a given bandwidth utilization value in the target topology. To test the allowed combinations, besides the information about which capabilities are available in the network, the method needs to be informed about which capabilities can be combined or not, in order to avoid conflicts among them. Table 13 shows the necessary information of some capabilities examples.

Associated with this table, there are some policies to determine the possible combinations:

- Two network level capabilities performing similar actions cannot be used at the same time;
- Two or more capabilities to put devices to sleep cannot be applied at the same device, at the same time;
- Two capabilities to do link rate cannot be applied at the same device, at the same time.

These policies, combined with the capabilities information in Table 13, are defined in the system as an auxiliary table, the “Allowed Combinations Table”. This table is consulted in the step “Request possible capabilities combination” (Group 5 in Figure 25)). The possible capabilities or combinations, considering the capabilities described are:

- ALR in the interfaces of all nodes;
- Synchronized Coalescing in all nodes;
- Combination of ALR in the interfaces and Synchronized Coalescing in the nodes;
- SustNMS in the network;
- ElasticTree in the network;
- SustNMS in the network, plus ALR in the interfaces that remain powered on;
- ElasticTree in the network, plus ALR in the interfaces that remain powered on.

For each leaf of the preliminary tree PTA, the method puts together the information about the bandwidth utilization for that leaf and the equipment power profile to get the amount of electrical energy expected to be spent in a baseline operation mode, i.e., a mode in which no energy efficiency capability is applied. The same module, which is called **Analytical Solver** (Group 6 in Figure 25), calculates the expected energy consumption and packet loss for each bandwidth utilization applying the allowed capability combination. There is one Analytical Solver for each capability, modeling its behavior regarding energy savings and performance.

In more detail, the Analytical Solver needs the following inputs: (i1) Parameters necessary for the network to be configured (Group 1 in Figure 24); (i2) Topology of the network (Group 2 in Figure 25); (i3) Equipment power profiles (Group 3 in Figure 25); (i4) Randomly generated bandwidth utilization values (Group 4 in Figure 25), organized as a preliminary tree (PTA) with the leaves corresponding to the different bandwidth utilization scenarios (Group 5 in Figure 25), as exemplified in Figure 25. The amount of leaves in the tree must be defined before generating the random bandwidth values. In our method, the number of leaves is defined empirically, by partitioning the interval between zero and the maximum value of available bandwidth in an arbitrary number of sub-intervals. Other strategies can be used, noting that since the method uses interpolation in the later stage, increasing the number of leaves gives more accuracy. That is, the more leaves, the more likely is that the method will be applying the best capability (calculated by a utility function, as described below) for the instantaneously measured bandwidth value.

After deciding how many bandwidth values the tree will have, the method defines the exact sub-intervals by executing a random numbers generator and considering the refined policies. For instance, if the policy says to apply capabilities only if the bandwidth is smaller than 50% of the maximum workload of the network, the method will generate only random numbers smaller than maximum load * 50%. The last input, (i5) is the available energy efficiency capabilities along with the possible combinations information (Group 5 in Figure 25).

The Analytical Solver processing part will perform the steps described in the Algorithm 1 (Group 6 in Figure 25). The output is the Preliminary Tree B (PTB) composed by leaves with bandwidth utilization values and a set of capabilities along with the expected savings and packet losses, as exemplified in Figure 27.

Algorithm 1: Building the Preliminary Tree B (PTB)

```
Require: $Inputs\ i1, i2, i3, i4, i5$
  for all nodes in the network do
    $energyBaseline \leftarrow energyBaseline + nodeWatts$
  end for
  for all bandwidth utilization value (leaf) in the preliminary tree PTA do
    for all allowed capabilities combination do
      $energyAfterSavings \leftarrow CapabilityAnalyticalSolver.energy$
      $energySavings \leftarrow energyBaseline/energyAfterSavings$
      $packetLosses \leftarrow CapabilityAnalyticalSolver.packet\_losses$
    end for
  end for
```

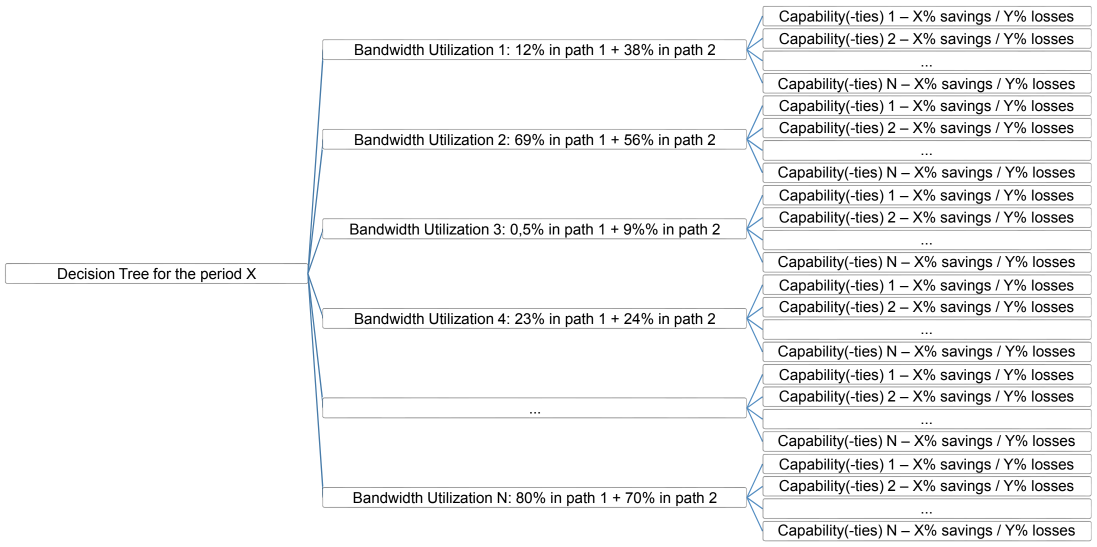

Decision Tree for the period X
Bandwidth Utilization 1: 12% in path 1 + 38% in path 2
Bandwidth Utilization 2: 69% in path 1 + 56% in path 2
Bandwidth Utilization 3: 0,5% in path 1 + 9%% in path 2
Bandwidth Utilization 4: 23% in path 1 + 24% in path 2
...
Bandwidth Utilization N: 80% in path 1 + 70% in path 2
Capability(-ties) 1 – X% savings / Y% losses
Capability(-ties) 2 – X% savings / Y% losses
...
Capability(-ties) N – X% savings / Y% losses

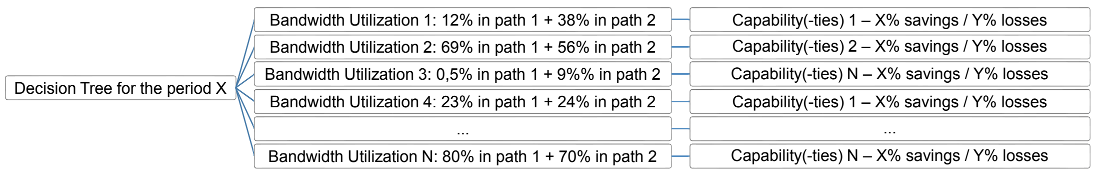
Figure 27: Preliminary Tree B (PTB) Example with expected savings and losses for each bandwidth utilization and associated capability(-ties)

Based on the estimated values of savings and packet losses for the topology and the equipment power profile calculated for each leaf of PTB by the Analytical Solver, the Utility Function (UF) determines the grade for each energy efficiency capability(-ties) combination using the formula (Group 7 in Figure 25). Considering the Utility Function (UF) results, the method will build the "Final Decision Tree" (FDT), in which each leaf is composed by a bandwidth utilization value and an associated capability(-ties), the one which achieved the highest grade in the UF, as illustrated in Figure 28. These steps are described in Algorithm 2. For each period of the day, one different Final Decision Tree is expected.

Algorithm 2: Building the Final Decision Tree (FDT)

```
Require: $PTB$
  for all bandwidth utilization value (leaf) in the preliminary tree PTB do
    for all allowed capabilities combination c do
      for all nodes k do
        UF[c]=$pl * \frac{1}{\frac{\sum_{k=0}^{n} EnergyAfterSavings_{Router\ k}}{\sum_{k=0}^{n} EnergyBaseline_{Router\ k}}}$
      end for
    end for
    SelectedCapability = max UF[c]
  end for
```

The Final Decision Trees are received by the last method module, which will train the network to react differently considering the bandwidth utilization (Group 8 Figure 24). The required inputs for this step are: (i6) Final Decision Tree FDT composed by leaves with bandwidth utilization values and the best capability(-ties) for the bandwidth utilization, as exemplified in Figure 28; (i7) the network itself equipped with a Network Management System to provide metrics such as bandwidth utilization, energy, and packet loss. After, the method builds the Final Decision Tree using an Interpolation tool (FDTI).


Decision Tree for the period X
Bandwidth Utilization 1: 12% in path 1 + 38% in path 2
Capability(-ties) 1 – X% savings / Y% losses
Bandwidth Utilization 2: 69% in path 1 + 56% in path 2
Capability(-ties) 2 – X% savings / Y% losses
Bandwidth Utilization 3: 0,5% in path 1 + 9%% in path 2
Capability(-ties) N – X% savings / Y% losses
Bandwidth Utilization 4: 23% in path 1 + 24% in path 2
Capability(-ties) 1 – X% savings / Y% losses
...
...
Bandwidth Utilization N: 80% in path 1 + 70% in path 2
Capability(-ties) N – X% savings / Y% losses
Figure 28: Final Decision Tree (FDT) Example with the selected capability(-ties)

This tree has three levels: the root, the leaves with bandwidth utilization values and, associated to each bandwidth utilization value, one leaf with the best capability (-ties) for the given bandwidth scenario. The method uses the class DecisionTreeClassifier from the Scikit tool (PEDREGOSA et al., 2011) to train the

FDT so that it will be able to interpolate bandwidth values, that is, predict the target values when no existing bandwidth leaf matches exactly the bandwidth measured during operation. This constitutes the Final Decision Tree with Interpolation (FDTI).

The time to build the FDTI relates to the DecisionTreeClassifier class in the Scikit Tool. Learning an optimal decision tree is an NP-Complete problem (SCIKIT-LEARN, n.d.). According to the tool documentation, the algorithms there use heuristics “such as the greedy algorithm where locally optimal decisions are made at each node”. The Scikit implementation is said to have a total cost of $O(n_{features}n_{samples}log(n_{samples}))$ (SCIKIT-LEARN, n.d.). In our case, $n_{features}$ relate to the built FDT, and $n_{samples}$ to the PTA, with the randomly generated bandwidth values.

After building the FDTI, the method can handle all bandwidth values, even those that were not specifically calculated during the method. Algorithm 3 describes the FDTI usage while operating the network. Figure 29 represents the operation of the Final Decision Tree with Interpolation.

To apply the capabilities in the network means issuing a set of Netconf or OpenFlow instructions that will configure the management parameters, such as link rate or state (sleep/powered on). For instance, to apply ALR in the nodes means changing the link rates between pairs of nodes using Ethernet data rates (e.g. 10 Mbps) after a handshake between each pair of nodes. As detailed later in Chapter 5, in this work we used GreenSDN (RODRIGUES et al., 2015) to enforce the decisions and

Algorithm 3: SOS Operation

```
while TRUE do
    $Wl$ ←Network bandwidth utilization
    if $Wl_{n+1}$ differs from $Wl_n$ then
        SelectedCapability(-ties) ← FDTI ($Wl$)
        Enforce SelectedCapability(-ties) (See details below).
        Optional step: Measure and show savings.
        Optional step: Measure and show losses.
    end if
end while
```

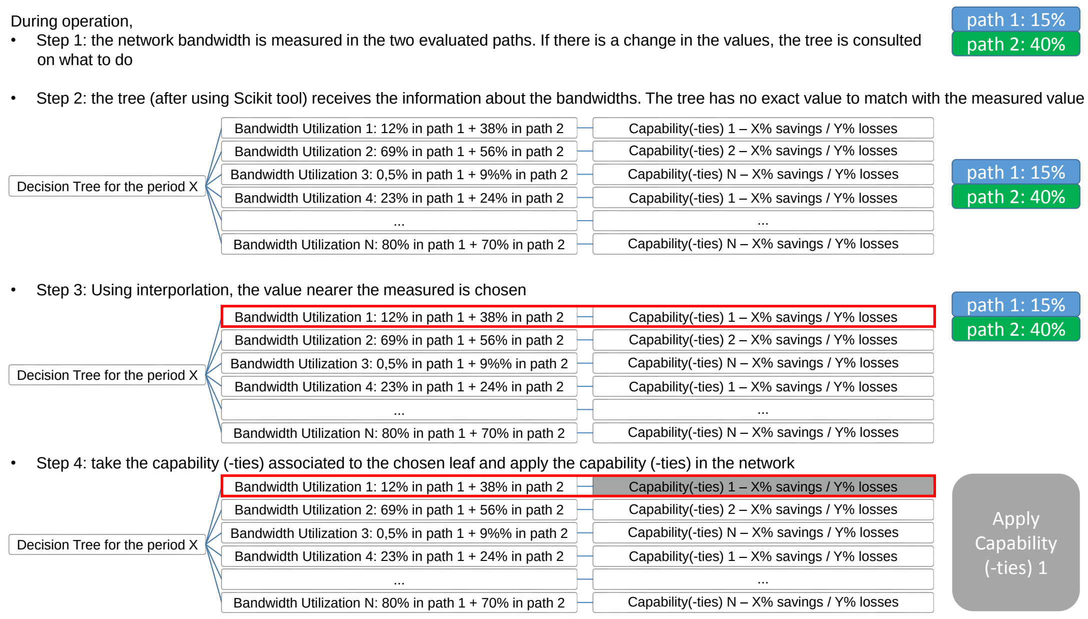

During operation,
• Step 1: the network bandwidth is measured in the two evaluated paths. If there is a change in the values, the tree is consulted on what to do
path 1: 15%
path 2: 40%
• Step 2: the tree (after using Scikit tool) receives the information about the bandwidths. The tree has no exact value to match with the measured value
Decision Tree for the period X
Bandwidth Utilization 1: 12% in path 1 + 38% in path 2
Bandwidth Utilization 2: 69% in path 1 + 56% in path 2
Bandwidth Utilization 3: 0,5% in path 1 + 9%% in path 2
Bandwidth Utilization 4: 23% in path 1 + 24% in path 2
...
Bandwidth Utilization N: 80% in path 1 + 70% in path 2
Capability(-ties) 1 – X% savings / Y% losses
Capability(-ties) 2 – X% savings / Y% losses
Capability(-ties) N – X% savings / Y% losses
Capability(-ties) 1 – X% savings / Y% losses
...
Capability(-ties) N – X% savings / Y% losses
path 1: 15%
path 2: 40%
• Step 3: Using interpolation, the value nearer the measured is chosen
path 1: 15%
path 2: 40%
• Step 4: take the capability (-ties) associated to the chosen leaf and apply the capability (-ties) in the network
Apply Capability (-ties) 1

Figure 29: Example of the Final Decision Tree with Interpolation (FDTI) operation

validate the proposed method. GreenSDN is an SDN-based test environment which emulates different energy savings protocols.

To apply SC followed by ALR starts by configuring the nodes to perform traffic bursts and sleep while buffering data. This configuration involves SC parameters as “tOn” (time the equipment will be fully operational) and “DutyCyle” (percentage of time the equipment must remain sleeping). ALR will then be applied during the operational (not sleeping) periods of SC, reducing the link rate.

To apply SustNMS in the network means considering all existing paths in the topology, performing green traffic engineering to consolidate traffic in some of the paths and then put the unused devices to sleep.

The decision trees are constructed offline, that is, without affecting the network operation. Only after the trees are built, they are deployed in the network. The decision trees should be recalculated, that is, the method should be run again after (i) a specified period of time; or (ii) after a trigger. The possible triggers are, among others: when a new node is installed; when a new connection is added; when a new protocol is enabled on existing nodes or controller; when the network administrator gets an indication that the traffic pattern has changed, such as when a new enterprise application is introduced in a private cloud datacenter; or when a new customer using a significant number of virtual machines in a public cloud datacenter is added.

4.6 Addressing Dynamicity

The method should also be able to deal with dynamicity aspects, as described in Chapter 3. As described in Section 4.1, the Information Models support time dynamicity by considering a TimePeriodCondition. And a different policy for each period must be defined, as exemplified in Section 4.3. For each policy (and for each period of the day), one different Final Decision Tree is expected. For instance, the network administrator must define a policy for the day, and another for the night; or one for the business hours, one for lunch time, and another for the night.

Another type of dynamicity that can be discussed is during operation, in terms of reacting to load variation. The method reaction time is logarithmic in time, but the underlying infrastructure takes some more time to enforce the decision. That means it would not react instantaneously, depending on the amount of information to be processed and on the controller in use.

In order to avoid unnecessary and costly changes, a threshold can be used. For instance, if the load variation is smaller than X%, SOS will keep the current decision. If the variation gets greater than this threshold, then SOS will select and enforce other capabilities.

This strategy comes at a cost of not running the optimal solution all the time, but would ensure a consistent operation. The X value can be defined based on the network traffic characteristics.

4.7 Scalability Analysis

There are two aspects that must be discussed when it comes to evaluate SOS regarding scalability: the time to choose another capability(-ties) (another leaf in a Decision Tree) and enforce the decision in the infrastructure, represented in Algorithm 3; and the time to recalculate the decision trees, which happens, as described in Section 4.5, after a specified period of time or a trigger.

Considering the **time to choose another capability(-ties)**, the complexity of using the tree built using the Scikit Tool is $O(log(l))$ (SCIKIT-LEARN, n.d.), in which l in the number of elements in the tree (in the SOS case, the number of randomly generated bandwidth values). Although finding the right leaf for a certain bandwidth utilization value is fast (logarithmic in complexity), the time to apply all the necessary configuration changes in the network in the worst case is done linearly with the number of nodes. Although the complexity itself is still linear, the configuration tasks could be executed in multiple parallel threads on the controller, making the execution faster.

Regarding the **time to recalculate the decision trees**, as detailed in Section 4.5, the decision trees of SOS are constructed offline, that is, without affecting the network operation. Only after the trees are built, they are deployed in the network to provide the best combination of capabilities given a network state (flows, usage) in a conflict-free operation. So, even in scenarios with hundreds or thousands of nodes, the operation would not be affected while the decision trees are under construction.

The recalculation depends on the number of bandwidth values randomly generated (bv), nodes (n), and capabilities allowed combinations (c). Table 14 details the complexity of each building block, using the groups in Figures 24 and 25 as reference.

In sum, recalculating the Decision Trees is exponential in complexity. Given that the capabilities are mutually exclusive within one node, the best solution is to keep using a greedy, per-node testing approach to check possible savings and losses

Table 14: Scalability Analysis

Step	**Group in Figures 24,25**	**Complexity**
Get information	Groups 2, 3, 4 and 5	$O(1)$
Build PTA	Conclusion of Group 5	$O(n)$
Build PTB (Algorithm 1)	Group 6	$O(c^n)$
Build FDT (Algorithm 2)	Group 7	$O(bv * c)$
Build FDTI	Group 8	depends on the tool, as Scikit

according to the node-specific power profile for that capability(-ties). However, one must consider that the capabilities combination strategy, represented in Table 13, reduces the necessary steps by not executing the calculations among incompatible capabilities.

Such greedy approach does not affect the applicability of SOS, as the calculation of the Decision Trees are not performed in real-time and parallel threads could be spawned for the execution. For instance, the Decision Trees (DT) are independent from each other, and there is one DT for each policy time condition (different periods of the day). Each policy could be refined and its correspondent DT built without waiting for the other Decision Trees. Another option is to parallelize the Analytical Solvers calculations. For instance, the SC Analytical Solver can calculate the expected savings and losses in parallel to the ALR Analytical Solver, or the SustNMS Analytical Solver.

Some strategies to avoid recalculating the whole Decision Tree are also possible. For instance, if one capability is no longer available, it is possible to recalculate only the leaves in which this capability was the previous selection.

4.8 Chapter Final Remarks

In this Chapter, the SOS method was presented in detail, including its sequence diagram and the supporting parts, such as the sustainability-oriented information models and the proposed Utility Function. All steps were exemplified and discussed.

For the SOS prototype, the tables for the Table Lookup refinement need to be implemented based on the Information Models, the Decision Trees need to be built and demand an interpolation tool to decide what to do when the bandwidth usage is different from the ones in the trees. A testbed able to emulate energy efficiency capabilities must be used to support the decisions enforcement. The next Chapter presents the method implementation details.

5 SOS IMPLEMENTATION ENVIRONMENT

In this Chapter, the method implementation is detailed, including the architecture, the XMLs used for the refinement and algorithms. The test environment in which the experiments were run is also presented.

5.1 Method Implementation Architecture

As a proof of concept, this work describes a prototype implementation of the SOS (Sustainability Oriented System) method. The method was implemented for an SDN (Software Defined Network) environment, as it is shown in Figure 30. As it can be seen in this figure, the method modules are grouped in two parts: the application and the controller.

The policies translation was implemented in module 1. It uses the sustainability-oriented information models and outputs the necessary information for the orchestration part (refined down from the high-level policies). The energy efficiency capabilities are combined, and the Utility Function is used to choose the best option in module 2. This module outputs the Final Decision Tree (FDT), input to module 3, which corresponds to the Scikit tool to build the Final Decision Tree with Interpolation (FDTI), a decision tree able to interpolate the measured bandwidth utilization and select which action to take. Module 4 brings this final decision tree with interpolation to the controller, making it act as a policy decision point (module 5). This module keeps listening to the network metrics and deciding what to do based on the tree information. The decisions are enforced in the switches.

In parallel, a set of services modules supports the operation, providing information

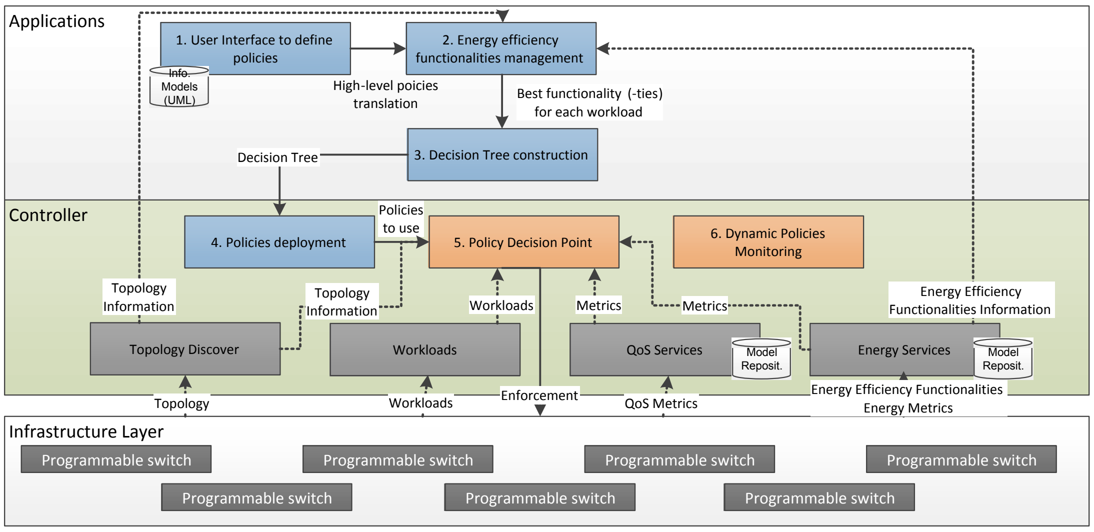


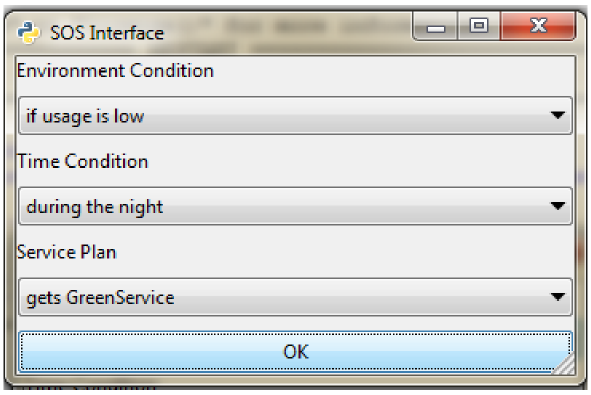
Figure 30: Method Implementation Architecture

about the network topology, the bandwidth utilization values, performance measured as packets lost, and energy consumption of each part of the network. This last service is able to take the power profile equations from the Model Repository and calculate the expected Watts spent by each equipment given a certain workload.

5.2 Method Implementation Details

The method was implemented in Python 2.7, using XML (eXtensible Markup Language) to represent the information in the necessary steps. The first steps comprise the Table Lookup translation, which translates high-level policies to the network level using tables to relate objects. The relationship among the objects of the policies is defined in the Sustainability-Oriented Information Models. The user defines the high-level policies in the SOS Policies Interface, depicted in Figure 5.2.

The TableLookup.py module translates the high-level policies defined by the user in the Interface using the files depicted in Figure 32, Figure 33, and Figure 34. The output is recorded in a text file, exemplified in Figure 35, used as an input to the next module, which needs information from the high-level. This concludes the steps performed in the module 1 shown in Figure 30.


SOS Interface

Environment Condition

if usage is low

Time Condition

during the night

Service Plan

gets GreenService

Figure 31: SOS Policies Interface

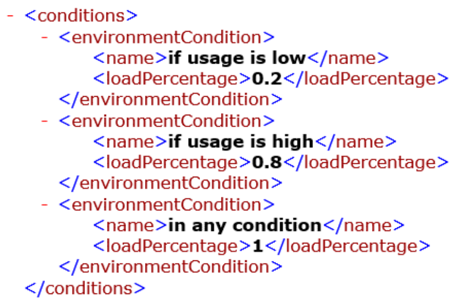
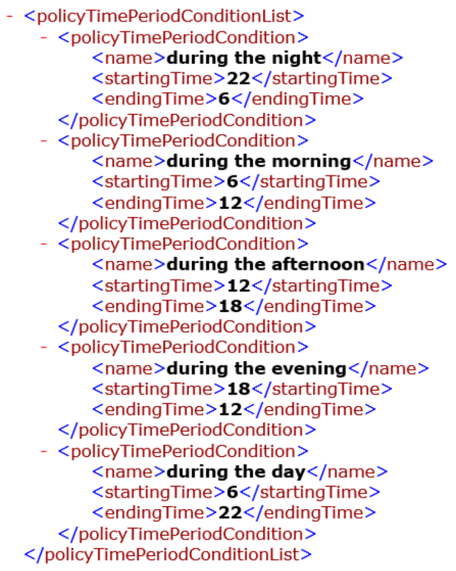

```
- <conditions>
  - <environmentCondition>
      <name>if usage is low</name>
      <loadPercentage>0.2</loadPercentage>
    </environmentCondition>
  - <environmentCondition>
      <name>if usage is high</name>
      <loadPercentage>0.8</loadPercentage>
    </environmentCondition>
  - <environmentCondition>
      <name>in any condition</name>
      <loadPercentage>1</loadPercentage>
    </environmentCondition>
  </conditions>
```

Figure 32: Environment Condition XML

The next steps are related to information gathering so that the best capability (-ties) can be selected. The network topology depicted is represented in an XML. The devices power profiles, necessary to calculate the amount of Watts each equipment dissipates are represented in the file PowerProfileTable.xml, illustrated in Figure 36.

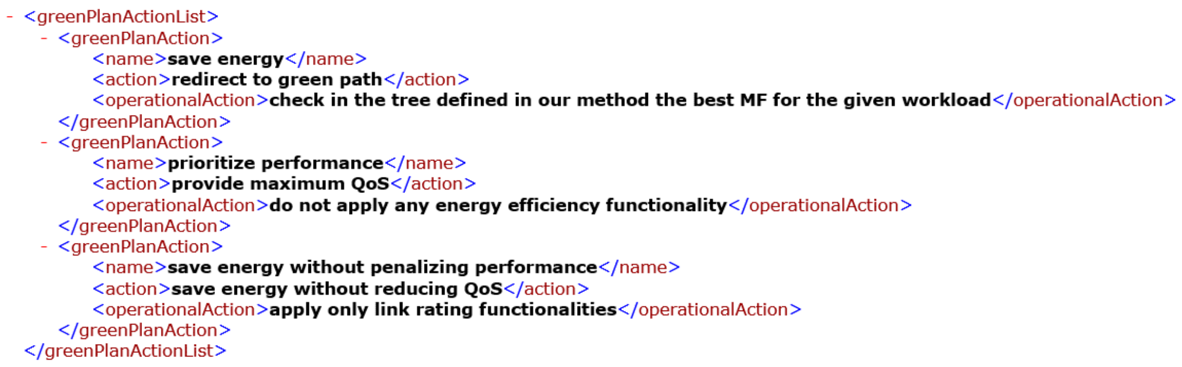
The next step is to generate the random workloads used to build the first Preliminary Tree A (PTA). Algorithm 4 describes the bandwidth utilization values generation. The output is another XML file, relating the network topology with the generated workloads, as illustrated in Figure 37, corresponding to the Preliminary Tree A (PTA).

The only information still missing to module 2 is the possible combinations for the energy efficiency capabilities. The combinations are implemented as an extra table: "Allowed Combinations Table". Since there are two policies being considered, in this step the method has two XMLs: *capabilitiesCombinationAllGreen.xml*, *capabilitiesCombinationOnlyRating.xml*. The first depicts all the combinations



```
- <policyTimePeriodConditionList>
  - <policyTimePeriodCondition>
      <name>during the night</name>
      <startingTime>22</startingTime>
      <endingTime>6</endingTime>
    </policyTimePeriodCondition>
  - <policyTimePeriodCondition>
      <name>during the morning</name>
      <startingTime>6</startingTime>
      <endingTime>12</endingTime>
    </policyTimePeriodCondition>
  - <policyTimePeriodCondition>
      <name>during the afternoon</name>
      <startingTime>12</startingTime>
      <endingTime>18</endingTime>
    </policyTimePeriodCondition>
  - <policyTimePeriodCondition>
      <name>during the evening</name>
      <startingTime>18</startingTime>
      <endingTime>12</endingTime>
    </policyTimePeriodCondition>
  - <policyTimePeriodCondition>
      <name>during the day</name>
      <startingTime>6</startingTime>
      <endingTime>22</endingTime>
    </policyTimePeriodCondition>
  </policyTimePeriodConditionList>
```

Figure 33: Policy TimeCondition XML



```
- <greenPlanActionList>
  - <greenPlanAction>
      <name>save energy</name>
      <action>redirect to green path</action>
      <operationalAction>check in the tree defined in our method the best MF for the given workload</operationalAction>
    </greenPlanAction>
  - <greenPlanAction>
      <name>prioritize performance</name>
      <action>provide maximum QoS</action>
      <operationalAction>do not apply any energy efficiency functionality</operationalAction>
    </greenPlanAction>
  - <greenPlanAction>
      <name>save energy without penalizing performance</name>
      <action>save energy without reducing QoS</action>
      <operationalAction>apply only link rating functionalities</operationalAction>
    </greenPlanAction>
  </greenPlanActionList>
```

Figure 34: Green Plan Action XML

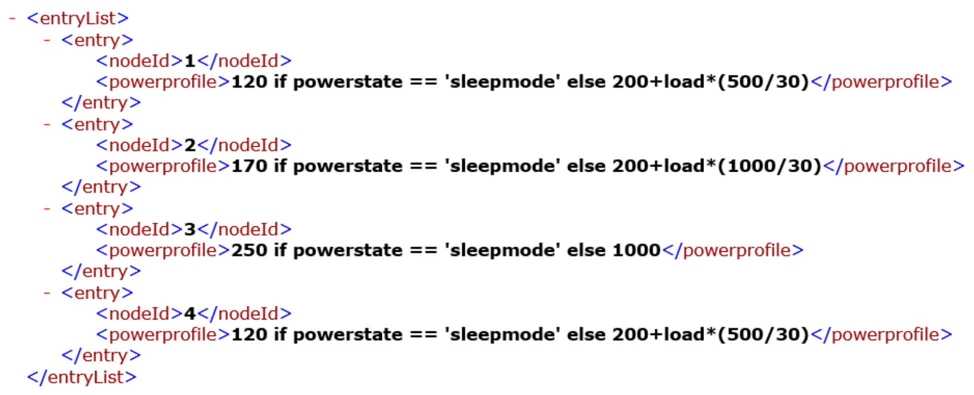
considering all capabilities allowed, as depicted in Figure 38. The other shows what can be done considering that only link rating is allowed (in case of a policy for the day, for instance, with more traffic to handle), as depicted in Figure 39.

With the entire information ready to use, the module will use the Analytical Solvers (described in Chapter 4) to estimate savings and losses for each pair workload - capability (-ties). Considering the existing capabilities available, three were selected

```
['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating capabilities']

['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best capability(-ties) for the given workload']
```

Figure 35: Module 1 output: the refined policies



```
- <entryList>
  - <entry>
      <nodeId>1</nodeId>
      <powerprofile>120 if powerstate == 'sleepmode' else 200+load*(500/30)</powerprofile>
    </entry>
  - <entry>
      <nodeId>2</nodeId>
      <powerprofile>170 if powerstate == 'sleepmode' else 200+load*(1000/30)</powerprofile>
    </entry>
  - <entry>
      <nodeId>3</nodeId>
      <powerprofile>250 if powerstate == 'sleepmode' else 1000</powerprofile>
    </entry>
  - <entry>
      <nodeId>4</nodeId>
      <powerprofile>120 if powerstate == 'sleepmode' else 200+load*(500/30)</powerprofile>
    </entry>
  </entryList>
```

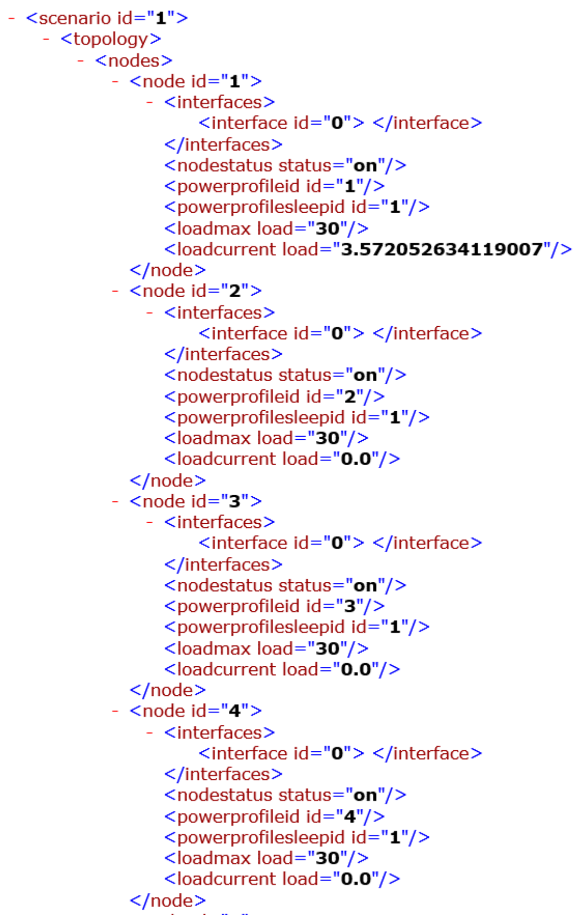
Figure 36: Power Profile Table XML

to be implemented, each one as a representative of a different scope: ALR (component scope), SC (device scope), and SustNMS (network scope).

ALR puts interfaces to sleep considering the Ethernet rates: 1Gbps, 100Mbps, and 10Mbps. According to (RICCA et al., 2013), ALR can save up to 21% on the studied equipment. Ricciardi et al. (RICCIARDI et al., 2011) studied the capability and discovered that the energy spent after reducing the link rate depends on the interface native 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 high, since it spends much less time to wake up the interfaces (microseconds order of magnitude, while waking up a node from a sleep mode in a

Algorithm 4: Random bandwidth utilization values generation

```
$R1 \leftarrow loadmax\ of\ the\ first\ Router$
{Distributes the load according to the first router}
for all Paths in the Topology do
    for all Nodes do
        $random \leftarrow a\ random\ number\ between\ 0\ and\ 1$
        $loadcurrent \leftarrow loadcurrent + R1 * random$
    end for
end for
{Adjusts according to the maxload of each node in the path}
$aux \leftarrow 1$
for all Nodes do
    $aux \leftarrow max(loadcurrent\ or\ aux)/loadmax$
    $loadcurrent \leftarrow loadcurrent/aux$
end for
```



```
- <scenario id="1">
  - <topology>
    - <nodes>
      - <node id="1">
        - <interfaces>
            <interface id="0"> </interface>
          </interfaces>
          <nodestatus status="on"/>
          <powerprofileid id="1"/>
          <powerprofilesleepid id="1"/>
          <loadmax load="30"/>
          <loadcurrent load="3.572052634119007"/>
        </node>
      - <node id="2">
        - <interfaces>
            <interface id="0"> </interface>
          </interfaces>
          <nodestatus status="on"/>
          <powerprofileid id="2"/>
          <powerprofilesleepid id="1"/>
          <loadmax load="30"/>
          <loadcurrent load="0.0"/>
        </node>
      - <node id="3">
        - <interfaces>
            <interface id="0"> </interface>
          </interfaces>
          <nodestatus status="on"/>
          <powerprofileid id="3"/>
          <powerprofilesleepid id="1"/>
          <loadmax load="30"/>
          <loadcurrent load="0.0"/>
        </node>
      - <node id="4">
        - <interfaces>
            <interface id="0"> </interface>
          </interfaces>
          <nodestatus status="on"/>
          <powerprofileid id="4"/>
          <powerprofilesleepid id="1"/>
          <loadmax load="30"/>
          <loadcurrent load="0.0"/>
        </node>
```

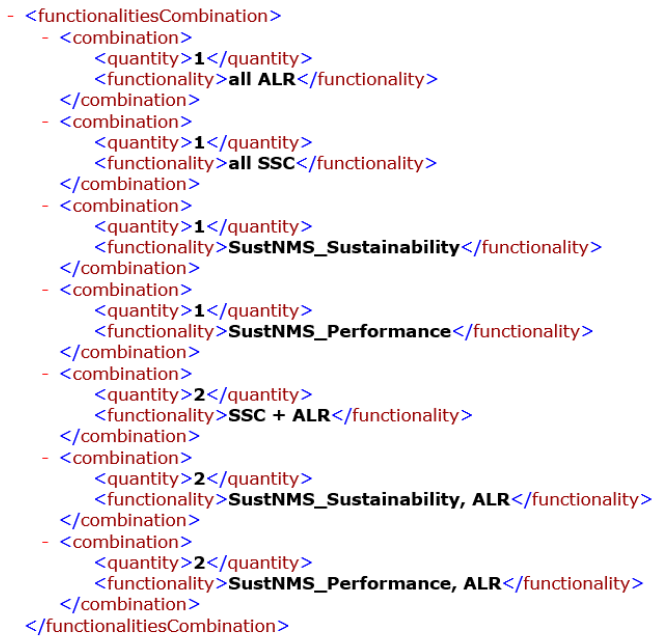
Figure 37: Preliminary Tree A XML

network scope capability can take seconds to minutes, depending on the equipment). It is also interesting to use in conjunction with other capability, in those nodes that remain powered on (for instance, SustNMS with ALR). The ALR Analytical Solver is described in Algorithm 5. The SC Analytical Solver is described in Algorithm 6.

The SustNMS Analytical Solver followed the algorithm in (JANUARIO et al.,

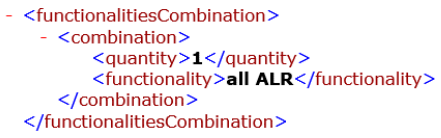
Algorithm 5: ALR Analytical Solver

```
Require: $Inputs\ packets\ per\ second,\ power\ profiles$
  $minRate \leftarrow 10Mbps$
  for all Devices do
    $maxRate \leftarrow maximum\ device\ capacity$
    if Packets per second < minRate then
      $result \leftarrow power\ consumption\ with\ ALR\ savings\ based\ on\ power\ profile$
    else
      $result \leftarrow power\ consumption\ device\ normal\ rate$
    end if
  end for
```



```
- <functionalitiesCombination>
  - <combination>
      <quantity>1</quantity>
      <functionality>all ALR</functionality>
    </combination>
  - <combination>
      <quantity>1</quantity>
      <functionality>all SSC</functionality>
    </combination>
  - <combination>
      <quantity>1</quantity>
      <functionality>SustNMS_Sustainability</functionality>
    </combination>
  - <combination>
      <quantity>1</quantity>
      <functionality>SustNMS_Performance</functionality>
    </combination>
  - <combination>
      <quantity>2</quantity>
      <functionality>SSC + ALR</functionality>
    </combination>
  - <combination>
      <quantity>2</quantity>
      <functionality>SustNMS_Sustainability, ALR</functionality>
    </combination>
  - <combination>
      <quantity>2</quantity>
      <functionality>SustNMS_Performance, ALR</functionality>
    </combination>
  </functionalitiesCombination>
```

Figure 38: XML representing the Capabilities Combination for the Night



```
- <functionalitiesCombination>
  - <combination>
      <quantity>1</quantity>
      <functionality>all ALR</functionality>
    </combination>
  </functionalitiesCombination>
```

Figure 39: XML representing the Capabilities Combination for the Day

2013). The combination ALR/SC has an specific analytical solver, described in Algorithm 7. The combination SustNMS/ALR was implemented in another way: the SustNMS Analytical Solver outputs the list of necessary routers to remain on. Over this list, ALR is applied if possible, when the workload is smaller than 10 Mbps.

The module now has all the information necessary to apply the Utility Function and choose the best capability (-ties). This concludes module 2 steps. Module 3 builds the decision tree (one or more than one, according to the number of policies defined in the user interface). The Module uses the information from Module 2 and calls the Scikit Tool to build the Final Decision Tree with Interpolation (FDTI), as depicted in Figure 29, which is then deployed in the controller. From this point on, the Algorithm 2 is operating on the network.

Algorithm 6: SC Analytical Solver

```
Require: $Inputs\ packets\ per\ second,\ power\ profiles$
  $tOn \leftarrow duration\ of\ the\ period\ with\ the\ device\ active\ in\ milliseconds$
  $DutyCycle \leftarrow percentage\ of\ cycle\ time\ the\ device\ must\ remain\ active$
  {tOff = (tOn / DutyCycle) - tOn}
  $threshold \leftarrow number\ of\ packets\ to\ deactivate\ SC\ (adaptive\ behavior)$
  for all Devices do
    if Packets per second < threshold then
      $resultOn \leftarrow power\ consumption\ device\ on$
      $resultOff \leftarrow power\ consumption\ device\ off$
      $result \leftarrow resultOn * tOn + resultOff * tOff$
      $buffer \leftarrow size\ of\ the\ buffer\ in\ number\ of\ packets$
      if packets per second > buffer then
        $Calculate\ packet\ losses$
      else
        $No\ losses$
      end if
    else
      $result \leftarrow power\ consumption\ device\ on$
      {Does not sleep}
    end if
  end for
```

5.3 Validation Environment

To test and validate the proposed method, this work used the GreenSDN testbed proposed in our research by Rodrigues et al. (RODRIGUES et al., 2015) to emulate the environment. This thesis' author contributed with the architecture definition, graphical interfaces, power profiles definition and implementation.

Simulation is a popular experimentation method, but it lacks realism and involves lots of assumptions to reproduce the environment. Popular simulation tools are ns-2, ns-3 or Opnet (LANTZ; HELLER; MCKEOWN, 2010) (PEDIADITAKIS; ROTSOS; MOORE, 2014). Emulation, on the other hand, enables the reproduction of real world scenarios, being considered more adequate than simulation to reproduce networks in a more realistic way (SIATERLIS; GARCIA; GENGE, 2013). Among the emulation tools, Mininet is the most popular among the recent efforts to fit the network tools in low hardware requirements (PEDIADITAKIS; ROTSOS; MOORE, 2014). It is

Algorithm 7: SC in conjunction with ALR Analytical Solver

```
Require: $Inputs\ packets\ per\ second,\ power\ profiles$
  $tOn \leftarrow duration\ of\ the\ period\ with\ the\ device\ active\ in\ milliseconds$
  $DutyCycle \leftarrow percentage\ of\ cycle\ time\ the\ device\ must\ remain\ active$
  {tOff = (tOn / DutyCycle) - tOn}
  $threshold \leftarrow number\ of\ packets\ to\ deactivate\ SC\ (adaptive\ behavior)$
  for all Devices do
    if Packets per second < threshold then
      $resultOn \leftarrow power\ consumption\ device\ on$
      $resultOff \leftarrow power\ consumption\ device\ off$
      $result \leftarrow resultOn * tOn + resultOff * tOff$
      $buffer \leftarrow size\ of\ the\ buffer\ in\ number\ of\ packets$
      if packets per second > buffer then
        $Calculate\ packet\ losses$
      else
        $No\ losses$
      end if

      $minRate \leftarrow 10Mbps$
      $maxRate \leftarrow maximum\ device\ capacity$
      if Packets per second < minRate then
        $result \leftarrow power\ consumption\ with\ ALR\ savings\ during\ SC\ tOn$
      else
        $result \leftarrow power\ consumption\ device\ normal\ rate$
      end if
    else
      $result \leftarrow power\ consumption\ device\ on$
      {Does not sleep}
    end if
  end for
```

readily available and considering experiments replication, is one of its main strengths. Mininet has a CLI interface easy to deploy with a good walk-through documentation and provides good software support to deploy OpenFlow.

The network is emulated on the Mininet Virtual Machine (MVM) using the POX controller to manage its actions and the Iperf tool to generate the traffic on the network. The OpenFlow 1.0 protocol does the communication between the data-plane and the controller. This OpenFlow version was used because it was the only one available with the POX controller and the more stable at the time this work was developed. The POX controller is implemented using Python 2.7. The emulation environment is depicted in Figure 40. SOS is located in the Green Application Control module. As described in

Figure 30, the method deploys the Final Decision Tree with Interpolation (FDTI) inside the controller, which has the support services to provide the necessary information.

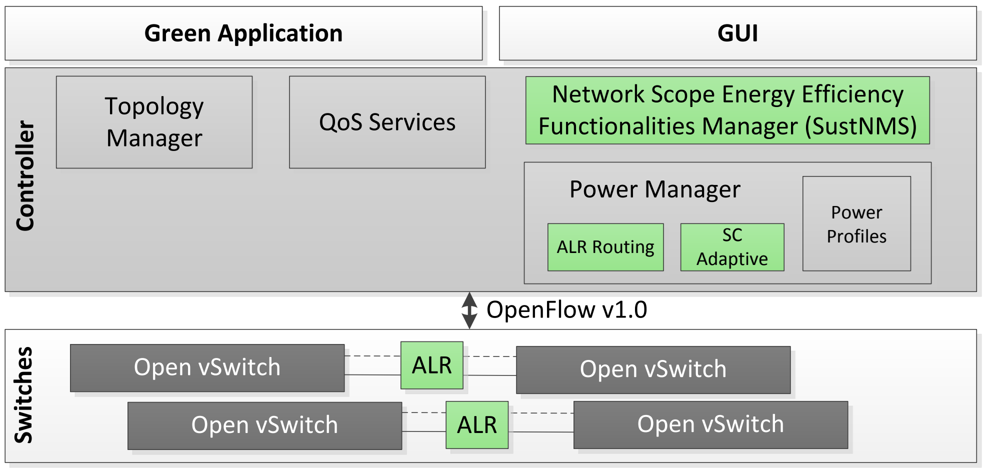


Figure 40: GreenSDN emulation environment (RODRIGUES et al., 2015)

The algorithm used by the GreenSDN controller to perform the emulation is depicted in Algorithm 8. It describes the Controller functionality responsible for applying the green capabilities selected by the Green Application Control (in this work, by the SOS method). The controller takes the underlying network topology and the flows from an XML file. It also creates the services that will support the operation, such as the QoS and the Power Management monitors. The controller then starts monitoring the network, receiving from the Application Control the capabilities that must be enforced. The necessary metrics are also monitored (energy consumption and packet loss), and the GUI renders what is happening in the network.

In the experiments, a load proportional Power Profile (PP) was used for all nodes. There is also a Power Profile for sleeping periods (PP$_{sleeping}$) and adapted Power Profiles for ALR (PP$_{ALR}$) and SC (PP$_{SC}$). The following equations represent the different behaviors expected:

$$PP = Power_{baseline} + (\frac{Power_{variable}}{Load_{\max}})workload \tag{5.1}$$

$$PP_{sleeping} = Power_{standby} \tag{5.2}$$

$$PP_{ALR} = PP - 15 \tag{5.3}$$

$$PP_{SC} = PP_{sleeping}tOff + PPtOn \tag{5.4}$$

Equation (1) represents a load proportional power profile in Watts composed by a baseline consumption (fixed part) plus a variable part, composed by the power part that

Algorithm 8: GreenSDN controller implementation (RODRIGUES et al., 2015)

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

varies according to the workload and the maximum load the equipment can handle. Equation (2) represents the fixed power dissipated by the equipment while sleeping. Equation (3) represents the consumption with ALR activated in the equipment links. As described in Section 5.2, ALR can save up to 21% of the total energy consumed by the equipment. This work adopted a more conservative rate, 15%, as studied by (RICCIARDI et al., 2011). The last Equation, (4), represents the consumption with SC activated. To calculate the consumption, it takes into account the time the equipment remain asleep multiplied by the power sleep power profile described in Equation (2) plus the time the equipment remains operating multiplied by the load proportional power profile described in Equation (1).

5.4 Chapter Final Remarks

In this Chapter, the SOS prototype architecture and implementation details were presented. The XML used in the refinement steps, the Analytical Solvers algorithms and the power profiles were also described. The method was validated using GreenSDN, the first work to present an open, easy to replicate environment to validate the use of energy efficiency capabilities in SDN environments.

In the next Chapter, this environment will be used to evaluate the SOS considering the Utility Function results, the selection of one or a combination of energy efficiency capabilities considering different network situations, the decision trees application and dynamicity aspects.

6 SOS EXPERIMENTS AND VALIDATION

In this Chapter, the results of the SOS method deployment are presented using the test environment described in the previous Chapter. The Utility Function selection is validated and the tests results are compared with the expected outcomes from the Analytical Solvers in Section 6.2. Section 6.3 demonstrates the additional savings possible when two capabilities are combined. Section 6.4 presents the complete use case to check the tree decisions being applied, and Section 6.5 describes the dynamicity experiments. Section 6.6 contains further details of the step-by-step execution of the implemented method, starting the test environment, defining policies, considering different workloads and periods of the day. The demonstration was outlined in (RIEKSTIN et al., 2015c).

6.1 Experiments Setup: topology and power profiles

The network topology of the experiments, depicted in Figure 41, was inspired by the core part of the RNP network (Rede Nacional de Ensino e Pesquisa, the Brazilian NREN - National Research and Education Network).

The power profiles used in the experiments were based on those ones proposed by (JANUARIO et al., 2013). The power profiles equations are the following:

$$PP_{proportional500} = 200 + \left(\frac{500}{30}\right) * workload \tag{6.1}$$

$$PP_{proportional500-sleeping} = 120 \tag{6.2}$$

$$PP_{proportional500+ALR} = 200 + \left(\frac{500}{30}\right) * workload - 15\% \tag{6.3}$$

$$PP1_{proportional500+SC} = 120 * tOff + (200 + \left(\frac{500}{30}\right) * workload) * tOn \tag{6.4}$$

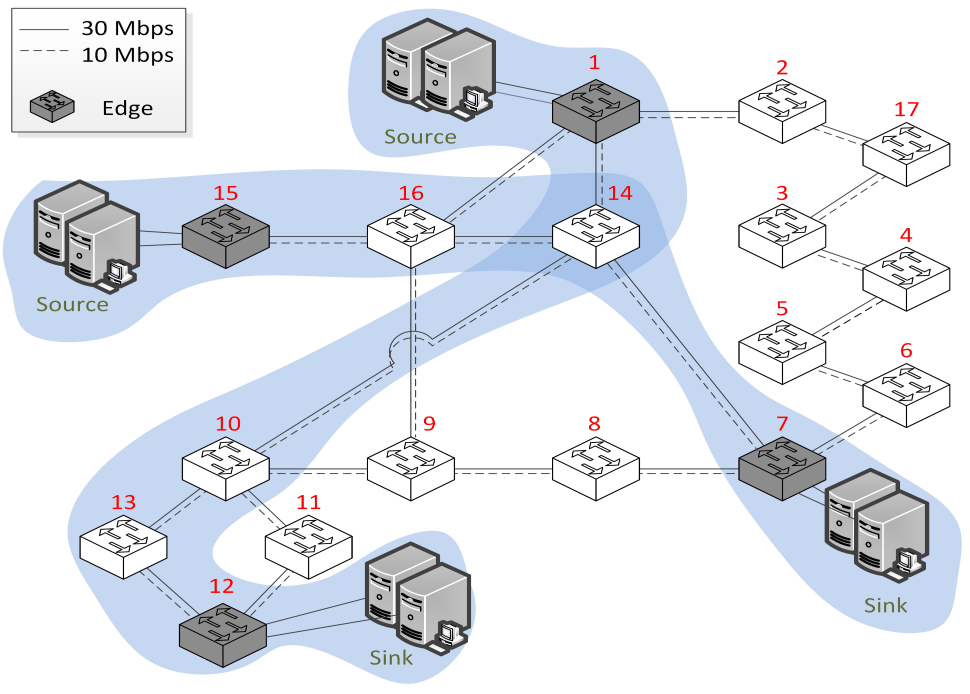


Figure 41: The network topology used in the SOS method proof-of-concept

In Section 6.3, two additional power profiles are added for showing the additional savings possible when combining two energy efficiency capabilities for different equipment, another load proportional power profile (PP$_{proportional1000}$), and one not load proportional (PP$_{fixed}$):

$$PP_{proportional1000} = 200 + \left(\frac{1000}{30}\right) workload \tag{6.5}$$

$$PP_{proportional1000-sleeping} = 170 \tag{6.6}$$

$$PP_{fixed} = 1000 \tag{6.7}$$

$$PP_{fixed-sleeping} = 250 \tag{6.8}$$

6.2 Utility Function Validation

In order to check the Utility Function (UF) selection, the Analytical Solvers results applied to this UF (ALR, SC, SustNMS Sustainability Policy, or SustNMS Performance Policy) were evaluated using the power profile PP$_{proportional500}$. In some cases, the maximum grade is achieved by more than one scenario, i.e., scenarios with different pairs bandwidth utilization value/capability (-ties). There is a good example of the Utility Function selection in the scenarios depicted in Table 15. This table lists all possible combinations of capabilities for one bandwidth utilization value. In this example, applying SustNMS-Sustainability saves more energy, with the final power dissipated 2663.45W, smaller than 2786.91W from SustNMS-Performance. However, SustNMS-Sustainability presents packet losses and, for this reason, loses points with the pl value 0.9. As the final result, the SustNMS-Performance Utility Function grade got the greater value, and so this option is selected. As a consequence, the system will not save as much energy as possible, but will not loose so many packets.

Table 15: Utility Function validation example

Capabilities	**With savings(W)**	**Baseline (W)**	**PL**	**UF Result**
ALR and/or SSC	3371,95	3371,95	1	1,00
SustNMS-Performance	2786,91	3371,95	1	1,21
SustNMS-Sustainability	2663,45	3371,95	0,9	1,14

6.3 Orchestration Experiment of Two Capabilities Combination

To check the combinations of capabilities and respective savings and losses, a test environment in which SustNMS is applied alone and in conjunction with ALR was emulated using the power profiles $\mathrm{PP}_{proportional500}$, $\mathrm{PP}_{proportional1000}$, and PP_{fixed}. The experiment was performed running two flows: one from the leftmost source W in Figure 41, to the sink E on the rightmost side, and another from the topmost source N to the bottommost sink S. Figure 42 depicts the Analytical Solver results for $\mathrm{PP}_{proportional500}$ in four situations:

- The baseline with no load, that is, the energy spent when there is no energy efficiency capability being applied;
- The baseline without energy efficiency capabilities being applied with two 10 Mbps flows;
- SustNMS being used in the scenario with two 10 Mbps flows; and
- The orchestration scenario, in which SustNMS is applied in conjunction with ALR. This happens because the 10 Mbps workload allows ALR application, bringing more savings.

Figure 43 represents the same situations, but measured directly in the emulation environment. One can see that the results are similar. In both cases, the application of just one capability brings 15% of savings, while applying both bring 20% reduction. The amount of savings produced by energy efficiency orchestration can be as high as the maximum savings brought by the second capability. In the example, ALR was applied after SustNMS usage, in the switches that remained powered on. Besides, a conservative scenario in which ALR can save 15% of energy was assumed, but this number can reach 21% (RICCIARDI et al., 2011).

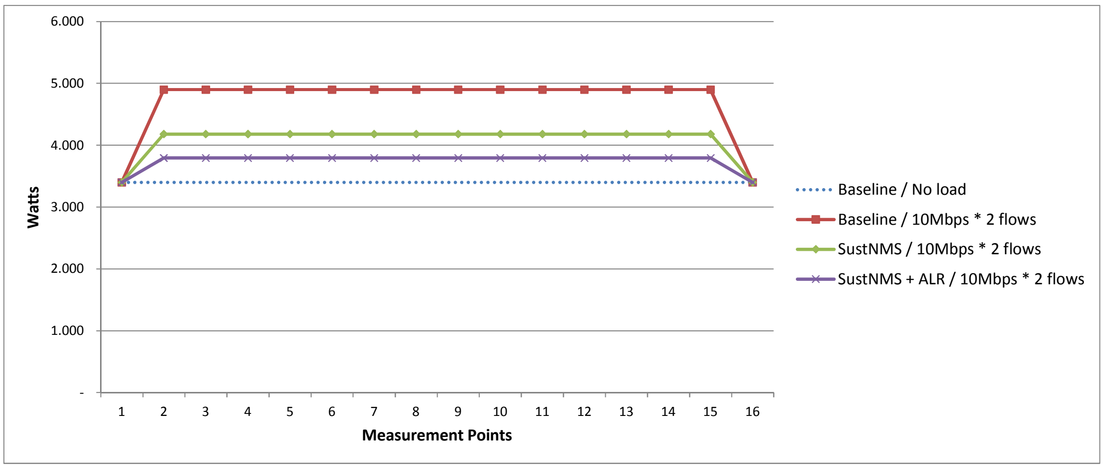


Figure 42: Analytical solver results for two 10 Mbps flows for PP$_{proportional500}$

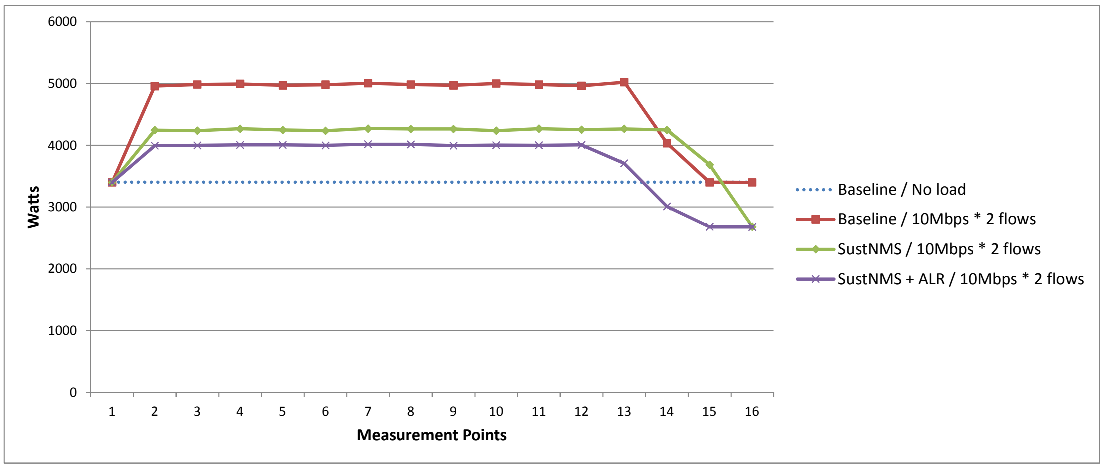


Figure 43: Emulation results for two 10 Mbps flows for PP$_{proportional500}$

Considering the second power profile, PP$_{proportional1000}$, applying SustNMS brought near 11% of savings. Applying ALR in conjunction, brought additional 10% savings, as illustrated in Figure 44.

If the power profile is not load-proportional, the savings can be higher. Figure 45 depicts a scenario in which every switch spends 1000W regardless the load. The savings applying SustNMS is 47%. Orchestration brings 53% savings.

One important consideration is the scenario with high loads, for example, two 20 Mbps flows for PP$_{proportional500}$. Energy efficiency capabilities may be activated only during idle periods in the network. Therefore, it is expected they will not present significant savings in high load scenarios. Besides, ALR, for instance, would not make

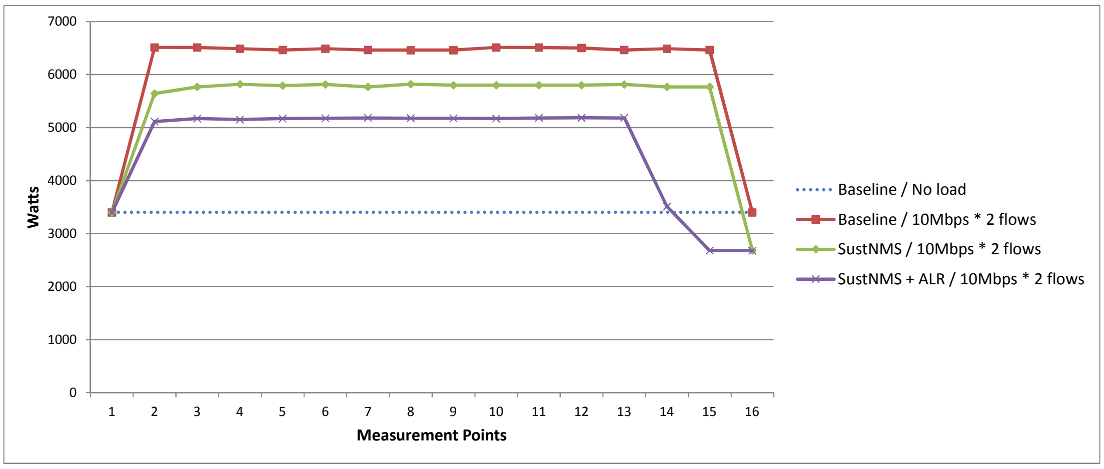


Figure 44: Emulation results for two 10 Mbps flows for $\mathrm{PP}_{proportional1000}$

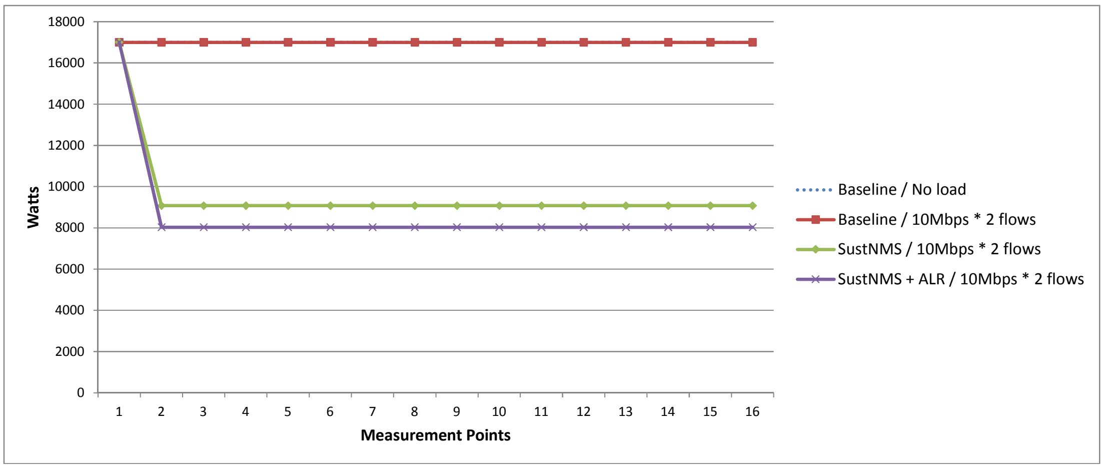


Figure 45: Results when the energy consumption is nor proportional to the load with PP_{fixed}

any difference, since it reduces link rates only according to the Ethernet rates, and, in this case, the workload is higher than the 10 Mbps reduced link rate. The graph uses a dashed line for the SustNMS with ALR scenario to show it has the same results as applying only SustNMS as depicted in Figure 46. The power profile used was the load proportional $\mathrm{PP}_{proportional500}$.

6.4 Validating the Use of Decision Trees

To validate the tree decisions, the results from the Utility Function module, which used the predictions from the Analytical Solvers, were compared with the

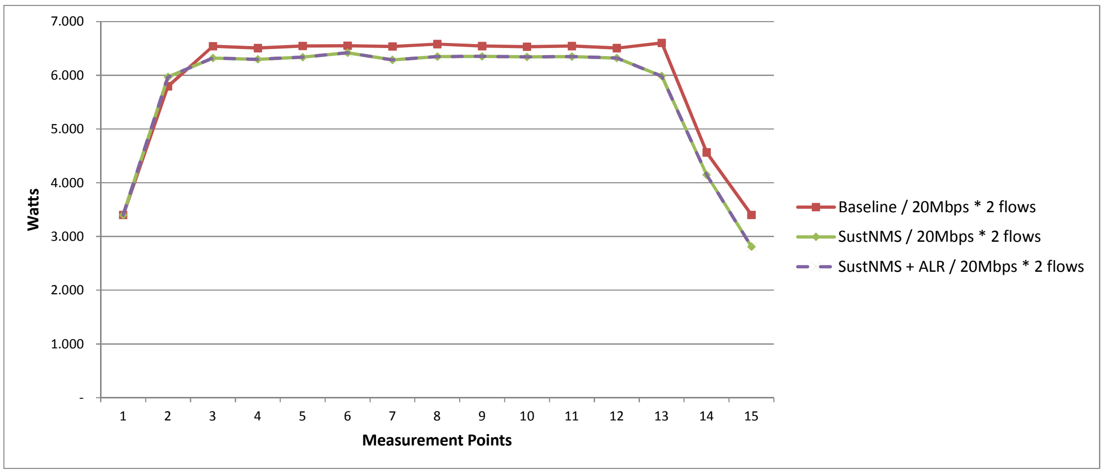


Figure 46: Results for a high load scenario with load proportional power profiles for $PP_{proportional500}$

results while running the experiments in the emulation environment using the power profile $PP_{proportional500}$. Table 16 shows the selected capabilities for each randomly generated set of bandwidth usage values in an experiment with 10 generated scenarios, considering the equipment and the network topology depicted in Figure 41.

The first column shows the selected capabilities for each scenario. For simplicity, they are not differentiated by scope in the table (ALR is applied to components, SC to nodes, SustNMS to the network). For instance, when written “ALR and/or SC”, it means that the nodes and its components have ALR and/or SC enabled, according to the workload each node is handling. The other 17 columns represent the bandwidth usage in each node, randomly generated. One can observe that the values are always smaller than 30Mbps, the maximum load of the nodes used in the experiment. According to Figure 41, the paths in use are composed by the following nodes:

- Path 1: 1, 14, 10, 13, 12
- Path 2: 15, 16, 14, 7

From the Utility Function results, when both flows are smaller than 12 Mbps, the expected capabilities combination (the one with the highest Utility Function grade) is

Table 16: Utility Function selections for the given scenarios using the power profile $PP_{proportional500}$

Selected	Device Bandwidth Usage in Each Node (in Mbps)																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ALR and/or SC	3,6	0	0	0	0	0	4,4	0	0	3,6	0	3,6	3,6	8,0	4,4	4,4	0
SustNMS Perf.	24,4	0	0	0	0	0	24,8	0	0	24,4	0	24,4	24,4	30	24,8	24,8	0
SustNMS Sust.	14,2	0	0	0	0	0	12,4	0	0	14,2	0	14,2	14,2	26,5	12,4	12,4	0
ALR and/or SC	6,2	0	0	0	0	0	10,5	0	0	6,2	0	6,2	6,2	16,7	10,5	10,5	0
SustNMS Sust.+ALR	4,4	0	0	0	0	0	13,2	0	0	4,4	0	4,4	4,4	17,7	13,2	13,2	0
ALR and/or SC	11,7	0	0	0	0	0	10,8	0	0	11,7	0	11,7	11,7	22,5	10,8	10,8	0
ALR and/or SC	5,9	0	0	0	0	0	10,8	0	0	5,9	0	5,9	5,9	16,7	10,8	10,8	0
ALR and/or SC	10,3	0	0	0	0	0	0,1	0	0	10,3	0	10,3	10,3	10,5	0,1	0,1	0
SustNMS Sust.	12,4	0	0	0	0	0	12,5	0	0	12,4	0	12,4	12,4	25,0	12,5	12,5	0
SustNMS Sust.+ALR	14,2	0	0	0	0	0	6,7	0	0	14,2	0	14,2	14,2	20,9	6,7	6,7	0

SC with ALR. If one of them is greater than 12 Mbps, SustNMS-Sustainability is the expected selection, with ALR being applied if one of the flows is smaller than 10Mbps. When the sum of both flows surpasses 30 Mbps, node 14 will be the bottleneck and will lose many packets. In this case, the Utility Function ended up selecting SustNMS-Performance that is, it is better to wake up some additional devices and save less, in order to loose fewer packets, ensuring more QoS. In this case, switches 8 and 9 will be waken up.

Considering the expected decision tree behavior, the following experiments were performed in the testbed: (1) Flow 1: 2Mbps, Flow 2: 2Mbps; (2) Flow 1: 14Mbps, Flow 2: 6Mbps; (3) Flow 1: 15Mbps, Flow 2: 15Mbps; (4) Flow 1: 24Mbps, Flow 2: 24Mbps. Note that none of them is equal the workloads depicted in Table 16, so that it is possible to check the Scikit tool interpolation. Table 17 details what happened in each case. All the experiments had the result as expected.

Figure 47 and Figure 48 show how the graphics of the SOS method prototype are. The first window illustrates the network topology represented in the method in conjunction with the capabilities being applied and where they are being applied. The

Table 17: Decision Tree Results

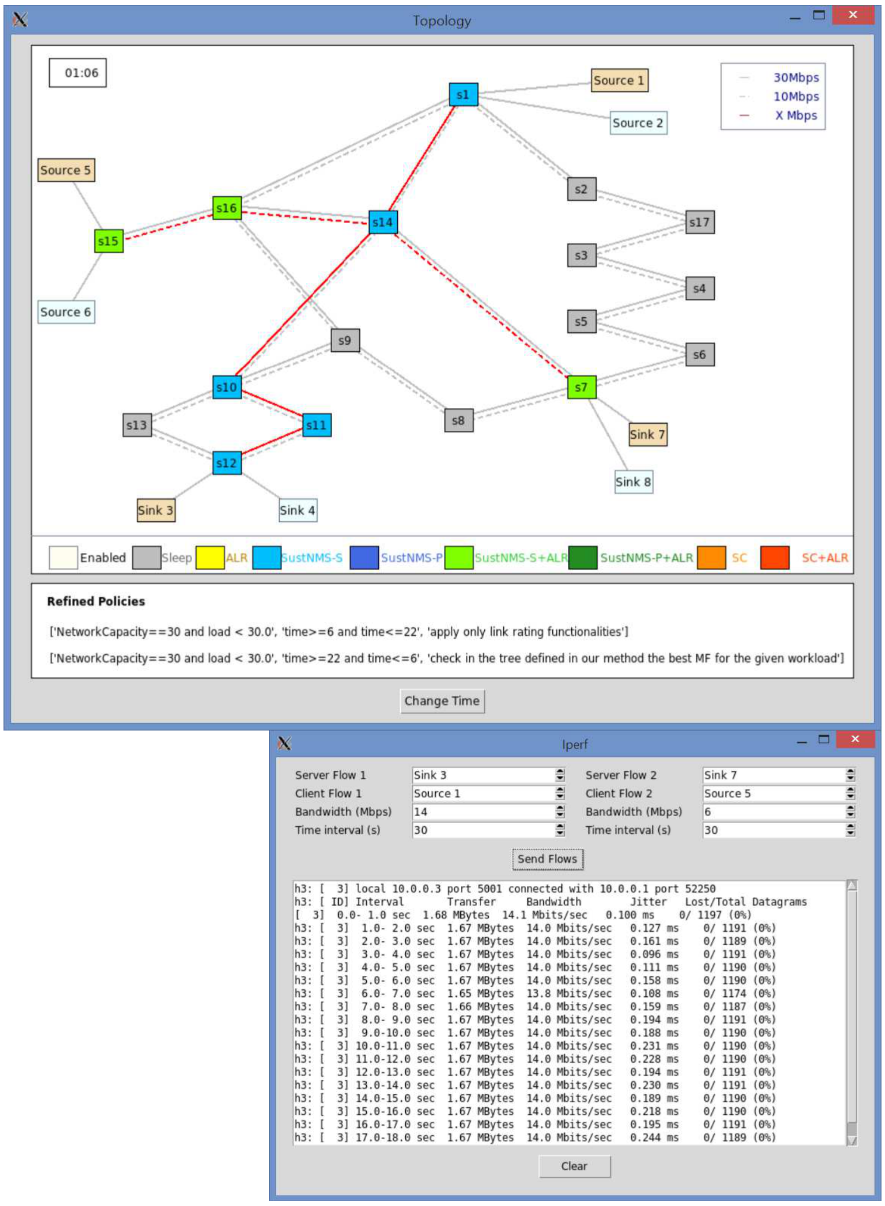
Flow 1 / Flow 2	**Expected from the Utility Function**	**Happened in the Emulation Environment**
2Mbps / 2 Mbps	SC + ALR in both flows	SC + ALR in both flows
14 Mbps / 6 Mbps	SustNMS-Sustainability+ALR in the flow with 6Mbps, only SustNMS-Sustainability in the flow with 14Mbps	SustNMS-Sustainability+ALR in the flow with 6Mbps, only SustNMS-Sustainability in the flow with 14Mbps
15 Mbps / 15 Mbps	SustNMS-Sustainability in both flows	SustNMS-Sustainability in both flows
24 Mbps / 24 Mbps	SustNMS-Performance in both flows	SustNMS-Performance in both flows

refined policies are also presented. The second window shows the iPerf generated flows. Figure 47 depicts the interface for the method running with the selection of the capabilities for the scenario in which one flow has 14Mbps and the other, 6Mbps. Figure 48 depicts the result for the 24 Mbps flows, in which SustNMS-Performance is selected. Note that, considering the possible losses, two extra switches are turned on, s8 and s9.

6.5 Checking Dynamicity

The last requirement to be validated is dynamicity. Among the terms refined from the high-level policies, is the period of the day the energy efficiency capabilities will take place (during the night and during the day), and which capabilities are allowed in each case. For instance, during the night, all capabilities are allowed, while, during the day, only link rating capabilities are allowed, as exemplified in Figure 49 and Figure 50. This can be the determined because the link rating capabilities enforcement are in hundreds of milliseconds order of magnitude while sleeping can take seconds to be fully operational. Besides, under high utilization scenarios, rate adaption suits better than sleeping, as, for instance, Nedevschi et al. (NEDEVSCHI et al., 2008) presented.

SOS reacts differently in these two pre-defined periods. Figure 49 illustrates the SOS graphical user interface during the night, applying different capabilities (in the example, SC pus ALR), and Figure 50 presents the method operating during the day,



Refined Policies

['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

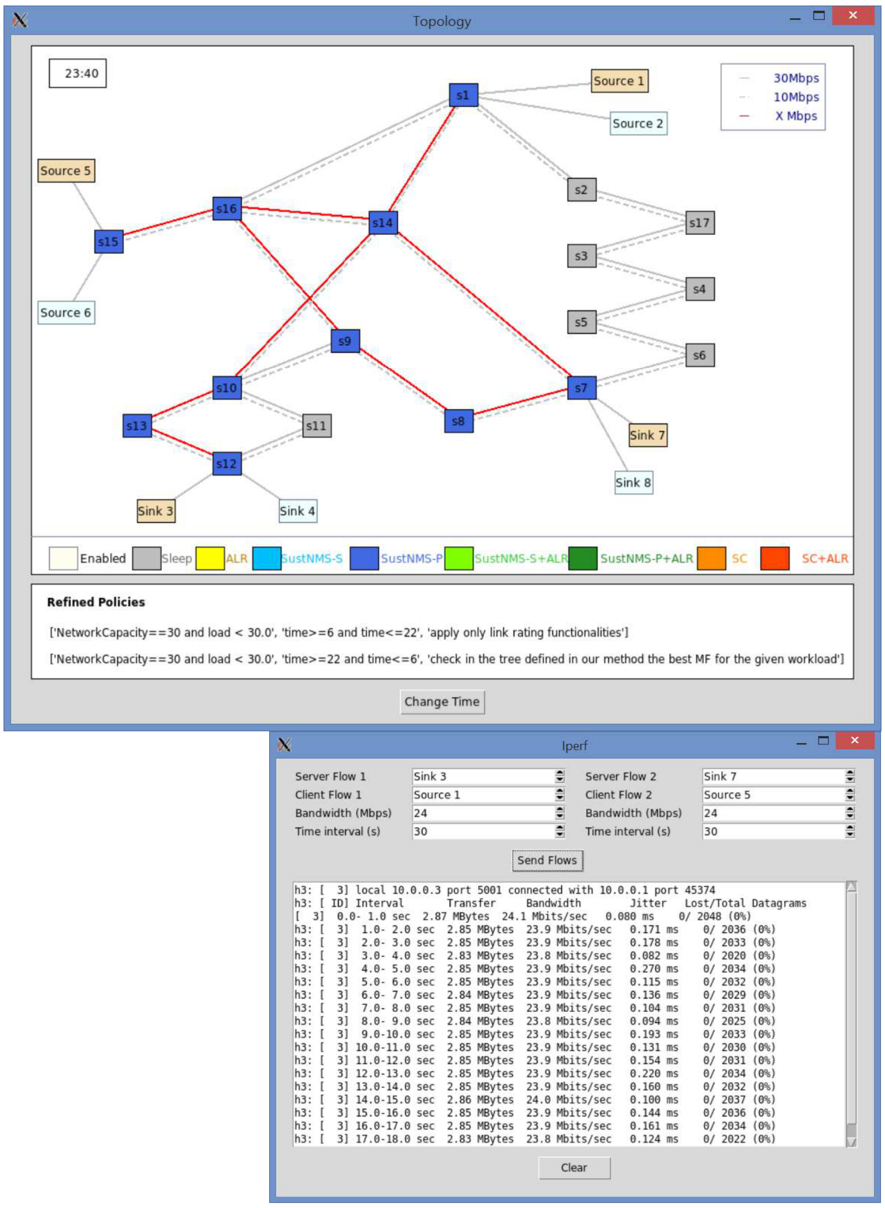
['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	14	Bandwidth (Mbps)	6
Time interval (s)	30	Time interval (s)	30

```
h3: [  3] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 52250
h3: [ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec  1.68 MBytes  14.1 Mbits/sec   0.100 ms    0/ 1197 (0%)
h3: [  3]  1.0- 2.0 sec  1.67 MBytes  14.0 Mbits/sec   0.127 ms    0/ 1191 (0%)
h3: [  3]  2.0- 3.0 sec  1.67 MBytes  14.0 Mbits/sec   0.161 ms    0/ 1189 (0%)
h3: [  3]  3.0- 4.0 sec  1.67 MBytes  14.0 Mbits/sec   0.096 ms    0/ 1191 (0%)
h3: [  3]  4.0- 5.0 sec  1.67 MBytes  14.0 Mbits/sec   0.111 ms    0/ 1190 (0%)
h3: [  3]  5.0- 6.0 sec  1.67 MBytes  14.0 Mbits/sec   0.158 ms    0/ 1190 (0%)
h3: [  3]  6.0- 7.0 sec  1.65 MBytes  13.8 Mbits/sec   0.108 ms    0/ 1174 (0%)
h3: [  3]  7.0- 8.0 sec  1.66 MBytes  14.0 Mbits/sec   0.159 ms    0/ 1187 (0%)
h3: [  3]  8.0- 9.0 sec  1.67 MBytes  14.0 Mbits/sec   0.194 ms    0/ 1191 (0%)
h3: [  3]  9.0-10.0 sec  1.67 MBytes  14.0 Mbits/sec   0.188 ms    0/ 1190 (0%)
h3: [  3] 10.0-11.0 sec  1.67 MBytes  14.0 Mbits/sec   0.231 ms    0/ 1190 (0%)
h3: [  3] 11.0-12.0 sec  1.67 MBytes  14.0 Mbits/sec   0.228 ms    0/ 1190 (0%)
h3: [  3] 12.0-13.0 sec  1.67 MBytes  14.0 Mbits/sec   0.194 ms    0/ 1191 (0%)
h3: [  3] 13.0-14.0 sec  1.67 MBytes  14.0 Mbits/sec   0.230 ms    0/ 1191 (0%)
h3: [  3] 14.0-15.0 sec  1.67 MBytes  14.0 Mbits/sec   0.189 ms    0/ 1190 (0%)
h3: [  3] 15.0-16.0 sec  1.67 MBytes  14.0 Mbits/sec   0.218 ms    0/ 1190 (0%)
h3: [  3] 16.0-17.0 sec  1.67 MBytes  14.0 Mbits/sec   0.195 ms    0/ 1191 (0%)
h3: [  3] 17.0-18.0 sec  1.67 MBytes  14.0 Mbits/sec   0.244 ms    0/ 1189 (0%)
```

Figure 47: SOS Method applying SustNMS Sustainability + ALR for a workload of 14 Mbps / 6 Mbps (emulation environment)

applying only ALR, since only link rating capabilities are allowed. It is also important to note that ALR is only applicable when the workload is smaller than 10Mbps.



Refined Policies

['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	24	Bandwidth (Mbps)	24
Time interval (s)	30	Time interval (s)	30

```
h3: [  3] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 45374
h3: [ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec  2.87 MBytes  24.1 Mbits/sec   0.080 ms    0/ 2048 (0%)
h3: [  3]  1.0- 2.0 sec  2.85 MBytes  23.9 Mbits/sec   0.171 ms    0/ 2036 (0%)
h3: [  3]  2.0- 3.0 sec  2.85 MBytes  23.9 Mbits/sec   0.178 ms    0/ 2033 (0%)
h3: [  3]  3.0- 4.0 sec  2.83 MBytes  23.8 Mbits/sec   0.082 ms    0/ 2020 (0%)
h3: [  3]  4.0- 5.0 sec  2.85 MBytes  23.9 Mbits/sec   0.270 ms    0/ 2034 (0%)
h3: [  3]  5.0- 6.0 sec  2.85 MBytes  23.9 Mbits/sec   0.115 ms    0/ 2032 (0%)
h3: [  3]  6.0- 7.0 sec  2.84 MBytes  23.9 Mbits/sec   0.136 ms    0/ 2029 (0%)
h3: [  3]  7.0- 8.0 sec  2.85 MBytes  23.9 Mbits/sec   0.104 ms    0/ 2031 (0%)
h3: [  3]  8.0- 9.0 sec  2.84 MBytes  23.8 Mbits/sec   0.094 ms    0/ 2025 (0%)
h3: [  3]  9.0-10.0 sec  2.85 MBytes  23.9 Mbits/sec   0.193 ms    0/ 2033 (0%)
h3: [  3] 10.0-11.0 sec  2.85 MBytes  23.9 Mbits/sec   0.131 ms    0/ 2030 (0%)
h3: [  3] 11.0-12.0 sec  2.85 MBytes  23.9 Mbits/sec   0.154 ms    0/ 2031 (0%)
h3: [  3] 12.0-13.0 sec  2.85 MBytes  23.9 Mbits/sec   0.220 ms    0/ 2034 (0%)
h3: [  3] 13.0-14.0 sec  2.85 MBytes  23.9 Mbits/sec   0.160 ms    0/ 2032 (0%)
h3: [  3] 14.0-15.0 sec  2.86 MBytes  24.0 Mbits/sec   0.100 ms    0/ 2037 (0%)
h3: [  3] 15.0-16.0 sec  2.85 MBytes  23.9 Mbits/sec   0.144 ms    0/ 2036 (0%)
h3: [  3] 16.0-17.0 sec  2.85 MBytes  23.9 Mbits/sec   0.161 ms    0/ 2034 (0%)
h3: [  3] 17.0-18.0 sec  2.83 MBytes  23.8 Mbits/sec   0.124 ms    0/ 2022 (0%)
```

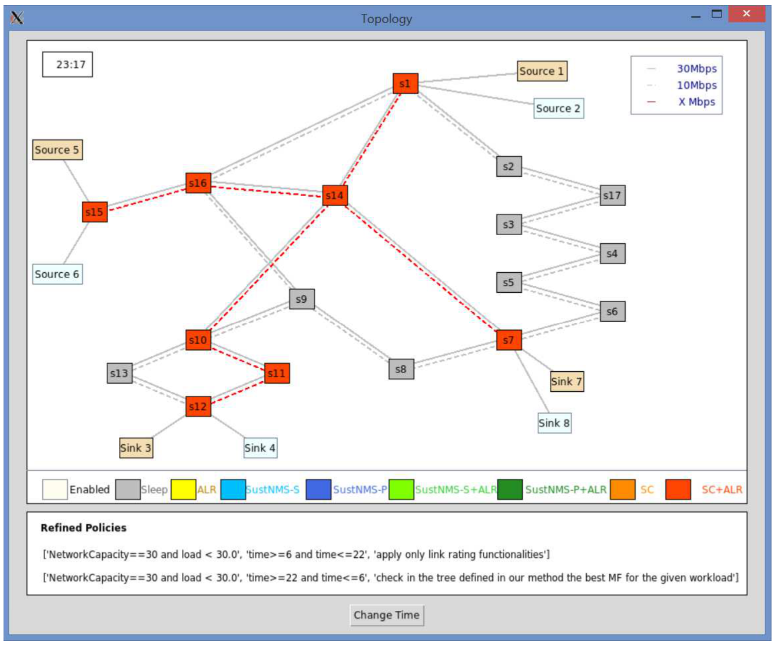
Figure 48: SOS Method applying SustNMS Performance for a workload of 24 Mbps / 24 Mbps (emulation environment)

6.6 SOS Demonstration

In this Section, SOS is demonstrated, starting the test environment, defining policies, considering different workloads and periods of the day. The video of this experiment can be accessed *here*[1].

The first steps relate to the environment configuration and setup, starting the

[1]https://vimeo.com/134749090

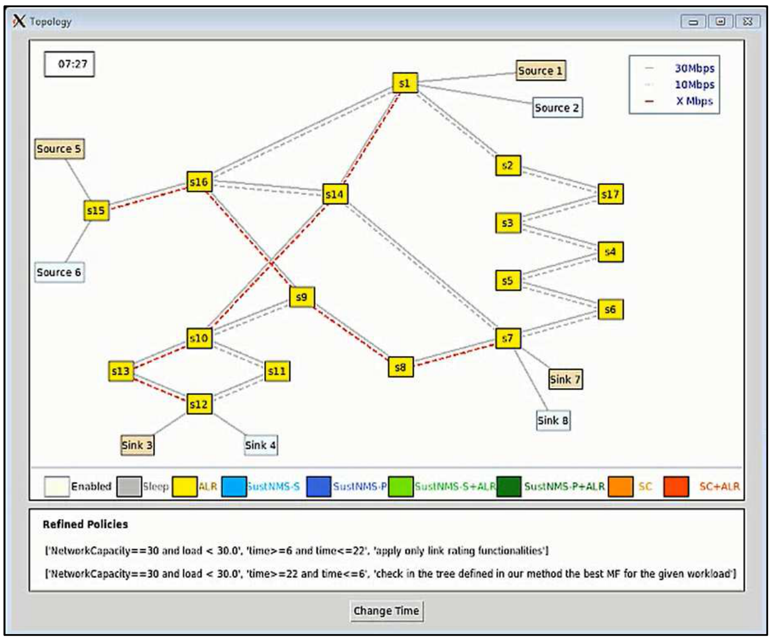


Refined Policies

['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

Figure 49: SOS method operating during the night, allowing any combination of capabilities



Refined Policies

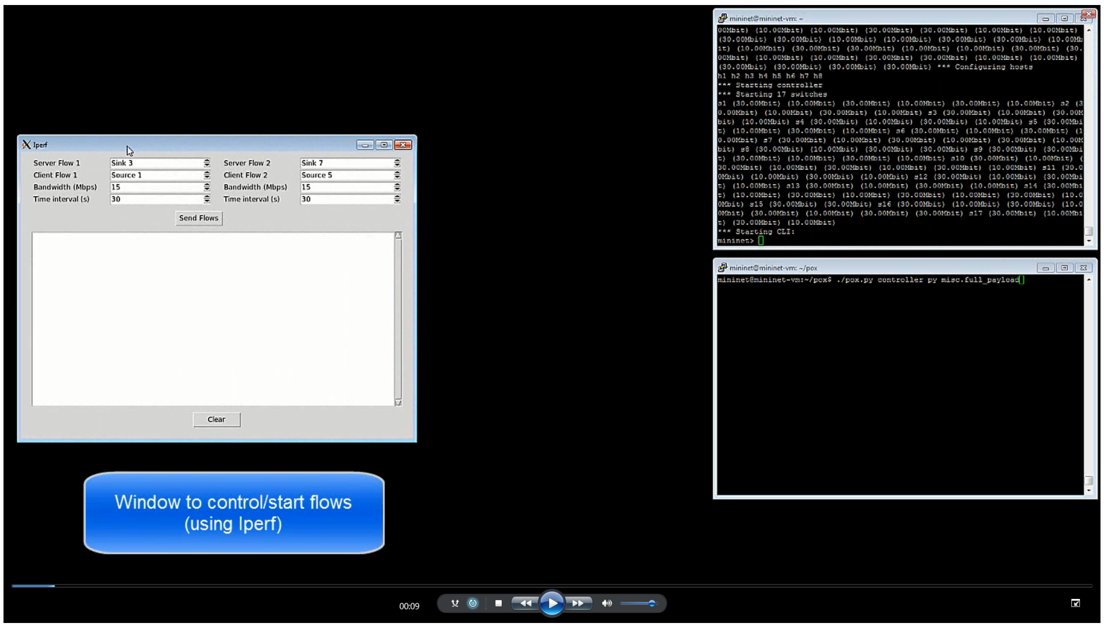
['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

Figure 50: SOS method operating during the day, allowing only link rating capabilities

topology and the switches. The flows to test the method are controlled using Iperf and there is a window to generate the different flows, as illustrated in Figure 51. The controller is then started.

The first step after the environment is ready to use is to define the business policies in the user interface, as illustrated in Figure 52. The decision tress are then built



Iperf

Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	15	Bandwidth (Mbps)	15
Time interval (s)	30	Time interval (s)	30

Send Flows

Clear

mininet@mininet-vm: ~

```
00Mbit) (10.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) (10.00Mbit) (10.00Mbit)
(30.00Mbit) (30.00Mbit) (10.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) (10.00Mb
it) (10.00Mbit) (30.00Mbit) (30.00Mbit) (10.00Mbit) (10.00Mbit) (30.00Mbit) (30.
00Mbit) (10.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) (10.00Mbit) (10.00Mbit)
(30.00Mbit) (30.00Mbit) (30.00Mbit) (30.00Mbit) *** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
*** Starting 17 switches
s1 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s2 (3
0.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s3 (30.00Mbit) (10.00Mbit) (30.00M
bit) (10.00Mbit) s4 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s5 (30.00Mbi
t) (10.00Mbit) (30.00Mbit) (10.00Mbit) s6 (30.00Mbit) (10.00Mbit) (30.00Mbit) (1
0.00Mbit) s7 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00M
bit) s8 (30.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) s9 (30.00Mbit) (30.00Mbi
t) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s10 (30.00Mbit) (10.00Mbit) (
30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s11 (30.0
0Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s12 (30.00Mbit) (10.00Mbit) (30.00Mbi
t) (10.00Mbit) s13 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s14 (30.00Mbi
t) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.0
0Mbit) s15 (30.00Mbit) (30.00Mbit) s16 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.0
0Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) s17 (30.00Mbit) (10.00Mbi
t) (30.00Mbit) (10.00Mbit)
*** Starting CLI:
mininet>
```

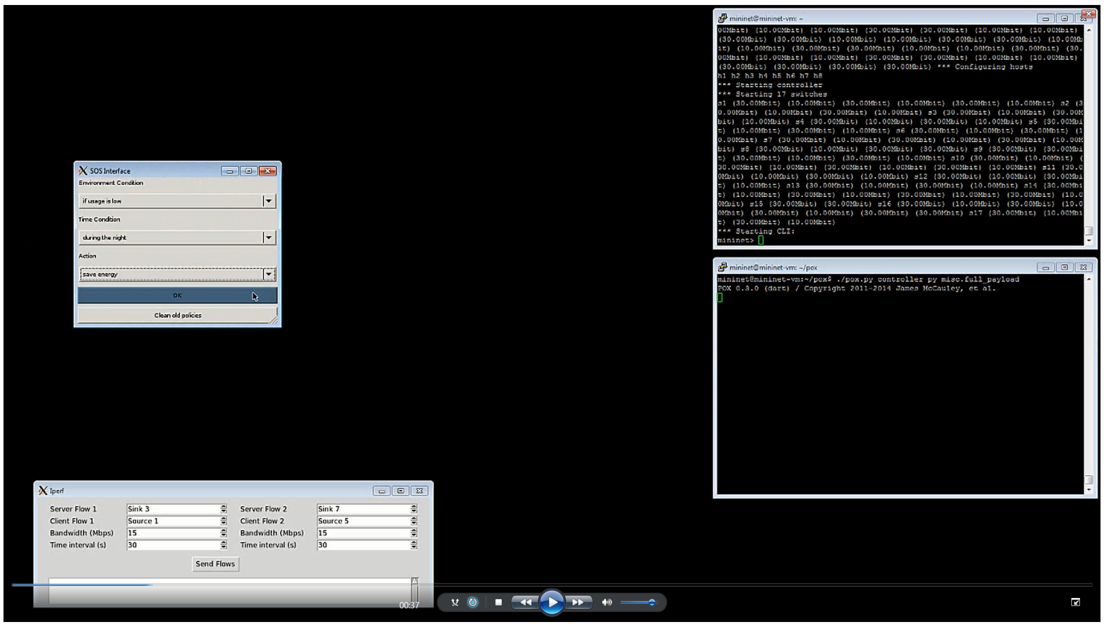
mininet@mininet-vm: ~/pox

```
mininet@mininet-vm:~/pox$ ./pox.py controller py misc.full_payload
```

00:09

Figure 51: SOS Setup

according to the policies. They will act as a policy decision point. When the network conditions change, we expect a different capability to be selected, or a combination of capabilities.



SOS Interface

Environment Condition

if usage is low

Time Condition

during the night

Action

save energy

OK

Clean old policies

mininet@mininet-vm: ~

```
00Mbit) (10.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) (10.00Mbit) (10.00Mbit)
(30.00Mbit) (30.00Mbit) (10.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) (10.00Mb
it) (10.00Mbit) (30.00Mbit) (30.00Mbit) (10.00Mbit) (10.00Mbit) (30.00Mbit) (30.
00Mbit) (10.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) (10.00Mbit) (10.00Mbit)
(30.00Mbit) (30.00Mbit) (30.00Mbit) (30.00Mbit) *** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
*** Starting 17 switches
s1 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s2 (3
0.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s3 (30.00Mbit) (10.00Mbit) (30.00M
bit) (10.00Mbit) s4 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s5 (30.00Mbi
t) (10.00Mbit) (30.00Mbit) (10.00Mbit) s6 (30.00Mbit) (10.00Mbit) (30.00Mbit) (1
0.00Mbit) s7 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00M
bit) s8 (30.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) s9 (30.00Mbit) (30.00Mbi
t) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s10 (30.00Mbit) (10.00Mbit) (
30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s11 (30.0
0Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s12 (30.00Mbit) (10.00Mbit) (30.00Mbi
t) (10.00Mbit) s13 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) s14 (30.00Mbi
t) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.0
0Mbit) s15 (30.00Mbit) (30.00Mbit) s16 (30.00Mbit) (10.00Mbit) (30.00Mbit) (10.0
0Mbit) (30.00Mbit) (10.00Mbit) (30.00Mbit) (30.00Mbit) s17 (30.00Mbit) (10.00Mbi
t) (30.00Mbit) (10.00Mbit)
*** Starting CLI:
mininet>
```

mininet@mininet-vm: ~/pox

```
mininet@mininet-vm:~/pox$ ./pox.py controller py misc.full_payload
POX 0.3.0 (dart) / Copyright 2011-2014 James McCauley, et al.
```

Iperf

Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	15	Bandwidth (Mbps)	15
Time interval (s)	30	Time interval (s)	30

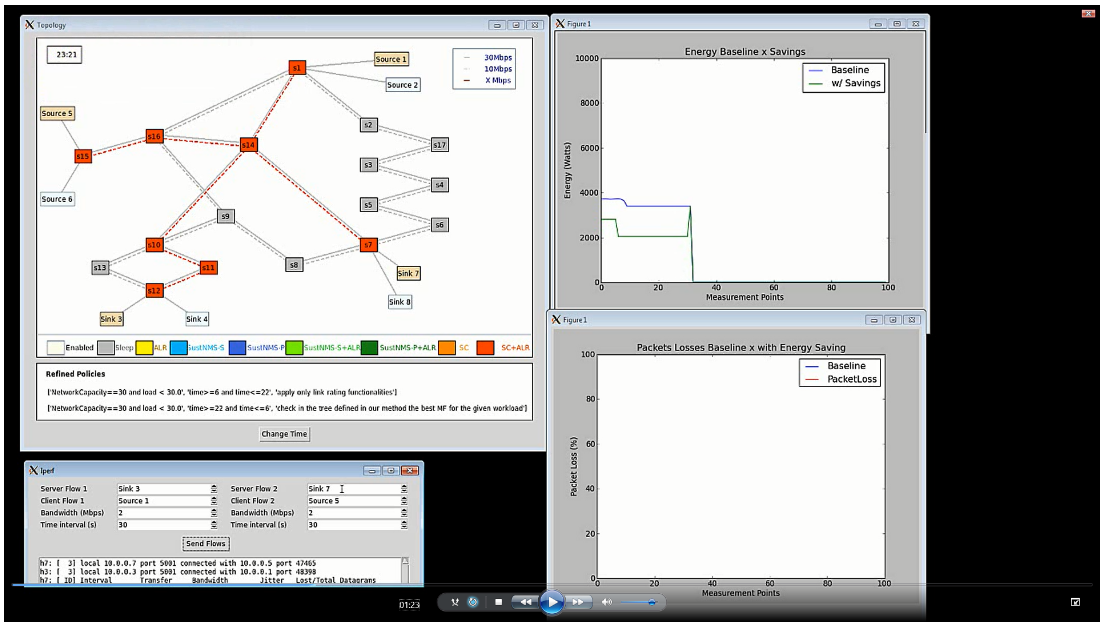
Send Flows

00:37

Figure 52: Definition of Policies in the SOS Interface

After this, as illustrated in Figure 53, it is possible to check the topology with its sources and sinks, the links being used, the capabilities being applied (represented by the different colors), and the two policies examples, one for the the night, another for the day, translated using Table Lookup. It is also possible to check the graphs showing energy savings and packet losses, if they happen.

The experiment starts with two small flows of 2 Mbps each. The SOS method selects the best configuration for this scenario. In this case, SC, a sleeping capability, was applied in conjunction with ALR, a link rating capability, in a complementary way, as illustrated in Figure 53. When the network condition change, other capabilities will be selected.



Refined Policies

['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

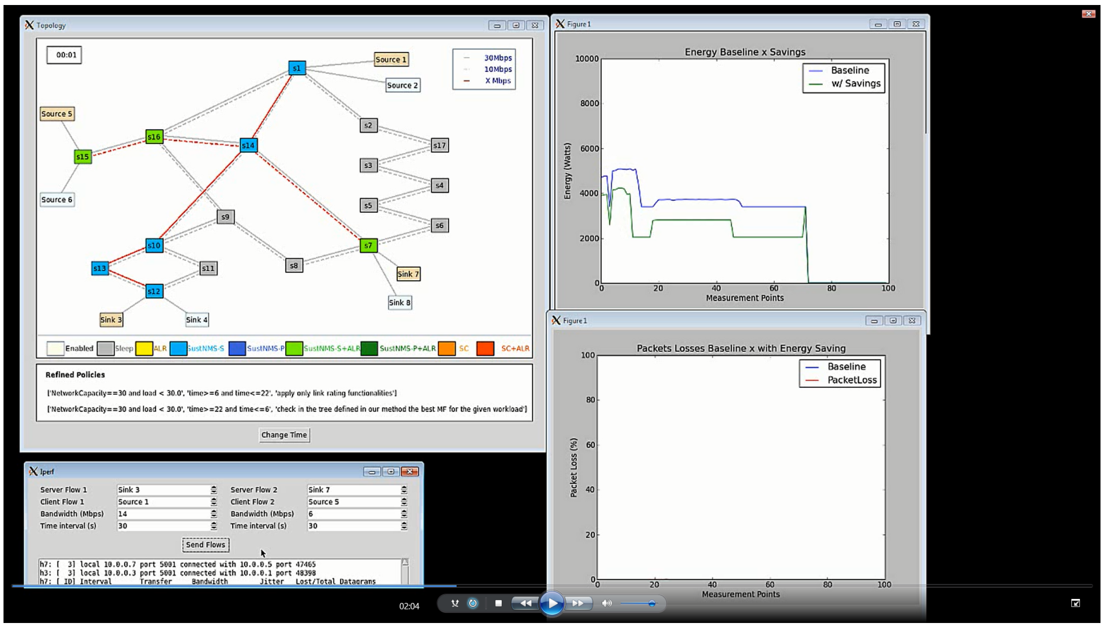
Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	2	Bandwidth (Mbps)	2
Time interval (s)	30	Time interval (s)	30

```
h7: [  3] local 10.0.0.7 port 5001 connected with 10.0.0.5 port 47465
h3: [  3] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 48398
h7: [ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
```

Figure 53: SOS selection for two 2Mbps flows: SC with ALR

With two bigger flows, one smaller and another bigger than 10 Mbps, 6 Mbps and 14 Mbps, SustNMS Sustainability, that is, with the policy which prioritizes energy savings, is the best option, plus ALR in the 6 Mbps path. Remember that ALR applies only for flows smaller than 10 Mbps. Then we have two different situations: one flow with SustNMS Sustainability, other with SustNMS Sustainability plus ALR, as illustrated in Figure 54.

With two flows bigger than 10 Mbps, 15 Mbps each, but still smaller than the maximum load the network can handle, SustNMS Sustainability is the best option for



Refined Policies

['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

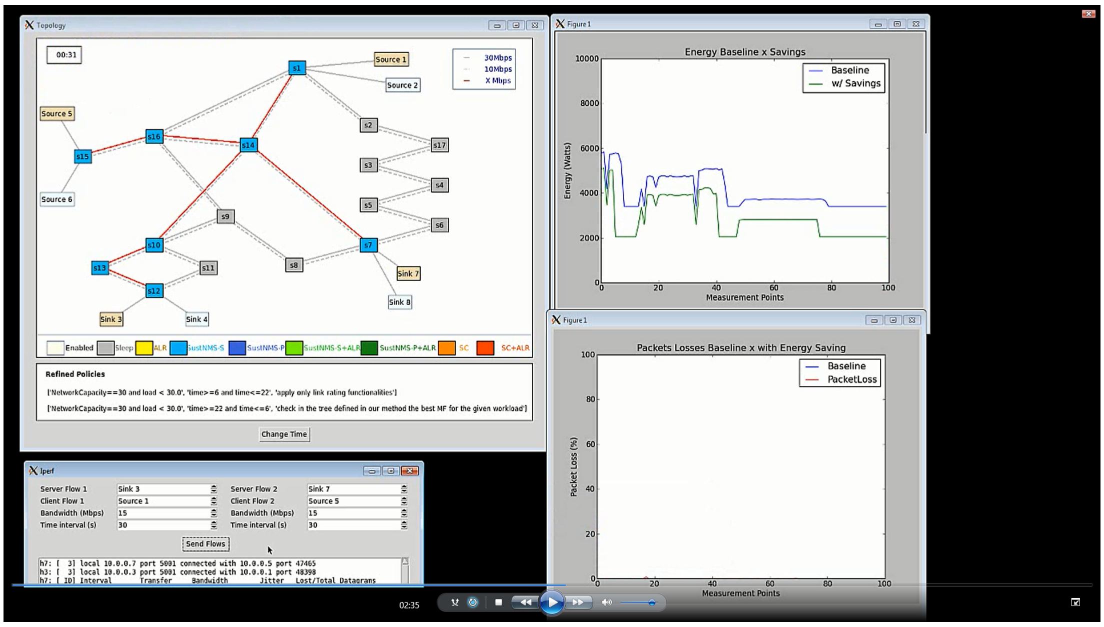
Iperf

Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	14	Bandwidth (Mbps)	6
Time interval (s)	30	Time interval (s)	30

```
h7: [  3] local 10.0.0.7 port 5001 connected with 10.0.0.5 port 47465
h3: [  3] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 48398
h7: [ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
```

Figure 54: SOS selection for two different flows, one of 14 Mbps and another of 6 Mbps: SustNMS Sustainability, and SustNMS Sustainability with ALR

both paths, as illustrated in Figure 55



Refined Policies

['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

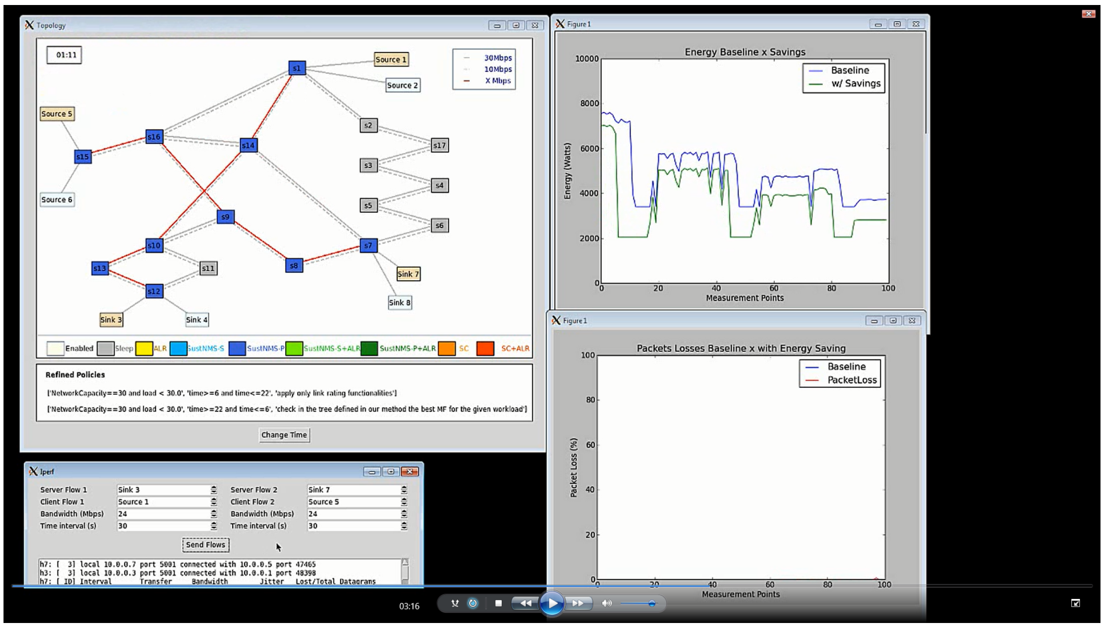
Iperf

Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	15	Bandwidth (Mbps)	15
Time interval (s)	30	Time interval (s)	30

```
h7: [  3] local 10.0.0.7 port 5001 connected with 10.0.0.5 port 47465
h3: [  3] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 48398
h7: [ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
```

Figure 55: SOS selection for two 15 Mbps flows: SustNMS Sustainability

When the two flows are bigger than the maximum load the network can handle, for instance, 24 Mbps each, SustNMS Performance is the best option, with additional switches. Note that the node 14 is the bottleneck. The devices can handle 30 Mbps, but the node 14 is in the two paths, and so it has to handle 48 Mbps. In this case, SOS selected to apply SustNMS Performance, and two extra switches were turned on, which means that, for this case, considering the Utility Function, it is better to save less in order not to lose to many packets. Figure 56 illustrates this case.



Refined Policies

['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

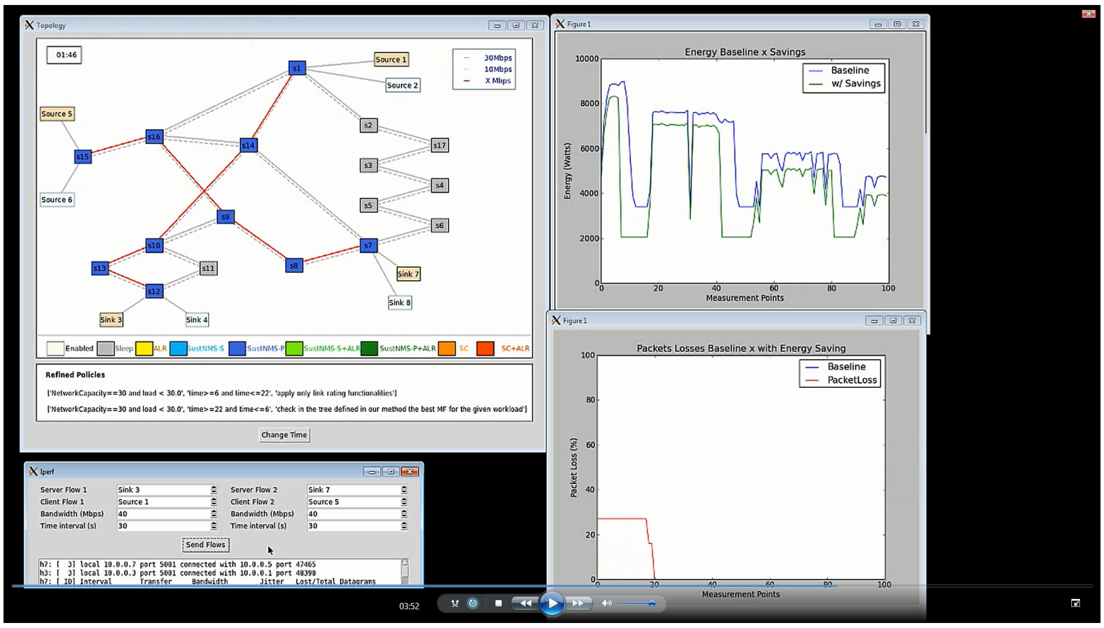
Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	24	Bandwidth (Mbps)	24
Time interval (s)	30	Time interval (s)	30

```
h7: [  3] local 10.0.0.7 port 5001 connected with 10.0.0.5 port 47465
h3: [  3] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 48398
h7: [ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
```

Figure 56: SOS selection for two 24 Mbps flows: SustNMS Performance, turning on two extra switches

To show the losses that happen in extreme cases, we then show what happens when two 40 Mbps flows are in place. Even turning on the extra switches, the losses are inevitable, as illustrated in Figure 57. The packet losses can be checked in the graph in the bottom right.

These decisions were for the night policy, which allows any capability to be applied (or combination of capabilities). In the topology window, in the top-left corner, it is possible to check a timer representing the hour. When it is day, another policy has to be enforced. And another decision tree will decide what to do in each workload. In the case illustrated in Figure 58, two 2 Mbps flows are in place and all nodes are with ALR applied, reducing the energy consumption. This happens because the policy for the day says that only ALR must be enforced, and ALR is allowed when the load is smaller than 10Mbps.



Refined Policies

['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

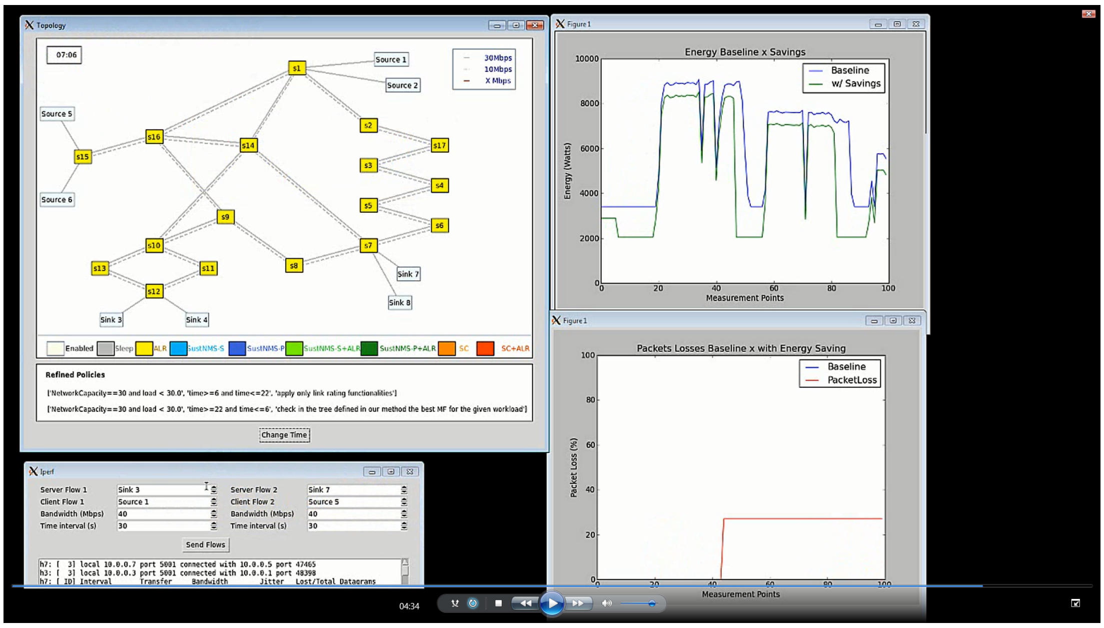
['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

Iperf

Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	40	Bandwidth (Mbps)	40
Time interval (s)	30	Time interval (s)	30

```
h7: [  3] local 10.0.0.7 port 5001 connected with 10.0.0.5 port 47465
h3: [  3] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 48398
h7: [ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
```

Figure 57: SOS selection for two 40 Mbps flows: SustNMS Performance, but losses were inevitable



Refined Policies

['NetworkCapacity==30 and load < 30.0', 'time>=6 and time<=22', 'apply only link rating functionalities']

['NetworkCapacity==30 and load < 30.0', 'time>=22 and time<=6', 'check in the tree defined in our method the best MF for the given workload']

Iperf

Server Flow 1	Sink 3	Server Flow 2	Sink 7
Client Flow 1	Source 1	Client Flow 2	Source 5
Bandwidth (Mbps)	40	Bandwidth (Mbps)	40
Time interval (s)	30	Time interval (s)	30

```
h7: [  3] local 10.0.0.7 port 5001 connected with 10.0.0.5 port 47465
h3: [  3] local 10.0.0.3 port 5001 connected with 10.0.0.1 port 48398
h7: [ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrans
```

Figure 58: SOS during the day: applying only ALR when the flows are smaller than 10 Mbps

6.7 Chapter Final Remarks

Using a Utility Function combining QoS and energy efficiency, in conjunction with Table 13 that presents additional energy efficiency capabilities information to determine the possible combinations, it was possible to orchestrate different capabilities, bringing more savings and ensuring a conflict-free operation. By using policy refinement, the method has information from the business level, being tied to the business strategies, and with the support of sustainability-oriented information models, the method supports energy efficiency capabilities enforcement and time dynamicity. This addresses the primary objective of this work, to present a method that overcomes the challenges in orchestrating energy efficiency capabilities considering sustainability-oriented policies refinement.

Recently, different approaches to improve the infrastructures energy efficiency have been proposed. However, such energy management capabilities were designed to operate autonomously and independently from each other. Besides operating the network in a conflict-free manner, the orchestration of energy efficiency capabilities performed by the SOS method allows to combine more than one capability at the same time in the nodes. Such coordination leads to greater savings than enabling just one capability, as demonstrated by the experiments. Besides, the method can choose which capability (-ties) is (are) the best for a given scenario, thus turning on the one that has the best results. Before SOS, there was no available method that could orchestrate or coordinate energy efficiency capabilities, or consider business directives in such operation in an automated way.

The existing refinement methods focus on translating policies from a high to a lower level of abstraction, supporting policy analysis, resources discovery, and dynamicity. Despite the possibility of extending them in order to comprise energy efficiency capabilities translation, such methods do not perform energy efficiency capabilities orchestration. That is, they are neither able to combine more than one capability to save more energy, nor to ensure a conflict-free operation. They are not able to do such orchestration for other domains either, such as QoS or security, which can be one of the SOS method extensions for deployment in production networks, with more than one capability type.

The proposed method was validated through different experiments, testing the Utility Function, checking the extra savings when combining more than one capability, the decision tree interpolation and dynamicity aspects.

7 FINAL CONSIDERATIONS

Energy efficiency capabilities help operators and service providers to reduce operational costs and GHG emissions in their ICT infrastructures. Sustainability-oriented network management policies can help by bringing business directives into the network, turning the management more automated and less error prone, also reducing operational costs. In this regard, this work proposed a method able to orchestrate energy efficiency capabilities considering sustainability-oriented policies refinement, enabling a more energy efficient and automated infrastructure.

To the best of the author's knowledge, this is the first work that comprises the complete refinement of such policies including the orchestration of energy efficiency capabilities. The method has the following advantages:

- Coordination of energy efficiency capabilities allowing the operator to optimize the energy consumption.
- Besides the possibility of saving more energy, the orchestration ensures a conflict-free operation. A conflicting operation could lead to undesired behavior, failures, and, consequently, reduced quality of service. Besides, applying a capability not suited to the current bandwidth utilization value might lead to congestion or packet loss.
- Business-level directives, refined down to the device and instance policy levels, in an automated way, bring high-level goals to the network operation. Such automation turns the management task easier, less manual, and less prone to errors.

7.1 Result Analysis

The method implemented in the Sustainability-Oriented System (SOS) is able to orchestrate energy efficiency capabilities considering business directives. The proposed Utility Function was validated comparing its results with the Analytical Solvers results for savings and packet losses. In this case, it was possible to check that the result is affected by the losses and, in the exemplified case, even the capability that brings more savings, ended up losing to the other that saves less, but loses no packets. It is important to note that the proposed Utility Function also ensures connectivity: in a case in which the losses are greater than 0.5%, the Utility Function will result in zero. Therefore, there will not be a case in which all equipment is turned off to maximize the savings.

To check the capabilities combination, SustNMS was combined with ALR, and more savings were achieved, both in a load proportional scenario (corresponding to more modern equipment) and in a not load proportional scenario (corresponding to legacy equipment), except in the case with a heavy load in the network. Energy efficiency capabilities are motivated by the idle periods in the network. Therefore, it is expected they will not present significant savings in high load scenarios.

To validate the use of the decision trees, this work presented the results expected from the Utility Function module and tested the decisions for different bandwidth usages, showing that the expected capabilities were selected for the expected intervals after the interpolation performed to deploy the Final Decision Tree with Interpolation (FDTI). To conclude, the dynamicity aspects were demonstrated, showing that the system behaves differently considering the time period.

7.2 Applicability Analysis

In this Section, the SOS applicability in production scenarios is analyzed regarding the equipment readiness for green capabilities and a standard for green networking capabilities.

7.2.1 Enforcing Green Capabilities in Production Networks

To work on production networks, the enforcement of the capabilities selected by SOS depends on the underlying equipment, as in any green capabilities enforcement case. Many devices are already able to enter standby modes, or scale down the working speed depending on the current utilization, thus saving energy. However, the management of these devices are not always sustainability-oriented (BOLLA et al., 2011). Some proposals deal with this drawback, such as the network scope ones detailed in Chapter 2. Building a network infrastructure able to support sustainability features is key to achieve the benefits of energy efficiency features (COSTA et al., 2012).

7.2.2 Mapping to the GAL Standard

In order to fully support heterogeneous devices, considering the complete hierarchical structure inside each node (all components of an equipment), SOS could be written as a GAL application, the Green Abstraction Layer standard from ETSI (European Telecommunications Standards Institute). GAL "provides a way to expose green networking capabilities of devices toward the network control plane" (BOLLA et al., 2014). The network scope capabilities often treat a network device as node in a graph, whose edges represent the links between nodes. According to the authors, the hierarchical structure of a node supported by GAL would overcome such gap, combining local and network control loops and optimizing the device energy consumption. Besides, the abstraction layer would make it easier to manage a heterogeneous environment. GAL-enabled applications "can be developed easily and installed in telecom management systems to manage the energy efficiency of network devices".

GAL supports discovery of available energy efficiency capabilities and devices, provisioning energy efficiency capabilities and monitoring devices and metrics. GAL models each possible state as an Energy-Aware State (EAS), combining sleeping and power scaling actions. The possible groups of states are: standby; maximum performance and power consumption; power scaling; and power scaling and standby. Among the standby possible states are the fully active and the fully off. The fully active can have *n* power scaling sub-states, from maximum to minimum performance (and maximum savings). The existing standards EMAN (Energy Management) and ACPI (Advanced Configuration and Power Interface), besides representing energy-aware states as GAL does, are predefined and non-extensible.

Each EAS must have a set of associated attributes, divided in three categories: the obligatory power consumption, also obligatory network performance, and the optional transition features, such as the time intervals needed to move from one state to another. The obligatory attributes are similar to the SOS ones: the Analytical Solvers provide information about the expected savings for each capability and about the network performance represented by packet losses, while GAL uses maximum throughput. The transition information would be an extension for SOS to be able to consider it in the decision. As it is now, only the policies comprise such concern, for instance, allowing only ALR (and not sleeping capabilities) to operate during high load periods (as "during the day" in the experiments), in accordance to some studies that indicate that rate adaption is better under high loads, as presented by Nedevschi et al. (NEDEVSCHI et al., 2008).

As indicated in the GAL standard workflow (ETSI, 2014), SOS could be integrated as follows:

- A Network Control Policy (NCP), able to understand energy-aware metrics, acquires the current and available EASes from all nodes;
- Considering the available traffic and network performance constraints, SOS would use its Decision Trees to define a new network configuration. The capabilities chosen would then be applied to update the network parameters and the parameters to be set in each node. To deploy the Decision Trees, the SOS should be able to understand the possible EASs combinations. They can be represented in conjunction with the capabilities. For instance, SC has two different EASes, on and sleeping, and its own Local Control Policy (LCP) to control bursts, and ALR has two or more EASes related to link rating that could be combined with SC "on" mode;
- The NCP and the LCP interact to ensure both are working in the same way;
- The LCP iterates down to the bottom-level LCP to apply the request;
- The lower-level LCPs configure the EASes of each component.

7.3 Main Contributions

The main contributions of this work are:

- The definition of sustainability-oriented policies;
- The identification of the requirements that a policy refinement process should address in order to tackle energy efficiency;
- A method to orchestrate energy efficiency capabilities, supported by a Utility Function that combines sustainability-oriented and performance aspects, able to

choose the best capability (or a combination of capabilities) for a given scenario, ensuring the adequate quality of service; and

The additional contributions of this work are:

- An analysis of the existing refinement methods in light of the elicited requirements;
- An approach to refine sustainability-oriented policies from a business level down to the network level;
- The development of a prototype intended as a proof-of-concept of the proposed method in an emulated environment.

7.4 Future Work

As future work, this thesis lets the adaption of requirements (ii), determination of the resources that are required for the policy execution; (iii) validation and verification if the refined policy is in accordance with the high-level policy; and (iv), policy conflicts detection and resolution, as mentioned in Chapter 4. This work focused on the requirements that change more with sustainability-oriented policies: (i) translation of high-level policies into enforceable policies; (v) addressing policies dynamicity; (vi) ways to express sustainability-oriented policies in a standardized fashion; and (vii) energy efficiency capabilities orchestration.

As another limitation, which is also let as future work, dynamicity was partially addressed. Two dynamicity issues related to the Policy Continuum Network Level were described: time, considered in the Information Models; and change in scenario (when a node migrates, for instance). The work from Monsanto et al. (MONSANTO et al., 2013) addresses part of the second requirement. The authors suggest using parameterized static policies that can be recomputed as the network environment changes. This is similar to what was proposed in the patent US7617304 (DEVARAKONDA; HERGER, 2009). In case of a change in the scenario, a policy to change a parameterized policy can be further developed. E.g., " *if networkNew different from networkOld, x = 5*". After a node migration, " *change x=3*". This means there is a need to consider parameterized policies, as well as a “master” policy that changes policies. This master policy should be implemented as a supervisor module, able to keep checking the aforementioned condition to change a policy.

As another possible SOS future work, one could examine ways of reducing the number of Decision Trees and the allowed capabilities combination (leaves in the trees) that need to be generated for SOS through different types of optimization strategies. SOS could also be tested against larger scenarios to go further in the scalability check. Or could be implemented in production scenarios, such as the ones presented in Chapter 3.

Another possible future work is the evaluation of the applicability of the SOS method in different environments, such as in wireless and photonics networks, considering topology changes and error rates. The capabilities combination table (Table 13) is able to support different capabilities and would be one of the parts of the method that should consider the new capabilities. Also important to analyze is the application of SOS in environments owned by different providers. This should start by defining contracts among the different players.

From the applicability analysis, some other future works can be envisioned. SOS and GAL could be adapted to work together, considering a heterogeneous environment and Energy Aware States (EAS) for capabilities combination.

Three possible extensions to SOS are envisioned: support more granular information from the energy efficiency capabilities; support an expanded refinement, comprising other types of capabilities; and support cloud infrastructures, including compute and storage infrastructures.

7.4.1 More Granular Information from Capabilities

Regarding more granular information, policies could benefit from the ability to control the behavior details of the capabilities in the network or in the device levels of the Policy Continuum. This could be achieved by configuring each capability parameter according to the goals defined in high-level policies.

For this purpose, the translation process should identify some specific parameters such as the maximum losses allowed or the minimum savings expected in the network. The method should then be able to configure the parameters of the green capabilities that best match the requirements or even eliminate the capabilities that do not match the losses or savings goals. For instance, SC with DutyCycle 10% or SC with DutyCycle 50% have different savings and losses results. The Analytical Solver could test both and choose not only applying SC, but applying SC with DutyCycle 10% and discarding SC with DutyCycle 50% because it saves less.

7.4.2 Expanded Refinement to Comprise QoS and Access Control in Conjunction with Energy Efficiency

The SOS method can be extended to support an expanded refinement, including orchestration of QoS and access control capabilities and additional metrics. The Analytical Solvers for energy efficiency capabilities could be extended to consider latency, ensuring it will not exceed a defined threshold, or ensure a minimum allocated bandwidth. Such expansion would bring more complex calculations, but could give an even better answer for the question about which capability (or combination of capabilities) is the best for a given network situation.

7.4.3 SOS and Network Functions Virtualization

Going further, SOS could be expanded to be applied in other scenarios, such as for NFV (Network Functions Virtualization), to orchestrate energy efficiency capabilities not only for networks, but also for other parts, such as compute and storage.

A good NFV use case was presented in the OpenStack Atlanta Summit in May, 2014: a NFV customer submits a backup request. The NFV provider returns back with information about the time to schedule the job. In parallel, the provider starts triggering events to ensure the SLA and also to save energy when possible, e.g., powering down the servers after the job is done (UDUPI; DUTTA; KRISHNAN, 2014).

The authors present the Smart Scheduler in OpenStack, a smart resource placement solution that takes into account the constraints involved, like a "universal decision making engine". SOS could be placed inside such engine to help achieving more energy savings, by saying which capability is the best for each part of the infrastructure considering the service constraints. The SOS method could consider the other infrastructure parts in two ways:

- As a series system, with one independent Utility Function for each part, plus a priority. For instance, compute has priority 1, storage, 2, and network, 3, meaning that the network should consider the other decisions in order to choose its own actions, as represented in Figure 59;
- As one complex Utility Function that comprises all necessary decisions, as represented in Figure 60.

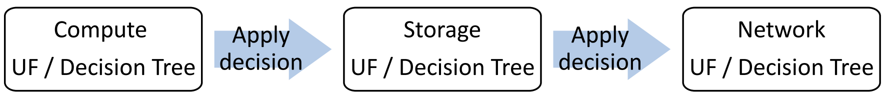


Figure 59: Support to other infrastructures as a series system

Considering the first, simpler case, the decision made for compute and storage

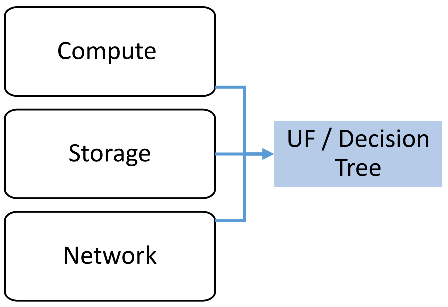


Figure 60: Support to other infrastructures as one complex Utility Function and Decision Tree

will be treated as inputs for the network decision to be made. Figure 61 represents in a generic way the available capabilities, and the possible combinations among them.

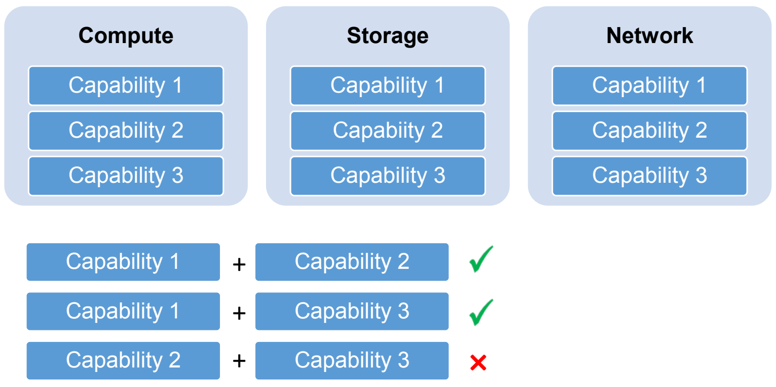


Figure 61: Compute, Storage and Network generic capabilities and allowed combinations

Examples of capabilities are the OpenStack Neat for dynamic virtual machines consolidation (BELOGLAZOV; BUYYA, 2014), ACPI for CPU or hard drives, or SSD caching for storage (CABRERA et al., 2013).

Each part has its own Utility Function, which will consider some random utilization values in conjunction with the capabilities combination expected savings and performance reduction, if any. Each part will then have a decision tree deployed, as exemplified in Figure 62.

Over each Decision Tree, the Scikit tool should be used to result in the Final Decision Tree with Interpolation for each part of the infrastructure. While operating, when a change in the utilization occurs, the series system will check the decision for compute, then for storage, and then for network.

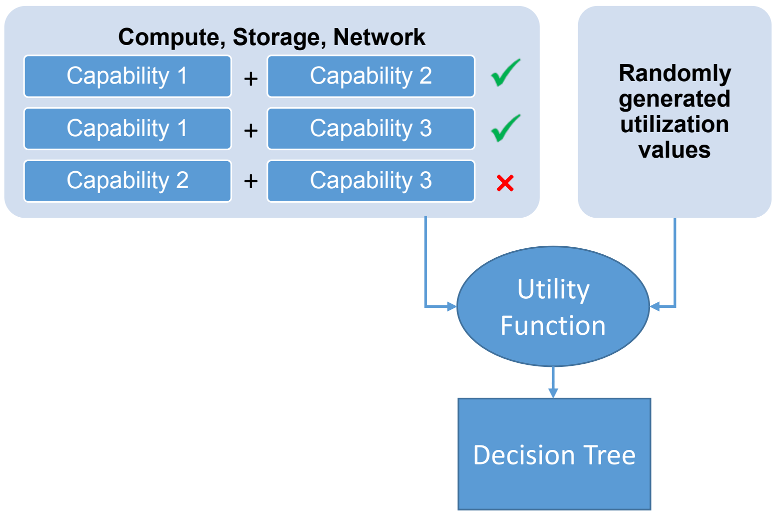

Compute, Storage, Network
Capability 1 + Capability 2 ✓
Capability 1 + Capability 3 ✓
Capability 2 + Capability 3 ✗
Randomly generated utilization values
Utility Function
Decision Tree

Figure 62: Compute, Storage and Network will have different Decision Trees

REFERENCES

ABTS, D. et al. Energy Proportional Datacenter Networks. In: *Proceedings of the 37th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2010. (ISCA '10), p. 338–347. ISBN 978-1-4503-0053-7.

AGRAWAL, D. et al. Policy management for networked systems and applications. In: *Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International Symposium on*. Nice, France: IEEE Computer Society, 2005. p. 455–468.

AKRAM, H.; PASCAL, B.; THIERRY, G. Controlled Stochastic Petri Net Model for End-to-end Network QoS Provisioning in Middleware-based Multimedia and Real-time Systems. In: *Proceedings of the 44th Annual Simulation Symposium*. San Diego, CA, USA: Society for Computer Simulation International, 2011. (ANSS '11), p. 111–118. ISBN 1-930638-56-6.

BALLAND, P.; HINRICHS, T. *Congress: a System for Declaring, Auditing, and Enforcing Policy in Heterogeneous Cloud Environments*. 2014. Available at: <https://www.openstack.org/summit/openstack-summit-atlanta-2014/session-videos/presentation/congress-a-system-for-declaring-auditing-and-enforcing-policy-in-heterogeneous-cloud-environments>. Accessed in: 08 mar. 2015.

BANDARA, A. K.; LUPU, E. C.; RUSSO, A. Using event calculus to formalise policy specification and analysis. In: *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*. Lake Como, Italy: IEEE Computer Society, 2003. p. 26–39.

BARACHI, M. E. et al. A multi-service multi-role integrated information model for dynamic resource discovery in virtual networks. In: *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. Shanghai, China: IEEE Computer Society, 2013. p. 4777–4782.

BARROSO, L.; HOLZLE, U. The case for energy-proportional computing. *Computer*, v. 40, n. 12, p. 33–37, Dec 2007. ISSN 0018-9162.

BEIGI, M. S.; CALO, S.; VERMA, D. Policy transformation techniques in policy-based systems management. In: *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*. Yorktown Heights, New York: IEEE Computer Society, 2004. p. 13–22.

BELLER, A.; JAMHOUR, E.; PELLENZ, M. Defining Reusable Business-Level QoS Policies for DiffServ. In: SAHAI, A.; WU, F. (Ed.). *Utility Computing*. Berlin, Germany: Springer Berlin Heidelberg, 2004, (Lecture Notes in Computer Science, v. 3278). p. 40–51. ISBN 978-3-540-23631-3.

BELOGLAZOV, A.; BUYYA, R. OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds. *Concurrency and Computation: Practice and Experience*, June 2014. ISSN 1532-0634. Available at: <http://onlinelibrary.wiley.com/doi/10.1002/cpe.3314/abstract>.

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.; KHAN, S.; ZOMAYA, A. Green data center networks: Challenges and opportunities. In: *Frontiers of Information Technology (FIT), 2013 11th International Conference on*. Islamabad, Pakistan: IEEE Computer Society, 2013. p. 229–234.

BOLLA, R. et al. Enabling backbone networks to sleep. *Network, IEEE*, v. 25, n. 2, p. 26–31, March 2011. ISSN 0890-8044.

BOLLA, R.; BRUSCHI, R.; DAVOLI, F. Energy-Aware Resource Adaptation for Next-Generation Network Equipment. In: *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. Hawaii, USA: IEEE Computer Society, 2009. p. 1–6. ISSN 1930-529X.

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.

____. A northbound interface for power management in next generation network devices. *Communications Magazine, IEEE*, IEEE Computer Society, v. 52, n. 1, p. 149–157, January 2014. ISSN 0163-6804.

BOLLA, R.; BRUSCHI, R.; RANIERI, A. Performance and power consumption modeling for green COTS software router. In: *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*. Bangalore, India: IEEE Computer Society, 2009. p. 1–8.

BOUTABA, R.; AIB, I. Policy-based Management: A Historical Perspective. *J. Netw. Syst. Manage.*, Plenum Press, New York, NY, USA, v. 15, n. 4, p. 447–480, Dec. 2007. ISSN 1064-7570.

BRADSHAW, J. et al. The KAoS policy services framework. In: *Eighth Cyber Security and Information Intelligence Research Workshop (CSIIRW 2013)*. Oak Ridge, TN, USA: ACM, 2013.

BRADSHAW, J. M.; USZOK, A.; MONTANARI, R. Policy-Based Governance of Complex Distributed Systems: What Past Trends Can Teach Us about Future Requirements. In: ____. *Adaptive, Dynamic, and Resilient Systems*. Boca Raton, FL, USA: CRC Press, 2014. p. 259–283.

CABRERA, J. et al. Energy-Efficient Cloud Storage using Solid-State Drive Caching. In: *11th USENIX Conference on File and Storage Technologies, 2013. FAST'13*. San Jose, California: UNENIX, 2013. Available at:

<https://www.usenix.org/system/files/fastpw13-final30.pdf>. Accessed in: 01 mar. 2015.

CARVALHO, T. C. M. B. et al. Towards Sustainable Networks - Energy Efficiency Policy from Business to Device Instance Levels. In: *ICEIS*. Wroclaw, Poland: SciTePress, 2012. p. 238–243.

CHARALAMBIDES, M. et al. Dynamic Policy Analysis and Conflict Resolution for DiffServ Quality of Service Management. In: *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*. Vancouver, Canada: IEEE Computer Society, 2006. p. 294–304. ISSN 1542-1201.

CHO, K. *WIDE-TRANSIT 150 Megabit Ethernet Trace 2008-03-18 (Anonymized)*. Otemachi, Tokyo, Japan: IMDC, 2008. Available at: <http://imdc.datcat.org/collection/1-05L8-9=WIDE-TRANSIT-150-Megabit-Ethernet-Trace-2008-03-18-Anonymized->. Accessed in: 20 mar. 2015.

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 OSPF-Integrated Routing Strategy for QoS-Aware Energy Saving in IP Backbone Networks. *Network and Service Management, IEEE Transactions on*, v. 9, n. 3, p. 254–267, September 2012. ISSN 1932-4537.

COOK, G. et al. *Clicking Clean: How Companies are Creating the Green Internet*. Washington, D.C., USA, 2014. Available at: <http://www.greenpeace.org/usa/Global/usa/planet3/PDFs/clickingclean.pdf>.

COSTA, C. H. et al. SustNMS: Towards service oriented policy-based network management for energy-efficiency. In: *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*. Pisa, Italy: IEEE Computer Society, 2012. p. 1–5.

CRAVEN, R. et al. Policy refinement: decomposition and operationalization for dynamic domains. In: *Network and Service Management (CNSM), 2011 7th International Conference on*. Paris, France: IEEE Computer Society, 2011. p. 1–9.

DAMIANOU, N. *A Policy Framework for Management of Distributed Systems*. Thesis (PhD) — Imperial College London, 2002. Available at: <http://pubs.doc.ic.ac.uk/policy-framework-distrib-systems/>.

DAMIANOU, N. et al. The ponder policy specification language. In: *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*. London, UK, UK: Springer-Verlag, 2001. (POLICY ’01), p. 18–38. ISBN 3-540-41610-2. Available at: <http://dl.acm.org/citation.cfm?id=646962.712108>.

DARIMONT, R.; LAMSWEERDE, A. van. Formal Refinement Patterns for Goal-driven Requirements Elaboration. In: *Proceedings of the 4th ACM SIGSOFT Symposium on Foundations of Software Engineering*. New York, NY, USA: ACM, 1996. (SIGSOFT ’96), p. 179–190. ISBN 0-89791-797-9. Available at: <http://doi.acm.org/10.1145/239098.239131>.

DAVY, S.; JENNINGS, B.; STRASSNER, J. The Policy continuum-Policy Authoring and Conflict Analysis. *Comput. Commun.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 31, n. 13, p. 2981–2995, Aug. 2008. ISSN 0140-3664.

DEVARAKONDA, M.; HERGER, L. *Method, apparatus, computer program product and web-enabled service providing dynamically adjustable policies*. Google Patents, 2009. US Patent 7,617,304. Available at: <https://www.google.com/patents/US7617304>.

DMTF. *CIM | DMTF Standard*. n.d. Available at: <http://www.dmtf.org/standards/cim>. Accessed in: 22 feb. 2015.

DURHAM, D. et al. *fThe COPS (Common Open Policy Service) Protocol*. 2000. Available at: <http://tools.ietf.org/html/rfc2748>. Accessed in: 19 oct. 2014.

EMERSON-ELECTRIC. *Energy Logic: Reducing Data Center Energy Consumption*. 2009. Available at: <http://www.cisco.com/web/partners/downloads/765/other/Energy_Logic_Reducing_Data_Center_Energy_Consumption.pdf>.

ERICSSON. *Ericsson Energy and Carbon Report - On the Impact of the Networked Society*. Stockholm, Sweden, 2013. Available at: <http://www.ericsson.com/res/docs/2013/ericsson-energy-and-carbon-report.pdf>.

ETSI. *Environmental Engineering (EE): Green Abstraction Layer (GAL); Power management capabilities of the future energy telecommunication fixed network nodes*. Sophia Antipolis Cedex, France, 2014. Available at: <http://www.etsi.org/deliver/etsi_es/203200_203299/203237/01.01.01_60/es_203237v010101p.pdf>.

FOSTER, N. et al. Languages for software-defined networks. *Communications Magazine, IEEE*, v. 51, n. 2, p. 128–134, February 2013. ISSN 0163-6804.

GARG, S. K.; BUYYA, R. Green cloud computing and environmental sustainability. *Harnessing Green IT: Principles and Practices*, p. 315–340, 2012.

GeSI. *SMARTer 2020: The Role of ICT in Driving a Sustainable Future*. Brussels, Belgium, 2012. Available at: <http://gesi.org/SMARTer2020>. Accessed in: 19 jun. 2014.

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.

HELLER, B. et al. Elastictree: Saving energy in data center networks. In: *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2010. (NSDI'10), p. 17–17. Available at: <http://dl.acm.org/citation.cfm?id=1855711.1855728>.

HU, J.; FU, Y. Research on Policy Refinement Method. In: *Cyberworlds, 2008 International Conference on*. Hangzhou, China: IEEE Computer Society, 2008. p. 800–804.

IHMC. *KAoS project*. n.d. Available at: <http://ontology.ihmc.us/kaos.html>. Accessed in: 18 feb. 2014.

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

JUNG, G. et al. Generating Adaptation Policies for Multi-tier Applications in Consolidated Server Environments. In: *Proceedings of the 2008 International Conference on Autonomic Computing*. Washington, DC, USA: IEEE Computer Society, 2008. (ICAC '08), p. 23–32. ISBN 978-0-7695-3175-5.

KACHRIS, C.; TOMKOS, I. Power consumption evaluation of all-optical data center networks. *Cluster Computing*, v. 16, n. 3, p. 611–623, 2013. ISSN 1386-7857.

KAGAL, L. *Rei: A Policy Language for the Me-Centric Project*. Palo Alto, CA, USA, 2002. Available at: <http://www.hpl.hp.com/techreports/2002/HPL-2002-270.pdf>.

KAHLOUL, L. et al. Modeling and verification of rbac security policies using colored petri nets and cpn-tool. In: ZAVORAL, F. et al. (Ed.). *Networked Digital Technologies*. Berlin, Germany: Springer Berlin Heidelberg, 2010, (Communications in Computer and Information Science, v. 88). p. 604–618. ISBN 978-3-642-14305-2.

KEPHART, J. Research challenges of autonomic computing. In: *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*. St. Louis, Missouri, USA: IEEE Computer Society, 2005. p. 15–22.

KLIAZOVICH, D.; BOUVRY, P.; KHAN, S. DENS: Data center energy-efficient network-aware scheduling. In: *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*. Hangzhou, China: IEEE Computer Society, 2010. p. 69–75.

KLINGERT, S.; SCHULZE, T.; BUNSE, C. GreenSLAs for the energy-efficient management of data centres. In: *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking*. New York, NY, USA: ACM, 2011. (e-Energy '11), p. 21–30. ISBN 978-1-4503-1313-1.

KOCH, T.; KRELL, C.; KRAEMER, B. Policy definition language for automated management of distributed systems. In: *Proceedings of the 2Nd IEEE International Workshop on Systems Management (SMW'96)*. Washington, DC, USA: IEEE Computer Society, 1996. (SMW '96), p. 55–. ISBN 0-8186-7442-3. Available at: <http://dl.acm.org/citation.cfm?id=829511.830381>.

KOUTITAS, G.; TASSIULAS, L.; VLAHAVAS, I. *Energy Efficiency Monitoring in Data Centers: Case Study at International Hellenic University*. 2012. UTH and

IHU (CI)-WP4. Available at: <http://www.fp7-trend.eu/system/files/content-public/248-trend-friends-workshop-gent-14-february-2012-presentations/wp4energyefficiencymonitoringinsmallscaledatcentersuthkoutitas.pdf>.

LAMBERT, S. et al. Worldwide electricity consumption of communication networks. *Opt. Express*, OSA, v. 20, n. 26, p. B513–B524, Dec 2012.

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. Available at: <http://doi.acm.org/10.1145/1868447.1868466>.

LIAO, B.-S.; GAO, J. An automatic policy refinement mechanism for policy-driven grid service systems. In: *Grid and Cooperative Computing-GCC 2005*. Beijing, China: Springer, 2005. p. 166–171.

LOBO, J.; BHATIA, R.; NAQVI, S. A Policy Description Language. In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1999. (AAAI '99/IAAI '99), p. 291–298. ISBN 0-262-51106-1. Available at: <http://dl.acm.org/citation.cfm?id=315149.315308>.

LOBO, J. et al. Logic programming, knowledge representation, and nonmonotonic reasoning. In: BALDUCCINI, M.; SON, T. C. (Ed.). Berlin, Heidelberg: Springer-Verlag, 2011. p. 280–299. ISBN 978-3-642-20831-7. Available at: <http://dl.acm.org/citation.cfm?id=2001078.2001097>.

LOPEZ, D.; KRISHNAN, R. R.; FIGUEIRA, N. *Policy Architecture and Framework for NFV Infrastructures*. 2015. Available at: <http://tools.ietf.org/html/draft-norival-nfvrg-nfv-policy-arch-02>. Accessed in: 08 mar. 2015.

LYMBEROPOULOS, L.; LUPU, E.; SLOMAN, M. An adaptive policy based framework for network services management. In: *Journal of Network and Systems Management*. NY, USA: Plenum Press, 2003. p. 277–303.

MACKENZIE, A.; WICKER, S. Game theory in communications: motivation, explanation, and application to power control. In: *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*. [S.l.: s.n.], 2001. v. 2, p. 821–826 vol.2.

MALMODIN, J. et al. Greenhouse gas emissions and operational electricity use in the ict and entertainment & media sectors. *Journal of Industrial Ecology*, Blackwell Publishing Inc, v. 14, n. 5, p. 770–790, 2010. ISSN 1530-9290.

MANRAL, V. *Benchmarking Power usage of networking devices*. USA, 2010.

MAULLO, M.; CALO, S. Policy Management: An Architecture and Approach. In: *Systems Management, 1993, Proceedings of the IEEE First International Workshop on*. Los Angeles, CA, USA: IEEE Computer Society, 1993. p. 13–26.

MEER, S. Van der et al. Autonomic networking: Prototype implementation of the policy continuum. In: *Broadband Convergence Networks, 2006. BcN 2006. The 1st International Workshop on*. Vancouver, Canada: IEEE Computer Society, 2006. p. 1–10.

MOFFETT, J. D.; SLOMAN, M. S. Policy hierarchies for distributed systems management. *Selected Areas in Communications, IEEE Journal on*, IEEE, v. 11, n. 9, p. 1404–1414, 1993.

MONSANTO, C. et al. Composing Software-defined Networks. In: *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2013. (nsdi'13), p. 1–14. Available at: <http://dl.acm.org/citation.cfm?id=2482626.2482629>.

MOORE, B. (Ed.). *Policy Core Information Model (PCIM) Extensios*. Fremont, CA, USA, 2001. Published: Internet Requests for Comments. Available at: <http://www.rfc-editor.org/rfc/rfc3060.txt>.

MOORE, B. *RFC3640 - Policy Core Information Model (PCIM) Extensions*. United States, 2003. Available at: <http://www.ietf.org/rfc/rfc3460.txt>.

MOSTOWFI, M.; CHRISTENSEN, K. Saving energy in LAN switches: New methods of packet coalescing for Energy Efficient Ethernet. In: *Green Computing Conference and Workshops (IGCC), 2011 International*. Orlando, FL, USA: IEEE Computer Society, 2011. p. 1–8.

NASCIMENTO, V. T. et al. Sustainability Information Models for Energy Efficiency Policies. Under Development. 2015.

NEDEVSCHI, S. et al. Reducing Network Energy Consumption via Sleeping and Rate-adaptation. In: *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2008. (NSDI'08), p. 323–336. ISBN 111-999-5555-22-1. Available at: <http://dl.acm.org/citation.cfm?id=1387589.1387612>.

NGUYEN, K.-K. et al. Powering a data center network via renewable energy: A green testbed. *Internet Computing, IEEE*, v. 17, n. 1, p. 40–49, Jan 2013. ISSN 1089-7801.

OPENSTACK. *OpenStack Congress*. n.d. Available at: <https://wiki.openstack.org/wiki/Congress>. Accessed in: 08 mar. 2015.

____. *OpenStack Open Source Cloud Computing Software*. n.d. Available at: <https://www.openstack.org/>. Accessed in: 08 mar. 2015.

OZEL, O.; UYSAL-BIYIKOGLU, E. Network-wide energy efficiency in wireless networks with multiple access points. *Transactions on Emerging Telecommunications Technologies*, v. 24, n. 6, p. 568–581, 2013. ISSN 2161-3915.

PEDIADITAKIS, D.; ROTSOS, C.; MOORE, A. W. Faithful Reproduction of Network Experiments. In: *Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. New York, NY, USA:

ACM, 2014. (ANCS ’14), p. 41–52. ISBN 978-1-4503-2839-5. Available at: <http://doi.acm.org/10.1145/2658260.2658274>.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, JMLR. org, v. 12, p. 2825–2830, 2011.

PERRY, M.; BAUER, M. Policies, Rules and Their Engines: What do They Mean for SLAs? In: NEGOITA, M. G.; HOWLETT, R. J.; JAIN, L. C. (Ed.). *Knowledge-Based Intelligent Information and Engineering Systems*. Springer Berlin Heidelberg, 2004, (Lecture Notes in Computer Science, 3213). p. 1164–1170. ISBN 978-3-540-23318-3, 978-3-540-30132-5. Available at: <http://link.springer.com/chapter/10.1007/978-3-540-30132-5_158>.

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. Available at: <http://dl.acm.org/citation.cfm?id=2039912.2039944>.

RIEKSTIN, A. C. et al. No more electrical infrastructure: Towards fuel cell powered data centers. In: *Proceedings of the Workshop on Power-Aware Computing and Systems*. New York, NY, USA: ACM, 2013. (HotPower ’13), p. 5:1–5:5. ISBN 978-1-4503-2458-8. Available at: <http://doi.acm.org/10.1145/2525526.2525853>.

____. 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.

____. Orchestration of energy efficiency capabilities in networks. *Journal of Network and Computer Applications*, p. –, 2015. ISSN 1084-8045. Available at: <http://www.sciencedirect.com/science/article/pii/S1084804515001435>.

____. 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.

____. 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.

RNP - Rede Nacional de Ensino e Pesquisa. *Traffic | RNP*. 2015. Available at: <http://www.rnp.br/en/services/connectivity/traffic>. Accessed in: 17 mar. 2015.

RODRIGUES, B. 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.

RUBIO-LOYOLA, J. *Towards the Policy Refinement Problem in Policy-based Management Systems: A synthesis study*. VDM Publishing, 2008. ISBN 9783836486934. Available at: <http://books.google.com/books?id=hO0HNQAACAAJ>.

RUBIO-LOYOLA, J. et al. A methodological approach toward the refinement problem in policy-based management systems. *Communications Magazine, IEEE*, IEEE, v. 44, n. 10, p. 60–68, 2006.

SCHLENK, R. et al. Taxonomy of dynamic power saving techniques in fixed broadband networks. In: *Photonic Networks, 14. 2013 ITG Symposium*. Leipzig, Germany: VDE, 2013. p. 1–8.

SCHVANEVELDT, R.; COHEN, T. Abductive reasoning and similarity: Some computational tools. In: IFENTHALER, D.; PIRNAY-DUMMER, P.; SEEL, N. M. (Ed.). *Computer-Based Diagnostics and Systematic Analysis of Knowledge*. Springer US, 2010. p. 189–211. ISBN 978-1-4419-5661-3. Available at: <http://dx.doi.org/10.1007/978-1-4419-5662-0_11>.

SCIKIT-LEARN. *Decision Trees*. n.d. Available at: <http://scikit-learn.org/stable/modules/tree.html>. Accessed in: 16 apr. 2015.

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.

SLOMAN, M. Policy Driven Management For Distributed Systems. *Journal of Network and Systems Management*, v. 2, p. 333–360, 1994.

SNIR, Y. et al. *Policy Quality of Service (QoS) Information Model*. 2003. Available at: <http://tools.ietf.org/html/rfc3644>. Accessed in: 08 apr. 2014.

SRIVASTAVA, V. et al. Using game theory to analyze wireless ad hoc networks. *Communications Surveys Tutorials, IEEE*, v. 7, n. 4, p. 46–56, Fourth 2005. ISSN 1553-877X.

STRASSNER, J. *Policy-Based Network Management: Solutions for the Next Generation*. 1. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003. ISBN 1558608591.

TWIDLE, K. et al. Ponder2-A Policy Environment for Autonomous Pervasive Systems. In: *Policies for Distributed Systems and Networks, 2008. POLICY 2008. IEEE Workshop on*. Palisades, NY: IEEE Computer Society, 2008. p. 245–246.

UDUPI, Y.; DUTTA, D.; KRISHNAN, R. *Increasing Network and Energy Efficiency via Optimized NFV Placement in Openstack Clouds*. 2014. Available at: <https://www.openstack.org/summit/openstack-summit-atlanta-2014/session-videos/presentation/increasing-network-and-energy-efficiency-via-optimized-nfv-placement-in-openstack-clouds>. Accessed in: 08 mar. 2015.

UDUPI, Y.; SAHAI, A.; SINGHAL, S. A classification-based approach to policy refinement. In: *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*. Munich, Germany: IEEE Computer Society, 2007. p. 785–788.

USZOK, A. et al. Toward a flexible ontology-based policy approach for network operations using the kaos framework. In: *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*. Baltimore, MD, USA: IEEE Computer Society, 2011. p. 1108–1114. ISSN 2155-7578.

____. New developments in ontology-based policy management: Increasing the practicality and comprehensiveness of kaos. In: *Policies for Distributed Systems and Networks, 2008. POLICY 2008. IEEE Workshop on*. Palisades, NY: IEEE Computer Society, 2008. p. 145–152.

VELDT, K. van der et al. Carbon-aware path provisioning for NRENs. In: *Green Computing Conference (IGCC), 2014 International*. Dallas, TX, USA: IEEE Computer Society, 2014. p. 1–7.

VERIZON. *2013 Corporate Responsibility Supplement*. New York, NY, USA, 2013. Available at: <http://responsibility.verizon.com/assets/docs/cr-report-supplement-2013.pdf>. Accessed in: 05 may 2014.

VERMA, D. C. *Policy-Based Networking: Architecture and Algorithms*. Thousand Oaks, CA, USA: New Riders Publishing, 2000. ISBN 1578702267.

VOELLMY, A. et al. Maple: Simplifying SDN Programming Using Algorithmic Policies. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 43, n. 4, p. 87–98, Aug. 2013. ISSN 0146-4833.

WAN, Z.; TAHA, W.; HUDAK, P. Event-driven frp. In: KRISHNAMURTHI, S.; RAMAKRISHNAN, C. (Ed.). *Practical Aspects of Declarative Languages*. Berlin, Germany: Springer Berlin Heidelberg, 2002, (Lecture Notes in Computer Science, v. 2257). p. 155–172. ISBN 978-3-540-43092-6. Available at: <http://dx.doi.org/10.1007/3-540-45587-6_11>.

WANG, X. et al. A Survey of Green Mobile Networks: Opportunities and Challenges. *Mob. Netw. Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 17, n. 1, p. 4–20, Feb. 2012. ISSN 1383-469X.

WATERS, G. et al. *Policy Framework Architecture*. 1999. Available at: <http://tools.ietf.org/html/draft-ietf-policy-arch-00>. Accessed in: 20 mar. 2014.

WBCSD; WRI. *The GHG Protocol for Project Accounting*. World Resources Institute, 2005. Available at: <http://ghgprotocol.org/files/ghgp/ghg_project_accounting.pdf>.

WESTERINEN, A. e. a. *Terminofdraflogy for Policy-Based Management*. United States, 2001. Available at: <https://www.ietf.org/rfc/rfc3198.txt>.

WIES, R. Using a classification of management policies for policy specification and policy transformation. In: *Proceedings of the Fourth International Symposium on Integrated Network Management IV*. London, UK, UK: Chapman & Hall, Ltd., 1995. p. 44–56. ISBN 0-412-71570-8. Available at: <http://dl.acm.org/citation.cfm?id=280136.280146>.

ZHANG, M. et al. GreenTE: Power-aware traffic engineering. In: *Network Protocols (ICNP), 2010 18th IEEE International Conference on*. Kyoto, Japan: IEEE Computer Society, 2010. p. 21–30. ISSN 1092-1648.