An Energy-Efficient Reservation Framework for Large-Scale Distributed Systems

Anne-Cécile Orgerie

N° d'ordre : 640 *N° attribué par la bibliothèque : ENSL640*

- École Normale Supérieure de Lyon -
Laboratoire de l'Informatique du Parallélisme

THÈSE

en vue d'obtenir le grade de

Docteur de l'Université de Lyon - École Normale Supérieure de Lyon

Spécialité : Informatique

au titre de l'École Doctorale Informatique et Mathématiques (ED 512)

présentée et soutenue publiquement le 27 septembre 2011 par

Mlle Anne-Cécile Orgerie

An Energy-Efficient Reservation Framework for Large-Scale Distributed Systems

Directeurs de thèse :	Mme.	Isabelle	GUÉRIN-LASSOUS	
	M.	Laurent	LEFÈVRE	
Après avis de :	M.	Pascal	BOUVRY	
	M.	Ken	CHRISTENSEN	
Devant la commission d'examen formé de :				
	M.	Pascal	BOUVRY	Membre/Rapporteur
	M.	Ken	CHRISTENSEN	Membre/Rapporteur
	Mme.	Isabelle	GUÉRIN-LASSOUS	Membre
	M.	Laurent	LEFÈVRE	Membre
	Mme.	Christine	MORIN	Membre
	M.	Chris	PHILLIPS	Membre

Remerciements

La thèse est une chance, saisis-la.

En tout premier lieu, je tiens à remercier Laurent et Isabelle de m'avoir donné cette chance. Pour avoir répondu à tous mes doodles et toutes mes innombrables questions, pour leurs encouragements, leur disponibilité, leur gentillesse, pour avoir été des modèles fiables et stimulants. J'espère avoir été à la hauteur de leurs attentes.

La thèse est un rêve, fais-en une réalité.

À toute thèse, son jury. Je remercie Ken Christensen et Pascal Bouvry d'avoir accepté d'être rapporteurs de ma thèse. Leur relecture attentive et leurs commentaires ont enrichi cette thèse.

Je remercie également Chris Phillips et Christine Morin d'avoir accepté d'être membres du jury. Leur expertise a permis de compléter harmonieusement cette équipe de volley-ball. Un grand merci à Frédéric Desprez qui s'est malheureusement retrouvé sur le banc de touche.

La thèse est un défi, fais-lui face.

Ces trois années de défi se sont bien déroulées grâce aux collègues qui ont bien voulu collaborer avec moi. Pour communiquer avec moi, ils ont dû endurer plusieurs langages que je pratique avec une relative habileté : anglais, perl, python, bash, C, php, html, css... En particulier, un grand merci à Marcos dont les conseils m'ont été très précieux et à Jean-Patrick et Dino qui ont courageusement accepté de travailler avec moi sur des développements logiciels.

La thèse est un devoir, accomplis-le.

Je tiens à remercier Isabelle, Olivier Glück, Anne Benoît, Florent Dupont, Fabien Rico et Guillaume Chelius pour m'avoir confié leurs étudiants et m'avoir dévoilé le métier d'enseignant sous un très beau jour. Merci aussi à Fanny, ma co-TDwoman de choc.

La thèse est un jeu, joue-le.

Pour avoir partagé le bureau GN1 Sud 341 et supporté mes posters de travers, merci à Jean-Christophe, Pierre-Solen et Mehdi.

La thèse est un mystère, perce-le.

Merci à tous les doctorants du LIP qui contribuent à faire du troisième étage et particulièrement du coin café un espace chaleureux où les questions métaphysiques sur la thèse ne trouvent certes aucune réponse, mais sont le sujet de débats animés et passionnés. Pour les thés et les tablettes de chocolat partagés, merci à Fanny, Matthias et Yves.

La thèse est une promesse, remplis-la.

Merci à tous les membres de l'équipe RESO : permanents (Isabelle, Laurent, Paulo, Jean-Patrick, Olivier et Thomas), ingénieurs (Olivier, Jean-Christophe et Matthieu), post-doctorant (Marcos) et doctorants dont certains m'ont supportée en mission (Fabienne, Ghilherme, Pierre-Solen, Sébastien, Romaric, Landry, Mehdi et Doreid). Tous contribuent à l'atmosphère sereine de cette équipe diversifiée et accueillante. Merci également aux "anciens" qui m'ont montré la voie : Dino et Narjess.

La thèse est un combat, accepte-le.

Je tiens à remercier la direction du laboratoire et en particulier Gilles Villard et Daniel Hirschkoff. Ils font du LIP en endroit où il fait bon vivre pour les doctorants et nous permettent de travailler dans de bonnes conditions.

Je remercie aussi la direction du département d'informatique et notamment Éric Fleury et Yves Robert. La foire aux cours du début d'année reste pour tous les moniteurs un grand moment de bataille collective et conviviale.

Un grand merci à Natacha Portier qui était reponsable des admissions lors de mon entrée à l'ENS de Lyon ; je m'y suis effectivement sentie "comme un poisson dans l'eau".

La thèse est une aventure, ose-la.

Pour m'avoir suivie dans toutes mes aventures avec patience, je remercie les assistantes du LIP (Sylvie, Caroline, Laetitia, Corinne, Sèverine, Evelyne, Damien et Marie). Merci également aux MI-LIP (Jean-Christophe, Serge et Dominique) toujours très réactifs, même à distance. Merci aux ingénieurs de Grid'5000 et notamment Olivier Mornard pour leur disponibilité.

La thèse est bonheur, mérite-le.

Merci à tous ceux qui ont pu faire le déplacement pour venir me soutenir le jour du grand oral ainsi qu'à ceux qui auraient voulu être là.

Enfin, je remercie mes parents Marie-Agnès et Jean-Luc, mes frères Paul et Philippe ainsi que toute ma famille. Ils m'ont soutenue et supportée pendant toutes les péripéties de cette aventure avec beaucoup de patience et d'affection.

La thèse est la thèse, défends-la.

Que mère Teresa me pardonne d'avoir détourné son magnifique hymne à la vie.

Résumé

Depuis quelques années, économiser l'énergie est devenu un enjeu majeur dans les technologies de l'information et de la communication (TIC). Celles-ci représentent en effet 2% des émissions de CO2 de la planète, soit autant que l'aviation.

Les systèmes distribués (grilles, clouds, réseaux haute performance) constituent de gros consommateurs d'électricité. En effet, pour des besoins de haute disponibilité, leurs ressources sont allumées en permanence et notamment lorsqu'elles ne sont pas utilisées.

Les systèmes de réservation garantissent qualité de service et respect des contraintes de l'utilisateur. Ils permettent également une gestion plus fine des ressources. Pour limiter la consommation électrique des systèmes distribués et des réseaux dédiés, nous avons proposé un système de réservation de ressources efficace en énergie.

Mesurer et comprendre la consommation énergétique des systèmes distribués

Afin de mieux comprendre la consommation électrique des ressources de calcul, nous avons déployé une infrastructure de mesure sur le site lyonnais de Grid'5000 (une grille expérimentale française comprenant plus de 5000 cœurs distribués sur 9 sites géographiques). Cette infrastructure comprend des wattmètres qui, grâce à notre interface logicielle, nous permettent d'obtenir la puissance consommée par chacun des 135 noeuds à raison d'une valeur par seconde.

Nous avons ensuite analysé l'utilisation de cette grille et nous avons corrélé ces informations avec la consommation électrique. Cette analyse nous a permis de mettre en lumière plusieurs points : l'utilisation est faite de pics et de creux et la consommation des nœuds inutilisés est très élevée. Ainsi, éteindre ces nœuds lors des périodes creuses pourrait permettre d'économiser beaucoup d'énergie.

ERIDIS : Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems

Différentes techniques peuvent être employées pour réduire la consommation des ressources. Réduire la vitesse de fonctionnement des composants est une solution permettant d'économiser de l'énergie même lorsque les ressources sont utilisées. Éteindre des ressources ou des composants peut permettre des économies d'énergie conséquentes lorsque ces ressources ne sont pas utilisées. Cependant, ces extinctions ont besoin d'être coordonnées à un niveau supérieur pour être réellement efficaces. C'est pourquoi nous avons proposé ERIDIS (Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems), une infrastructure de réservation de ressources efficace en énergie.

ERIDIS agit au niveau du gestionnaire de ressources. Ses trois approches principales consistent à éteindre les ressources non utilisées, agréger les réservations et prédire les réservations futures dans le but d'éviter de trop fréquents allumages et extinctions.

Cette infrastructure générique a été adaptée aux grilles de calcul, aux clouds et aux réseaux filaires dédiés.

EARI : Energy-Aware Reservation Infrastructure

Les grilles de calcul et les centres de traitement de données sont de plus en plus utilisés pour des applications grandissant en taille et en complexité. Leur consommation énergétique devient un frein majeur au déploiement de nouveaux centres.

En nous appuyant sur l'analyse des traces d'utilisation de la plate-forme Grid'5000 sur l'année 2007, nous avons conçu une infrastructure qui prend en compte la dimension énergétique sans modifier les performances des utilisateurs. Cette infrastructure basée sur ERIDIS s'appelle EARI (Energy-Aware Reservation Infrastructure).

Nous avons validé cette infrastructure en "rejouant" les traces de Grid'5000 avec et sans utiliser EARI. Nos résultats montrent que nous aurions pu réduire la facture électrique de l'année 2007 d'environ 50% avec notre infrastructure, ce qui correspond à la consommation annuelle d'un village de 600 habitants.

GOC : Green Open Cloud

Les infrastructures de clouds sont des éléments de plus en plus incontournables dans l'Internet. En effet, ces environnements virtualisés offrent une isolation robuste qui permet aux utilisateurs de partager les mêmes ressources sans interaction nuisible. Cette flexibilité dans la gestion des ressources introduite par la virtualisation et la migration permet de regrouper les applications sur un plus petit nombre de ressources et donc d'économiser de l'énergie.

En nous appuyant sur ce constat, nous avons proposé GOC (Green Open Cloud), une architecture basée sur ERIDIS et adaptée aux infrastructures de clouds capables d'effectuer des migrations de machines virtuelles. Cette architecture a été validée expérimentalement sur notre plate-forme locale. Nous avons en particulier mesuré à l'aide de wattmètres le coût des opérations basiques d'une machine virtuelle : instanciation, démarrage, lancement de l'application, arrêt et migration.

HERMES : High-level Energy-aware Reservation Model for End-to-end networkS

Enfin, nous nous sommes intéressés au cas des réseaux dédiés : réseaux d'entreprises, réseaux interbancaires et réseaux de grilles de calcul et de clouds. En effet, ces réseaux sont régis par une seule entité, ce qui nous a permis de proposer un *overlay* appelé HERMES basé sur ERIDIS.

Notre première étape a été de modéliser la consommation énergétique des équipements réseau. En nous appuyant sur ce modèle baptisé ECOFEN (Energy Consumption mOdel For End-to-end Networks), nous avons proposé un simulateur basé sur NS2 (Network Simulator, le simulateur le plus utilisé actuellement dans la communauté réseau). Notre simulateur permet de simuler de grands réseaux avec des trafics réalistes et d'obtenir la consommation énergétique de chaque équipement en tenant compte du trafic et du type d'équipement utilisé (routeur de cœur, switch résidentiel, carte réseau, etc.).

Ensuite, nous avons adapté ERIDIS au cas des réseaux dédiés. Cette nouvelle infrastructure a été nommée HERMES (High-level Energy-aware Reservation Model for End-to-end networkS). Les algorithmes d'ordonnancement des réservations ont par exemple eu besoin d'être adaptés puisque pour faire une réservation de bout en bout, il faut réserver simultanément tous les liens du chemin à emprunter.

La validation d'HERMES s'est effectuée grâce à notre simulateur de réseau capable de gérer des réservations de bande passante BoNeS (Bookable Network Simulator). Les simulations ont été effectuées sur plusieurs architectures de réseaux typiques (réseau d'entreprise, réseau interbancaire et réseau de grille de calcul) utilisant des valeurs de consommations électriques réalistes et avec des trafics inspirés de la littérature.

Conclusion

ERIDIS est un modèle générique, robuste et adaptable permettant de mieux gérer l'énergie dans les systèmes de réservation de ressources. Nous avons montré son application à trois domaines différents : les grilles de calcul, les clouds et les réseaux dédiés. À travers l'étude et la validation des infrastructures proposées, nous avons montré que des économies d'énergie conséquentes peuvent être réalisées dans ces trois types d'infrastructures en utilisant ERIDIS.

Contents

List of Figures

List of Tables

The beginning of knowledge is the discovery of something we do not understand.

Frank Herbert

1 Introduction

1.1 A new challenge for large-scale distributed systems

Large-scale distributed systems – such as computing data centers, Grids, Clouds and dedicated networks – consist of vast collections of computing and storage resources interconnected through a network. These systems, with different levels of scalability, interoperability and respect to user constraints, have become the building blocks for numerous applications and services ranging from weather forecast to web search.

Over the years, several solutions have been proposed to manage IT resources in distributed systems and make them available to user applications. Computational grids first appeared in late 1990s as a promising hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [FK98]. For many years grid computing has been an active field of research whose main goals are, in addition to managing resources, to address issues related to resource heterogeneity, geographical distribution of elements, and their dynamic behavior [SF10a].

More recently, the maturity of virtualization techniques has led to the emergence of Clouds, which are a new paradigm increasingly becoming essential to IT services. Cloud computing aims to provide dynamic, reliable, customized and QoS guaranteed environments to end-users [WTK+08]. By benefiting from economies of scale, Clouds can efficiently manage and offer virtually unlimited numbers of resources, minimizing the costs incurred by organizations when providing Internet services.

Computer networks are the crucial elements that interconnect IT resources in these distributed systems. As the size of the systems increases and their traffic demands diversify, network resources are often stretched to their limits and, in many cases, become a performance bottleneck. When aiming at high performance, dedicated networks come to the rescue of distributed systems by providing a reliable infrastructure for large-scale applications that require high speed data transfers and quality of service [LW08].

Advances in the systems discussed above have historically been related to improving their performance, scalability and quality of service. Reservation mechanisms are common practice to guarantee that users receive the negotiated quality of service, which helps planning and executing applications with very specific constraints and deadlines [CRH08, SMLF09, PBK+06]. Users are thereby able to reserve computing nodes, virtual machines or bandwidth capacity for allocated time intervals. Moreover, in systems that support advance reservations, users can specify a reservation start time in the future. From the system administrator's point of view, reservations allow for more flexible and predictable resource management since the duration of a reservation is known at its submission.

In recent years, the energy consumed to power Information and Communication Technologies (ICT) has become a major concern. ICT is responsible for 2% of the global CO2 emissions, the equivalent to the aviation industry. In 2008, data centers accounted for over 3% of the electricity consumed in the US and between 1.5% and 2% of the global consumption. Moreover, the energy demands continue to rise at around 12% annually [Koo08]. If cloud computing were a country in 2007, it would have ranked 5th in electricity usage, placing itself between Japan and

India [Gre10b] as shown in Figure 1.1.[1] Moreover, although technology improves, the advent of petascale machines, cloud computing and peer-to-peer systems give hints that the energy required to power these systems is likely to rise [Sma08].

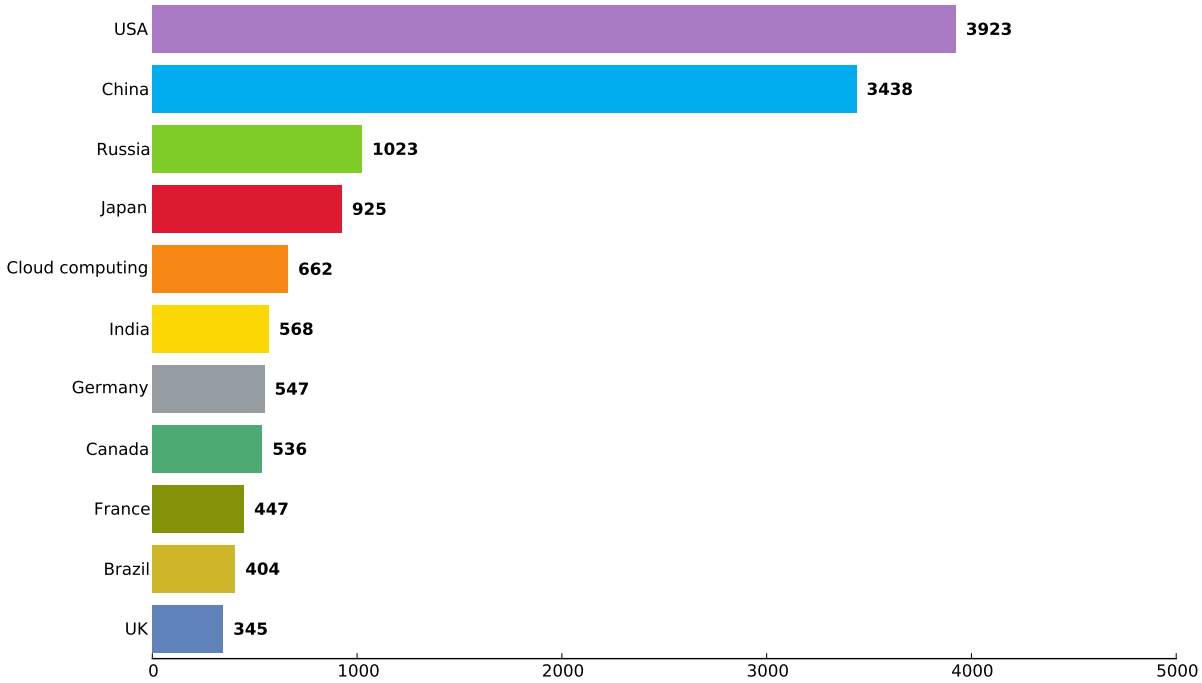

Figure 1.1: Electricity consumption for 2007 in billion kWh (from [Gre11]).

These disturbing figures show that it is, therefore, urgent to curb the rising energy consumption of ICT and drastically to increase the efficiency of large-scale distributed systems. However, improvements have to accommodate usage growth to avoid the Jevons Paradox; a situation where improving the efficiency can increase the usage, thereby enlarging the overall consumption [Gre11].

Improving the energy efficiency of large-scale distributed systems is, however, a challenging task. The main approaches for augmenting the efficiency are to employ hardware that consumes less (e.g. solid state drives and low-power processors) and to reduce the energy consumed by non-IT infrastructure such as air-conditioning (e.g. utilizing free cooling). In spite of their benefits, these gains do not address the resource wastage, the main source of inefficiency. Several of current ICT infrastructures employ large server farms and network equipment that are constantly powered on and operating at their maximum performance even when they are not in use.

1.2 Research problem and objectives

Large-scale distributed systems are major electricity consumers whose consumption is not yet optimized. This thesis investigates means to improve the energy-efficiency of large-scale distributed systems that employ resource reservations while respecting the required performance and user constraints. We focus on reservations because they enable more flexible management and capacity planning, which are useful assets to reduce the energy consumption of large-scale distributed systems. Although reservation frameworks have been developed for large-scale data centers, Grids, Clouds and dedicated networks, only a few take into account the energy as-

[1] The figures considers the data centers and telecommunication networks used by cloud-based computing platforms.

pect and they are exclusively used in Grid environments. Currently no unified energy-efficient reservation framework has been proposed to manage resources in these contexts.

Our objectives are to:

1. Investigate energy monitoring solutions for large-scale distributed systems.
2. Provide a fine-grain analysis of the energy usage of resources of large-scale distributed systems. This analysis will help us to identify where the energy is wasted and whether the energy-efficient solutions proposed in the literature are applicable to these systems.
3. Provide an architectural model that manages resources and user reservations of large-scale distributed systems in an energy-efficient and transparent way. This general model will exploit the similarities between the reservation models of different types of distributed networks. The user performance is not impacted by the energy-efficient policies of the model.
4. Adapt to and validate this model in three different contexts: (1) data centers and grids; (2) clouds; and (3) dedicated networks. As reservation-based systems already exist in large-scale data centers and Grids but not in Clouds and dedicated networks, the validations in those environments will consider scenarios focused on this usage.

1.3 Contributions

Based on the objectives defined previously, we present in this thesis the following contributions:

1. This thesis reviews and classifies the solutions proposed in the literature to reduce energy consumption of large-scale distributed systems.
2. We describe the infrastructure (equipment and software) deployed to monitor an experimental Grid site. This infrastructure enabling 135 computing nodes provides an external power measurement per second and per node [18]. Then, we analyze the usage and the energy consumption of the experimental Grid. This analysis over a period of several months provides us with better understanding and precise quantification of resource wastage [11, 25, 23]. It also allow us to put things in perspective to distinguish facts from myths for large-scale distributed systems dealing with resource heterogeneity [17].
3. We propose ERIDIS: an energy-efficient reservation framework for large-scale distributed systems. Based on reservation aggregation in terms of time and space (number of used resources), usage prediction and on/off power management, this unified theoretical framework can be adapted to reservation-enabled Grid, Cloud and wired network environments [1, 5]. Our working hypothesis is that reservation-based systems are more controllable and can be more optimized than best-effort systems.
4. This thesis presents EARI: an energy-efficient reservation framework for large-scale data center and grid resources adapted from ERIDIS. This infrastructure is validated using replay mechanisms with real Grid traces. Comparing to non energy-aware resource management, it allows important energy savings depending on the platform load [15, 14, 4, 19].
5. We investigate the energy cost of virtual machines during basic operations (boot, run and halt) and live migration [24, 7]. This led us to propose GOC: a Cloud framework adapted from ERIDIS and validated through a prototype implementation test on real nodes. Using live migration for a better load consolidation and proxying techniques to ensure the network presence of sleeping hosts, GOC is compared to current resource management in Clouds [13, 12, 3].

6. This thesis presents ECOFEN: and energy model and simulator for evaluating power consumption and testing energy-efficient algorithms in large-scale networks. Based on the well-known network simulator NS2, it is enhanced with energy-efficient features for networking equipment described in the literature [16]. This preliminary work allowed us to propose HERMES: a bandwidth reservation framework adapted from ERIDIS to dedicated networks. HERMES is designed for high-performance networks and thus, does not deal with wireless networks which are more used in access networks. Centralized and decentralized power management techniques are compared in terms of number of control messages and energy consumption. The HERMES framework is validated through simulation-based results since rate adaptation on on/off techniques are not yet present in networking equipment [22, 10, 9, 8, 2, 6].

1.4 Organization of the manuscript

The manuscript is organized as follows.

Chapter 2 reviews the literature on techniques and mechanisms to save energy in large-scale distributed systems, considering both computing and networking resources.

In Chapter 3, we present our experience in Grid monitoring and understanding its energy consumption. The deployed infrastructure, described in this chapter, focuses on the energy consumption of distributed resources.

Chapter 4 proposes our solution ERIDIS: Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems. It details the architecture and algorithms designed to save energy.

This solution is adapted to three different contexts:

- data centers and Grids in Chapter 5;
- Cloud environments in Chapter 6;
- dedicated networks in Chapter 7.

These three chapters present both an energy-efficient framework, which is the adaptation of ERIDIS to the particular context, and the validation of each framework using realistic assumptions.

We present conclusions and perspectives in Chapter 8.

Energy and persistence conquer all things.

Benjamin Franklin

2

Energy-efficiency in computing and networking resources: state of the art

Although energy consumption has always been a key factor in sensor networks and battery-constrained devices, it has only recently become an issue for other ICT systems. The Gartner's annual hype cycle for emerging technologies[1] shows this phenomenon: green IT first appeared in 2008, directly at the peak of the wave; in 2009 it was on the declining side of the wave, and surprisingly by 2010 it had disappeared.

From the start, improving performance has been one of the main goals of research in distributed computing, from hardware [Moo98] to middleware design. However, it has become impossible to ignore the energy consumption of distributed systems. On a small scale, the cost of the energy consumed by a server often exceeds its purchase cost [Bar05]. On a large scale, data centers are reaching power delivery limits, and new infrastructures are being set up near power stations to make use of the maximum power they can deliver [Gre11].

In general, the energy that a system consumes comprises two components:

- A fixed (or static) part that depends on the system's size and on the type of the used component (computing, storing and network elements); it is due to leakage currents present in any powered system.

- A variable (or dynamic) part that results from the usage of computing, storage, and network resources; it is due to the activity and the clock rates.

The energy consumption E of an element depends on its power consumption P over time t. For a given time interval T, the energy is given by:

$$E(T) = \int_0^T P(t)\ dt \tag{2.1}$$

In a distributed system, a large amount of energy is generally wasted by various computing and networking equipment – such as PCs, switches, routers, and servers – because they typically remain fully powered on even when idle. Due to scaling effects, even a small wastage of energy at the component level can have global consequences on the energy efficiency of the distributed system. For instance, in 2010 the Green Grid consortium carried out a survey about unused servers in 188 data centers, mostly located in the United States [Gre10a]. They estimate that on average 10% of the servers are never used, hence wasting energy.

Reducing the energy consumption of large-scale distributed systems is a challenging issue that should be addressed at different levels: infrastructure, nodes, and hardware components. Hence, the rest of this chapter is organized as follows. Section 2.1 discusses the different approaches for saving energy in computing resources. Section 2.2 classifies the research work on energy savings for wired networks. Finally, Section 2.3 presents concluding remarks.

[1] `http://blogs.gartner.com/hypecyclebook/2010/09/07/2010-emerging-technologies-hype-cycle-is-here/`

2.1 Energy-efficiency of computing resources

Data centers and supercomputers are made up of large numbers of servers concentrated in a small area. For example, *Tianhe-1A* located at the National Supercomputing Center in Tianjin, China, is the fastest supercomputer in the world (2.566 PFLOPS) and consumes 4.04 MW of power[2]. At a price of $0.10 per KWh, the cost to power this supercomputer is around $3.5 million per year (about 2.5 million euros). Such power requirements and costs are generally limiting factors to the scalability of supercomputers and data centers.

The development of supercomputers has long been driven by performance as attested by the top500 list[3]. The green500 list[4] has attempted to reorganize the top500 ranking considering the energy efficiency of supercomputers, trying to raise awareness about their huge power requirements [FS09].

In these large computing infrastructures, energy can be saved at different levels. However, the first, and difficult step, towards reaping these savings is to figure out how much power each individual component consumes. It is mandatory to know how much energy resources consume in order to design and evaluate new energy-efficient architectures and algorithms.

2.1.1 Measuring and modeling the energy consumption

The energy consumption of computing resources can be either determined by wattmeters or estimated via energy models. Wattmeters can be completely external equipment or components integrated into Power Distribution Units (PDUs) and temperature sensors. Regarding power measurements, energy sensors integrated into the components offer a smaller granularity than external wattmeters. Deploying energy sensors or wattmeters can be costly if it is not done when the whole infrastructure (cluster or data center) is set up.

Using energy models to estimate the consumption of components or entire infrastructures is less costly than deploying wattmeters. However, the models should be lightweight not to interfere with the energy consumption they try to estimate. Models can estimate the energy consumption of racks, devices, processes, services, etc. For example, *PowerTOP*[5] is a Linux software utility released by Intel whose goal is to "find the software component(s) that make your laptop use more power than necessary while it is idle".

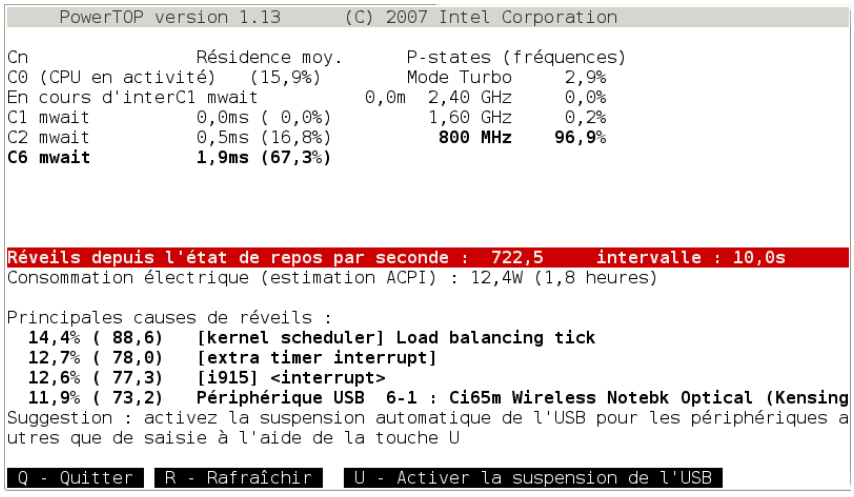

Figure 2.1: Screenshot of PowerTOP running on a laptop.

[2]Nvidia article http://pressroom.nvidia.com/easyir/customrel.do?easyirid=A0D622CE9F579F09&version=live&prid=678988&releasejsp=release_157
[3]Top500 supercomputing sites http://www.top500.org/
[4]Green500 list http://www.green500.org/
[5]PowerTOP http://www.linuxpowertop.org/powertop.php

Figure 2.1 shows a screenshot of PowerTOP running on a laptop. It provides the time spent at each Central Processing Unit (CPU) state (15.9% for the C0 state for example), the number of wake-ups per second (722.5 in this case), an estimation of the power usage using the ACPI (12.4 W), the top causes for wake-ups (in this case, the first one is the kernel scheduler) and tips to reduce the power consumption (to enable the option to automatically disable USB ports when not in use). To estimate the power usage, PowerTOP uses Advanced Configuration and Power Interface (ACPI)[6].

As processors are among the most consuming components of computers, several solutions have been proposed to evaluate their consumption at different levels [CBBA10], including:

- Cycle level estimation where the power consumption of each processor unit is estimated at each clock cycle.
- Instruction level power analysis in which the power consumption of processor instructions are summed to estimate the energy consumed by a program.
- Power analysis at the functional level based on the analysis of the processor architecture.
- System level power estimation considering the average power of an instruction multiplied by the execution time to obtain the program's energy consumption.

In [FWB07], the authors model the energy consumption according to a CPU's activity, whereas another approach consists in deducing the consumption by using *event-monitoring counters* included in modern processors such as Pentium 4 [MB06].

The power P consumed by a processor can be expressed mathematically as the sum of its static power P_{static} and its dynamic power $P_{dynamic}$ [WvLDW10]. $P_{dynamic}$ can be presented as follows:

$$P_{dynamic} = ACV^2 f$$

where A is the percentage of active gates, C the total capacitance load, V the supply voltage and f the frequency [GFC05].

Another research issue is to estimate the power consumed by applications. In [SF10b], when providing a model to predict the power consumption and performance of the LINPACK HPL benchmarks[7], the authors concluded that maximum energy efficiency is not always achieved at the highest performance. This leads to the question: how to measure energy-efficiency? For infrastructures such as data centers, the most common metric, introduced by the *Green Grid*,[8] is the *Power Usage Effectiveness* (PUE), which is defined as:

$$PUE = \frac{Total\ Facility\ Power}{IT\ Equipment\ Power}$$

Another common metric is the *Data Center Infrastructure Efficiency* (DCiE) [Gre07]:

$$DCiE = \frac{1}{PUE} = \frac{IT\ Equipment\ Power}{Total\ Facility\ Power} \times 100\%$$

These two metrics indicate how much power is used by the IT infrastructure and hence, how efficient are the cooling system and other non-IT resources. Other metrics are available, such as

[6] Advanced Configuration and Power Interface (ACPI) is a standard co-developed by Hewlett-Packard, Intel, Microsoft, Phoenix Technologies and Toshiba. Its goals is to reduce computer power consumption by switching off its components. The operating system is in charge of managing the power supply of each component.

[7] High Performance LINPACK (HPL) `http://www.netlib.org/benchmark/hpl`

[8] The Green Grid is a non-profit, open industry consortium of end-users, policy-makers, technology providers, facility architects, and utility companies collaborating to improve the resource efficiency of data centers and business computing ecosystems (`http://www.thegreengrid.org/`).

the *performance per watt*, used to produce the *Green500* list [FS09] and expressed in Floating Point Operations Per Second (FLOPS) per watt.

Sun Microsystems has proposed a new three-criterion metric called *Space, Watts and Performance* (SWaP) to evaluate server efficiency:

$$SWaP = \frac{Performance}{Space \times Power\ Consumption}$$

They provide a web utility to compute a server's energy efficiency[9].

To evaluate the energy efficiency of an application or middleware, it is interesting to use as criteria the QoS metrics related to the evaluated software. For task scheduling in a Cloud environment, the authors of [YL11] utilize the energy-delay product. They show that computing nodes that are at most three times slower than the fastest node should be discarded from the Cloud system to achieve an optimal energy-delay product.

Another solution is to use benchmarks to compare the efficiency of different architectures and software. Examples include the *SPECpower*[10], a benchmark that evaluates the power and performance characteristics of volume server class and multi-node class computers; and JouleSort, a benchmark to evaluate the trade-off between power and performance of computing nodes by sorting a fix number of records using as little energy as possible [RSRK07].

Once the energy consumption of individual computing resources is known, researchers can design and implement new techniques to reduce energy consumed by the overall infrastructure. In a grid context, different solutions can be applied to both the node and the grid levels. The following sections discuss research on these two levels.

2.1.2 Node optimizations

This section analyzes energy-efficient solutions that work at the node level, which can in turn lead to great energy savings at the grid level.

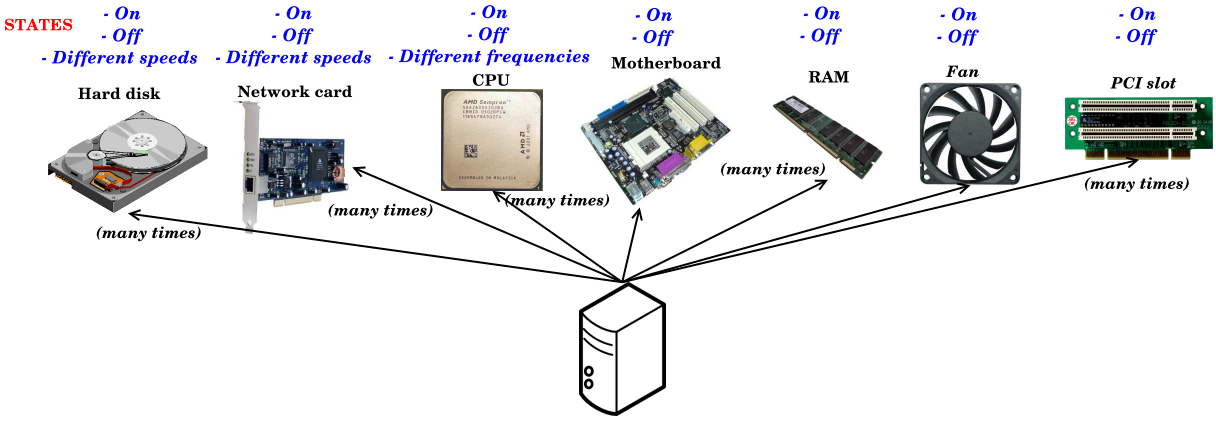

Figure 2.2: Possible states per node component.

A computing node comprises several components, each of which can be optimized to save energy as shown in Figure 2.2. The figure also shows which components can be switched off or put in lower power modes. A node can typically be in one of several states, from fully-on to fully-off, which have their own energy consumptions. For example, a CPU has an off-state and several on-states that correspond to different frequencies and voltages [SRH05].

Several studies focus on minimizing the consumption of specific components, such as Network Interface Cards (NICs) [GCN05], disks [AAF+09] and CPUs [DD06]. Table 2.1 shows results

[9]SWaP `http://www.europe.access.com/x64_swap.html`
[10]SPECpower `http://www.spec.org/power_ssj2008/`

on breaking up the energy consumed by a typical rack server into the consumptions of its components [FWB07].

Component	Peak power	Count	Total	Percentage
CPU	40 W	2	80 W	37.6 %
Memory	9 W	4	36 W	16.9 %
Disk	12 W	1	12 W	5.6 %
PCI slots	25 W	2	50 W	23.5 %
Motherboard	25 W	1	25 W	11.7 %
Fan	10 W	1	10 W	4.7 %
System total			213 W	

Table 2.1: Component peak power breakdown for a typical server (from [FWB07]).

The motherboard, a high consuming component, can be turned off only if the entire node can (sleep state). This section details node-level techniques to curb the energy consumption of computing resources. These techniques comprise switching off resources (using sleep states), configuring different voltages and frequencies for the CPU, improving the energy-efficiency of software and using energy-efficient hardware and low-level capabilities.

Sleep state

While the first and natural idea to save energy is to shut down idle nodes [CAT+01], this raises a problem: how to wake them up when required? Wake-On-LAN, a mechanism implemented on Ethernet cards, allows a distant user to wake up a PC by sending specific packets via the network [GCN05]. However, under this scheme the Ethernet card must be powered at all times.

The Intelligent Platform Management Interface (IPMI) is a standard hardware that operates independently from the operating system and allows administrators to manage a system remotely through a direct serial connection or via a LAN connection[11]. This interface can also be used to switch nodes on and off remotely [Lea06].

Suspend to disk techniques can be used to reduce the energy consumed during wake-up and booting periods. When a node suspends to disk, all the main memory contents are saved in file on the hard disk drive, thus preserving the state of the operating system (open applications, documents, etc.). All of the node's components are turned off and, at the next state switch, the node will load the hibernate file, restoring the previous state.

However, the sleep state can only be used to save energy when the node is not used. Other techniques can be more efficient in cases where nodes are constantly switched on.

Dynamic Voltage and Frequency Scaling

For a few years, laptop processors have been able to adjust their working frequency and power consumption to conserve battery life. This technology, called Dynamic Voltage and Frequency Scaling (DVFS) [SRH05], is commonly available on recent High Performance Computing (HPC) nodes present in the large-scale data centers, Grids and Clouds, but it is seldom exploited.

Under DVFS, *P-states* (processor performance states) define the different frequencies supported by a processor. The several P-states – P0, P1, P2. . . Pn where n is processor dependent – enable power savings. For example, under P3 the processor will run slowly and use less power than under P1. On Linux, the *CPUfreq*[12] infrastructure allows the control of P-states by *governors* that decide which available frequency, between minimal and maximum, must be chosen. The available governors include the *on-demand*, which adjusts the frequency automatically; the

[11] IPMI `http://www.intel.com/design/servers/ipmi/`

[12] CPUfreq `http://www.kernel.org/pub/linux/utils/kernel/cpufreq/cpufreq.html`

performance that chooses the highest frequency; the *user-space*, which allows for setting the frequency manually; the *conservative*, which increases the frequency progressively (unlike the on-demand governor which skips to the maximal frequency when the system is in full load).

C-states correspond to CPU idle states, except for C0, which is an operational state. In C3 the processor can be turned off and will have to be reset to carry out instructions again, whereas C4 refers to the deeper sleep state. The higher the number, the less energy the processor consumes and the longer it takes to become active again. While keeping the processor idle for a long period can allow for power savings, it is necessary to reduce CPU wake-ups by disabling services and processes not strictly required. From the Linux kernel version 2.6.24, there is a new feature called Dynamic ticks or tick-less kernel (NO_HZ) which allows to wake up the processor only when required. This way, the processor is not woken up often just to realize that there is no work to do.

Powernow! and *Cool'n'Quiet* technologies from AMD and *Speedstep* technology from INTEL implement the P-states. They can reduce tension depending on the frequency and deactivate unused processor parts.

Software improvements

This section presents software improvements that reduce the energy consumed by computing resources.

Nowadays, developers of drivers, kernel modules and distributed applications should be more careful when designing code if they wish to reduce the energy consumed by their systems. For example, waiting loops and active polling may frequently wake up the CPU. As shown by PowerTOP (Figure 2.1), many applications wake the CPU up hundreds of times per second and sometimes unnecessarily[13].

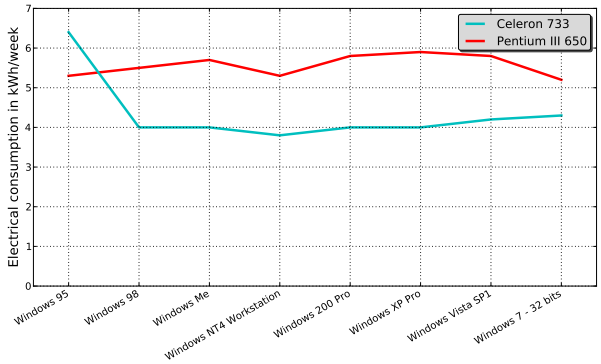

Figure 2.3: Power consumed by several versions of Windows for two different PCs (from [Eco11]).

When concerning software, the Operating System (OS) is the first element to consider. OSs have heterogeneous power consumptions and can be optimized to consume less energy. Even different versions of the same OS do not present the same power consumption as illustrated in Figures 2.3 and 2.4, which show the power consumed by several Windows versions and Linux kernels respectively. The studies for Windows [Eco11] and Linux [Les10] have similar methodologies. The consumption of a node under different versions of the considered OS presents non negligible variations depending on the version in use.

Moreover, an OS can regulate its activity and energy consumption to meet thermal or energy constraints, a task that is commonly done through the standard ACPI [Ste98].

The Basic Input Output System (BIOS) – the very first software called when a system boots – is stored in an (EEP)ROM on the motherboard and contains a set of basic functions to initialize

[13]Examples of such applications are provided on the PowerTOP website: `http://www.linuxpowertop.org/known.php`

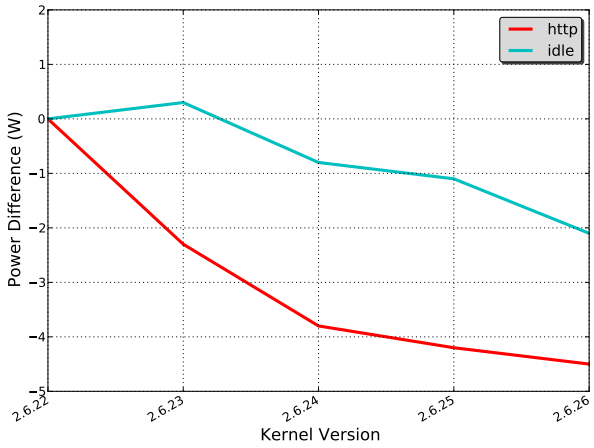

Figure 2.4: Power consumed by different Linux kernels (from [Les10]).

hardware, run diagnostics and search for bootable devices where the OS might be available. Although manufacturers develop a BIOS for each motherboard they design, the default setup is generally used to support all the configurations (OS) and checks for all the possible devices, thus wasting time and energy for nothing.

In a similar way, initializing the Universal Serial Bus (USB) host adapter takes time. HPC nodes often have several USB interfaces, but they are hardly used. To reduce the time to boot, major BIOS setups can disable USB ports and avoid their initialization. Moreover, as shown by PowerTOP (Figure 2.1), USB ports frequently wake up the CPU even if they are not in use. These factors also concern other server components that are generally not used in data centers (e.g. RS-232 serial ports, bluetooth modules and wireless cards) and that could be disabled.

Energy-aware hardware capabilities

As shown in Figure 2.2, manufacturers can improve the energy efficiency of all server components (e.g. power supply, fans and disks) by providing solutions that work at the hardware level. For example, HPC nodes commonly use hard disk drives – a scenario that may change as SSD become more affordable – whose energy consumption can be reduced by spinning platters down [CPB03]. However, spinning platters up induces a peak power and takes time. Hence, the objective is generally to minimize disk accesses to keep platters spun down as long as possible [CPB03].

In addition to improving the energy-efficiency of hardware components, manufacturers should allow individual components such as PCI slots and CPU cores, to be put into sleep state when not in use (independently from the motherboard). During computing phases for example, network cards may not be required and could hence be turned off.

Finally, manufacturers should focus their efforts on increasing the number of voltages, frequencies and speeds available for each component (CPUs, disks, NICs) making it able to adapt its working conditions to the load. This could leads closer to proportional computing [BH07].

2.1.3 Grid and data center power management

This section describes techniques for power management at the scale of a data center or grid. Some of these techniques coordinate at a wide scale the node-level schemes described in the previous section. The explored research areas include using green sources, thermal management of computing resources, workload consolidation and energy-aware task scheduling at the middleware level.

Green sources

The first way to save energy at a data center is to locate it close to where the electricity is generated, hence minimizing transmission losses. For example, Western North Carolina, USA, attracts data centers with its low electricity prices due to abundant capacity of coal and nuclear power following the departure of the region's textile and furniture manufacturing [Gre11]. As of writing, this region has three super-size data centers from Google, Apple and Facebook with respective power demands of 60 to 100 MW, 100 MW and 40 MW [Gre11].

Other companies opt for *greener* sources of energy. For example, Quincy (Washington, USA) supplies electricity to data facilities from Yahoo, Microsoft, Dell and Amazon with its low-cost hydroelectrics left behind following the shutting down of the region's aluminum industry [Gre11]. Several renewable energy sources like wind power, solar energy, hydro-power, bio-energy, geothermal power and marine power can be considered to power up super-sized facilities.

Another green evolution is to use *free cooling* which consists in cooling equipment with outside air [PV10]. As cooling accounts for about 33% of the power used in a data center [GHMP08], this technique leads companies to locate their facilities in regions and countries with cold climate, such as Sweden[14]. Another free cooling technique is to use sea water, such as in the new Google data center in Hamina, Finland[15].

In spite of these approaches, numerous data facilities have already been built and cannot move. Grid environments, on the other hand, can still take advantage of multiple locations to use green sources of energy with approaches such as *follow-the-sun* and *follow-the-wind* [FLR+09]. As sun and wind provide renewable sources of energy whose capacity fluctuates over time, the rationale is to place computing jobs on resources using renewable energy, and migrate jobs as renewable energy becomes available on resources in other locations.

Thermal management

Thermal issues have been raised first because they are the most direct consequences of increasing the number of transistors on processor chips. These issues and energy consumption are interrelated since decreasing the heat production will reduce energy consumption. For this reason many algorithms deal with both energy and thermal issues [PSBG02, SBP+05, MB06].

Figure 2.5, from an HP technical report [PSBG02], presents the typical case of an infrastructure with a PUE of 1.5, meaning that cooling by itself consumes half the amount of power used by the computing resources. The authors present a solution, called thermal load balancing [SBP+05], that takes advantage of the different clusters' location in the Grid and assigns workload based on the thermal management infrastructure and the seasonal and diurnal variations of temperature [PSBG02]. It takes as example two sites of the same Grid, one located in New Delhi and another in Phoenix. During Summer, the external temperature in New Delhi reaches its peak at midday, at which time it is night in Phoenix and the temperature is lower. Hence, it is preferable to place the workload in Phoenix and use less cooling capacity than in New Delhi.

Workload consolidation

Workload consolidation [CD01, CAT+01], investigated in previous work [DCA+03, USC+08, VAN08a, CAT+01, KCH09, KKH+08, JJH+09, SKZ08], consists in running multiple tasks on the same physical machine in order to reduce the number of nodes that are switched on. A key component of these systems is to monitor and estimate the workload of applications or the arrival of user requests. Several techniques have been applied to estimate the load of a system,

[14] "Safe, Green and Cool" `http://www.investsweden.se/world/Industries/ICT/Data-centers/`

[15] `http://www.google.com/corporate/datacenter/efficient-computing/efficient-data-centers.html`

such as exponential moving averages [BJR94], Kalman filters [Kal60], auto-regressive models, and combinations of methods [KN01, CAT+01].

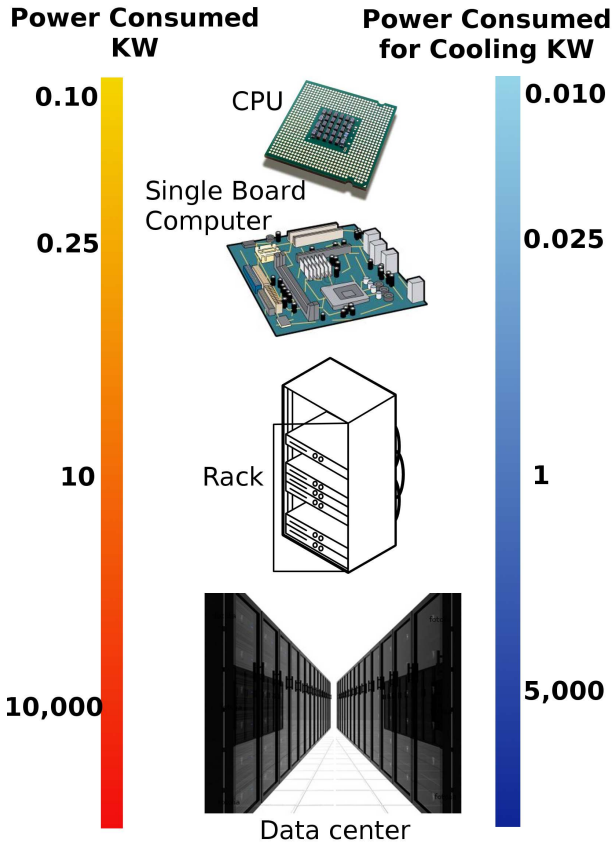

Figure 2.5: Power consumed by cooling (from [PSBG02]).

Fitted with workload-estimation techniques, Grid systems provide schemes to minimize the energy consumed by the underlying infrastructure while minimizing costs and violations of Service Level Agreements (SLAs). Chase *et al.* [CAT+01] introduced MUSE, an economy-based system that allocates resources of hosting centers to services aiming to minimize energy consumption. Services bid for resources as a function of delivered performance whilst MUSE switches unused servers off. Kalyvianaki *et al.* [KCH09] introduced autonomic resource provisioning using Kalman filters. Kusic *et al.* proposed a look-ahead control scheme for constantly optimizing the power efficiency of a virtualized environment [KKH+08]. With the goal of maximizing the profit yielded by the system while minimizing the power consumption and SLA violations, the provisioning problem is modeled as a sequential optimization under uncertainty and is solved using the look-ahead control scheme.

In some cases, consolidating workload on fewer nodes may increase the overall energy consumed by the platform if unused nodes are not switched off. In [FPK+05], the authors show that for some parallel applications, one can save energy and time by executing a program on more nodes at a slower speed rather than on fewer nodes at the fastest speed. Similarly, sometimes using the least power consuming state of processors is more energy consuming than exploring parallelism and as many processors as possible to complete an application faster [dLJ06]. Hence, parallelism should be implemented carefully.

Parallel applications with unbalanced load can benefit from using DVFS at nodes with small tasks while they wait to synchronize with nodes with heavier load [KFL05]. As the middleware can influence the energy consumed by the platform as it maps users' tasks and physical resources, it constitutes a great leverage to improve the energy efficiency of distributed systems. However,

although workload consolidation techniques often rely on task scheduling algorithms, energy-efficient scheduling does not always aim at consolidating tasks on fewer nodes, as shown in the next subsection.

Moreover, for all on/off algorithms, an unnecessary wake-up wastes energy by creating power spikes when the node is woken up and put into sleep mode again. Such algorithms should thus be carefully designed in order not to shut down nodes unnecessarily.

Energy-aware task scheduling

A significant number of algorithms has been proposed for energy-aware scheduling. These algorithms differ on whether they are designed for divisible tasks [CAT+01] [WvLDW10], whether they require synchronization [JG06], if they use DVFS [WvLDW10], or whether they work on homogeneous clusters [MRZ+03] [YCKT09]. They are generally classified into three categories [ZC08]: off-line scheduling based on *a priori* task information [YDS95], on-line scheduling, which is purely dynamic [KR10, ZC08] and hybrid approaches including an off-line phase where the slack is greedily absorbed and dynamic algorithms operating in the online phase [HJ08, SK04].

In [CD01], the authors want to minimize the number of joules per operation. The resource manager gets a set of awake nodes and minimizes its size as much as possible. When a task ends on a node, the manager tries to move the other tasks from this node to the other running nodes. If a new task arrives, the manager tries to put it on the awake nodes, while the other nodes remain off. This algorithm includes no load balancing mechanisms, so some nodes may wore out prematurely while others remain unused.

Under certain scenarios, it is also possible to negotiate the performance degradation with the user (mainly in terms of execution time) to save more energy. Such an approach is described in [WvLDW10] where users accept, for example, an increase of 10% in task execution time in order to reduce energy consumption.

Some task scheduling algorithms use DVFS techniques [JG06, SRH05, FWB07], which allows for energy savings when the nodes are not fully utilized. DVFS can be used during the execution of non-critical tasks [CMKR05, WvLDW10] or during communication phases of MPI applications [LFL06]. In this case, the processor frequency is adjusted depending on the CPU utilization. Another solution is to use the user-perceived latency, the delay between user input and computer response, to drive voltage scaling [YZJ05].

As outlined by the variety of the proposed solutions, scheduling algorithms should be designed for the workload they have to manage (e.g. web servers, computing jobs). For periodic real-time tasks, Aydi *et al.* prove that the optimal solution consists in using the CPUs at either full capacity or at the minimum speed if the utilization is under 100% [AMAMM01]. In addition to performance goals, energy-efficient job placement algorithms can take into account load balancing [MB06], thermal management [PSBG02, SBP+05] and network connections [CHL+08].

2.1.4 Virtualization techniques

Clouds, already used by numerous companies, differ from Grids as explained in [BYV+09] and can be part of the next-generation data centers with virtualized nodes and provisioning on demand. As of writing, Salesforce.com handles 54,000 companies and their 1.5 million employees using only 1,000 servers [Ham09]. Different enterprises, such as IBM [BMQ+07], also support and provide Cloud infrastructures and services to customer companies.

Virtualization is a key feature of Clouds that can improve the efficiency of large-scale distributed systems [TBL09]. It is now widely used to provide a large number of computing resources and minimize the energy consumption of Cloud infrastructures – an issue as urgent as the energy consumption of data centers and grids [NS07, SLB07, TBL09].

Even if virtualization adds a software layer that consumes energy [TCH+08], it allows for finer load consolidation on a virtualized node [SKZ08] and provides live migration techniques [TDG+06] to strengthen load aggregation. However, these techniques have a cost [OLG10] and should be carefully studied in order to reduce the overall consumption of Clouds.

This section details the energy gains of virtualization techniques at the Virtual Machine (VM) level, using VM migration techniques, and at the Cloud level.

Virtual Machines

Virtualization brought along ideas on energy management [TBL09, HLM06]. As nodes can be virtualized and host several virtual machines, virtualization can address some of the limitations in cooling and power delivery faced by large-scale distributed systems.

The overhead posed by VM technologies [CG05] has decreased over the years, which has expanded their appeal for running high performance computing applications [TMM06] and turned virtualization into a mainstream technology for managing and providing resources for a wide user community with heterogeneous software-stack requirements.

While the macro-level resource management performs actions that generally take into account the power consumption of a group of resources or the whole data center, at the host-level the power management is performed by configuring parameters of the hypervisor's scheduler, such as throttling of Virtual CPUs (VCPU) and using other OS specific policies. In the proposed architectures, hosts generally run a local resource manager that is responsible for monitoring the power consumption of the host and optimizing it according to local policies. The power management capabilities available in virtualized hosts have been categorized as [NS07]: "soft" actions such as CPU idling and throttling; "hard" actions like DVFS; and consolidating in the hypervisor. CPU idling or soft states consist in changing resource allotments of VMs and attributes of the hypervisor's scheduler (e.g. number of credits in Xen's credit scheduler) to reduce the CPU time allocated to a VM so that it consumes less power. Hard actions comprise techniques such as scaling the voltage and frequency of CPUs. Consolidation can also be performed at the host-level where the VCPUs allocated to VMs can be configured to share CPU cores, putting unused cores in idle state, hence saving the energy that would otherwise be used by the additional core to run a VM.

Nathuji and Schwan [NS07] presented VirtualPower, a power management system for virtualized environments that explores both hardware power scaling and software-based methods to control the power consumption of underlying platforms. VirtualPower exports a set of power states to VM guests that allow them to use and act upon these states thereby performing their own power management policies. The soft states are intercepted by Xen hypervisor and are mapped to changes in the underlying hardware such as CPU frequency scaling according to the virtual power management rules. The power management policies implemented in the guest VMs are used as "hints" by the hypervisor rather than executable commands. They also evaluate the power drawn by cores at different frequency/voltage levels and suggest that such technique be used along with soft schemes.

VM-based resource management systems such as Eucalyptus [NWC+08] and OpenNebula [FVG+08], allow users to instantiate and customize clusters of virtual machines atop the underlying hardware infrastructure. When applied in a data center environment, virtualization can allow for impressive workload consolidation. For instance, as Web applications usually present variable user population and time-variant workloads, virtualization can be employed to reduce the energy consumed by the data center environment through server consolidation whereby VMs running different workloads can share the same physical host.

Migration

Virtualization needs powerful resource management mechanisms [GIYC06] to benefit from migrating, pausing and resuming VMs. The design of resource-management policies is challenging (NP-hard problem) and dynamic. Live migration [CFH+05] greatly improves the capacities and the features of Cloud environments: it facilitates fault management, load balancing, and low-level system maintenance. Migration implies more flexible resource management as virtual machines can move from one host to another. It offers a new stage of virtualization by removing the concept of locality in virtualized environments.

However, this technique is complex and more difficult to use over MAN/WAN [TDG+06] than within a cluster. IP addressing is a problem since the system should change the address of the migrated virtual machine which does not remain in the same network domain. Moreover, it impacts the performance of VMs by adding a non negligible overhead [VBVB09].

Cloud computing

Current Web applications demand highly flexible hosting and resource provisioning solutions as explained in [SSvdBZ09]. The rising popularity of social network Web sites, and the desire of current Internet users to store and share increasing amounts of information (e.g. pictures, movies, life-stories, virtual farms) have required scalable infrastructure. Benefiting from economies of scale and recent developments in Web technologies, data centers have emerged as a key model to provision resources to Web applications and deal with their availability and performance requirements. However, data centers are often provisioned to handle sporadic peak loads, which can result in low resource utilization [IDE+06] and wastage of energy [HSMR09].

A Cloud computing environment can scale dynamically by allocating virtualized resources that are often provided as services over the Internet [Hay08]. Clouds open up new horizons where anything is considered a service (infrastructure, platform, software, computing, storage) and provide advantages such as cost and reliability. However, customers commonly worry about security and loss of sensitive data when using services from Cloud providers such as Amazon[16]. Accounting is another key challenge as providers need to be competitive and remain economically viable.

The ever-increasing demand for cloud-based services does raise the alarming concern on the energy consumed by data centers. Recent reports [PCGL07] indicate that energy consumption is becoming dominant in the Total Cost of Ownership (TCO). In 2006, data centers represented 1.5 percent of the total US electricity consumption. By 2011, the energy consumed by data centers could double [Sil08] leading to more carbon emissions. Electricity becomes the new limiting factor for deploying data center equipment.

A range of technologies can be utilized to make cloud computing infrastructures more energy efficient, including better cooling technologies, temperature-aware scheduling [MCRS05, FWB07, PSBG02], Dynamic Voltage and Frequency Scaling (DVFS) [SRH05], and resource virtualization [TBL09]. The use of VMs [BDF+03a] brings several benefits including environment and performance isolation; improved resource utilization by enabling workload consolidation; and resource provisioning on demand. Nevertheless, such technologies should be analyzed and used carefully for really improving the energy-efficiency of computing infrastructures [MLVH+02].

Consolidation is directly linked with energy management in clouds since it aims to manage jobs and assign them to physical nodes. Consolidation algorithms have to strike a balance of performance, resource utilization and energy consumption by exploring resource heterogeneity and application affinities [SKZ08, FRM+10].

By consolidating the workload of user applications into fewer machines, unused servers can potentially be switched off or put in low energy consumption modes. Yet attracting virtual-

[16] `http://aws.amazon.com`

ization is, its sole use does not guarantee reductions in energy consumption. Improving the energy efficiency of Cloud environments with the aid of virtualization generally calls for devising mechanisms that adaptively provision applications with resources that match their workload demands and utilizes other power management technologies such as CPU throttling and dynamic reconfiguration; allowing unused resources to be freed or switched off.

Existing work has proposed architectures that benefit from virtualization for making data centers and Clouds more energy efficient. The problem of energy-efficient resource provisioning is commonly divided into two sub-problems [LZLH09]: at micro- or host level – discussed earlier – power management techniques are applied to minimize the number of resources used by applications and hence reduce the energy consumed by an individual host; and at a macro-level, generally a Resource Management System (RMS) strives to enforce scheduling and workload consolidation policies that attempt to reduce the number of nodes required to handle the workloads of user applications or place applications in areas of a data center that would improve the effectiveness of the cooling system. Some of the techniques and information commonly investigated and applied at a macro- or RMS-level to achieve workload consolidation and energy-efficient scheduling include:

- Applications workload estimation;
- The cost of adaptation actions;
- Relocation and live-migration of virtual machines;
- Information about server-racks, their configurations, energy consumption and thermal states;
- Heat management or temperature-aware workload placement aiming for heat distribution and cooling efficiency;
- Study of application dependencies and creation of performance models; and
- Load balancing amongst computing sites.

Although consolidation fitted with load forecasting schemes can reduce the overall number of resources used to serve user applications, the actions performed by RMSs to adapt the environment to match the application demands can require the relocation and reconfiguration of VMs. That can impact the response time of applications, consequently degrading the QoS perceived by end users. Hence, it is important to consider the costs and benefits of the adaptation actions [VAN08a]. For example, Gueyoung *et al.* [JJH+09] have explored a cost-sensitive adaptation engine that weights the potential benefits of reconfiguration and their costs. A cost model for each application is built offline and to decide when and how to reconfigure the VMs, the adaptation engine estimates the cost of adaptation actions in terms of changes in the utility, which is a function of the application response time. The benefit of an action is given by the improvement in application response time and the period over which the system remains in the new configuration.

Moreover, consolidation raises the issue of dealing both with necessary redundancy and placement geo-diversity. Cloud providers such as Salesforce.com, who offer to host entire websites of private companies [Ham09], do not want to lose entire websites due to power outages or network access failures. Hence, outages and blackouts should be anticipated and taken into account in resource management policies [SV09].

We conclude this section by summarizing the existing approaches in Table 2.2. This classification follows the organization adopted in this section and replaces each presented work in its main research area. Our framework for large-scale data centers and Grids, detailed in Chapter 5, acts at the Grid level and uses consolidation techniques and scheduling algorithms. Our

framework for Clouds presented in Chapter 6 works at the management system level and uses migration techniques.

Node level	Sleep state	[CAT+01] [GCN05] [Lea06]
	DVFS	[SRH05] [KFL05] [WvLDW10] [LFL06] [CMKR05] [YZJ05]
	Software improvements	[Eco11] [Les10] [Ste98]
	Hardware capabilities	[CPB03]
Infrastructure level	Green sources	[Gre11] [FLR+09]
	Thermal management	[PSBG02] [SBP+05] [MB06]
	Workload consolidation	[CD01] [DCA+03] [USC+08] [VAN08a] [KKH+08] [JJH+09] [SKZ08] [FPK+05] [dLJ06]
	Task scheduling	[JG06] [MRZ+03] [YCKT09] [ZC08] [YDS95] [HJ08] [SK04] [FWB07] [AMAMM01] [MB06] [CHL+08]
Virtualized environments	Virtual machines	[NS07] [SLB07] [TBL09] [TCH+08] [CG05] [HLM06]
	VM migration	[CFH+05] [TDG+06] [VBVB09]
	Cloud level	[NWC+08] [FVG+08] [BDF+03a] [MLVH+02] [LZLH09] [VAN08a] [JJH+09] [FRM+10]

Table 2.2: Classification of the work on energy-efficiency

2.2 Wired networking resources

The number of Internet users has increased 5 fold between 2000 and 2009[17]. In less than twenty years, the Web has become an essential means of communication for private companies, governments, institutions and other organizations.

The ever increasing number of Internet hosts calls for high performance end-to-end networks, which in turn increases the topology complexity, the number of core components to ensure performance and reliability, and consequently, the energy consumed by the networks [BO09]. In [BBDC11], the authors forecast that the energy consumption of telecommunication networks will grow 2.5 times by 2018 compared to 2009 (Figure 2.6).

In [GS03], with coarse approximations, Gupta & Singh have stated that transmitting data through wired networks takes more energy (in bits per Joules) than transmitting it via wireless networks. Energy is indeed one of the main concerns of wireless networks, whereas it is currently not the case of wired networks because they are not battery constrained. However, wireless networks are not renowned for their energy-efficiency, and thus, the fact that wired networks are not better is eloquent.

The energy issue is becoming more present in wired networks because of the need for maintaining network connectivity at all times [CGNG04]. This ever increasing demand in energy can

[17] source: `http://www.internetworldstats.com/stats.htm`

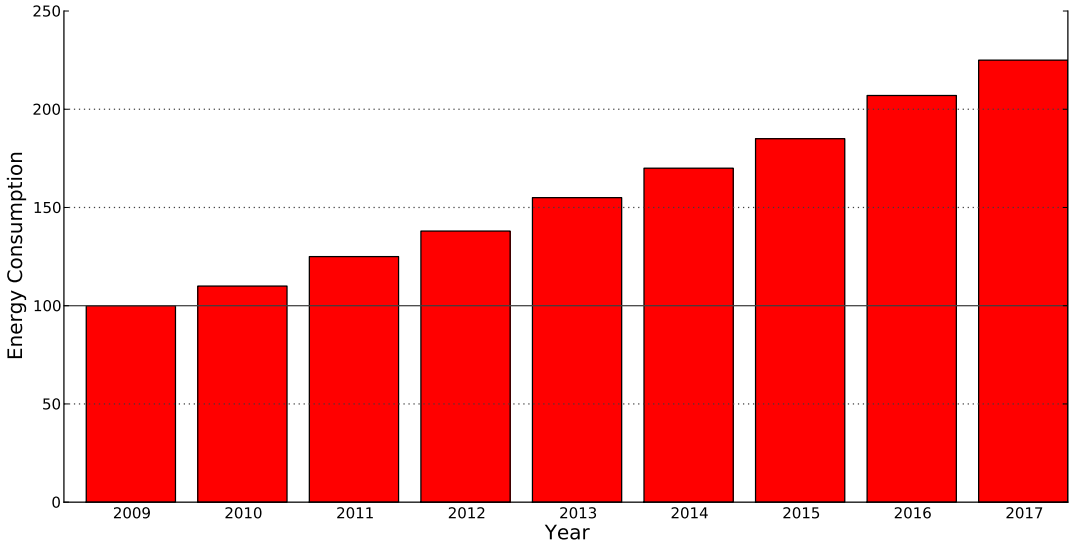

Figure 2.6: Energy consumption forecast for the optical networks (from [BBDC11]).

yet be greatly reduced. Studies have shown for a few years that network links, especially at the network edges, are lightly utilized [CGNG04, Odl03].

Moreover, the difference in terms of power consumption between an idle and a fully utilized Ethernet link is negligible [GCN05]. These observations have led researchers to propose several approaches to take advantage of link under-utilization and reduce the idle power consumption.

The energy consumption of networks is not only incurred by powering networking equipment (routers, switches, links, hubs, etc.), but also by end-hosts that demand high availability and full-time connectivity even if the network is not used.

Current research in energy conservation in wired networks aims to reduce the energy consumption of networking devices while guaranteeing the same Quality of Service (QoS) to users. This *same-for-less* approach is transparent to users whereas network managers, designers and providers should be aware of network usage and energy consumption in order to provide green network solutions.

Several methodologies can be used at the macro level with routing strategies and traffic aggregation and at the micro level with hardware improvements in NICs and switches, for example. However, interoperability and backward compatibility with existing protocols and products is also needed.

2.2.1 Measuring and modeling the energy consumption

Similar to computing infrastructures, before being able to save energy with new technologies and mechanisms, researchers and designers need to know how energy is consumed and wasted by network equipment. This preliminary analysis is key to understand how energy can be saved and to design energy models of network equipment that will be used to validate new hardware components and new algorithms.

Estimating and modeling the energy consumed by the Internet

The Internet is this familiar network that encompasses millions of private and public networks with different scales and functions, linked by various electronic and optical networking technologies. Due to its heterogeneity and to privacy and security constraints, the Internet's topology and density are hard to evaluate. As a result, precisely estimating the energy consumed by the Internet remains a challenge.

R. Bolla *et al.* have decomposed the Italian network into home, access, metro/transport and core networks [BBC+11]. They forecast the power consumption for typical network equipment by type of network, and the total number of equipment in the considered overall network (Figure 2.3). This allows us to deduce the energy consumption of the overall network and the portion induced by each its parts. Hence, a home network element is the least consuming, but is the most important contributor to the overall energy consumed by the network (79%) because of the large number of such equipment. Therefore, reducing the energy consumption of the least consuming appliance can still represent a lot of energy due to the scaling effect. Similar results are presented in [CML+10] based on the topology of an Italian Internet Service Provider (ISP).

	power consumption (W)	number of devices	overall consumption (GWh/year)
Home	10	17,500,000	1,533
Access	1,280	27,344	307
Metro/Transport	6,000	1,750	92
Core	10,000	175	15
	Overall network consumption		1,947

Table 2.3: 2015-2020 network forecast: device density and energy requirements in the business-as-usual case (BAU). Example based on the Italian network (from [BBC+11]).

Baliga *et al.* model the energy consumption of the Internet as a function depending on the access rate [BHT07]. According to their results, with a peak access rate of 100 Mbps, the Internet accounts for 1% of the electricity consumed in broadband enabled countries; a consumption that can increase up to 4% with a peak access rate of 1 Gbps.

On the client-side, Bianzino *et al.* estimate the power consumption of end-user PCs while browsing the Web depending on the hardware platform, operating system, browser and website. They point out that tabbed browsing cause several scripts to run in parallel while users typically interact with only one tab at any given time, thus wasting energy.

Estimating and modeling the energy consumed by certain types of networks

Due to the Internet diversity, it is easier to focus on a type of network (access networks, interconnection networks, backbone networks, etc.). As each network has its own role and well identified users, by focusing on a specific network type, parameters such as traffic, global architecture, types of interconnecting components are more easily determined.

In addition, constraining the study to a type of network allows to obtain a more precise view of the energy consumption of networking devices, and enables the development of energy consumption models for specific scenarios. In the following sections, we detail models for different types of networks. This organization in types does not cover the entire range of network categories, it just reflects the currently most explored research areas.

Optical networks

The growth of optical-network usage reflects the increasing demand in bandwidth by new Internet applications (e.g. e-science applications). The energy consumption of optical networks is thus becoming an important issue. In [BE09b], the authors proposed a model expressing the energy required to transmit an optical bit across a Wavelength Routed Node (WRN). Their basic assumption is that an optical network consumes energy in two cases: when transmitting an optical bit over fiber and when a WRN switches an optical signal. This model, which takes the bit error rate into account, is also applied in [BE09a] with Optical Burst Switched (OBS) networks that allow the dynamic sub-wavelength switching of data.

In [BAH+09], Baliga *et al.* present a power consumption model for optical IP networks, which they extrapolate to estimate the energy consumed by the Internet. Similarly to the Internet, Optical IP networks are split into three main parts: the access network, the metropolitan and edge network, and the network core. The power consumption model is described as a function of the access rate to users, and so, the global network consumption is the sum of the per user power consumption for all the components in the network. Thus, this model relies greatly on traffic estimation. A representative example is taken for each type of component and the power consumption of this component under typical load is included in the model to compute the total network consumption.

Backbone networks

Instead of studying the underlying network technology of backbones which are mainly optical networks [ZCTM10], the authors of [CMN09] consider a real IP backbone network with real traffic generated by one of the largest ISPs in Italy. The ISP network is divided into four levels: core, backbone, metro and feeder nodes. Even though core nodes are the most power consuming (per node), they represent less than 10% of the total energy consumption while feeders represent more than 65%, backbone nodes account for 19% and metro nodes for around 6%. These values do not consider air conditioning costs, which commonly increase the total power consumption by 40% to 60%.

Access networks

Access networks, also known as the *last mile*, connect end users to the telecommunication networks. The authors of [LKWG11] divide telecommunication networks into operator's network and home network, and provide energy consumption forecasts for these two parts. They show that access networks (including home networks and the access section of operator's networks) are the most consuming portion.

A consumption model of access networks is proposed in [BAS+08]. The authors consider several technologies: fiber to the home using Passive Optical Networks (PONs)[18], fiber to the node (FTTN)[19] combined with Very high speed Digital Subscriber Line (VDSL)[20], Point-to-Point (PtP) optical systems[21] and Worldwide Interoperability for Microwave Access (WiMAX)[22]. This analysis shows that the most energy efficient access technology that they considered is PON for low access rates and PtP for very high access rates, and FTTN gives the worst power per user and energy consumption per bit when it is operating at full capacity.

Interconnection networks

Interconnection networks are another well studied type of network used to relay data and control traffic between computation and storage elements in parallel computer systems [SPJ06]. The energy consumption model presented in [SEP05] for such network type is based on estimating the utilization of links and output buffers. Indeed, contrary to other technologies, in these networks, links are the dominant power consumers.

A holistic approach is presented in [KLY+05] where the authors present a power consumption model of the entire cluster interconnections based on energy profiles of the three major components: switches, NICs and links. This model leads them to propose energy-efficient designing techniques.

[18] A Passive Optical Network (PON) is a point-to-multipoint fiber in which unpowered optical splitters are used to enable a single optical fiber to serve multiple premises.

[19] Fiber To The Node (FTTN) is an architecture using a copper local loop linked with fiber at a distribution node.

[20] Very-high-bitrate DSL is a technology for high speed DSL over copper cables.

[21] Point-to-Point (PtP) optical systems designates fully optical based access networks.

[22] Worldwide Interoperability for Microwave Access (WiMAX) is a standard-based technology enabling the delivery of last mile wireless broadband access.

Theoretical graph models

A consensual and commonly adopted way to represent a network topology is to model it as an undirected graph where the vertices are the nodes and the edges are the bidirectional links [CCLM09, HFPJ10]. Each vertex and each edge has an associated power cost function with parameters varying from one to another using random theory. As the network topology is often unknown, random graph theory is used to generate graphs with particular properties that match the observed properties of these networks. Random graph theory can also be used to estimate the number of devices that can eventually be powered off to save energy [CCLM09].

Measuring and modeling the energy consumed by network devices

The most precise view of the energy consumed by wired networks is focused on network devices as it is possible to plug energy sensors at this small scale. Wired networks consist of several devices such as routers, switches, bridges, optical repeaters, hubs, firewalls, links, coaxial cables, optical fibers, twisted pair wires and network interface cards.

Each type of device has its own characteristics (architecture and functionalities) and services. Thus, each of these devices presents an energy consumption that is influenced by various parameters such as device type, traffic, number of connected devices, manufacturer, number of switched-on interfaces (ports for routers and switches), used protocols and QoS services (which can add processing time if high level operations such as flow identification and packet inspecting are required) and energy saving modes. The following sections describe models for different types of network devices. Here we cover only the most studied range of devices.

Routers

A router is a device that interconnects two or more networks and forwards data packets from one network to another (routing and forwarding tasks). Following the OSI model, the router is a layer 3 device.

Several models have been proposed to describe the energy consumed by a particular device. For example, the energy consumed by routers is studied in [SPJ06, WPM02, CSB+08] and different parameters are used to model this consumption. A general model for router power consumption depending on the number of active line cards is proposed in [CSB+08]. This model is based on real experiments with different line cards.

In [SPJ06], the authors take a different approach where the router actions are broken up into 7 basic operations: input buffer write, routing decision, virtual-channel allocation, switch arbitration, input buffer read, switch traversal and outgoing link traversal. Each operation dissipates dynamic power. Switch arbitration and allocation are not considered since they are negligible. Each of the remaining operations is detailed and mathematical formulas are provided for each power consumption. These models are integrated into their online power estimator embedded in each router in order to dynamically monitor local power consumption. Hence, these models replace an actual expensive wattmeter for each networking device. One can hope that such energy sensors will be directly included in future router architecture and will be available to router administrators.

The two previous power models are generic and can be applied to any router. In [WPM02], the authors design a power consumption model for CMOS (complementary metal-oxide-semiconductor) routers based on the switch capacitance and switching activity. To estimate the switch capacitance, the model includes FIFO buffers, crossbar switch and arbiters modeling, since these three components are the basic building blocks of routers. This model is then applied to two different commercial routers: the integrated Alpha 21364 and the IBM 8-port 12X Infiniband router. The model validation is compared with success to designers' estimates.

The crossbar switch is one of the possible switching fabric architectures. Switch fabrics are responsible for an important part of the router power consumption; 90% in the case of the IBM Infiniband router and 26-35% for the Alpha 21364 (input buffers contribute 46-61% of total

power) [WPM02]. In [YMB02], four widely-used switch fabrics are studied, namely crossbar, fully-connected, Banyan and Batcher-Banyan. The authors show that the fully connected switch has the lowest power consumption, and that the relation between power consumption and traffic throughput is almost linear for the crossbar, fully-connected and Batcher-Banyan networks.

Switches

A switch, commonly viewed as a layer 2 device in the OSI model, connects network segments in LANs. Quantitative models of energy consumption in several optical switching devices are provided in [Tuc10b]. These models decompose the device's consumption into the consumption of its hardware components (e.g. supply, transistors). The authors study two classes of switches: linear analog switches which pass the input signal to the appropriate output without altering the waveform of the signal, and digital switches that operate at the bit level and generally incorporate highly nonlinear and logic devices.

In [HCP09], the authors use linear regression to model the relation between the measured power consumption and the injected traffic in typical residential and professional switches. Some results are surprising: the Netgear residential switch (Netgear FS608v2, 8 ports) consumes less energy under high bandwidth values, and the 3Com professional switch (3Com 3824, 24 ports) achieves the optimum ratio between energy consumption and bandwidth between 100 and 1000 kbit/s. As a conclusion, they show that energy consumption and bandwidth are linked, but that the dependence is quite small and not linear. The number of end host devices plugged on the switch has a bigger impact on its consumption. For example, when the HP professional switch (HP ProCurve 2810-48G, 48 ports) is fully occupied (connected to 48 computers), its power consumption doubles.

This heterogeneous behavior has led Mahadevan *et al.* to propose a benchmarking framework in order to compare the power characteristics of a variety of network devices (routers and switches) [MSBR09a]. The power model for each device is based on the base chassis power, the number of linecards, the number of active ports, the port capacity, the port utilization, the Ternary Content Addressable Memory (TCAM) and the firmware. Per-port traffic characteristics are also taken into account by the traffic generator module: packet size, inter-packet delay and IP options set in the packet. This benchmarking framework is able to predict the power consumed by different switches currently used in data centers within a 2% error margin.

Digital Subscriber Line (DSL) systems

With the ubiquity of broadband residential Internet services and the increasing number of Internet users (around 1.8 billions in December 2009[23]), the number of deployed ADSL modems is exploding. The authors of [NB07] study the power consumption of typical ADSL modems under various load conditions, including minimum load (no physical wires plugged to the device), idle (with physical wires plugged but no user-initiated traffic), loaded (with various packets per second rates). For two Linksys ADSL modems (AG041 and WAG54G), heavy link load tends to decrease the power consumption compared to the idle state, which is not the case of the Cisco 837 modem. Hence, these findings are similar to the previous results on switches; devices are not really efficient when idle.

In DSL systems, the Line Drivers (LD) are responsible for 46% of the total power consumption of an ADSL2+ line [GNMP09]. Guenach *et al.* formulates a model for LD power consumption as a function of transmitted power [GNMP09]. In fact, LD power consumption highly depends on the Aggregate Transmit Power (ATP) of DSL lines.

Generic mathematical models for networking devices

As we go from core networks to access networks, the bandwidth capacity and number of elements per node decrease (number of other devices per node), but the number of deployed nodes increases. As both the number of nodes and their bandwidth capacity affect the energy

[23] source: `http://www.internetworldstats.com/stats.htm`

consumption, possible energy savings are greater at both core and access levels. In fact, in 2000 in U.S. [GS03], 95,000 LAN switches were deployed consuming 3.2 TeraWatt hours per year; and 3,257 routers were deployed consuming 1.1 TeraWatt hours per year.

However, when no empirical measurements of energy consumption are available for a given network device, researchers use generic mathematical models to evaluate their algorithms and frameworks. Such models express the relation between traffic load and energy consumption with simple functions, including linear, cubic, logarithmic, step functions [RGM09]. These energy profiles may not be realistic [MSBR09a] as they represent an ideal situation. Ideally, the power consumed should be proportional to the load, and so a proportional model is often used and referred to as the power consumption model for future networking devices [MSBR09a].

As seen in this subsection, each type of network device can be subject to a specific energy consumption model. However, similar devices from different manufacturers, with the same functionalities and capacities, can have different energy consumption profiles due to heterogeneous hardware components. No extensive study exists that gathers energy consumptions of a wide and representative range of switches and routers under widely varying traffic conditions and power management techniques.

Several models have been proposed for the different network devices [WPM02, AK08, SPJ06], for different types of networks [BAH+09, CMN09, BAS+08], and for the whole Internet [BO09, BHT07, CCLM09]. Combined with real energy measurements, they allow researchers to validate their new frameworks and algorithms.

2.2.2 Hardware and low-level optimizations

A way to reduce the energy consumption of a network is to increase the efficiency of its equipment. For that reason manufacturers of network devices are constantly proposing *green* routers and switches [AK08].

D-Link[24], Cisco[25], Netgear[26] are among the manufacturers proposing new *green* functionalities in their products such as adapting the power transmission to the link length and to the load, power off buttons, and more energy efficient power supplies.

These new products come often as result of green initiatives (GreenTouch[27], GreenStar Network[28], ECR initiative[29], etc.) and study groups (such as the IEEE 802.3 Energy Efficient Ethernet Study Group[30]) which aim to standardize and to enforce new regulations in terms of energy consumption for network equipment.

Several hardware solutions have been proposed from improving hardware capabilities (e.g. power saving by cable length, smart cooling FAN, disk spin down and DVFS) to new functionalities (e.g. power on/off button, low power mode, port auto power down).

Power saving according to a cable length is similar to the power saving modes of wireless devices that adjust their radio range according to the distance of their neighbors. The specifications of typical Ethernet twisted-pair cables specify a maximum length of 100 meters. Hence, router and switch ports are basically set for the maximal cable length, but at homes and enterprise environments, they are most commonly used with cables measuring a couple of meters.

Networking devices can also benefit from low-level improvements coming from research on energy-efficient computing resources [GS07b]. Power on/off buttons enable home equipment to be manually switched off when they are not used. Wake On LAN techniques allow them to be switched on and off remotely.

[24] http://dlinkgreen.com/energyefficiency.asp
[25] http://www.cisco.com/en/US/products/ps10195/index.html
[26] http://www.netgear.com/NETGEARGreen/GreenProducts/GreenRoutersGateways.aspx
[27] http://www.greentouch.org/
[28] http://www.greenstarnetwork.com/
[29] http://www.ecrinitiative.org/
[30] http://grouper.ieee.org/groups/802/3/eee_study/index.html

When ports are not used at their full capacity, they can be put in low power modes and when they are not used at all (no plugged link), they can be automatically powered down. This latter improvement is based on sleeping methods detailed in the next section.

Hardware optimizations also include re-engineering the network equipment to use more energy-efficient technologies and to reduce the complexity of current devices [Rob09].

2.2.3 Shutdown: sleeping methods

As discussed earlier, network links – especially those at the edges of a network – are lightly utilized [CGNG04, Odl03]. Therefore, to save energy researchers have proposed techniques to switch off network equipment (i.e. put them into sleeping mode) when they are not used [GS03, CMN08]. Despite its benefits, this technique raises several problems including connectivity loss and long re-synchronization periods. In addition, constantly switching equipment off and on can be more energy consuming than keeping them on all the time.

New mechanisms have been designed to address these issues. Examples of mechanisms include proxying techniques to maintain connectivity [NC10] and to quickly re-synchronized both ends of a link [GS07a]. Here we first review the main techniques and algorithms for switching off network devices. Then, we describe proxying techniques to enable longer switching-off periods.

On/off and sleeping techniques and frameworks

A seminal work dealing with on/off approaches has been performed by Gupta & Singh [GS03], where they proposed two solutions to reduce energy wastage:

- To switch off network interfaces of LAN switches during packet inter-arrival times, which requires to extend the interface's memory buffers to minimize packet loss.
- To modify routing protocols in order to aggregate all traffic going through parallel routes into one route during low-traffic periods and thus, to enable unused network links to be switched off.

The first solution, uncoordinated and passive, takes advantage of idle periods in traffic, whereas the second, coordinated and active, tries to increase these idle periods. Therefore, the first solution requires to improve low-level capabilities, whereas the second deals with the control plane. These approaches are not straightforward since the network presence of sleeping elements should be maintained. According to the experiments detailed in [GS07b], links of an on-campus backbone switch can be powered off from 40% to more than 80% of the time.

Observing that current network devices do not have power management primitives, the authors of [BBCL11] propose standby primitives (sleeping mode) for backbone devices using virtualization capabilities of layer 2 protocols (mainly represented by MPLS and Ethernet) and completely transparent to layer 3 protocols (such as IP). A network node is devoted to collecting traffic load information from routers, applying necessary reconfigurations, and switching elements on and off to meet QoS constraints[31]. This solution also requires periodic wake-ups of sleeping hardware for failure detections.

In [HWX+11], the authors propose reconfigurable routers to deal with traffic and route aggregation and management of routers under sleeping states. Dynamic Ethernet Link Shutdown (DELS) is presented in [GS07a]. It takes sleeping decisions based on buffer occupancy, the behavior of previous packet arrival times and a configurable maximum bounded delay.

The problem of finding the minimum set of nodes and links to be powered on while guaranteeing full connectivity and maximum link utilization is NP-hard and has been modeled in [CMN08]. This problem is also studied in [YSS10] with different QoS constraints (hop limit,

[31] QoS described in terms of maximum link utilization and backup availability while allowing the largest number of line cards to sleep

bandwidth limit reliability and stability). Numerous heuristics have been proposed to solve it [GS07a, NPI+08, SP03, CMN08, CMN09, RGM09], but most do not consider the practical problems of the on/off approach: switching on and off requires time, it leads to a network reconfiguration because of topology change, and a wake-up method is required to determine how and when nodes and links should be switched on again. To solve the latter problem, several solutions can be envisaged such as a periodic wake-up [NPI+08], an automatic wake-up on sensing incoming traffic [GS03], or a "dummy" packet sent to wake up a neighboring node [GGS04].

Instead of a sleeping mode, the IEEE 802.3az task group[32] proposes in [CRN+10] Low Power Idle (LPI), a standard that has been adopted in September 2010 [RCRM11]. The basic idea – to transmit data as fast as possible to spend as much time as possible in low power idle mode – is based on the following principles: the highest rate provides the most energy-efficient transmission (in Joules/bit) and the LPI mode consumes minimal power. The LPI transition is initiated by the transmitter. Then, a periodical refreshment is done to detect failures and facilitate fast transition. Transmissions are deferred to pre-defined time as shown in Figure 2.7 presented in [RMHL10].

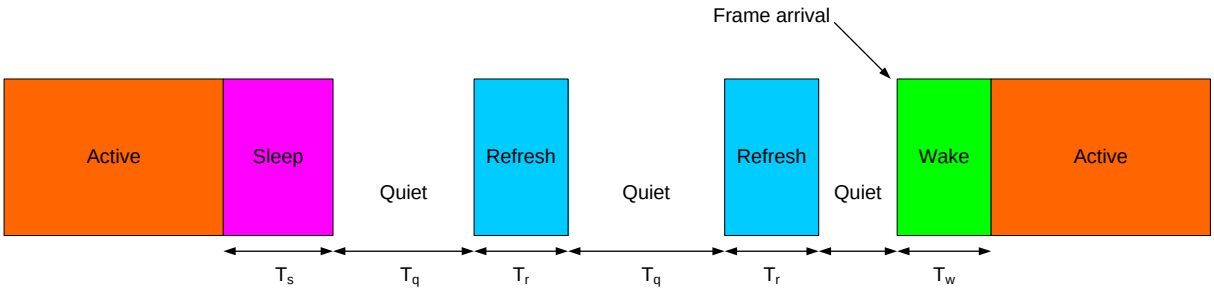

Figure 2.7: Transitions between the active and sleep modes in Energy-Efficient Ethernet (from [RMHL10]).

T_s is sleep time (the time needed to enter sleep mode); T_w is wake-up time (the time required to exit sleep mode). The transceiver spends T_q in the quiet (energy-saving) period but also has short periods of activity (T_r) to refresh the receiver state.

Proxying techniques

In 1998, a study showed that PCs are generally powered on even when they are not in use; an observation that has led Irish & Christensen [IC98] to look for a solution consisting in completely powering off PCs during idle periods. Even though powering off network elements such as links or switches is not yet commonplace, switching off PCs include powering off their NICs, and thus dealing with loss of network connectivity.

A way to address these issues is to use a proxy to answer to unimportant messages on behalf of a node and to wake-up the node when required. This solution, detailed in [CGNG04, GCN05, JCN08], is based on a Network Connectivity Proxy (NCP) that handles network presence requests such as ARP, ICMP and DHCP, and keeps connections alive. This proxy solution can be implemented in the NICs themselves as proposed in [SGRO+08]. Agarwal *et al.* propose an improved NIC that answers to network requests while the node is in suspend-to-RAM state [AHC+09]. Another approach illustrated by [ASG10] is to use dedicated on-demand proxy servers.

Taking advantage of low link utilization and path redundancy, the shutdown approach uses idle periods in network traffic. Energy savings can also be made during low-demand period with the slowdown approach.

[32] The IEEE 802.3az (Energy Efficient Ethernet) task group develops standards for reducing power consumption of Ethernet devices (`http://www.ieee802.org/3/az/public/index.html`).

2.2.4 Slowdown: adapting to the needs

The difference in power consumption between an idle and a fully utilized Ethernet link is negligible [GCN05]. To keep communicating NICs synchronized, idle bit patters are continuously transmitted even when no data is sent. As a consequence, the power consumption levels of non-energy aware equipment are the same when idle and transmitting.

An exception are the 1000BASE-T specifications, designed to operate at 10 Mb/s, 100 Mb/s and 1 Gb/s in order to keep backward compatibility with previous specifications. It has been observed that running Ethernet links at lower data rates decreases the power consumption of NICs and switches [ZSGRG08].

In the same way, to reduce heat dissipation at local switching stations to which numerous ADSL2+ links lines are connected, the ADSL2+ specification is designed to support several power modes: L0, L2 and L3 [ITU]. Each mode leads to a different energy consumption of the link: 3 Watts for L0, 0.5 Watts for L2 and 0.3 Watts for L3. However, no standard policy has been adopted to determine when a link should switch to another power mode, and thus, these capabilities are not used [Gin05].

Similarly to computing resources, the energy consumption of networking devices is not proportional to the usage [GCN05]. To get closer to proportionality, the transmission rate of Ethernet devices can be adapted to the load, like with DVFS techniques. This technique is called Adaptive Link Rate (ALR) [GCN05, GCS06]. A high and a low buffer thresholds are determined; when the buffer occupancy reaches the high buffer, the link rate is switched to a higher value, and when it goes under the low buffer, the link rate is decreased. The difficulty lies in finding the good values for these thresholds in order to avoid packet losses and oscillations since rate switching takes times. An ALR prototype is implemented in [ZSGRG08] showing that switching times are in the order of milliseconds.

A similar approach, called Dynamic Adjustment of Link Width (DAWL), is proposed by Alonso *et al.* [AMSL04]. The thresholds are also based on link utilization. Nedevschi *et al.* provide a rate adaptation mechanism [NPI+08]. As outlined by the authors, the distribution of operating rates and their corresponding power consumption greatly influence the efficiency of slowdown techniques.

2.2.5 Coordination: network-wide management and global solutions

The optimization, shutdown and slowdown approaches focus on specific network elements. However, energy efficiency improvements have also to be made at wider scales. For example, routing algorithms [CSB+08, RGM09] and network protocols [IC98, BC09] can be improved to save energy.

Coordinated power management schemes benefit from previously cited techniques, such as on/off and adapting rate techniques, and take decisions at a wider scale. This implies that they make greater energy savings. For example, in low-demand scenarios with network redundancy, entire network paths can be switched off, and the traffic is routed on other paths [IOR+10, SPJ06].

Figure 2.8, from [BBDC11], shows different options for the same transfer scenario:

(a) no power-aware optimizations;

(b) shutdown approach;

(c) slowdown approach;

(d) shutdown and slowdown approaches combined.

Both shutdown and slowdown approaches increase the total transfer time and reduce the energy consumption. The combination of these two techniques gives the best results in terms

of energy consumption and the worst with regard to transfer time. However, depending on the QoS constraints, this delay can be acceptable and can be shortened using traffic prediction. An analysis of these two methods is provided in [NPI+08] showing that both methods can be beneficial depending on the power profile of the network equipment and the utilization of the network itself.

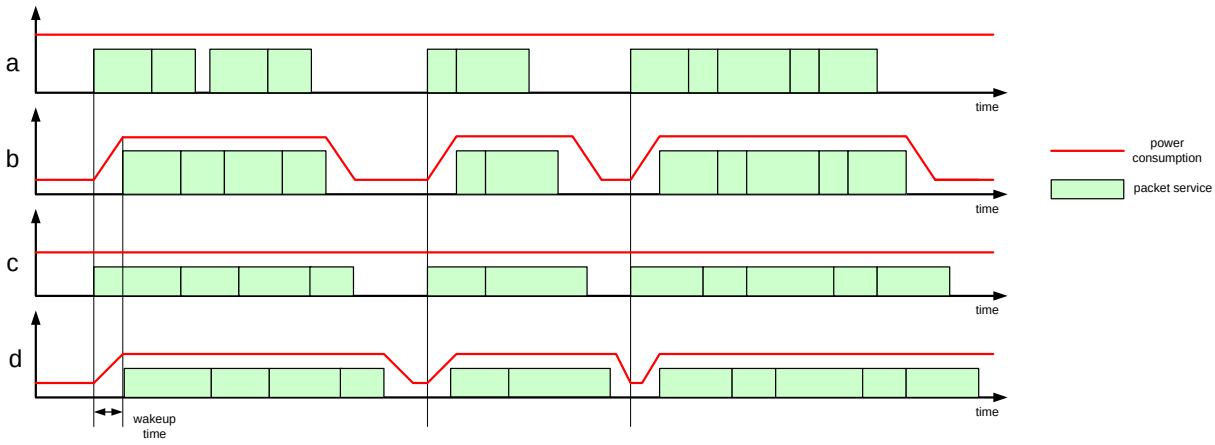

Figure 2.8: Packet service times and power consumptions in the following cases: (a) no power-aware optimizations, (b) only idle logic, (c) only performance scaling, (d) performance scaling and idle logic (from [BBDC11]).

Among the solutions to improve the energy efficiency at a network scale, we distinguish three categories: the green design of network protocols which concern improvements of current protocols to make them more energy-efficient; the clean-slate approaches which propose new energy-aware network architectures, and the energy-aware frameworks which consider the entire network consumption as the quantity to optimize.

Green design of network protocols

Each network layer has its own protocol and associated energy cost.In [WS04], the authors analyze the energy consumption of TCP and isolate the cost of each operation (e.g. copies, computing checksums). They show that several techniques can be employed to reduce the consumption, such as zero copy, maintaining the send buffer on the NIC and copying data in large chunks.

In 1998, Irish & Christensen [IC98] proposed a "Green TCP/IP" with a connection-sleep option; an option that works on the basis of client-server applications with connection-oriented communications. It allows a power-managed TCP/IP client to notify a TCP/IP server that it is going to a sleeping power mode. The server thus keeps the connection alive, but it does not send any packet to the sleeping client.

Energy savings can also be made at the routing protocol level. In [CEL+10], the authors adapt Open Shortest Path First (OSPF) for allowing routers to power off some network links during low traffic periods. A power-aware routing protocol for IP routers using line card and chassis reconfiguration is proposed in [CSB+08]. A centralized routing and on/off management is described in [GS09]; it presents good QoS results in terms of delay, packet loss and jitter.

At the application level, the BitTorrent[33] protocol is also the subject of considerable research work [ACGP09, AGP10, BC09]. In [ACGP09, AGP10], the authors consider using a proxy to delegate the download operations while powering off the user's node, while in [BC09], the authors modify the BitTorrent protocol to save energy.

[33] BitTorrent is a peer-to-peer communication protocol for file sharing.

Traffic grooming is used in optical wavelength routing network to group flows to avoid optical-electronic-optical conversions (OEOs) and thus saving energy. In [ZSC10], the authors compare static and dynamic grooming policies under various loads, and the best approach depends on the considered scenario. The authors of [VHDGV+10] show that optical end-to-end grooming consumes about half the power of the link-by-link grooming on a realistic scenario.

Clean-slate approaches

The IP was designed in the 1970s. By 1980, around 200 hosts were connected and nowadays, there are around 2.198 billion Internet users in the world[34]. New network protocols need years to be deployed and hence, they may have been designed to traffic conditions that are not the case by the time they are deployed. Clean-slate approaches disregard current design constraints (such as inter-operability) and propose new innovative architectures that improve the QoS of current and future applications.

These new architectures comprise bufferless routers [HJL09], optimal network design for specific applications [BAHT09], synchronous IP switching to synchronize router's operations and schedule traffic in advance [BO09], pure optical switching architectures [Tuc10b] and routing algorithms dealing with flows instead packets [Rob09].

Energy-aware frameworks

To be energy efficient, network-wide solutions should be:

- adapted to the network topology (e.g. redundancy, multi-paths);
- adapted to the traffic (e.g. bursts, always low);
- adapted to the scenario usage (e.g. P2P, Web servers);
- realistic in terms of technology (e.g. compatibility, interoperability);
- scalable, reliable, fast, fault-tolerant, efficient and secure.

Coordinated power management schemes benefit from previously cited techniques, such has on/off and adapting rate techniques, and take decisions at a wider scale, which implies that they make greater energy savings. For example, in low-demand scenarios with network redundancy, entire network paths can be switched off, and the traffic routed on other paths [IOR+10, SPJ06, SWHK08, CM10]. However, these solutions require major changes such as dynamic routing protocols that can handle shut down nodes and links, a centralized management to coordinate switch off [GS09] and higher-level cooperation between Internet Service Providers (ISP) and Content Providers (CP) [CM10].

Network virtualization is also a promising solution to enable energy savings. In [TAG+11], the authors propose an energy-aware planning of virtual infrastructure (VI)[35]. They show that their framework can save up to 40% of energy. Virtual ROuters On the Move (VROOM), a network management primitive is presented in [WKB+08]. It simplifies the management of routers by virtualizing them and allowing them to freely move from one physical node to another, thus avoiding logical topology changes. The migration of virtual routers can also be used to save energy [CP11].

To summarize, Table 2.4 proposes a synthesis of the research work discussed in this section. The contributions are categorized by network level and type of approach. The three first approaches (hardware, shutdown and slowdown) are applied at the node level whereas the last one

[34] source: `http://www.worldometers.info/`

[35] The objective of VI planning is to identify the topology and determine the virtual resources required to implement a dynamically reconfigurable VI based on both optical network and IT resources.

(coordination) is applied at the network level. The framework presented in Chapter 4 enters in the network design category because it is based on reservation-based networks that differ from current networks.

		Access networks	Core and backbone networks	Network design	Applications
Hardware	Improving and re-engineering	[GS07b]	[AK08] [Rob09]		
Shutdown	On/off and sleeping	[GS03] [NPI+08] [GGS04]	[BBCL11] [HWX+11] [CMN08] [YSS10] [SP03] [CRN+10] [GS07a]		
	Proxying	[IC98] [CGNG04] [JCN08] [SGRO+08] [AHC+09] [ASG10]			[GCN05]
Slowdown	Rate adaptation	[ZSGRG08] [GCS06] [NPI+08]	[AMSL04]		
Coordination	Network protocols	[IC98] [WS04]	[CEL+10] [ZSC10] [VHDGV+10]	[CSB+08] [GS09]	[ACGP09] [AGP10] [BC09]
	Clean-slate		[BO09]	[HJL09] [Tuc10b]	[BAHT09]
	Frameworks	[CCLM09] [SPJ06]	[CMN09] [CM10]	[SWHK08] [RGM09] [WKB+08] [TAG+11] [CP11]	

Table 2.4: Taxonomy of the energy-efficient works per network level and approach

2.3 Conclusion

This chapter discussed approaches for computing and networking resources to save energy in large-scale distributed systems. The existing solutions have been classified in several levels.

These energy-efficient methods can be combined to have a further impact on energy consumption without impacting application performance and resource utilization. However, the main leverage to reduce the electric bill and the carbon footprint is still to increase the energy awareness of users, network providers and designers. That remains the best way to enforce good (green) practices and to encourage standardizations on this subject.

No law or ordinance is mightier than understanding.

Plato

3

Instrumenting and Understanding Power Consumption of Large-Scale Distributed Systems

Chapter 2 has discussed various available techniques to reduce the energy consumption of distributed systems. As observed, in the quest to find means to reduce consumption, the first step is to identify how energy is consumed by these systems.

Targeting mostly application performance, distributed systems have constantly increased in size and in computing power of their resources. The power supply requirements of these systems have increased in a similar fashion, which has raised concerns about the energy they consume. It thus becomes essential to instrument and monitor such systems in order to understand how energy is used. The instrumentation can provide application and system designers with the information they need to understand the energy consumed by their applications and design more efficient mechanisms and policies for resource management. However, monitoring large-scale distributed systems such as Grids remains a challenge.

When setting up an infrastructure to understand how distributed systems consume energy, our requirements included (1) wattmeters that can provide one measurement per second and per device to offer a fine-grained view of consumption; and (2) software tools to store, manage, and expose this information. When our study on energy consumption started, neither the wattmeters nor the software tools were available.

As of writing, servers embed several energy sensors that can monitor the power consumed by internal components such as CPUs. However, the servers of our platform, acquired in 2006, did not possess such sensors. Therefore, we opted to use external wattmeters specially designed by a SME that produces electrical equipment. Although nowadays Power Distribution Units (PDUs) are equipped with components that can monitor the power consumption, when we set up our infrastructure, the available PDUs did not offer the granularity and precision that we required for our experiments.

With regard to software, although various solutions have been proposed for monitoring the usage of resources (e.g. CPU, storage and network) to improve the performance of applications [ZS05, ADBF+05, TF04], only a few systems monitor the energy usage of entire infrastructures [SV09]. Therefore, we designed our own software components for the energy-sensing infrastructure. When deciding on the required data management techniques, the following goals were considered:

- Increase users' awareness on the energy consumed by their applications;
- Improve the design of user applications;
- Obtain data to advance Grid middleware;
- Help managers in implementing power management policies; and
- Create an energy-data repository for further studies.

This infrastructure provides a deeper understanding of the energy consumption of computing resources, particularly about the relation between consumption and load.

The rest of this chapter is organized as follows. We present our experimental testbed for Grid'5000 in Section 3.1. Section 3.2 explains the architecture of our monitoring infrastructure, whereas Section 3.3 details the software used by this monitoring infrastructure. Section 3.4 starts the analysis of the power consumption by exposing some myths and facts about the node's consumption. We conclude this chapter in Section 3.5.

3.1 Grid'5000

Grid'5000 [CCD+05], depicted in Figure 3.1, is an experimental platform supported by the French Ministry of Research, regional councils, INRIA, CNRS and universities. It provides a nation-wide platform distributed across 9 sites, containing 30 clusters with more than 6,200 CPU cores, interconnected through 10Gb/s links supported by the Renater Research and Educational Network[1].

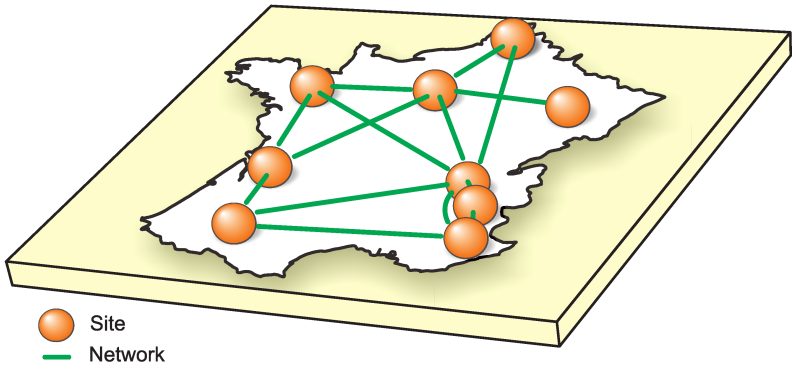

Figure 3.1: The Grid'5000 infrastructure in France.

This platform can be defined as a highly-reconfigurable, controllable and monitorable experimental Grid. Its utilization is specific: each user can reserve a number of nodes on which he deploys a system image; the nodes are entirely dedicated to the user during his reservation. The Resource Management System (RMS) used by Grid'5000 is called OAR [CDCG+05]; an open-source batch scheduler that provides a simple and flexible interface for reserving and using cluster resources[2]. Users can submit best-effort jobs, started by OAR when resources are available, and advance reservations where users specify the desired start time of their requests. Figure 3.2 illustrates one of the interfaces provided by the Grid'5000 API, showing the status of the platform. The API also gives users access to information about each node, such as CPU, memory, network, disk and swap usages. However, to obtain this information a daemon must run on each reserved node. As running this daemon is optional because it can interfere with user experiments, the information may not be available for all reservation requests.

Although Grid'5000 is different from a production infrastructure, the concerns on energy consumption are similar to those of production environments. Therefore, solutions proposed to tackle challenges in Grid'5000 can fit both experimental and production infrastructures.

3.2 Instrumenting a distributed infrastructure with energy sensors

The energy-sensing infrastructure,[3] deployed on three Grid'5000 sites (i.e. Lyon, Grenoble, and Toulouse) monitoring 160 computing and networking resources, comprises several wattmeters

[1] `http://www.renater.fr`

[2] `http://oar.imag.fr/index.html`

[3] The infrastructure is financially supported by the INRIA ARC GREEN-NET initiative and the ALADDIN/Grid'5000 consortium.

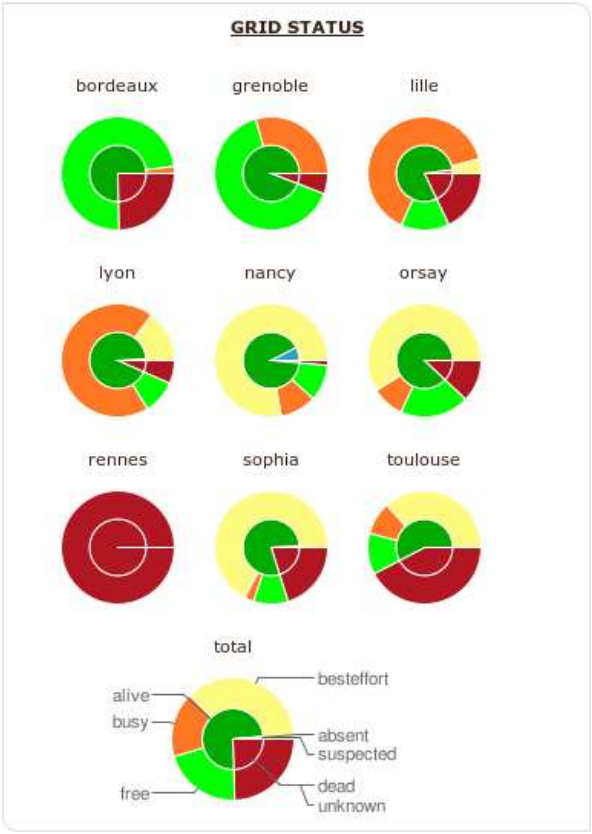

Figure 3.2: Example of information provided by Grid'5000 API.

manufactured by Omegawatt[4]. A dedicated server, connected to the wattmeters via a serial link, stores the gathered data and generates live graphs of energy usage.[5]

In the Lyon site, where the whole infrastructure is monitored, the wattmeters periodically measure the energy consumed by 135 nodes and a router. Four Omegawatt boxes where required; a 48-port box, two 42-port boxes and a 8-port box. Each box works as a multi-plug measurement device, where a physical host is connected to a port responsible for measuring its power consumption. A data collector (i.e. a remote server) communicates via a serial link (i.e. RS232) with the boxes in order to retrieve the data on energy consumption.

On the data collector, a daemon for each box is responsible for collecting the energy data. At every second, this daemon sends a data request – an hexadecimal code with a CRC – to the box to ask for the information on energy consumption of all the resources monitored by the box. The response is an hexadecimal code with a CRC in the form of "packets", each packet containing the instantaneous power consumption of six nodes. The software responsible for communicating with the wattmeters for obtaining the data has not been provided with the boxes. We had to design it by ourselves for collecting at every second the measurements of 135 nodes including CRC checking, hexadecimal decoding and storage of these values. The data gathering process should not exceed one second which is the interval between two measurements.

As the goal of instrumenting the Grid infrastructure has been to provide information on energy consumption to different groups of users and system administrators, several data management techniques had to be applied. For example, the energy data is stored in three different ways:

- Raw data: that includes a timestamp indicating when a measurement was performed

[4] `http://www.omegawatt.fr/gb/index.php`

[5] A data collector is responsible for collecting the data from the boxes every second.

(there is one log file per resource).

- Last values: one file (in memory file systems) per resource and the value is over-written at each second.
- Round-Robin Databases[6]: one file per monitored node.

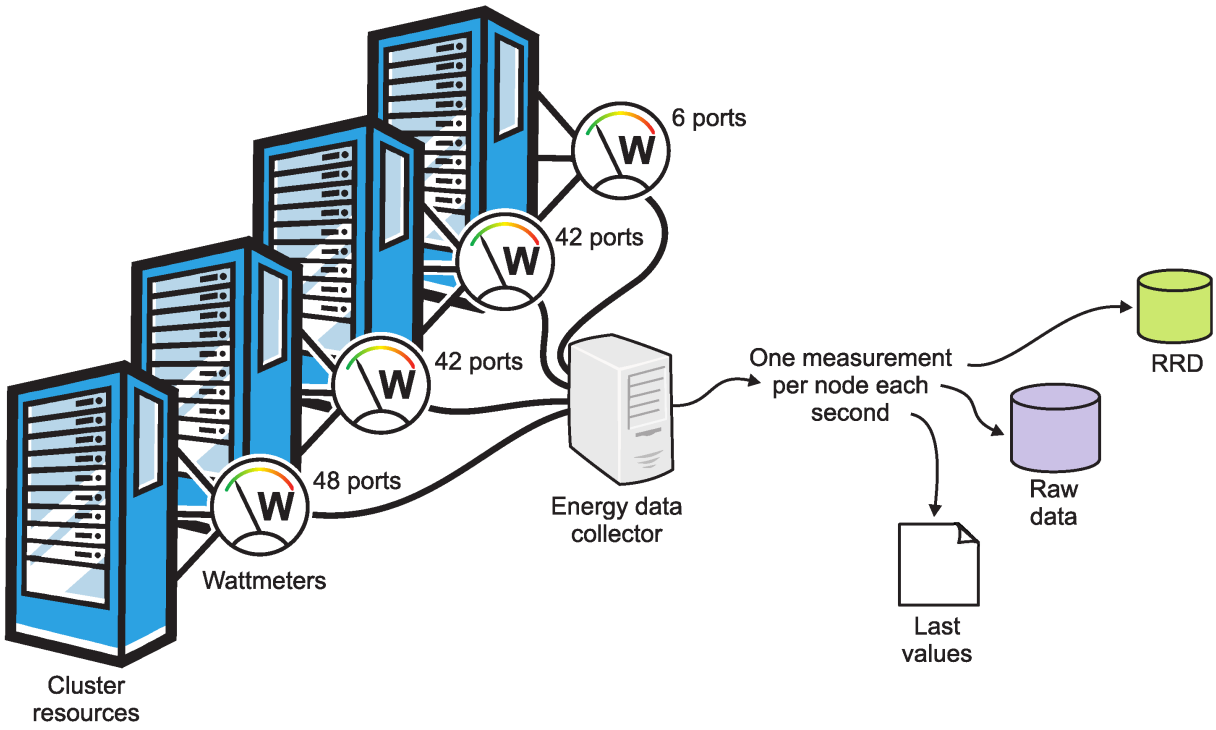

Figure 3.3: Energy data collector deployed on the Lyon site.

The data collection and storage process for the Lyon site is illustrated in Figure 3.3. As each user category has different requirements, the frequency of measurement is important and must be calibrated depending on what the users want to observe. For administrators who want to observe global energy trends, a few measurements per hour or per day can suffice. Users who want to understand the usage and the impact of applications on energy consumption, demand several measurements per minute. For middleware designers who want to understand peak-usage phenomena (during a system deployment phase for example), multiple measurements per minute are mandatory. Moreover, in this case the data must be stored to allow them to evaluate trends and patterns that can aid the improvement of middleware tools. Therefore, to fulfill the requirements of different user categories, in the Green Grid'5000 one live measurement (in Watts) is performed every second for each monitored resource with a precision of 0.125 Watts. One year of energy logs for a 150-node platform corresponds to approximately 70 GB of data.

These energy logs can be correlated with the reservation logs (provided by OAR) and the resource usage logs provided by the Grid'5000 API in order to have a more complete view on the energy usage.

3.3 Displaying information on energy consumption

Measuring energy consumption and storing the data are only part of the process when the goal is to reduce the amount of electricity consumed by an infrastructure. Ideally, the obtained

[6]RRDtool is the OpenSource industry standard, high performance data logging and graphing system for time series data (`http://oss.oetiker.ch/rrdtool/`).

information on energy consumption should drive optimizations in resource management systems; inform users of the cost of their applications in terms of electricity spent; and possibly offer means to raise the users' awareness and adapt their behavior so they can make more conscious choices when executing their applications. In addition, the information should be presented to users in a way they can measure the impact of their choices in terms of energy consumption when they run their applications differently from the manner they are used.

Towards achieving these goals, we have developed a set of tools for visualizing the data collected from instrumenting the Grid platform [DdAGLO10]. This section describes a graphical interface, termed as *ShowWatts*, for displaying the measurements in near real-time, and a list of Web pages and graphs for viewing historical data on energy consumption. The first interface allows users to view the instant energy consumption of the whole platform and check the immediate impact of executing an application on a set of resources; whereas the Web pages and graphs enable the analysis of energy consumption over larger periods and the identification of patterns and application behaviors. This work has been made in collaboration with Marcos Dias de Assunção and Jean-Patrick Gelas.

3.3.1 ShowWatts: live display

ShowWatts comprises a Java Swing application for displaying the energy consumption of the instrumented platform in nearly real-time, and a module responsible for aggregating the energy consumption information and transferring it to the node where it is displayed. It follows the client-server model where a daemon is launched on the server side and another on the client; the client daemon uses the "last values" described in Section 3.2. These two processes communicate through an SSH tunnel. The server side part is optimized to minimize the data volume sent: only the values that differ from the previous transfer are sent.

From a system administrator's point of view, this interface can be used to monitor the whole platform or the consumption of specific resources. The administrator can select the resources in which she is interested and view their share of the overall energy consumption. In addition, it is possible to view other statistics such as a rough estimate on the money spent with electricity.

Additionally, the application is customizable and can be changed to display information on other platforms. This is achieved by implementing the module that aggregates the consumption information and changing two XML files; one that describes general configuration and display preferences and another that specifies the monitored resources. For demonstrative purposes, we have implemented a reservation scheduler in python, which shows the impact in energy consumption when switching machines off and on, and aggregating reservation requests.

3.3.2 Portable web-based visualization

As mentioned earlier, in addition to displaying the instant energy consumption, visualization tools must provide means whereby users and system administrators can analyze past consumption. By visualizing the energy consumed by individual resources or groups of resources over larger periods, administrators and users can better know the behavior of their platform, design more energy-efficient allocation policies and investigate possible system optimizations.

We have developed a set of scripts that automatically generate Web pages and graphs for displaying the energy consumption of individual resources and groups of resources. Figure 3.4 shows an example of Web page containing a list of graphs with measurements taken from a Grid'5000 site. These graphs are updated every minute so that the user has an up-to-date view of the platform in terms of energy consumption.

As discussed in Section 3.2, to collect the energy measurements, we have developed a software to interact with the wattmeters and to store the received information in different formats: RRD databases and text files.

Figure 3.4: Web interface for energy usage visualization.

RRD databases offer the advantage of having a fixed size as they use only average values for generating graphs of long periods. They give an overview of the energy consumed by the platform. They provide different views of the nodes' energy consumption (i.e. per hour, per day, per week, per month and per year) wherein a view presents maximum, minimum and average values. The current value per node (in Watts) is also displayed and updated every second (by an ajax script) as well as the global value for the whole site. These tools are particularly suitable for administrators as they provide a quick overview of the whole platform with an in-depth access to the most recent values for each node.

3.3.3 Collecting and providing energy logs

While measuring the power consumption of a Grid is the first step to understand the energy usage of the platform, the following step is to collect and store this data and give users access to it.

The RRD aggregated views, are not precise enough to understand the link between running an application on a node and the energy it consumes. Hence, text files are used to store each measurement (i.e. for each node at each second with a timestamp). These text files are archived and compressed every week in order to reduce the disk space required (*logrotate* mechanism); this allows us to keep all the logs.

While collecting and storing energy logs is important, the main goal is to provide these logs to users, so that they can see the impact of their applications on energy consumption of the platform. This may increase users' awareness about the energy consumed by large-scale infrastructures. Therefore, the monitoring Website includes *log-on-demand* features. As shown in Figure 3.5, the Web portal entails a Web page where a user queries the energy consumption of servers by specifying the period, the time increment (i.e. value in seconds between two measurements) and the nodes. These tools are specially designed for the users and the middleware developers who want to precisely profile their applications in terms of energy usage and to correlate the energy data with resource (e.g. CPU, memory, disk, network, etc.) utilization to view the impact of their programming decisions on energy consumption.

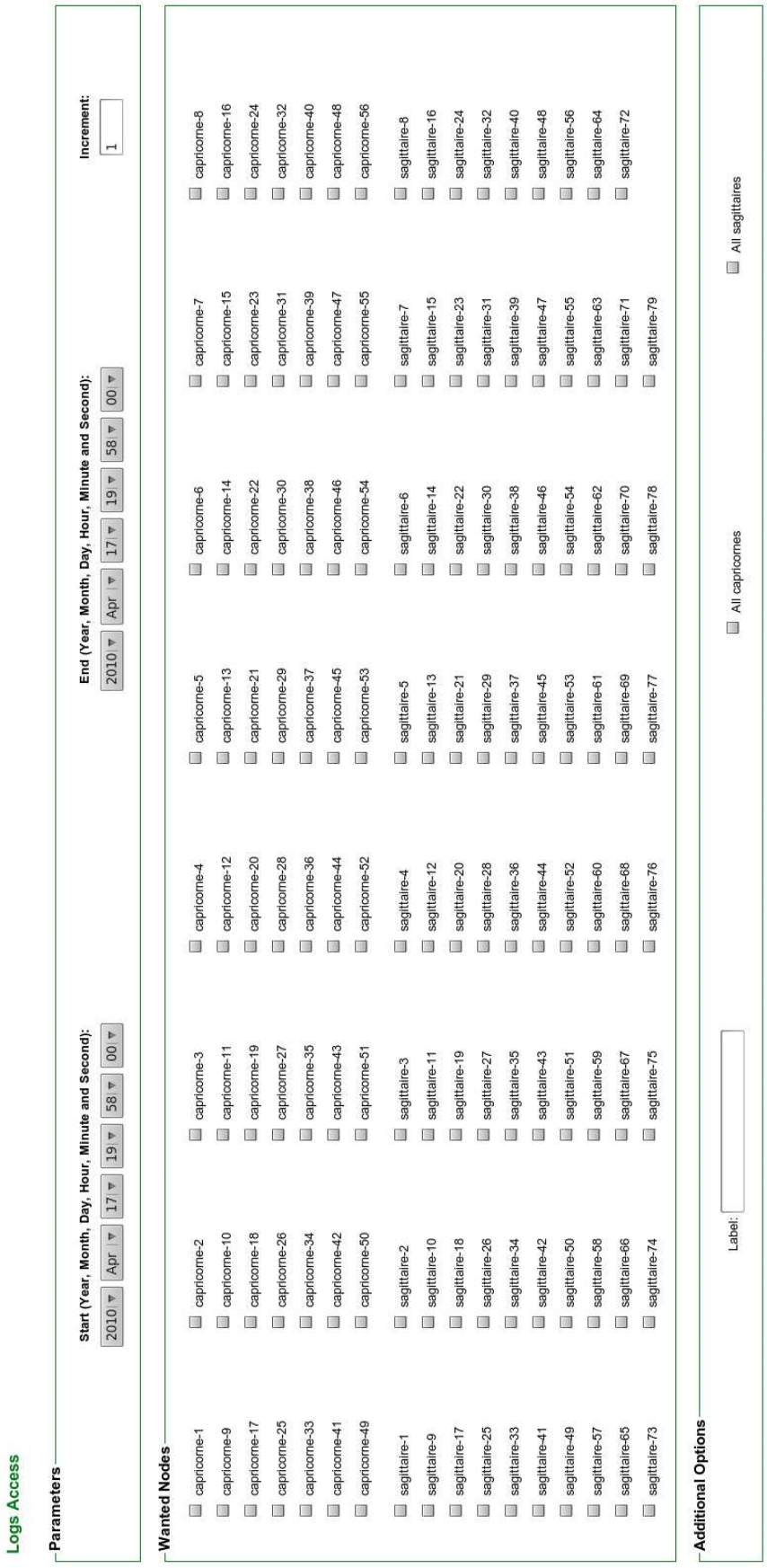

Figure 3.5: Web interface for energy-log requests.

The logs and graphs are available for all Grid'5000 users and administrators. Figure 3.6 presents the typical result of a log request. First, the global information of the request is given, such as the global consumption of the requested nodes during the given time frame. Moreover, for each node, the energy graph is provided to show the energy profile of the user's request.

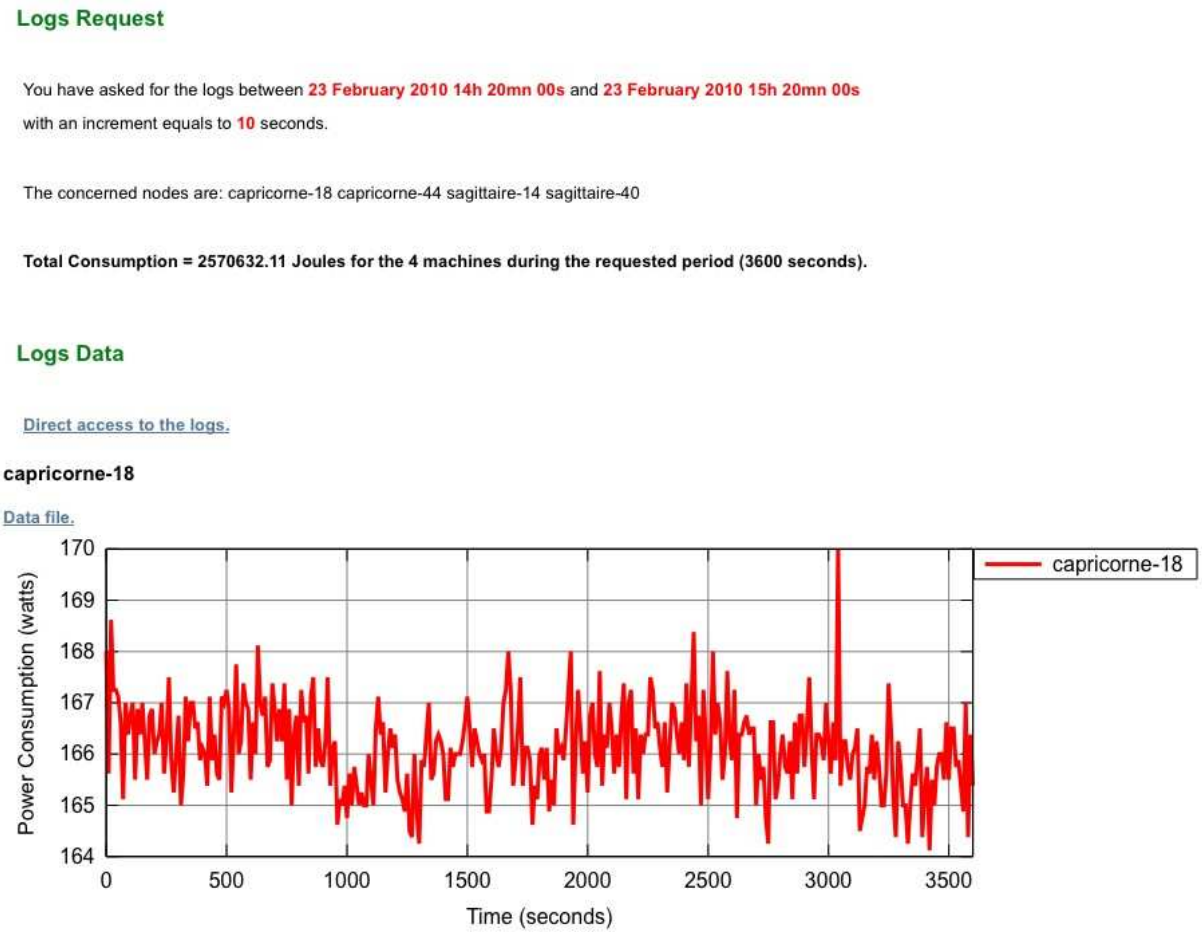

Figure 3.6: Web page displaying the result of an energy-log request.

As observed, having a configurable sensing infrastructure is one of the basic components mandatory for designing energy efficient software frameworks. The Green Grid'5000 is the first available large-scale platform whose energy is monitored per plug every second.

Several techniques have been applied to obtain, store and manage the data resulting from monitoring the energy consumed by the infrastructure. The different manners in which the data has been stored and reported aim to serve the needs of several categories of users, including application developers, middleware designers, system administrators and infrastructure managers. In addition to energy consumption data, obtaining and analyzing information from multiple sources (e.g. schedulers and resource utilization) is essential to understanding the usage of the infrastructure and applications' behaviors. This analysis can help system designers and managers to identify how to devise and implement techniques for reducing the cost of energy consumption. It also allows them to evaluate the return of investment of deploying the sensing infrastructure itself. At present, instrumenting a platform is expensive and its cost can surpass the savings achieved by striving to improve the energy efficiency of the overall infrastructure.

Once the sensing infrastructure is set, it can be used to examine how the energy is used in distributed systems.

3.4 Understanding the energy consumption of distributed resources

Numerous studies focus on reducing the energy consumption of large-scale infrastructures (e.g. data centers, Grids, Clouds) and improving these systems' energy efficiency. To address the challenges facing this field, research on energy efficiency must rely on serious and proved assumptions. When reviewing the state of the art, we observed basic and rapid hypotheses on energy consumption and efficiency of computing and networking infrastructures that risk being accepted as *de facto*. This section lists and clarifies some of these "myths" by confronting them with realistic measurements and analysis that we have made to understand the energy consumption of distributed resources[7].

3.4.1 Homogeneous servers and energy consumption

Substantial work assumes that homogeneous nodes have the same power consumption [SWHK08, KBK07], whereas in practice, we observe a different reality. We have dynamically collected the consumption in Watts of 3 sets of homogeneous servers: two IBM eServer 326 (2.0 GHz, 2 CPUs), two Sun Fire v20z (2.4 GHz, 2 CPUs) and two HP Proliant 385 G2 (2.2 GHz, 2 dual core CPUs). The external wattmeters described earlier were used to measure the power consumed by the servers.

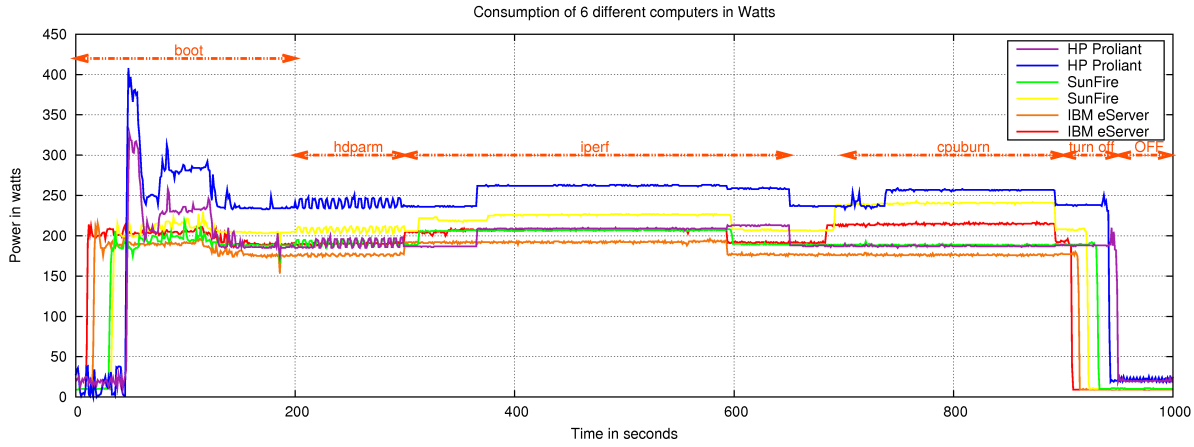

Figure 3.7: Consumption of 6 servers running typical applications.

Figure 3.7 shows that homogeneous servers have different consumptions when performing tasks that are representative of the life-cycle of servers. The tasks include nodes being switched off (but remain plugged to the socket), booting, intensively accessing the disk (`hdparm`[8]), undergoing intensive network communication (`iperf`[9]), or having intensive CPU usage (`cpuburn`[10]).

Figure 3.8 shows the servers' consumption when they are idle (performing no work, but switched on) for a long period of time – here termed as the *idle consumption* of a node. It is important to point out that the idle consumption of anode can account to up to 90% of the power it consumes when performing a task.

The consumption of a node does not only depend only on its architecture and the applications it is running, but also on its position on the rack and its temperature, for example. A cool node

[7] This list is not exhaustive.

[8] hdparm is a command line utility running under GNU/Linux to set or view the IDE hard disks' hardware parameters and stress the disk.

[9] Iperf is a commonly-used network tool to measure TCP and UDP bandwidth performance that can create TCP and UDP data streams.

[10] cpuburn is a tool designed to heavily load CPU chips in order to test their reliability.

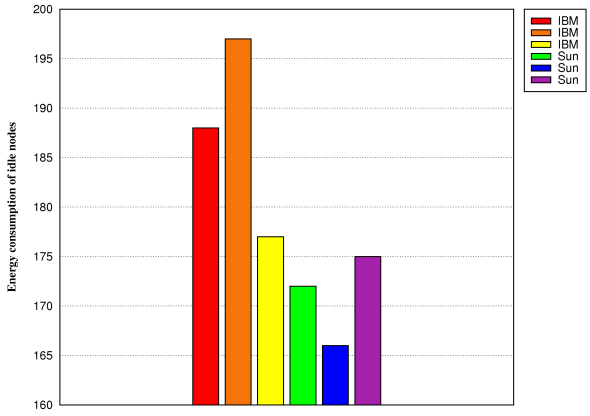

Figure 3.8: Power consumption (in Watts) of 6 nodes when they are idle.

will consume less energy since it will start its fans less often. According to [FWB07], fans can represent 5% of the consumption of a typical server.

We conducted another experiment to show that the consumption of a node can be influenced by many factors. We measured the consumption of a typical server (IBM eServer 326) in a rack full of running servers. Its idle consumption was 200 Watts. We then switched off all the nodes in the rack – but this one – and waited one hour to cool the nodes. The idle consumption of the node after the others cooled was 189 Watts which represents a 5,5% difference.

A node's consumption can be influenced by various and numerous factors which should not be neglected, since they lead to variations. The identification of these factors is hard and more work is required before we can propose mathematical models to explain the link between those factors and the variations in energy consumption. For example, we believe that mechanical parts such as fans and hard-disks can wear out and consume more energy over time. However, further investigation should be carried out to support this claim.

3.4.2 Non-linear relation between CPU load and energy consumption

To design and build new data centers and sustainable distributed systems, power-provisioning strategies are required. However, these strategies are hard to elaborate even in current large-scale distributed systems due to the lack of power-usage data. In fact, most facilities lack energy sensors and on-line power-monitoring tools and as a result, studying power usage in such infrastructures is a difficult task. For this reason, power models are often used to estimate the energy consumption.

We have seen in Section 3.4.1 that a node's energy consumption does not only depend on the node's architecture and on the application it runs. Other factors must be considered, and a more detailed analysis is required. This can be done by studying the energy usage of the node's main components (e.g., CPU, disk, network card). In [FWB07], the authors present a typical server's peak power consumption split among its components (CPU, memory, disk, PCI slots, motherboard, fan). These results show that CPUs are the most consuming components, but no component has a fixed energy consumption. The consumption depends on the load experienced by the component. To model the node's overall consumption, it is essential to understand the link between CPU load and energy consumption.

Several papers deal with modeling the CPU's energy consumption based on its load [SKWM01, BKWW03, FWB07]. In [SKWM01], authors present an energy model at the instruction level. In [FWB07], authors model the energy consumption according to CPU activity. Another approach consists in deducing it by using event-monitoring counters [BKWW03].

The common idea is often that the CPU's energy consumption is directly proportional to its load [FWB07, CHL+08, SWHK08]. We have performed several experiments showing that this

assumption does not hold for our servers. The experiments have used three IBM eServer 326 (2.0 GHz, 2 CPUs per node) and three Sun Fire V20z (2.4 GHz, 2 CPUs per node). All six nodes have two CPUs, thus a 200% usage means that the two CPUs are fully loaded. The CPU load is given by the system (we use the `htop`[11] and `sar`[12] commands).

On each of these six nodes, we apply successively three different types of workload that fully utilize the nodes. Hence, the CPU usage is the same for those three loads on the six nodes. We then compare the energy consumed by the six nodes during these three experiments.

The first experiment shows the electric consumption of the 6 nodes while a `stress`[13] tool is running. We find that the consumption increases by 13% to 19% compared to the idle consumption for each node (shown in Figure 3.8). During this experiment, the load of the CPUs reaches 200% on every node.

The second experiment is similar, but with two `cpuburns` (one for each CPU), which also represent a 200% load. We see that the energy consumption increases more than in the previous experiment, from 23% to 31%.

For the last experiment, we launch two `cpuburns` and an `iperf` simultaneously, causing the CPU utilization to reach 200%. However, we see that it only increases the electric consumption by 15% to 25% compared to the idle consumption. These values are smaller than in the previous experiment, although we added another task (the `iperf` application) which used the network card interface.

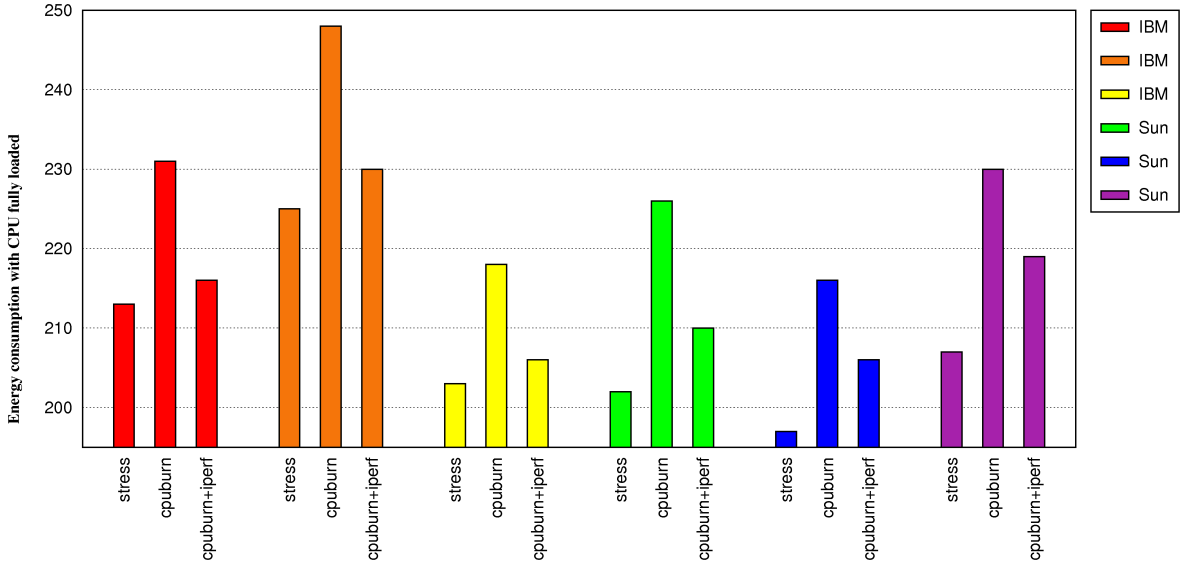

Figure 3.9: Comparison of the power consumption (in Watts) of three applications that fully load the CPU.

We have compared the results of the last three experiments in Figure 3.9. For three different experiments that fully load the CPUs, the electric consumption reaches widely different values for the two types of node architectures. The difference can reach 14% (compared to the idle consumption of the node), which is not negligible. This difference can be due to the heterogeneity of the load set by these three experiments.

Although CPU utilization has an impact on power consumption, the impact is not linear as stated in previous work [FWB07, CHL+08, SWHK08]. It is indeed not possible to have a linear

[11] *htop*, a system-monitor tool that produces a frequently-updated list of processes with their CPU-usage percentage.

[12] *sar* is a Solaris-derived system-monitor command used to display CPU activity.

[13] *stress* is a tool used to evaluate the perceived performance under heavy load. It imposes stress on the system using *sqrt* (CPU), *sync* (I/O), *malloc/free* (memory) and *write/unlink* (HDD) functions in our case.

function which has three values with the same abscissa (we have three different power consumption values for the same CPU load)! In [VAN08b], the authors observe the same phenomenon: the energy consumption model of each application is different.

In fact, the relation can be linear under two really restrictive (and not realistic) conditions: (a) the type of workload should be the same (similar utilization of the CPU not like our three kinds of workload); (b) the external environment must be the same (workload and temperature of the neighboring nodes, external temperature, location in the rack as seen in Section 3.4.1).

Moreover, even though tools like `top` show a CPU usage of 100%, CPU power consumption is not necessarily maximal. It may depend on the instruction set in use or the quantity of RAM size accessed by the running application. Then, relation between CPU load and energy consumption can not be linear and should not be used in energy models. It is also noticeable that using the CPU and the network card at full capacity can be less energy consuming than using only the CPU at full capacity.

3.4.3 Non-linear relation between network load and energy consumption

A lot of electricity is wasted to keep network hosts fully powered on, just to maintain their network presence. Proxying techniques can be a good candidate to solve this issue [JCN08, DCGG+09]. Several efforts aim to reduce the electric cost of networking devices, from network cards to switches and routers [GS03, GCN05]. Indeed, network cards are always powered on (as long as the node is on) even when they have nothing to do. It represents an important waste of energy with the scaling effects [GS03]. In [GCS06], the authors describe their algorithm called Adaptive Link Rate (ALR). It changes the link's data rate based on an output-buffer threshold policy. This algorithm does not affect the mean packet delay. This paper shows that the energy used with the different Ethernet link's data rates are not proportional to utilization.

In [HCP09], authors conclude their experiments by saying that the energy consumption's dependency on bandwidth is quite small and not linear. They even show that for some network devices, moving data consumes less energy than doing nothing. We have observed similar results with our office's switch. Hence, previous work stating that switch power consumption depends on the number of sent bits [YMB02] is not appropriate to model the energy consumed by current switches.

3.4.4 Non-linear relation between disk load and energy consumption

Storage represents a significant percentage of data center power [ZDD+04]. Thus, managing and reducing disk power consumption is a necessary target to decrease the total power consumption of large-scale distributed systems. Some work has been conducted to model and link disk performance and disk power usage [ZSG+03, AAF+09]. In [HSRJ08] the authors show that the power used for the standby and idle modes of disks can be reduced significantly.

The power consumption of disks is composed of a fixed portion (the idle state which includes the spindle motor) and a dynamic portion (I/O workload, data transfers, moving of the disk head during a seek operation) which represent about one third of the disk's total consumption [AAF+09]. Disks offer three types of functions: seeking, reading, and writing. Those three operations have different consumptions [ZSG+03] since they affect the dynamic portion.

In [AAF+09], the authors prove that the total power consumption of an enterprise disk drive is not a linear function of the number of I/Os per second. They show that less power is consumed relatively per each I/O (seek operation). This phenomenon is due to the fact that a larger number of concurrent I/O requests to the disk increase the internal disk queue and, therefore, I/Os can be reordered so as to shorten the seek distance and thus reduce power consumption.

We have seen through examples of the three main components (CPU, network interface, disk) of the nodes in large-scale distributed systems, that the link between the load and the

electric consumption is not linear, which complicates the problem of designing true models and makes it harder to find an optimal trade-off between performance and electric cost.

3.4.5 Energy consumption of switched-off nodes

To limit the energy wastage, nodes can be switched off when they are not used. Some of the existing work assumes that the consumption of switched-off nodes is zero (as stated in [RL03]). However, we observe that nodes still consume energy when they are off (Figure 3.10) – the energy consumed by plugged-node when it is off is called the *off consumption*.

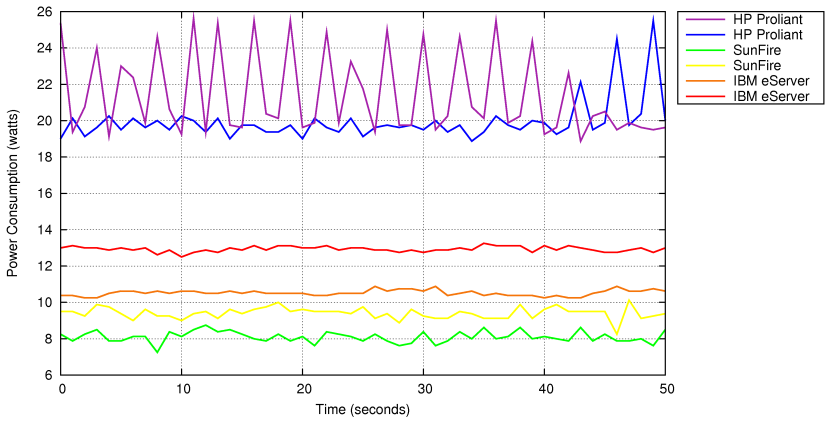

Figure 3.10: Consumption of 6 nodes when they are off.

For the HP Proliant servers, the off consumption represents on average 10% of their idle consumption. Thus, this off consumption is neither null nor negligible, nor stable, and depends on the architecture. This can be due to the card controllers embedded in those nodes which are used to wake them up remotely. These off consumptions have widely decreased over server generations, but there is still room for improvement. Indeed, for the two HP Proliant (385 G2), the off consumption represents 15% of the idle consumption. This off consumption must therefore be considered when designing algorithms and frameworks to manage resources in an energy-efficient way.

Booting and halting a node consume non-negligible amounts of energy by generating peaks of consumption (mostly for the boot start-up), as seen in Figure 3.7. These energy costs may be reduced by using suspend-to-disk or suspend-to-RAM techniques. In [Kno08], some experiments on two notebooks and one desktop machine show that the standby mode (ACPI S1 state, only a few parts of the board are switched off) is not energy efficient compared to the off state. However, it shows that the two notebooks consume as much energy when they are off than when they are in suspend-to-RAM mode. The suspend-to-RAM mode is often not available or configured on servers of large-scale infrastructures.

3.4.6 Energy cost of virtualization

Virtualization presently seems to be the most promising technique to reduce energy consumption in large-scale distributed systems [TBL09, NS07, SKZ08, SLB07]. A common criticism is that the virtualization layer increases the energy consumption for a given application since the hypervisor requires resources. We have found no work in the literature able to support nor deny this assumption.

To clarify this issue, we installed Xen 3.2[14] on a Sun Fire v20z. The idle consumption was identical to that with a GNU/Linux Debian distribution. We then launched a `cpuburn`

[14]Xen is a virtual-machine monitor that allows several guest operating systems to run concurrently on the same computer hardware.

on a virtual machine (512 MB of RAM and one virtual CPU) and observed that the energy consumption is the same with or without virtualization (the difference is in fact less than 1 Watt).

Our next experiment was to launch two `cpuburn` on the same machine without virtualization and two `cpuburn` on two different virtual machines. We also obtained the same consumptions (the difference is less than 1 Watt). Finally, we did the same experiment with an `iperf` and observed that, inside a virtual machine, this intensive networking application consumes 7 Watts more on average (this represents 4% of the idle consumption). This effect is due to the poor management of the network allocation in Xen 3.2. The next versions are expected to solve this issue.

We conducted the same experiment with a HP Proliant 385 G2 and with Xen Server 5.0 (which is more recent than Xen 3.2) and a Debian. The idle consumptions with the Debian and with a virtual machine on Xen Server 5.0 were still identical. We then did the experiment with a `cpuburn` in a virtual machine (512 MB and one virtual CPU) and a `cpuburn` in a Debian. There was a 2% difference between the two consumptions (the virtual machine experiment was the least consuming). These experiments lead us to conclude that for our servers, virtualization with a recent hypervisor does not imply any energy overhead for computing tasks. However, this study does not consider the time overhead which should be induced by the CPU overhead in the device-driver domain [CG05] (a virtual machine cannot always run on an entire physical CPU, the resources are shared).

As pointed out in [TCH+08], virtualization can still lead to a waste of resources since the hypervisor needs resources and this reduces the possibility to put more virtual machines on the same node. This issue is often not taken into account in the design of energy-efficient frameworks for virtualized distributed systems.

3.5 Conclusion

As observed, having a configurable sensing infrastructure is one of the basic components mandatory for designing energy efficient software frameworks. The different manners in which the data has been stored and reported aim to fulfill the needs of several categories of users, including application developers, middleware designers, system administrators and infrastructure managers [DdAGLO10]. In addition to energy consumption data, obtaining and analyzing information from multiple sources (e.g. schedulers and resource utilization) is essential to understanding the usage of the infrastructure and applications' behaviors. This analysis can help system designers and managers to identify how to devise and implement techniques for reducing the cost of energy consumption. It also allows them to evaluate the return of investment of deploying the sensing infrastructure itself. At present, instrumenting a platform is expensive and its cost can surpass the savings achieved by striving to improve the energy efficiency of the overall infrastructure.

Understanding the energy consumption of large-scale system infrastructures is also a complex but mandatory step to reduce the electric bill. Indeed, it is counter-intuitive, but load and energy consumption are not directly correlated. That makes the task of designing realistic power models even harder. Another crucial conclusion of this analysis is that the idle power consumption of computing resources is really high. Hence, during idle periods, the energy wastage is consequently high – a waste that can be reduced with on/off techniques. However, these techniques should be deployed cleverly because switching resources on and off consumes energy.

We have created a website[15] to share six months of logs from this infrastructure. These logs contain measurements on power consumption carried out every second for each node and usage data (user reservations and downtime).

[15] `http://www.ens-lyon.fr/LIP/RESO/ict-energy-logs/`

It is certainly not the least charm of a theory that it is refutable.

Friedrich Nietzsche

4

Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems

There are several techniques available to improve the energy efficiency of large-scale distributed systems, but their coordination requires a unified framework. To this end, this chapter presents the Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems (ERIDIS). This framework is based on algorithms to switch off unused resources, reservation aggregation to increase inactivity periods (and switch resources off for longer), and load prediction techniques to avoid switching resources off unnecessarily [OL11c, OL11b].

In reservation-based systems, users should specify the number of required computing or networking resources, the duration of their requests and a deadline. That is, for networking resources, the user reserves a certain bandwidth from a source to a destination for a given interval. This mechanism is typically suited for bulk data transfers, file replications and backup transfers. For computing resources, the user reserves virtual machines or cores for a specific timeframe. In both cases, the user defines a deadline before which the reservation should end. These systems allow for more flexible and predictable resource management. ERIDIS takes advantage of these favorable characteristics to save energy.

This chapter details the architecture and the components of ERIDIS. The following chapters present how ERIDIS is applied to three different scenarios, namely data centers and Grid resource management (Chapter 5); virtualized environments and cloud resource management (Chapter 6); and large-data transfers in dedicated networks (Chapter 7).

This chapter is organized as follows. Section 4.1 introduces reservation-based systems. Section 4.2 presents the general architecture of ERIDIS, whereas its reservation model is explained in Section 4.3. Section 4.4 details the management of resource requests. The energy-efficient scheduling of incoming requests is described in Section 4.5. The energy-efficient resource management is shown in Section 4.6. Section 4.7 describes how ERIDIS estimates the energy consumption in order to improve the energy efficiency of the resource management, while Section 4.8 states the prediction algorithms used to avoid useless on/off cycles. ERIDIS is provided with re-planning capacity, outlined in Section 4.9, to optimize the energy consumption. Finally, Section 4.10 concludes the chapter.

4.1 Reservation-based systems

Jobs are the programs that users want to execute on distributed systems. *Reservation-based* systems imply that users submit resource requests specifying bounds in terms of both duration (reservation length) and number of resources. Hence, these systems are typically not suitable for long jobs or jobs with unpredictable duration. Some reservation-based systems support *advance reservations* (i.e. users can specify job start times in the future) in addition to best-effort or on-demand provisioning.

Reservation mechanisms are widely used in large-scale distributed systems [SMLF09, CRH08, PBK+06] to provide users with guarantees on quality of service, including the respect of deadlines and specific hardware and software constraints. Distributed systems without reservation

mechanisms often present two issues, namely over-provisioning (such as in the public switched telephone network) and degraded service for critical applications or traffic (such as in the Internet) [FKL+99].

In environments that support reservations, users can specify given constraints regarding time and required resources (e.g. a deadline before which the job should be completed, a start time, hardware and software requirements and resource co-allocation). These constraints are essential for some High-Performance Computing (HPC) applications requiring quality of service from allocated computing and/or network resources. Examples of this kind of application include identifying and minimizing highway congestion in real-time, interactive and supervised medical imaging, and data replication in Content-Delivery Networks (CDN).

Reservations also allow for more flexible and predictable resource management as the length of each reservation can be known during submission, simplifying the task of scheduling algorithms. These mechanisms improve capacity planning and predictability, leading to a fair load distribution and optimal resource utilization [SF10a].

4.1.1 Grid and Cloud reservation systems

Grid and Cloud facilities offer resource sharing in domains that range from scientific and engineering applications to enterprise and utility computing [Maj09]. Several of the existing frameworks support advance reservations for QoS purposes, since it has been theoretically proved that advance reservations reduce the variance of job execution times [MAD+05], hence making the execution times more predictable. However, these techniques compromise the resource utilization [CBK10].

Resource reservation infrastructures

Globus Architecture for Reservation and Allocation (GARA) [FKL+99] has been one of the first systems providing advance reservation of heterogeneous resources. It has demonstrated the applicability of reservation systems in Grids.

The architecture of a Resource Management System (RMS) can be centralized, distributed or hierarchical. While centralized architectures, such as Condor[1], produce optimal schedule, distributed frameworks, such as Globus[2], are more scalable, fault-tolerant and flexible [ET05]. Hierarchical approaches try to strike a balance between the two approaches. For example, in [MAD+05], the authors propose a two-tiered system for reserving resources on Grids.

One of the purposes of designing a reservation infrastructure is to allow co-allocation [SVF06]; the simultaneous utilization of several resources by the same application. This property, among other benefits, helps minimize the makespan of Grid workflows [PW10].

In [SKD07], the authors compare reservation models and best-effort mechanisms. They show that advance reservations outperform best-effort policies when the resources are under high utilization and/or the applications have significant task sizes (requiring more than one processor).

Scheduling policies

Scheduling policies allocate resources to tasks. A *resource broker* is the distributed system's component responsible for managing the resources and applying the scheduling policy. An extensive set of research work is available on the design of scheduling policies as shown in Section 2.1.3.

[1] Condor is a specialized workload management system for compute-intensive jobs: `http://www.cs.wisc.edu/condor/`

[2] Globus is an open source toolkit for building computing grids: `http://www.globus.org/`

Until recently, the scheduling policies of Grid and Cloud resource brokers were based entirely on traditional techniques such as First In First Out (FIFO) and Round Robin [Mei08]. However, elaborate scheduling can consider several parameters including resource heterogeneity [CRH08] and required QoS [HSL02].

Real-time task scheduling is one of main issues in designing distributed systems, a problem generally known to be NP-hard [PSA97]. While satisfying the maximal number of user requests, scheduling algorithms often consider other objectives such as to: maximize resource and application utility [SVF06], minimize the total time to delivery (from job submission to job end) for each job [ET05], optimize combined resource utilization [YZL+10], maximize the provider's profit [Mei08], maximize reliability [RV11] and minimize the total resource provisioning cost [CLN11]. Scheduling problems considering multiple criteria are often NP-hard as well and the proposed solutions are commonly based on heuristics. In [YSLQ10], the authors propose a multi-criterion scheduling heuristic called N-variable constraints algorithm to answer this need of multi-criterion algorithms.

Advance reservations make scheduling algorithms more complex since they increase the possibilities (reservations can be placed in the future without continuous resource utilization) and put roadblocks in the resource schedule (reservations can be moved once accepted). In [Maj09], the authors propose an *any-schedulability criterion*; a set of inequalities that must be satisfied for a set of advance reservations to be scheduled under any work-conserving scheduling policy.

Another issue with reservations systems is the estimation of task length. Solutions based on benchmarks have been proposed to estimate this duration and thus avoid either over-provisioning or under-provisioning [ET05].

Service-oriented brokers

Resource negotiation protocols accompany reservation mechanisms [CBK10]: the user submits a request, the reservation service checks whether the task requirements can be met considering existing reservations and replies to the user's request. These protocols can also be used to negotiate other QoS parameters [Sim10].

Users can indeed require specific QoS guarantees that do not concern the computing resources because reservations of computing resources can be combined with I/O access and storage [THK+10] and network reservations [YZL+10]. An infrastructure described in [YZL+10] proposes a cloud infrastructure framework – validated using an IPTV application – that discovers, selects, reserves and assigns combined computing and networking resources.

To make the management of these high-QoS reservations more flexible, some infrastructures allow less urgent reservations to be preempted[3]. Haizea, a solution based on leases supporting advance reservations in Clouds [SMLF09], is implemented on top of OpenNebula[4] and shows the effectiveness of this model.

Although reservations of network resources resemble computing reservations, separate solutions have been provided to solve the issues described above. The next section discusses the solutions dedicated to network infrastructures.

4.1.2 Network reservation systems: bandwidth provisioning

Network reservations guarantee that a given bandwidth capacity is provisioned between a source and a destination for a certain timeframe [BCSS10]. ESnet's OSCARS[5] (On-demand Secure Circuits and Advance Reservation System) and DRAGON[6] (Dynamic Resource Allocation via

[3]Reservations are suspended and resumed later.

[4]OpenNebula is an open-source cloud computing toolkit for managing heterogeneous distributed data center infrastructures: `http://www.opennebula.org/`

[5]OSCARS: `http://www.es.net/services/virtual-circuits-oscars/`

[6]DRAGON: `http://dragon.maxgigapop.net/twiki/bin/view/DRAGON/WebHome`

GMPLS Optical Networks) are example of bandwidth provisioning (i.e. network reservation) projects over scientific networks.

Bandwidth provisioning is often required for applications generating bulk data transfers (i.e. large amounts of data to be transferred), such as peer-to-peer networks [SBB04, HHX+05] and content delivery networks [BCMR04] with media servers that require timely transfers of content among servers [SKB98]. Bandwidth provisioning techniques can even be used on the Internet with water-filling techniques [LSRS09] which are compared in terms of performance and cost to the mail service *FedEx*.

Bandwidth Allocation and Routing Algorithms

The two main issues that arise when provisioning bandwidth are to allocate the bandwidth in time and in space, which are solved by bandwidth scheduling and path computation algorithms, respectively. Two basic provisioning modes are commonly distinguished [SRR+07, JLRS08, GLB08]:

1. on-demand mode: a connection request is made when needed, and it is then accepted or denied depending on the current bandwidth availability; arriving requests are queued until their allocation;

2. in-advance mode: a connection request is granted for future time-slots based on bandwidth-allocation schedules; arriving requests are scheduled in the future as soon as they arrive (system of agendas).

Two approaches can also be followed to provision bandwidth [PZSJ09]:

1. centralized: a central server maintains all resource availability information, and all reservation requests may retrieve availability information from the server;

2. distributed: each node maintains its resource availability information, reservation scheduling is done in a collaborative way.

The centralized approach does not scale well [ZDH01] while the distributed technique has issues with the processing time of requests. The network-management system is often called *bandwidth broker* [Bur05, ZDH01], and is in charge of the reservation requests' admission control.

Many problems related to bandwidth allocation and path computation are NP-complete. Lin *et al.* [LW09] describe two basic scheduling problems: fixed path with variable bandwidth and variable path with variable bandwidth with a view to minimize the transfer end time of a given data size. They prove that both problems are NP-complete and propose greedy heuristic algorithms to solve them. In [LW08], they consider two other problems dealing with multiple data transfers, where the bandwidth allocation and path computation algorithms are mainly inspired from the Dijkstra and Bellman-Ford algorithms [SRR+07, LW08, JLRS08].

Network Protocols

The usual network protocols are not adapted to dedicated networks since they are designed to work in a best-effort mode under congestion, failures and resource competition. Dedicated networks have different characteristics, among which high speed, reliability and high delay-bandwidth products. It has long been known that TCP is inefficient in this kind of environment [JBB92], and as result new protocols using UDP for data transfer and TCP for control have been developed, such as the reliable blast UDP [HLYD02] and SABUL [GG03]. Other emerging solutions include better adaptivity to maximize the data rate according to the receiver's capacity and to maximize the goodput by minimizing synchronous, latency-bound communication (Adaptive UDP [EHW08], FRTP: Fixed Rate Transport Protocol [ZMV04]). These protocols are implemented atop UDP as application-level processes.

Furthermore, specific reservation protocols have been developed, such as the ReSerVation Protocol (RSVP) [ZDE+93] where resources are reserved across a network for integrated services in QoS-oriented networks. RSVP also enables the design of upper-layer protocols to complete its functionality [SKB98].

Advance-Reservation Algorithms

As on-demand allocation of resources can be regarded as a special case of in-advance allocation [JLRS08], focusing on advance reservations does not restrict our scope. The idea of making advance reservations of network resources is not recent [Rei94], but the main issue preventing its wide usage is the unpredictability of routing behavior. However, the emergence of the Multi-Protocol Label Switching (MPLS) [RVC01] standard along with traffic engineering and explicit routing features, makes it possible to disconnect the reservation management from the network layer, thus leading to better interoperability among systems for managing bandwidth reservations. While MPLS ensures the support of on-demand capabilities at layer 3, Generalized MPLS (GMPLS) provides the same functions at layer 2 and 1.

Mainly two transfer scheduling techniques can be used for advance reservations: online scheduling where requests are processed as they arrive or periodic batch scheduling where they are scheduled periodically [LRS09]. The employed time models can be continuous [JLRS08, LRS09] or discrete [Bur05, RRX09] with fixed time slots (slices) during which the resource allocations are similar.

Several other issues related to advance bandwidth reservations have been explored including fault tolerance [BDF03b], rerouting strategies [BLS05], load-balancing strategies [XXG+10] and time-shift reservations [PZSJ09].

4.2 ERIDIS architecture

As writing of this thesis, none of the reservation-based systems described in the previous section aims at saving energy. This observation has led us to propose an Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems called ERIDIS.

ERIDIS acts at the resource management level to minimize the energy consumed by large-scale distributed systems with reservation mechanisms. It also supports advance reservations and comprises the following components:

- sensors that collect, in real-time, information on the energy consumed by resources and directly measure the impact of the taken decisions;
- allocation and scheduling algorithms that optimize the reservation placement;
- on/off features to put resources into sleep mode when they are not used;
- prediction algorithms to anticipate the workload;
- workload aggregation policies to avoid frequent on/off cycles of resources.

The premise behind ERIDIS is that parts of the computing and networking resources can be put in a sleep-state to consume less energy when they are not needed by user tasks. Through coordinated resource management, both the opportunity for allowing resources to sleep and the duration of these sleeping periods can increase.

Whether users have a computation or a data transfer to be performed, they will make reservation requests that consist of at least:

- the earliest possible start time, a deadline, the number of required computing resources, and a duration – in the case of a computing job;

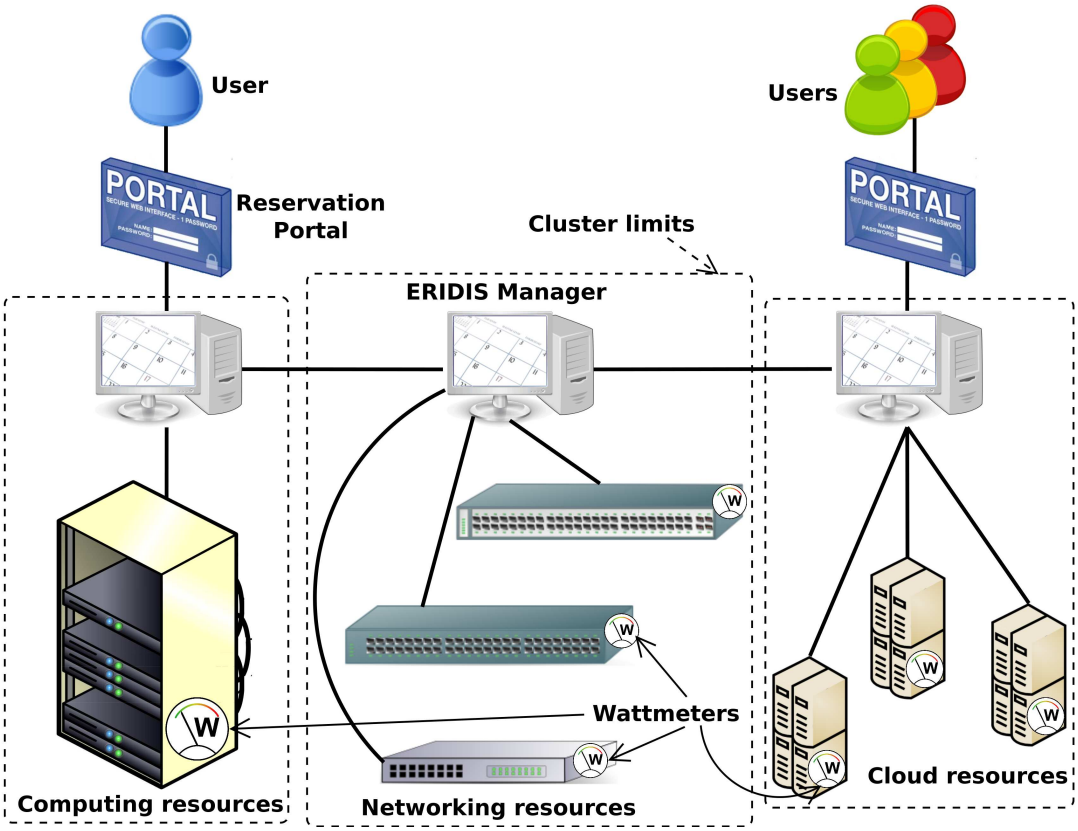

Figure 4.1: ERIDIS components

- the earliest possible start time, a deadline, a data volume to be transferred (in GB for example), the source node, and the destination node – in the case of a networking job.

Other constraints can be included such as a given computing environment or a specific geographic location to process a computing job, or a bit-rate transmission profile (maximum possible bit-rate over time) for a networking job. These additional options are detailed in the next chapters.

Figure 4.1 presents the logical architecture of ERIDIS, where users are connected to a reservation portal, which they see as the ERIDIS *gateway*. Each portal is directly connected to an ERIDIS manager; a local resource manager in charge of managing a cluster of resources (either computing, Cloud or networking resources). For example, for a Grid infrastructure, each cluster (e.g., Grid site) has its own *ERIDIS resource manager* to keep the overall architecture scalable, fast and robust. Hence, users send their requests to their local gateway, which is part of the resource manager in charge of dealing with the other resource managers, if required.

Each resource managed by ERIDIS is monitored by energy sensors (i.e. wattmeters), that provide accurate and periodical measurements to the manager. Thus, as reservations complete, ERIDIS can compute the energy consumption of each reservation and provide this information to the user in order to increase his energy-awareness.

The structure of the ERIDIS manager is presented in Figure 4.2. The arrows represent the actions of modules on other modules or databases. Administrators and system designers can specify the green policies they want to use including on/off techniques and DVFS techniques. These choices are taken into account by the reservation scheduler and in the resource-management modules. The reservation scheduler as the name implies, schedules and allocates resources to incoming reservation requests whereas the resource management module puts resources into sleep state or wakes them up if required. All the components of the manager are described in detail as the working of ERIDIS is explained.

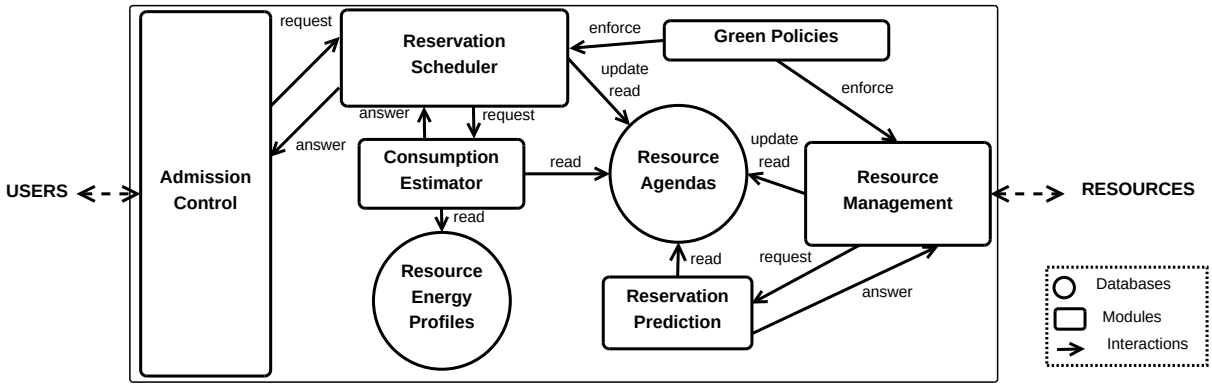

Figure 4.2: Structure of an ERIDIS manager.

4.3 The reservation model

A ERIDIS manager maintains an *agenda* for each resource it manages (Figure 4.2). The agenda stores information on future reservations concerning the resource, the different *states* (e.g. powered on but idle, turned off, booting, switching off, or partially or fully reserved) and duration of each state. The resource-management module updates the corresponding agendas if resources are down.

Figure 4.3 shows an example of resource agenda where a resource is switched off between two reservations and again once the second reservation completes. The date when a resource state changes is called an *event*, whereas the capacity can be either the number of cores or bandwidth capacity.

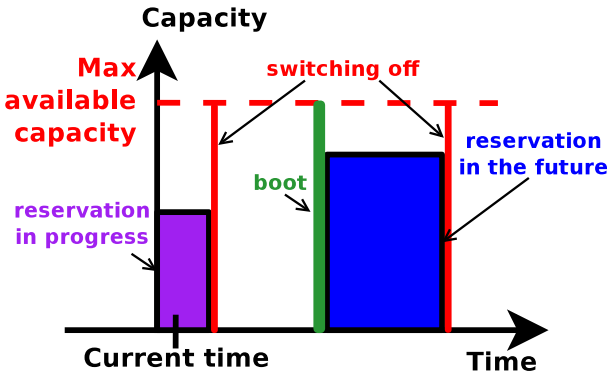

Figure 4.3: Example of an agenda of a given resource

To be more flexible and consume less space, this model uses continuous time to store the agendas rather than employing slotted time. The storage of the agendas is structured in a *time-capacity list* made of $(t[i], c[i])$ tuples, where $t[i]$ represents an event (time) and $c[i]$ is the percentage of resource used between this event and the next. For example, for a bandwidth reservation the capacity is the portion of reserved bandwidth, and for a computing job the capacity represents the percentage of CPU reserved for the job. Therefore, at any moment, a resource can know which percentage of its overall capacity is used, whereas specific capacity values are used to indicate power-on and shutdown periods.

The tuples are sorted in increasing order of $t[i]$. If $(t[n], c[n])$ is the last tuple, then it means that the resource is unused from $t[n]$ to ∞ $(c[n] = 0)$. The ERIDIS manager updates the agendas of the resources it manages.

In addition, the architecture of ERIDIS is semi-decentralized since each local manager gets full control of the resources it manages. Moreover, local managers communicate among them-

selves in a point-to-point manner without a global supervisor. For instance, in a Cloud context, if a user requires more resources than its local manager can offer, the local manager will request the number of resources it lacks from another manager. If the number of available resources is still insufficient, the local manager will ask another manager and so on, until it examines the whole list of managers[7]. When a new manager is added, it informs all the existing managers, so that they can update their lists.

This hierarchical organization, frequently utilized in large-scale distributed systems, ensures that the architecture is scalable, fast and robust since an ERIDIS manager is in charge of a limited number of resources and has a privileged access to other ERIDIS managers.

4.4 Management of the resource requests

A reservation resembles a lease contract between a resource provider and a user since the latter can utilize the reserved resources during a limited time interval that has been fixed during the negotiation process with the former. The negotiation comprises a three-step handshake process (Figure 4.4).

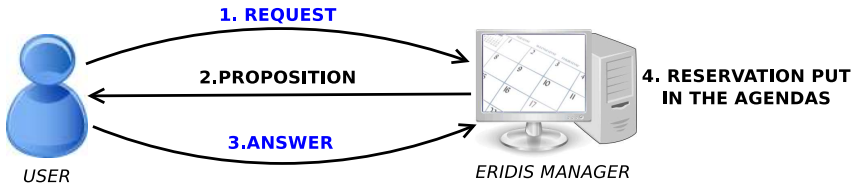

Figure 4.4: Reservation negotiation between the user and an ERIDIS manager.

First, the user sends a *request* that contains its resource and time constraints, i.e., at least the type and number of resources, the reservation duration, and a deadline. Second, the reservation manager proposes a solution, which can either fulfill the user's requirements or propose the earliest possible reservation start time if the deadline constraint cannot be respected (e.g. due to system load). If the request asks for too many resources, or does not respect the system's admission rules, it is rejected outright and the user is notified.

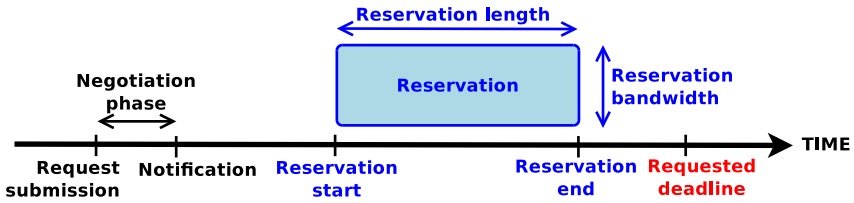

Figure 4.5: Reservation process.

Finally, if the reservation manager proposes a solution, the user has a fixed interval to accept or reject the offer, otherwise, the request is rejected. If the user accepts the offer, the manager updates all the concerned agendas by adding this new reservation. Algorithm 4.1 details the algorithm used to process the request, whereas the scheduling algorithm is detailed later (Algorithm 4.2).

The reservation-management system should guarantee that the negotiated reservation terms are well respected. It also has to dynamically schedule in time the reservations and allocate them the most appropriate resources. The sequence of operations is described in Figure 4.5.

Figure 4.6 presents the different reservation states. First, the user sends a request that is either rejected or accepted – in which case, the manager makes an offer. Then, the user either

[7]For this reason each ERIDIS manager is not necessarily linked to a user portal.

Algorithm 4.1 Request processing

```
If the request is not acceptable Then
  reject it;
Else
  launch the scheduling algorithm;
  If the request cannot be satisfied Then
    propose the earliest possible start time (given by the scheduling);
  Else
    send the offer to the user (given by the scheduling);
  wait for the answer;
  If there is no answer or the answer is no Then
    discard the request;
  Else
    place the reservation on each concerned agenda;
```

accepts or rejects the offer. If it is accepted, the reservation is scheduled, but it can be re-scheduled later before its start time if the manager decides to move it to accommodate a new request and save energy (process detailed in Section 4.9). This re-scheduling is still bound by the resource and time constraints negotiated between the user and the ERIDIS manager, so that no re-negotiation is allowed. During the reservation, the user has access to the reserved resources until the reservation's finish time.

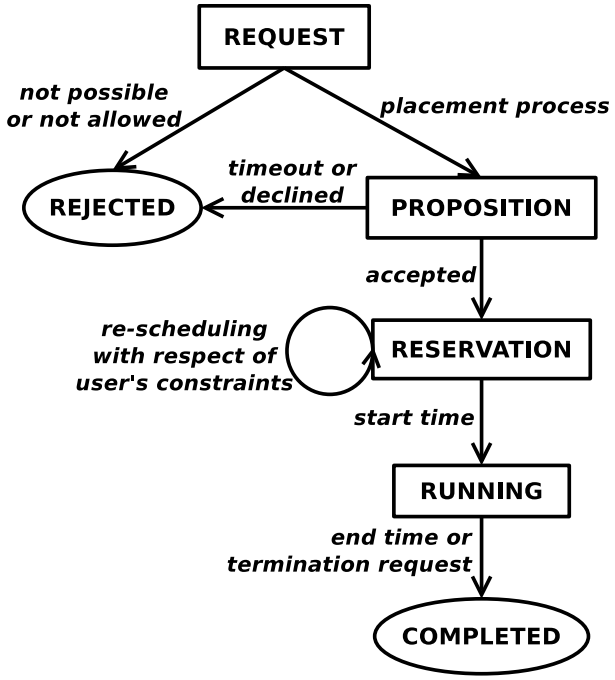

Figure 4.6: The different states in the management of a reservation.

4.5 Energy-efficient reservation scheduling

As an objective, the scheduling algorithm should provide either the most energy-efficient possibility in terms of both time placement and resource allocation, or the earliest possible start time if the deadline constraint cannot be satisfied. This algorithm is executed after the admission control process, on requests that respect the system's rules and that can be accepted.

If a reservation is possible, the scheduling algorithm stipulates a start time and provides a set of resources. Algorithm 4.2 details the scheduling method, where the first step is to find the earliest start time. As the request passes the admission control, the algorithm will determine whether the deadline constraint can be satisfied. If the earliest possible start time does not allow the reservation to end before its deadline, this new start time is proposed to the user, who can either accept or decline the offer.

Algorithm 4.2 Scheduling algorithm

find d the earliest possible start time with a satisfying set r of free resources;
$List = [(d, r)]$;
ForAll agenda events between d and (deadline - reservation duration)
  If the reservation can be placed before this event **Then**
    find the set of the least consuming free resources at that time;
    add the possible start time to *List* with the resource set;
  If the reservation can be placed after this event **Then**
    find the set of the least consuming free resources at that time;
    add this event to *List* with the set of resources;
If *List* contains only d **Then**
  return d and r;
Else
  ForAll dates in *List*
    estimate the reservation energy consumption if it starts at this date with the corresponding set of resources;
  propose the less energy consuming date and set of resources to the user;

The earliest possible start time is found by looking linearly at each event, starting from the current time, to see if a reservation satisfying the user's constraints can be placed before or after this event. Only the feasible solutions are examined, and this restriction of the solution space to certain dates (i.e., the events) ensures that the algorithm remains fast and scalable and that the solution found is energy-efficient, since the reservation has been aggregated with at least another one (by definition of the events).

When the earliest possible d has been found, the scheduling algorithm looks linearly at all the resources that can accept a reservation starting at d, and picks the N least consuming resources (taking into account the necessary switching on and off), where N is the number of resources required by the user. The energy-consumption estimation of a reservation is detailed in Section 4.7 (Consumption Estimator module in Figure 4.2). This set of N resources should satisfy the user's constraints (e.g., if he or she specified specific hardware or software constraints), and should be the least consuming set of resources among all the possible sets at d.

When the earliest possible solution (d and r the set of resources) has been found, the algorithm (Algorithm 4.2) searches if there is a better solution, in terms of energy consumption, after d but still before the deadline. Hence, the algorithm examines each event in the time interval between d and the deadline minus the reservation duration requested by the user. The process is similar to that used to find d and r. Each event of the time interval is considered, and for each event, the set of the N least consuming resources is determined. In the end, all these solutions are compared in terms of energy consumption and the best is chosen. This solution is exhaustive for an on-line model, since the algorithm tests all the possibilities that can be the least consuming ones due to the energy-saving properties of reservation aggregation. The resources are shared with other reservations, thus the optimal set of resources depends on the considered start time. As the available capacity of each resource varies over time (due to reservations), for each possible start time, the energy consumption estimation is used to pick the best solution.

4.6 Resource management and on/off algorithms

As reservations finish, the prediction algorithm forecasts the characteristics of the next reservation for the freed resources. If the predicted reservation of a resource is impending (in less than T_s seconds), the resource remains powered on (it would consume more energy to switch it off and on again), otherwise it is switched off.

Indeed, the simplest distributed on/off approach is to switch off resources as soon as the work completes, but it might not be the most energy-efficient solution. Figure 4.7 presents the two possible cases for a given period of inactivity: 1) the upper graph presents the case where the resource is switched off, stays off for a while and then is switched on again; 2) the lower graph shows the case where the resource stays idle for the entire interval.

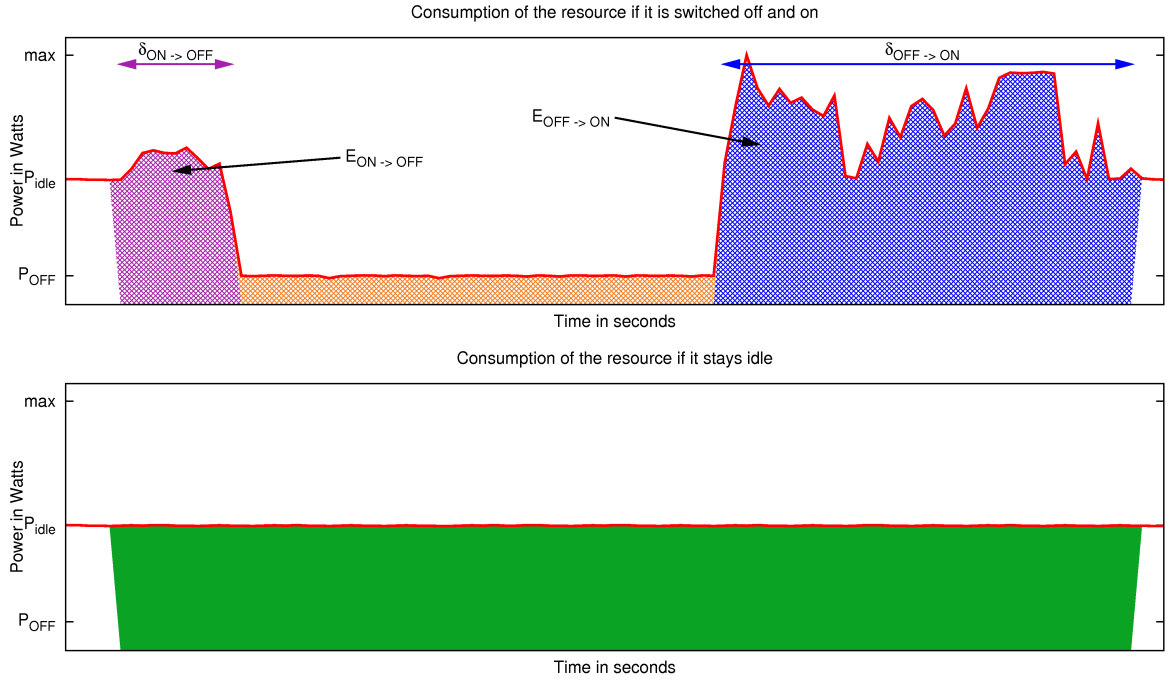

Figure 4.7: Definition of T_s.

The two colored areas represent the energy consumption for the two cases. We define as T_s the switching threshold since the energy consumptions of these two cases are equal for a given resource. This means that if the period of inactivity is greater than T_s, the most energy-efficient scenario is to switch off the resource and to switch it on again at the end. Otherwise, it is more energy efficient to let the resource idle.

Thus, for a given network resource, the formal definition of T_s is as follows:

$$T_s = \frac{E_{ON \to OFF} + E_{OFF \to ON} - P_{OFF}(\delta_{ON \to OFF} + \delta_{OFF \to ON})}{P_{idle} - P_{OFF}}$$

where P_{idle} is the idle consumption of the resource (in Watts), P_{OFF} the power consumption when the resource is off (in Watts), $\delta_{ON \to OFF}$ the duration of the resource shutdown (in seconds), $\delta_{OFF \to ON}$ the duration of the resource boot, $E_{ON \to OFF}$ the energy consumed to switch off the resource (in Joules) and $E_{OFF \to ON}$ the energy consumed to switch on the resource (in Joules). Then, at the end of a reservation, if one resource remains idle for more than T_s seconds, it is switched off.

The scheduling algorithm aims at aggregating the reservations as much as possible to save energy, especially the energy used to switch the resources on and off. However, switching the resources on and off can be difficult, demanding time and energy, and therefore when a resource

is switched off, it should stay off for a certain amount of time to save more energy than what is used when switching it off and then on again. Switched-off resources consume energy, but less than when they are idle (powered on but not in use) [OLG10]. Thus, a subtle balance needs to be found to ensure energy savings by switching resources off. That is why, at the end of each reservation, the resource manager determines if the freed resources should remain on or be switched off.

Algorithm 4.3 Reservation end

```
Provide the energy consumed by the overall reservation to the user;
ForAll reserved resources
  If this resource has an imminent reservation Then
    let it remain on;
  Else
    launch the prediction algorithm;
    If the resource is going to be used soon Then
      let it remain on;
    Else
      turn it off;
```

First, for each resource, the manager examines the agenda to see whether another reservation has been put just after it. That being the case, the resource stays on. Otherwise, the manager uses a prediction algorithm (detailed in Section 4.8) to estimate whether the resource is going to be used soon (Reservation Prediction module in Figure 4.2), in which case, it stays on; otherwise, the resource is switched off. This process is described by Algorithm 4.3. An *imminent* reservation is a reservation that will occur in a short period of time (smaller than T_s) and it is more costly to shut down the resource and to turn it back on than let it idle.

The problem here is to have a reliable and fast solution to remotely switch the resources on and off when necessary. This requires some hardware facilities on the resources and a dedicated infrastructure to access them. For example, in a Grid or Cloud context, IPMI facilities can be used [Lea06].

Moreover, each resource has a copy of its own agenda and thus, when it goes into sleep state, it uses a timer to know when it should wake up again. Hence, the previous mechanism to remotely switch on and off the resources is used only when a reservation – that was not planed when the resource went to sleep – occurs. Nevertheless, switching on/off operations should be fast enough to avoid impacting the reactivity of the whole system.

4.7 Energy-consumption estimates

As shown when describing the architecture, ERIDIS embeds energy sensors to monitor in real-time the energy consumption of the resources it manages. Therefore, at the end of each reservation, the ERIDIS manager reports the overall energy consumption of the reservation to the user, as a way to increase his energy awareness (Algorithm 4.3).

As we have seen previously, estimations of the energy consumption of some reservations are needed to take the best scheduling decisions. To make these estimations, the ERIDIS manager requires an energy profile for each resource that it manages. This energy profile contains information about:

- the energy and time required to switch the resource off ($\delta_{ON \to OFF}$),
- the energy and time required to switch the resource on ($\delta_{OFF \to ON}$),
- the mean power used by the resource when it is off (i.e., sleep state, P_{OFF}),

- the mean power used by the resource when it is idle (i.e., powered on but not used, P_{idle}),
- the relation between the capacity usage (in percentage) and the mean power usage of the resource (power consumption as a function of the resource utilization).

The ERIDIS manager draws up these energy profiles by using benchmarks and energy sensors (Resource Energy Profiles in Figure 4.2). These profiles are periodically updated (every month, for example) because hardware usage ("wear and tear"), heat or humidity conditions, among other factors, can impact the energy consumption of resources.

By exploiting these energy profiles, it becomes easy for the resource manager to estimate the energy consumption of a reservation. This estimation includes the energy cost to wake up and to switch off resources if it is necessary. Thus, if for example the reservation is stitched together with another one, the energy cost to wake up the resources is saved. Similarly, if a reservation shares some resources with another reservation because it does not require the full resource, the working energy cost of the resources is split between the two reservations according to the percentage of the resource that each of them is using.

4.8 Prediction algorithms

Prediction algorithms are used to ensure a good planning of the off-on cycles. They are required to predict the next reservation (number of required resources and start time). Our prediction algorithms are based on the recent history (the past part of the agenda). They are based on average values of past inactivity period durations and feedbacks which are average values of differences between the past predictions and the past corresponding real events in the agenda. More details are provided in Section 5.5 in the context of computing Grids.

For example, with computing resources, when a node is freed, the average value of the last few free-time intervals (when the node is not used) is computed, and it is assumed to be the value of the next free-time interval. This prediction algorithm has been tested in a Grid context in Chapter 5 with usage traces of a cluster belonging to the French experimental platform Grid'5000. In about 70% of the cases, the algorithm takes the good decision between switching off the resource and leaving it on. More generally, the energy gain is significant.

Prediction algorithms are bound to the workload that they try to characterize. Consequently, each type of workload requires a new algorithm to provide predictions as accurate as possible. For example, if it can be determined that the traffic present on a given network follows a Poisson law, this should be taken into account to design a specific prediction algorithm for this traffic.

Yet, accurate algorithms often involves high time complexity. So, a tradeoff should be find between response time and accuracy. Keeping in mind that those algorithms are used in a distributed manner on each resource after each reservation, we choose to give priority to the response time by using really simple algorithms that still provide acceptable results (more than 50% of good decisions, so more than the random choice). We plan to investigate other prediction algorithms for each kind of reservation system. However, this investigation will require real usage traces from production or commercial environments which are hard to obtain.

4.9 Energy consumption optimization: re-planning capacity

We have presented algorithms that work in a static way: once a decision has been taken, it cannot be changed. Our re-planning functionality is dynamic and allows reservations to be moved after their registration in the agendas while still guaranteeing the same QoS and respecting the user deadline. Indeed, off-line algorithms can lead to optimal solutions in terms of energy conservation because all the reservation requests are known from the beginning, which is not the case of our on-line algorithms. Off-line algorithms are hence not appropriate for real-time scheduling, and this re-planning capacity is a good trade-off between on-line and off-line scheduling.

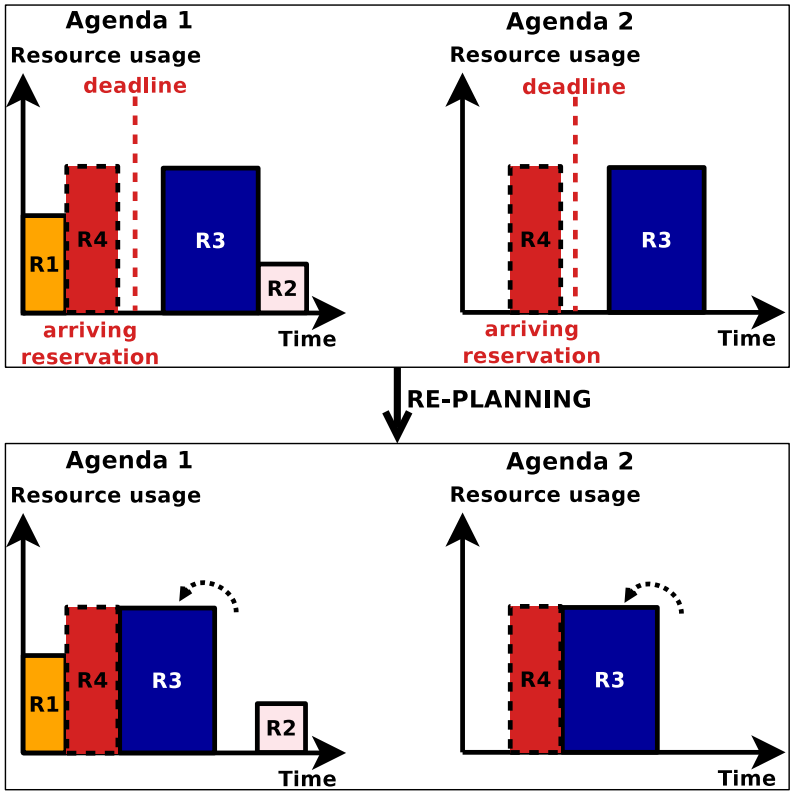

Figure 4.8: Re-planning capacity.

An example of how this re-planning functionality works with the agendas of two different resources before and after the re-planning operation is presented in Figure 4.8. Reservation $R4$ has just been stored into the two agendas and thus reservation $R3$ can be put just after it if it is more energy efficient than the former solution. This re-scheduling algorithm is launched after the acceptance of a new reservation.

4.10 Conclusion

Due to their size and heterogeneity, large-scale distributed systems require scalable, robust, fault-tolerant and energy-efficient resource-management infrastructures. Resource reservations in such systems guarantee quality of service to users and flexibility and predictability to management systems and administrators. Several reservation-based systems are available, but they do not consider energy consumption as a resource to optimize.

We have hence proposed ERIDIS: an Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems. ERIDIS can optimize the energy consumption of computing and networking resources and have a flexible and adaptive reservation management that satisfies user requirements through strict reservation policies. It also provides a unified framework to manage heterogeneous resources in an energy-efficient way. The ERIDIS architecture includes wattmeters to monitor the energy consumption of resources and provide this consumption to the resource manager and to the users. The resource manager uses this information to take energy-efficient decisions, specially in reservation scheduling to estimate the energy consumption of a reservation at several possible dates and hence choose the least consuming option. The algorithm aggregates reservations to increase the length of resource idleness. Indeed, ERIDIS switches off unused resources for energy saving purposes since idle resources are really consuming. Prediction algorithms are also employed to avoid useless off-on cycles.

As a proof of concept, we have adapted ERIDIS to three different application fields to

demonstrate its capabilities:

- data-center and Grid management with the Energy-Aware Reservation Infrastructure: EARI (Chapter 5),
- virtualized environments and cloud management with the Green Open Cloud: GOC (Chapter 6),
- data transfers in dedicated networks with the High-level Energy-awaRe Model for bandwidth reservation in End-to-end networkS: HERMES (Chapter 7).

The scientific mind does not so much provide the right answers as ask the right questions.

Claude Lévi-Strauss

5

Energy-Aware Reservation Infrastructure for Data Centers and Grids

5.1 Introduction

The energy consumption of data centers worldwide doubled between 2000 and 2006 [McK09]. Moreover, the incremental energy required by the US between 2008 and 2010 to power their data centers is equivalent to the amount generated by ten new power plants [McK09]. These alarming figures demonstrate the need for new technologies and infrastructures that increase the energy efficiency of large-scale distributed systems such as data centers and computing Grids.

One of the main challenges in making large-scale distributed infrastructures more energy-efficient is to reduce the energy wastage, caused by maintaining resources powered on even when they are not in use as seen on Chapter 2. We argue that Grids require energy-aware frameworks capable of switching off unused resources without impacting the performance and usage of user applications. To this end, we propose the Energy-Aware Reservation Infrastructure (EARI) [OLG08a, OLG08b, LO09] adapted from the ERIDIS model described on Chapter 4.

EARI is devoted to Grid infrastructures where users can reserve resources in advance specifying the duration of reservations, the number of required resources and the start time. When a reservation is accepted, the scheduler places it in its agenda.

This chapter is organized as follows. Section 5.2 presents our analysis of the usage of an experimental distributed platform. These results are correlated with energy measurements on one site in Section 5.3. Exploiting this information, we present the components of EARI in Section 5.4. Section 5.5 details the prediction models. We present validation results of the prediction algorithms and EARI in Section 5.6 and conclude in Section 5.7.

5.2 Understanding large-scale experimental grid usage

To better visualize stakes and potential energy savings, one needs to understand the resource utilization at different levels of the Grid (i.e. the grid itself, the clusters and the nodes). In this section, we analyze the resource reservations made by users of each of Grid'5000 sites over a year (i.e. 2008) [OL10]. Grid'5000 platform has been previously described in Section 3.1. The node reservation traces were obtained using the history provided by the Grid'5000 scheduler, OAR[1]. This information, consisting of nearly 1.2GBs of data, has been parsed to produce the statistics reported in this section.

By analyzing the usage history, we attempt to provide graphs and information that demonstrate the resource utilization of each Grid'5000 site. The time unit considered here is the second, a *resource* corresponds to a processor core and a *job* is a reservation. When reserving resources, a user informs the resource manager about characteristics of the reservation, including at least its start time, duration, and the number of required resources. The resource manager, responsible for admission control, verifies whether the job is compatible with previously accepted jobs

[1] OAR is a resource manager (or batch scheduler) for large clusters [CDCG+05].

and if it is, provides the user with a job ID.

When a resource is not available to the users, it can be in one of the following states:

- **dead**: the resource is down (e.g. due to a component failure);
- **suspected**: the resource does not work properly, but a failure has not been identified yet;
- **absent**: the resource is not physically present.

It is important to note that during the period over which the logs were collected, the infrastructure was constantly evolving, with new nodes being added and old ones being removed. For this reason, the results are percentages of the platform's capacity at the time of measurement (e.g. 100% of utilization at two different measurement times may not represent that the same number of resources were in used at both measurements). We do not present the results for the site located in Grenoble because it is used as testbed for Grid'5000 solutions and hence parts of the logs were not available.

5.2.1 The global view

Table 5.1 presents the results by site for 2008.

Site	Number of resources (cores)	Number of jobs (reservations)	Mean number of resources per job	Mean duration of a job (seconds)	Percentage of 'real' activity
Bordeaux	650	356222	7.44	2473.38	53.20%
Lille	618	344538	8.11	3154.58	72.89%
Lyon	322	138217	4.39	3723.55	69.27%
Nancy	574	74592	14.63	8912.82	60.08%
Orsay	684	92862	14.58	6246.07	57.82%
Rennes	714	58843	27.32	7069.33	64.58%
Sophia	568	58142	22.14	8767.35	81.51%
Toulouse	434	166191	6.29	2211.80	61.67%

Table 5.1: Job-related statistics

We observe that the usage varies greatly from one site to another (Table 5.1) in terms of number of cores, average number of cores per reservation, duration of reservations and the percentage of real work. This percentage represents the working time percentage excluding the dead and absent time periods during which the nodes do not consume any energy because they are unplugged.

The heterogeneous usage can be due to geographical purposes (the most distant sites are interesting to conduct communication experiments) and to hardware purposes: each site has different nodes with different architectures (storage, network capabilities...).

5.2.2 The site view

Figure 5.1 shows, as an example, the state of nodes at the site of Lyon with 322 cores. The black line indicates the number of reservations per week. For each week, the red areas represent the percentage of time during which cores are dead or shutdown; orange is percentage of time of suspected nodes; yellow is when nodes are absent; and green corresponds to nodes in use (i.e. a reservation is running).

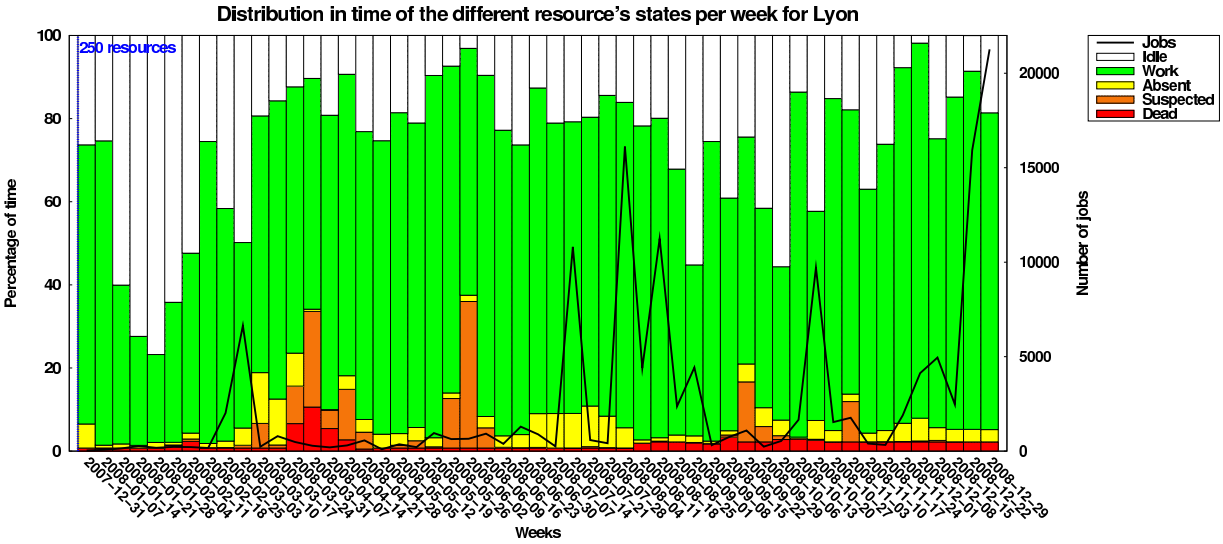

Figure 5.1: Percentage of nodes/time at each state for the Grid'5000's Lyon site.

The real percentage of work time at Lyon is 69.27%. Although during some weeks the usage is low, as shown in Figure 5.1, the real concern of such an experimental Grid is to handle periods of burst in work and communication specially before well-known deadlines, and such periods exist.

As the graph shows, there is no direct link between the number of reservations and resource utilization (i.e. the green areas), which means that reservations are really heterogeneous in terms of duration and number of used resources. The platform is utilized for a wide range of experiments from several research communities working in various domains, including distributed systems, middleware designers, high-performance computing and networking.

5.2.3 The node view

We identify at each week the two resources that namely correspond to the median (Figure 5.2) and the maximal (Figure 5.3) usage. The maximal resource has the greatest work time among the resources examined during the monitored period, whereas the median resource has the cumulative work nearest to the median value computed during the experiment.

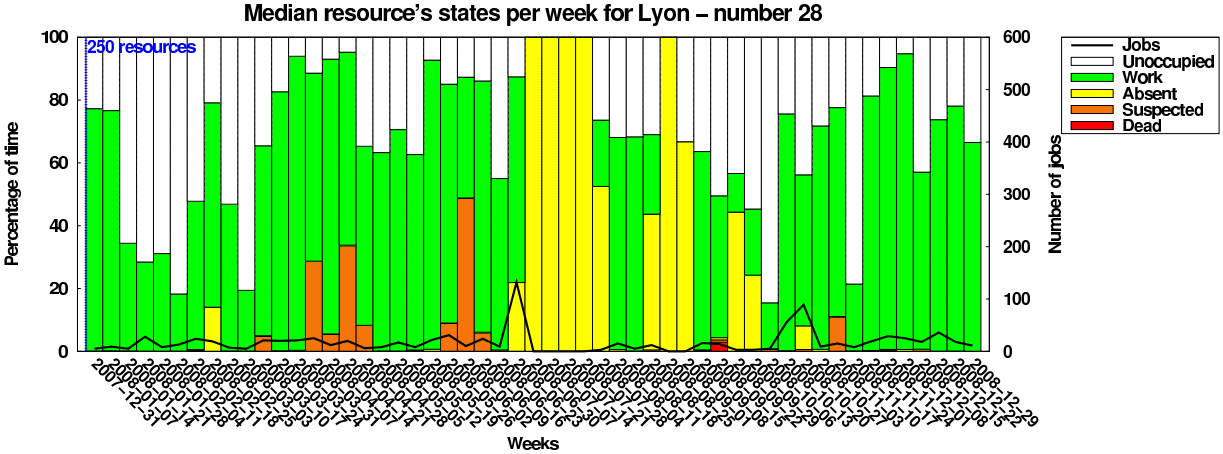

Figure 5.2: Median resource diagram for Grid'5000's Lyon site

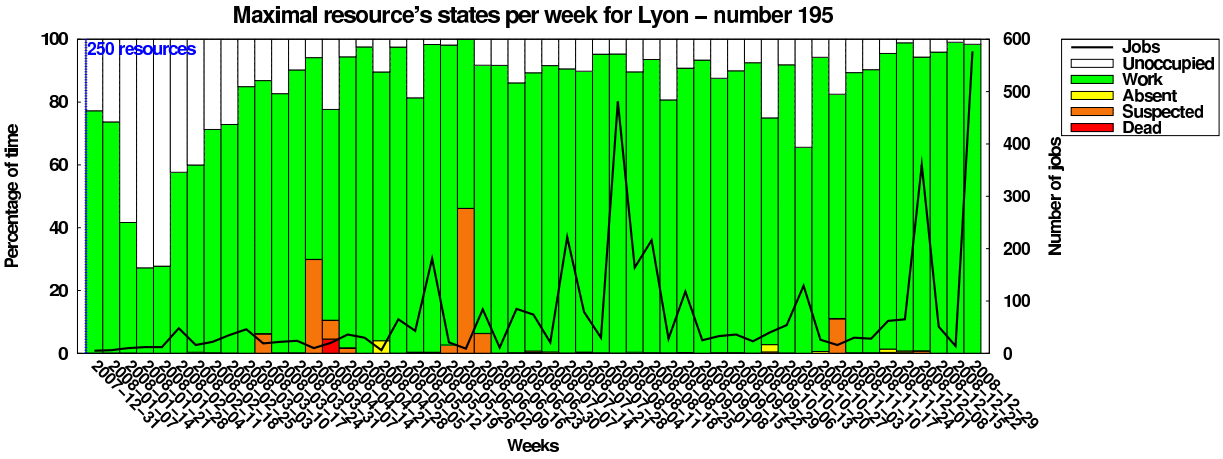

Figure 5.3: Maximal resource diagram for Grid'5000's Lyon site

From the usage of these two resources, we can infer that the utilization of resources in the same site varies greatly. While the maximal resource is used 82.3% of the time over a year, the median resource is used 51.1% of the time. This difference is partly due to failures since failed nodes are not considered in use, and nodes behaving strangely are less likely to be reserved. The other cause is that the scheduling system does not balance the load fairly among the nodes. Even if all the resources are necessary to support bursts in work, some nodes are largely unused.

The next section analyzes the correlation between resource usage and energy consumption. Since the monitoring infrastructure described in Chapter 3 was not available in 2008, we present information based on energy consumption logs extracted in 2010 focusing on the Lyon site, which is the only site whose power consumption is fully monitored.

5.3 Understanding the energy consumption of a site

We advocate that greater energy savings can be achieved if one understands how users utilize reserved resources and how much energy their reservations consume. The energy consumption data considered here spans six months, more precisely from 1st September 2009 to 27th February 2010 [DdAOL10].

5.3.1 Global energy consumption

The graph in Figure 5.4 shows the energy consumed by the servers of the Lyon site during six months; the overall consumption during this period was approximately 103.047 MWh. The intervals in green correspond to periods during which the energy consumption information is not available, either due to failures in the devices responsible for measuring the power consumption or down-times of the Grid infrastructure. Although the energy data of some of the nodes was available during these intervals, we maintain only the periods where data from all servers is at hand. The energy consumption of servers during days with failures shorter than two hours was obtained by using the average in KW from the available data as the consumption during the missing seconds.

Figure 5.4 also presents the resource utilization according to the reservation log obtained from the Resource Management System (RMS); the utilization indicates the percentage of reserved nodes, and hence does not imply that CPUs, storage or network resources were used by reservations at the same rate, as demonstrated in later sections. Considering the intervals under low resource utilization, one can observe that the static consumption of the platform is about 600 KWh. We also term the static consumption here as *idle consumption* since it corresponds to the electrical power drawn by servers when they are not actively executing user applications.

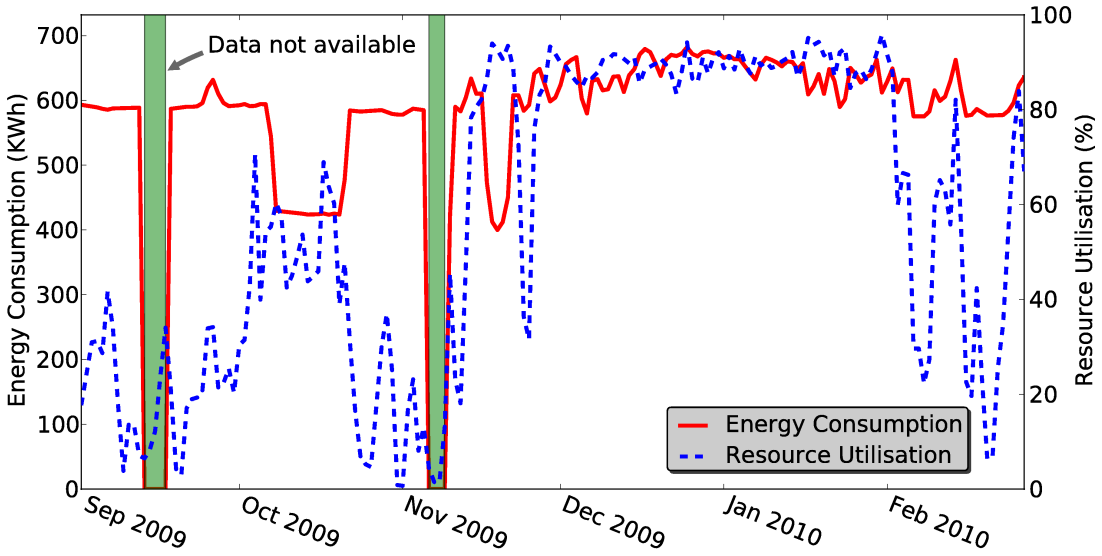

Figure 5.4: Energy consumption and utilization of nodes over six months.

Although the dynamic energy consumption is small when compared to the idle consumption, the graph indicates that during the months of January and February the former is proportional to resource utilization. However, there are troughs in the energy usage line when the consumption is much lower than the average.

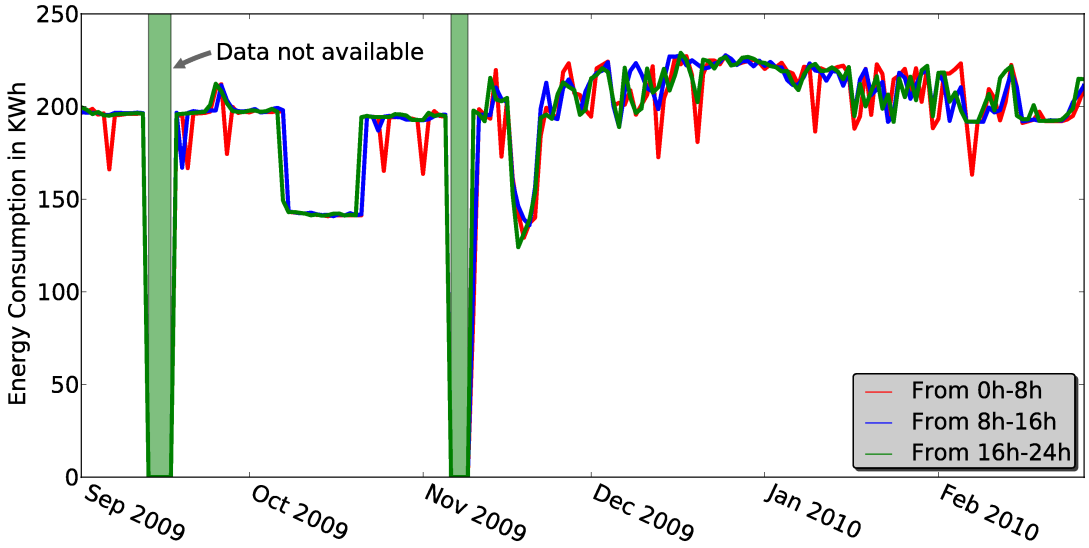

Figure 5.5: Energy consumption at different periods of the day.

To check whether the platform consumes different amounts of electrical power depending on the time of the day, we divide the day into three periods of eight hours each and plot the consumption of all servers during these different intervals. Figure 5.5 summarizes the results. It can be observed that the consumptions at different time intervals do not differ greatly because most of the energy consumed by the platform may be idle consumption (i.e. the minimum energy consumed by servers even when they are not executing user applications). This distance between static and dynamic energy consumptions would be larger if the platform employed more recent hardware with technologies such as CPU frequency scaling and auto power down of unused cores detailed in Chapter 2. In spite of the small dynamic energy consumption, a better understanding of resource usage patterns of reservation requests could lead to system optimizations for reducing the energy consumed. Hence, the next section provides details on the

energy consumed by the resource reservation requests.

5.3.2 Resource reservations

Table 5.2 summarizes the energy consumed by different categories of reservations. The second column shows the overall energy consumed by the reservations under each category, whereas the third column presents the average number of watts per node. The watts per node wn_j of a reservation request j are given by $wn_j = cs_j \div m_j \div p_j$, where cs_j is the energy consumed by reservation j, m_j is the number of nodes required by the reservation, p_j is the duration of j.

Table 5.2: Energy consumed by different reservation categories.

Number of nodes used	Number of reservations	Overall consumption in KWh	Average Watts per Node
1 to 2	3844	1529.77	197.23
3 to 5	681	1203.71	192.36
6 to 10	611	7384.92	200.44
11 to 30	7408	35371.17	216.49
31 to 70	205	9821.79	178.78
71 to 100	45	1918.33	185.91
101 to 135	50	5447.23	185.85

The average values in watts demonstrate that reservations that allocate up to 30 nodes tend to consume more energy. The highest average is given by reservations that require between 11 and 30 nodes. If one considers that resource utilization is proportional to dynamic energy usage, it can be observed that the low average watts of large reservations demonstrate that: small reservations are more resource intensive; and users may make large reservations to ensure exclusive use of a cluster or the whole infrastructure. This is a common practice for network experiments, where concurrent usage of the platform by multiple applications could compromise the tests and undermine the obtained results. This shows that optimizations can be made to allow unused machines to be switched off or placed on small consumption status when they are not in use.

5.3.3 The power consumptions of nodes

As discussed beforehand, the server infrastructure consumes a substantial amount of energy even when idle. This section attempts to quantify the static power consumed by the server infrastructure.

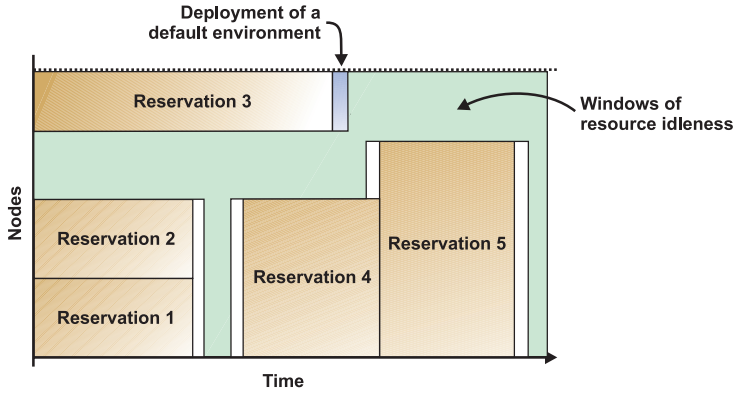

Figure 5.6: Obtaining the windows of resource idleness.

To compute the static power consumption of servers – i.e. the idle power consumption – only the power consumption measurements obtained during windows of resource idleness are utilized, as depicted in Figure 5.6. When computing the idle consumption of a node, all power measurements carried out during reservations for that node are discarded. We also disregard measurements within an interval of five minutes before and after reservations, and during environment deployment periods. Moreover, we ignore all power measurements carried out before 1st December 2009 in order to avoid events such as air-conditioning problems and large gaps in the power consumption data.

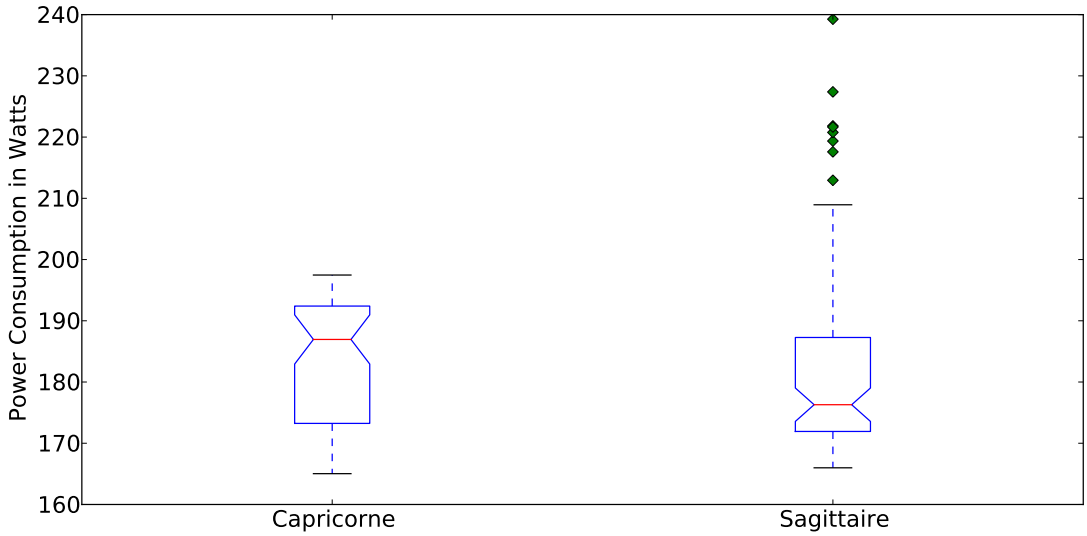

Figure 5.7: Average idle power consumption of servers.

Figure 5.7 reports the average idle power consumption of the two clusters located in Lyon. The heterogeneous idle consumption of sagittaire nodes was expected since some nodes of this cluster (i.e. from 70th to 79th) have been acquired after the remaining nodes and have higher power consumption. The different consumptions of capricorne nodes and the remaining of sagittaire's can depend on several factors as seen in Section 3.4.1. The idle power consumption of the nodes is not constant and is greatly impacted by the room temperature [TGV07, VBG09]. Hence, when a node is idle, its power consumption increases if the neighboring nodes are running and producing heat. The locality of a node in the rack seems to be another influencing factor because the air-conditioning pumps cold air into the room through the floor. Nodes located at the top of the rack tend to heat more and consequently consume more power. Moreover, some of the machines have been in production for several years, specially capricorne nodes, and some components have been replaced (e.g. hard-disks, memory and network cards). Although the information about the replaced components is not at hand, it certainly leads to a hidden heterogeneity that can also influence the idle power consumption of nodes.

To estimate the average dynamic power consumption of nodes – i.e. the consumption incurred by running user applications – we use the measurements obtained during the resource reservations. In addition, only measurements obtained after 1st December 2009 are considered when computing the averages and standard deviations. We term as "busy power consumption" of a node the average electrical power it draws during reservations. The "dynamic consumption" of a node is given as the difference between its busy and idle consumptions.

Figure 5.8 presents the busy and dynamic consumptions for the nodes of capricorne cluster. We can observe that both the average busy consumption and the standard deviations are higher than those presented in the idle consumption graphs. Although obvious, it shows the impact of resource usage on power consumption. However, as illustrated by the dynamic consumption bars, the power consumed by servers during the reservations is small when compared to the

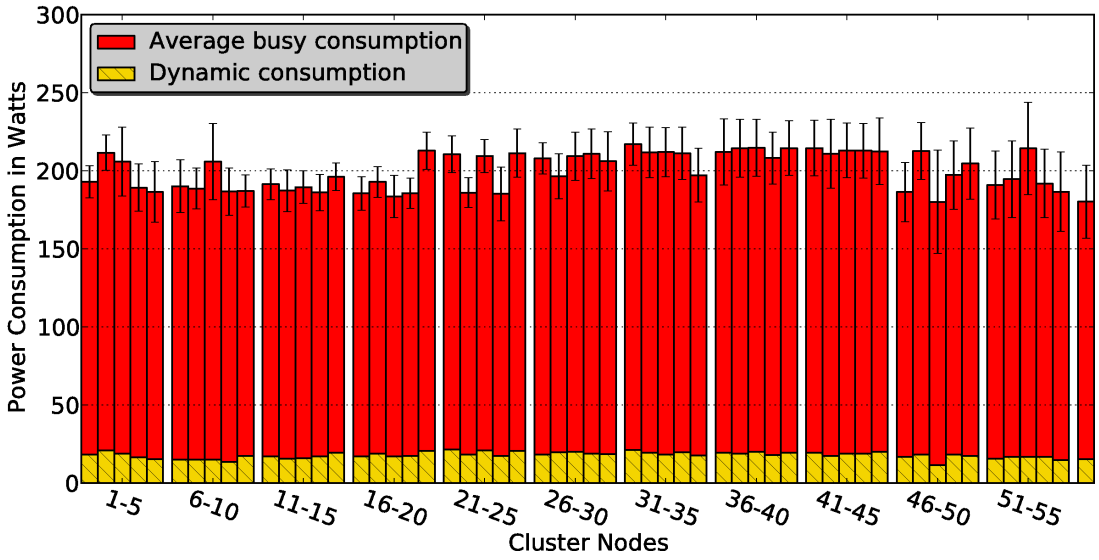

Figure 5.8: Average busy power consumption of capricorne's nodes.

overall power drawn by the servers.

The *application-driven energy consumption* – or dynamic consumption – is the share of energy consumed by using server resources during the reservations, hence disregarding the servers' idle consumptions. Calculating the application-driven consumption is important for evaluating allocation policies that attempt to curb the energy consumption by using server-level power management techniques such as CPU throttling.

The application-driven energy consumption for the six-month period was 1,823.47 KWh whereas the total amount of energy consumed by servers during the resource reservations was 59,688.53 KWh. Hence, the application-driven consumption accounts to approximately 3.05% of the energy consumed during the reservations. These results show that allocation policies using power management techniques such as CPU throttling are not appealing for the studied experimental infrastructure since the load posed by applications seem to be small when compared to the idle energy consumption. However, these results emphasize the need for management techniques that attempt to improve the energy efficiency of the platform by curbing the idle server consumption (e.g. approaches based on switching off unused resources).

5.4 Energy-Aware Reservation Infrastructure

5.4.1 EARI's architecture

As shown in the previous sections, Grids and data centers require energy-aware frameworks capable of switching off unused resources without impacting the performance and usage of applications. By adapting ERIDIS to this context, we propose the Energy-Aware Reservation Infrastructure (EARI) to manage the computing resources of Grids and data centers.

EARI's architecture, depicted in Figure 5.9 in the context of a cluster, is composed of a traditional Grid infrastructure with users, a portal, a scheduler and resource manager, and Grid resources. However, it also comprises a set of energy sensors plugged to the resources and an energy-aware manager responsible for applying the green policies.

A user reserves resources using the portal, whereas the scheduler processes and validates reservation requests. The scheduler also manages the infrastructure, giving users access to resources according to their reservations and the scheduling agenda. Sensors monitor and store data about the energy the resources consume; data that is used to compute "green" advices given to users to influence their reservation choices. The infrastructure also computes the consumption

profiles of past reservations and makes them available to users via the portal. Last but not least, EARI decides when to switch resources on and off by using prediction algorithms based on the resource agendas.

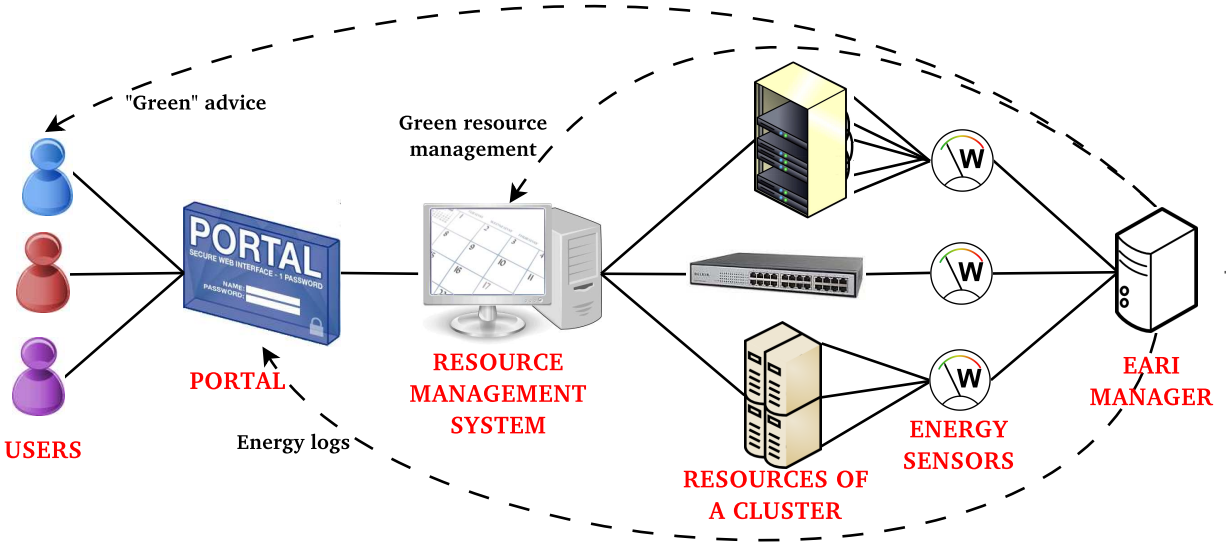

Figure 5.9: EARI components

Although each cluster or site has its own resource manager, all managers are linked and communicate in order to satisfy user requests.

5.4.2 Energy-efficient resource management

The idea when aggregating reservations is to offer several scheduling possibilities to users instead of just accepting or refusing their requests. To minimize the number of times resources are switched on or off (i.e. on/off cycles), EARI proposes to place a reservation after or before another reservation that is already in the agenda. By accepting an offer made by EARI, the user avoids unnecessary on/off cycles and hence saves the energy that would otherwise be consumed during these cycles.

The algorithm executed when reservations arrive is an adapted version of the ERIDIS scheduling algorithm (Algorithm 4.2). However, instead of testing all the scheduling possibilities like ERIDIS does, EARI tries four different solutions estimating the energy consumption of each solution. The user's request is denoted by $R = (l, n, t_0)$, where l is the duration of the reservation, n the number of resources and t_0 the start time (it equals the submission time by default).

The estimations are performed for R starting at:

- t_0 (or t_1, if t_0 is not possible; t_1 is the smallest possible start time);
- just after the next possible event, called t_{end};
- l seconds before the next possible event, termed as t_{start};
- during a slack period (time $\geq$ 2 hours and usage under 50%, see Section 5.5.4), at t_{slack}.

To perform these estimations, we need to compute: t_1, t_{end}, t_{start} and estimate t_{slack}. The complexity of this adapted scheduling algorithm is linear in the number of events in the agenda, contrary to the ERIDIS scheduling algorithm, which is quadratic. Therefore, although the EARI scheduling algorithm is not optimal, it is faster and specially designed to manage large numbers of resources and reservations.

Finally, the resource manager provides the estimations (i.e. energy consumptions and corresponding start times) to the user who then selects his favorite solution, still being able to disregard the advice given by EARI. The goal of the management is to glue reservations in order to avoid resource initialization and shutdown, which typically consume energy. However, EARI does not impose a solution on users, it just offers possibilities with different power consumptions and start times. The user chooses a solution depending on his energy-awareness and deadline.

Moreover, resources are not identical as they present different architectures and components, and hence do not consume the same amount of power.

The scheduler therefore tries to allocate resources with the smallest power coefficient, calculated depending on the mean power consumption of the resource during reservations over a great period of time. A resource that consumes less energy will have a smaller power coefficient and will hence take priority over other resources.

This allocation policy is employed when the user does not specify resource constraints. When the scheduler glues reservations together (by using t_{end} or t_{start}), it allocates the resources that are already awake (and gives priority to those with the smallest power coefficient). The power coefficient of a resource is calculated when it is added to the platform.

5.5 Predictions

EARI uses the same algorithm employed by ERIDIS as reservations complete to know whether the resources should be switched off (Algorithm 4.3).

The efficiency of this model, compared to a simple algorithm that puts resources into sleep state when they have no work to perform, resides in EARI's ability to predict accurately: the next reservation (number of required resources and start time), the energy consumed by a reservation and the slack period. In addition, the prediction algorithm should remain computationally simple in order to be efficient and applicable during reservation scheduling.

5.5.1 Predicting the next reservation

The start times of reservations made in recent past are used to estimate the arrival of the next reservation. The basic idea is to average out the characteristics of the six previous reservations of a site and use these averages as estimates for the next reservation. We use six reservations because this number gives the best results for the Grid'5000 traces (see Section 5.6). However, this parameter can be changed when EARI is employed in a different platform.

At time t, $R_0, \ldots, R_5$ represents the six previous reservations on a site, such that for $R_i = (l_i, n_i, t_i)$, l_i is the duration, n_i the number of resources and t_i the start time. These are the six reservations whose start times are the nearest to t – but not necessarily before t, and scheduled reservations can be taken into account. These reservations are ordered according to their start times (R_0 being the oldest).

The estimated start time t_e is the average of the five intervals between the start times of the six previous reservations added to t and a feedback:

$$t_e = t + 1/5[t_5 - t_0] + t_{feedback}$$

Similarly, n_e is the estimated number of resources required by the next reservation:

$$n_e = 1/5[n_1 + n_2 + n_3 + n_4 + n_5] + n_{feedback}$$

The accuracy of the prediction is essential to power management. If wrong estimations are made, resources either remain on waiting for imminent reservations that do not arrive or are turned off just before a reservation arrives, thus wasting the energy of one halting plus one reboot per resource.

5.5.2 Feedback on estimating the next reservation

The feedback is used to improve the energy efficiency of our approach since estimation errors are really penalizing in terms of energy consumption. The feedback is a corrective factor, calculated based on the three previous errors, that permits the algorithm to be really reactive to load variations.

When a reservation arrives, we compute the estimation errors. For example, at time t the reservation $R = (l_0, n_0, t_0)$ arrives, for which $R_e = (l_e, n_e, t_e)$ was the estimate. We denote by $Err_n = (n_0 - n_e)$ and $Err_t = (t_0 - t_e)$ the errors made by estimating respectively the number of resources and the start time of R.

Basically, if we predict the reservation too early, then $Err_t > 0$. Hence, Err_t is added to the next predicted start time to delay it by Err_t seconds. Then, we denote by $Err_t(a)$, $Err_t(b)$ and $Err_t(c)$ the three last errors for the predicted start time. The feedback $t_{feedback}$ is therefore:

$$t_{feedback} = 1/3[Err_t(a) + Err_t(b) + Err_t(c)]$$

$n_{feedback}$ is computed similarly.

5.5.3 Estimating the energy consumed by a reservation

The user, the resource type and the characteristics of the reservation $R = (l, n, t)$ are taken into account to estimate the energy consumption. The assumption made here is that a user has generally the same usage of resources during different reservations. Hence, we estimate the average power during working time per resource for each type of resource.

We denote by $R_a = (l_a, n_a, t_a)$, $R_b = (l_b, n_b, t_b)$ and $R_c = (l_c, n_c, t_c)$ the last three reservations made by a given user. $P_{real}(type_k, R_a)$ is the real average power per resource (in Watts) for reservation R_a – which requires resources of $type_k$ – during its work time. That is, if the reservation $R_a = (l_a, n_a, t_a)$ uses n_k resources of type $type_k$ and consumes $E_{real}(R_a, n_k)$ Joules (for its duration excluding the phases to setup and tear down resources); then:

$$P_{real}(type_k, R_a) = \frac{E_{real}(R_a, n_k)}{n_k \times l_a} = \frac{\sum_{i=1}^{n_k} E_{real}(R_a, N_k(i))}{n_k \times l_a}$$

where $E_{real}(R_a, N_k(i))$ is the total real energy (in Joules for the reservation R_a without transition times) consumed by the resource $N_k(i)$ whose type is $type_k$. We can then estimate the consumption of R. Supposing that there exists z resource types and n_k is the number of resources of type $type_k$, then estimated energy consumed by R during its working time is:

$$E_w(R) = l \times \left[\sum_{k=1}^{z} n_k \frac{(\sum_{\alpha \in \{a;b;c\}} P_{real}(type_k, R_\alpha))}{3} \right] + l \times n \times E\_feedback$$

We must add to this consumption, the costs of setting up and tearing down resources. To accomplish that, we should know the number of resources that are turned on (excluding those that are already on) and the number of resources that will be turned off when the reservation finishes. These values are provided by the agenda.

We denote by n_{ON}, the number of resources that should be turned on for a reservation (without taking into account those that are already on), and n_{OFF} the number of resources that should be turned off once the reservation completes (without taking into account those that will remain on to be used by the following reservation). Therefore, $E_e(R)$ the total energy consumed by R is:

$$E_e(R) = E_w(R) + \sum_{u=1}^{n_{ON}} E_{ON \to OFF}(u) + \sum_{v=1}^{n_{OFF}} E_{OFF \to ON}(v)$$

where $E_{ON \to OFF}(u)$ is the energy required to turn off the resource u and $E_{OFF \to ON}(v)$ the energy needed to turn on the resource v.

The different estimations made by the scheduling algorithm are more obvious now as they depend on the number of resources that are turned on and off as well as the resources that are used. As the number and types of available resources vary over time, different working consumptions are possible. The feedback is, as defined previously, the mean error made per resource during the consumption estimations for the past three reservations.

The scenario can be improved if, when a user submits a reservation, he specifies the type of tasks he or she will predominantly execute – e.g. network intensive services, parallel computations or benchmark launching. A consumption model can be designed per application type (and per resource type) and used in coordination with the other parameters (like the user, the resource type and the reservation characteristics) to estimate the energy consumption. This estimation relies on energy consumption measurements since one needs to know the energy profile of each reservation on each resource type.

5.5.4 Slack periods

A slack is a period longer than two hours[2] with a platform usage below 50%. Such periods typically happen during the night.

Information from the three previous days is used to estimate when the next slack period will occur. If there were no slack periods during the previous three days, we estimate that there will be no slack period on the day for which the prediction is made. At the time of the day h, to estimate the next slack period, we look for the start times of slack periods from the previous days which occur as close to h as possible. These times, which can be found using the history, are denoted by $h_{slack}(h, d-1)$, $h_{slack}(h, d-2)$ and $h_{slack}(h, d-3)$ (d stands for the current day). The start time of the next estimated slack period is the average of these values:

$$h_{slack} = 1/3[h_{slack}(h, d-1) + h_{slack}(h, d-2) + h_{slack}(h, d-3)]$$

To be really complete, our model should include the energy consumed to cool each resource. In fact, $P_{real}(type_k, R_a)$ (the real average power for reservation R_a for a resource whose type is a $type_k$) would include a fraction of the average power consumed by the cooling infrastructure during the reservation R_a proportional to its heat production. However, such a fraction is difficult to estimate, and that is why most power management systems do not take the cooling infrastructure into account.

5.6 Validation using real traces

To evaluate the proposed model, we implement a replayer of Grid'5000 traces that parses the raw data about reservations extracted from OAR (XML format), and replays the reservations using the EARI scheduling algorithm and different green policies. The idea is to re-run the Grid'5000 traces and to aggregate reservations (considering that users would have chosen one solution among the four proposed by EARI). Finally, we can compare this new scenario with the real sequence in terms of energy consumption and delay of the moved reservations. These experiments are based on the 2007 traces.

5.6.1 Evaluation of the prediction algorithm

To test the prediction algorithm, the first experiment respects the reservation requirements given by the user and does not move reservations. We use the 2007 Bordeaux logs for Figure 5.10 and Figure 5.11. Figure 5.10 shows the percentage of energy consumption of our model with prediction and the percentage of energy consumption of the model without prediction where T_s

[2]This period is a little longer than the average length of reservations on Grid'5000. Hence many reservations can fall within this period.

varies from 120 seconds to 420 seconds and P_{idle} (the power consumed by a single resource when it is idle) is 100, 145 or 190 Watts. 100% represents the energy consumption when the future is known – an ideal and unattainable scenario where no prediction algorithm is needed because we know when to turn resources on and off. Actually, this represents the theoretical lower bound for energy consumption.

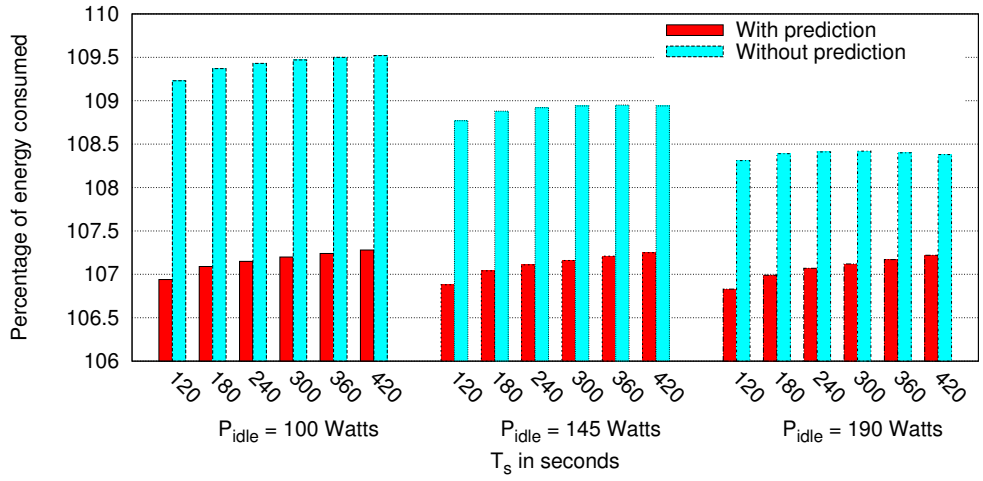

Figure 5.10: Percentage of energy consumption by using our model in relation to the energy consumed by knowing the future.

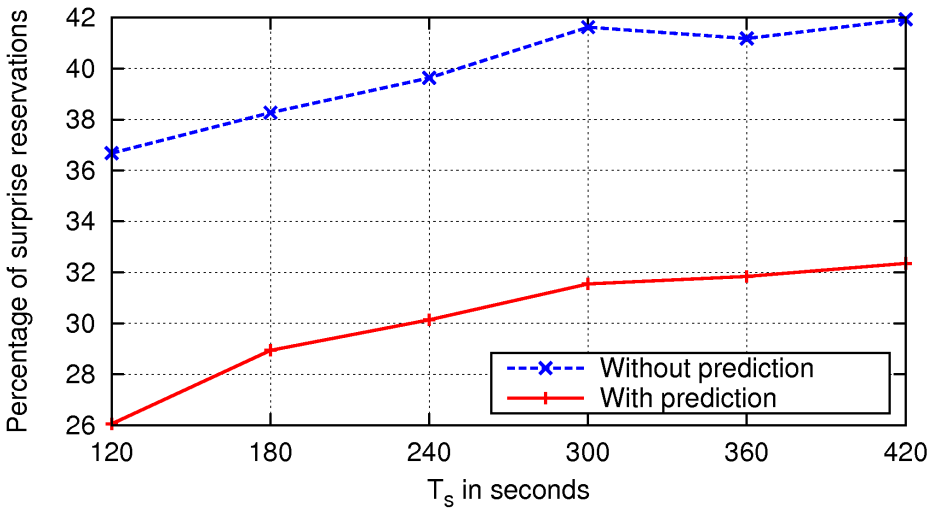

Figure 5.11: Percentage of surprise reservations in relation to total reservation number.

Based on the experimental measurements of Chapter 3, we set $P_{work} = 216$ Watts, $P_{OFF} = 10$ and $\delta_{ON \to OFF} + \delta_{OFF \to ON} = 110$ seconds. Based on the measurements of Figure 3.7, P_{idle} is set to 190 Watts, but it varies to simulate the future capacity of our model to shut down resource's components (e.g. a core or a disk). In the same way, T_s varies to simulate the future possibility to use hibernate modes (T_s is at least equal to the time needed to boot a resource plus the time to shut down it, so if we use suspend to disk or suspend to RAM mechanisms, it will decrease T_s).

Figure 5.10 shows that the total consumption with predictions is better than without predictions. Depending on the case, with the prediction algorithm, between 1.2% and 2.2% of the energy consumed by the entire site for a whole year can be saved.

Figure 5.11 shows the impact of surprise reservations with and without prediction. *Surprise reservations* are reservations that arrive less than T_s seconds after some resources are turned off – resources that could instead have been used to serve these arriving reservations. This is the reason why we are not closer to the known-future case in Figure 5.10. As expected, EARI predicts some of these reservations and has better results. Yet, it is not possible to achieve this goal perfectly (the future is not known!). The algorithm predicts if a reservation will occur in the next T_s period of time and it becomes less precise as T_s increases. These results prove that such a simple and fast prediction model can save significant amounts of energy compared to a naive on/off algorithm without prediction.

5.6.2 Evaluation of the aggregation technique

To evaluate EARI, we conduct experiments by replaying the 2007 traces. Reservations are moved along the time scale while respecting several policies. We evaluate the proposed techniques using data from four clusters' studied previously, representing four different workloads over a one-year period.

The following four diagrams (Figures 5.12, 5.13, 5.14 and 5.15), show the consumption with EARI in percentage compared to the consumption when all nodes are always powered on (current case). These consumptions are computed using the values found in Chapter 3. In the previous section, P_{idle} was set to 190 Watts. Here, we use three different values for P_{idle}, namely 100, 145 and 190 Watts. We consider that in near future, the idle consumption will decrease because we will be able to shut down unused cores or unused Ethernet cards (for example) automatically.

The graphs present for each P_{idle} the results of the six different policies in order to compare their energy savings. We fixed $T_s = 240$ seconds for these four diagrams (Figures 5.12, 5.13, 5.14 and 5.15) and present the ideal lowest bound, hereafter called "all glued". It is the ideal case where we glue all reservations, putting one after the other and switching the resources for the rest of the time. Hence, in this case we do not need any prediction and do not make prediction errors. This ideal case is not reachable because it assumes that we know all the reservations in advance, yet the future is unknown!

The prediction errors occur when we fail to predict an imminent reservation or when we predict a reservation that does not arrive. In the first case, we switch off resources that we will be needed in less than T_s seconds, wasting energy. In the second case, we let some resources run idle during T_s, which will be switched off after T_s seconds of inactivity.

Figure 5.12 presents the results for Bordeaux. As expected, in the three cases the *fully-green* policy consumes about 5% less energy. As expected too, the *user* policy consumes the most. We also notice that the *deadlined* policy is almost equivalent to the *50%-green* in terms of energy consumption.

It is important to notice that in the three cases, as P_{idle} varies, the "100%" varies as well since it corresponds to different amounts of energy in each of the three cases. In fact, this "100%" is computed as follows: the time spent to work times P_{ON} plus the time spent idle times P_idle. The times do not change between the three cases, but P_{idle} does.

Table 5.3 presents the average time that reservations have been delayed (res. stands for reservation) and the percentage of delayed reservations. For example, the *user* policy does not delay any reservation (so 0% of delayed reservations and 0 hours of the mean delay time). Moreover, these values do not depend on P_{idle}, but on T_s, and so we set $T_s = 240$ seconds for the previous diagram (Figure 5.12).

The percentage of delayed reservations for the *25%-green* policy is not 25% because of the reservations that we move. They can indeed take the place of reservations that do not move (the 75% that should follow the *user* policy), hence these reservations are put at the date nearest to the wished start time. The same is true for the *50%-green* and the *75%-green* policies.

In the case of the Bordeaux site, the mean delay time is not always relevant because we observe that during the three summer months all resources are down and EARI will delay some

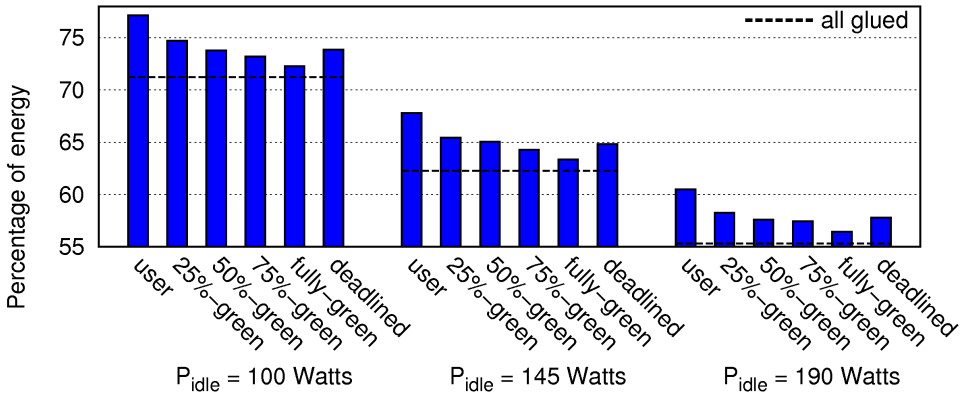

Figure 5.12: Energy consumption of Bordeaux with EARI with $T_s = 240$ s.

Policy	% of delayed res.	mean delay time
25%-green	40%	21 h.
50%-green	61%	33 h.
75%-green	79%	37 h.
fully-green	97%	218 h.
deadlined	68%	7 h.

Table 5.3: Statistics of delayed reservations for Bordeaux with $T_s = 240$ s.

reservations for three months.

Figure 5.13 presents the results for Lyon, which are similar to previous results (the *user* policy consumes the most and the *fully-green* the least), but they are more condensed. This is because Lyon has fewer reservations than Bordeaux, yet their work percentages are similar. Hence we can make fewer aggregations in Lyon because the reservations are bigger (larger number of resources proportionally compared to the total number of resources for each site) thus more difficult to move.

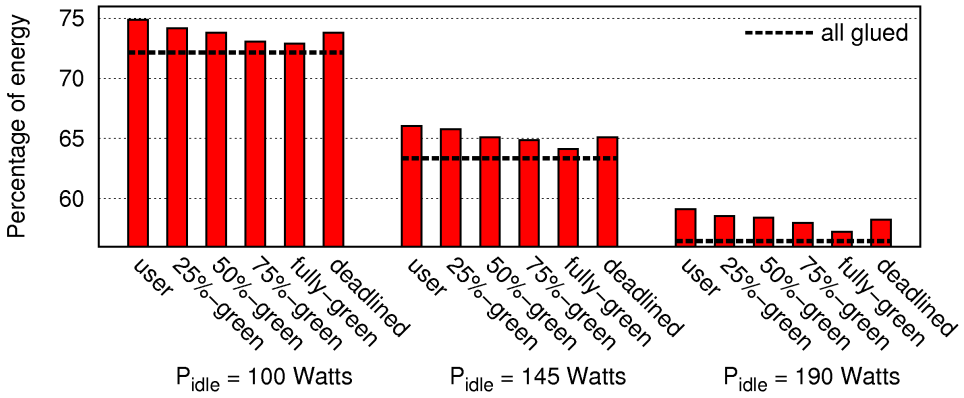

Figure 5.13: Energy consumption of Lyon with EARI with $T_s = 240$ s.

One can see in Table 5.4 that in the case of Lyon, our *fully-green* policy moves 99% of the reservations. This is normal, since moving some reservations causes a chain reaction resulting in moving most of the reservations. In this example, we see that the reservations are not delayed for more than 15 hours on average and this still leads to important energy savings.

Rennes (Figure 5.14) has the same problem: it presents big reservations in terms of both number of resources and length of reservations. Thus the reservations are hard to move and

Policy	% of delayed res.	mean delay time
25%-green	61%	12 h.
50%-green	75%	11 h.
75%-green	87%	12 h.
fully-green	99%	15 h.
deadlined	82%	7 h.

Table 5.4: Statistics of delayed reservations for Lyon with $T_s = 240$ s.

placed after or before other reservations. In this case, we still see that our *fully-green* policy consumes the least energy and is really near the *all glued* bound, so our prediction models and *fully-green* policy are efficient. Our prediction algorithm takes the right decision (to keep the resources on or to switch them off once a reservation finishes) in 70% of the cases on average.

Moreover, we see that sometimes our *75%-green* policy consumes more than the *50%-green* one. This is due to the random factor where moving a small reservation prevents us from moving a large one or several other reservations. This behavior is not energy-efficient, and so adding randomness does not necessarily lead to decrease in energy consumption.

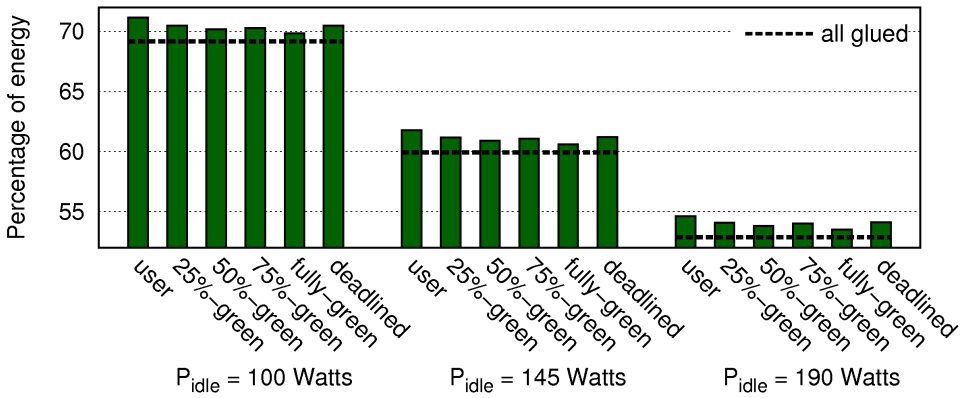

Figure 5.14: Energy consumption of Rennes with EARI with $T_s = 240$ s.

Table 5.5 shows the statistics for the delay, where in the case of Rennes, the mean delay is really short. Compared to the cluster size (714 resources), the reservations are not really big in terms of number of resources (55 on average), and so easier to move.

Policy	% of delayed res.	mean delay time
25%-green	44%	9 h.
50%-green	62%	9 h.
75%-green	81%	8 h.
fully-green	97%	7 h.
deadlined	73%	4 h.

Table 5.5: Statistics of delayed reservations for Rennes with $T_s = 240$ s.

In the present situation (where all the nodes are always powered on and consume 190 Watts when they are idle), we could save 73800 kWh only for Lyon by using the *fully-green* (considering only computing nodes and not networking devices and cooling infrastructure). This represents the consumption of a TGV[3] covering about 5000 km.

[3]The consumption of a TGV train (French high-speed train) is about 14.79 kWh per km.

The last diagram (Figure 5.15) presents the results for Sophia. It also confirms that our *fully-green* policy is the most efficient for different types of workload over a large period of time.

We notice that, even when our *fully-green* policy saves a small percentage of energy, this percentage represents a large amount of energy at the scale of an entire cluster during a whole year.

As for the two previous cases – Lyon and Rennes – the possible energy savings are smaller in percentage than in the case of Bordeaux. This is also because in the case of Bordeaux, our prediction algorithm works better than in these two cases. For example, for Sophia, the prediction algorithm takes the good decision (in terms of switching on and off the resources) in 63% of the cases. In future work we intend to improve the algorithm.

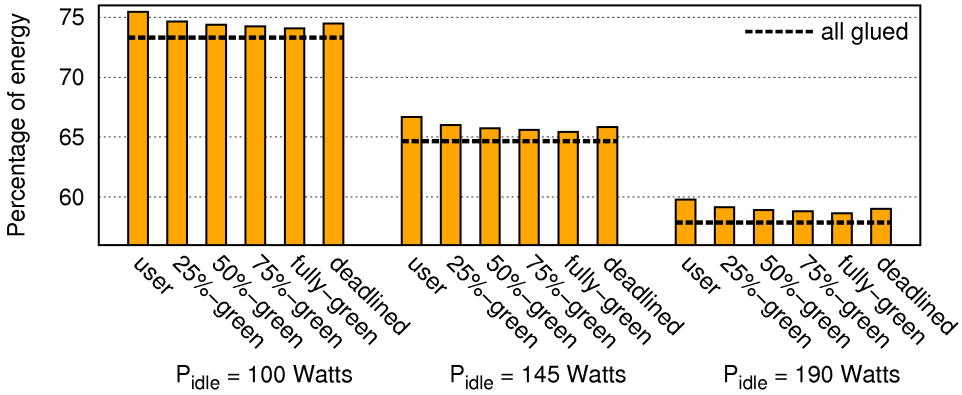

Figure 5.15: Energy consumption of Sophia with EARI with $T_s = 240$ s.

As in the other cases, we see in Table 5.6 that for Sophia, increasing the number of delayed reservations does not necessarily increase the time during which they are delayed. Indeed, the 40% of delayed reservations for the *25%-green* policy are delayed by 1 hour more than the 60% of delayed reservations for the *50%-green* policy.

Policy	% of delayed res.	mean delay time
25%-green	40%	8 h.
50%-green	60%	7 h.
75%-green	78%	14 h.
fully-green	94%	9 h.
deadlined	71%	5 h.

Table 5.6: Statistics of delayed reservations for Sophia with $T_s = 240$ s.

In another set of experiments we have looked at the impact of T_s on energy consumption. T_s can be changed to be more reactive, which means that resources remain idle for longer after reservations wait for new reservations to arrive. Thus, T_s is just the time that the resources wait for a predicted reservation once another reservation completes.

These experiments are shown in Figures 5.16 and 5.17 for Bordeaux and Lyon respectively. We have set $P_{idle} = 100$ Watts, a P_{idle} that we expected to be feasible in the future. T_s varies between 120 and 420 seconds by increments of 60.

We see in Figure 5.16 that the consumptions of the six policies are still similar to previous results. Moreover, we see that T_s has not a great impact on the global energy consumption. In fact, with larger T_s, the prediction errors can be compensated. If we fail to predict a reservation using a large T_s and if the reservation arrives in fact after the theoretical T_s, as defined in Section 4.6, we will not waste energy by definition of the theoretical T_s.

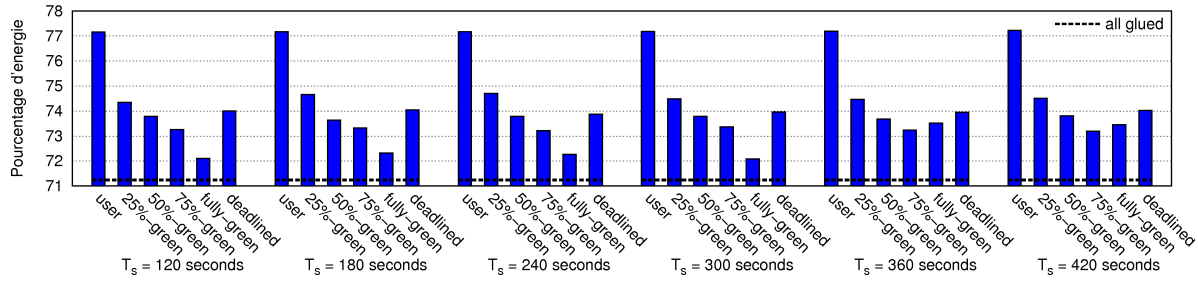

Figure 5.16: Energy consumption of Bordeaux with EARI with $P_{idle} = 100$ W.

However, we see in Figure 5.17 that these prediction errors can have huge consequences if they are combined with an unfavorable random factor (we have seen such case in Figure 5.14, the reservations that follow the *user* policy distort the prediction models and the algorithm always predict wrong imminent reservations, so the resources are often kept idle unnecessarily).

As a conclusion, we can say that a large T_s offers a better reactivity, but combined with prediction errors, it can reduce the energy efficiency.

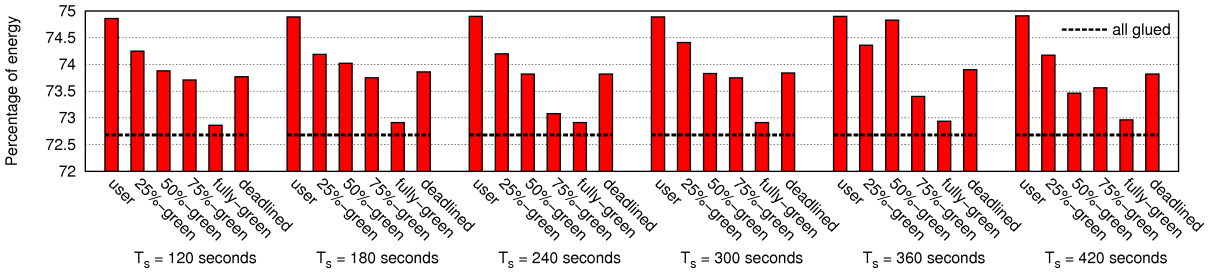

Figure 5.17: Energy consumption of Lyon with EARI with $P_{idle} = 100$ W.

One can wonder whether switching nodes on and off can damage resources. In fact, in the case of Grid'5000, they are already restarted as reservations begin since users need to deploy their own system images on the reserved nodes. As reservations complete, the resources are always rebooted and initialized with a standard image.

We have seen that all the policies are always more energy efficient than the *user* policy. All these results show that EARI can lead to significant energy savings even for the future Grids.

5.7 Conclusion

In this chapter, we have first studied the utilization of a real experimental platform, Grid'5000, in terms of both resource usage and energy consumption. This analysis reveals heterogeneity in terms of utilization and resource wastage.

Then, we have proposed EARI, an energy-efficient model, derived from ERIDIS, that reduces the global consumption of data centers and Grids. Based on workload consolidation, on/off algorithms and reservation prediction, EARI aims at reducing the energy consumption of the overall system it manages. This infrastructure is efficient and can be easily implemented and deployed.

We have presented results that validate this proposition and reveal that large energy savings are possible with minor impacts on application performance. These validations are based on the Grid'5000 traces previously studied. They show that using EARI could have saved more than 40% of the energy consumed in a year by the four studied sites. An extrapolation to the whole Grid'5000 platform shows that for one year, we could have saved 52% of the consumed energy. These energy gains represent the consumption of a French village with 600 inhabitants.

This work has led to the implementation of a *PowerSave mode* in OAR (the Grid'5000 batch scheduler) and green policies to switch off unused nodes [DCDdAG+10]. These energy-efficient features are currently under test on the Grid'5000 platform[4].

[4]Green features of OAR: `http://wiki-oar.imag.fr/index.php/Green_OAR`.

The cloud never comes from
the quarter of the horizon
from which we watch for it.

Elizabeth Gaskell

6 Green Open Cloud

6.1 Introduction

Cloud solutions have become essential to current and future Internet architectures as they provide on-demand access to virtually unlimited numbers of computing, storage and network resources. The elasticity of Clouds allows for the creation of computing environments that scale up and down according to the requirements of distributed applications.

Through economies of scale, Clouds can efficiently manage large sets of resources; a factor that can minimize the cost incurred by organizations when providing Internet services. However, as Cloud providers often rely on large data centers to sustain their business and supply users with the resources they need, the energy consumed by Cloud infrastructures has become a key environmental and economical concern. Data centers built to support the Cloud computing model can often rely on environmentally unfriendly sources of energy such as fossil fuels [Gre10b].

Clouds can be made more energy efficient through techniques such as resource virtualization and workload consolidation. After providing an overview of energy-aware solutions for Clouds, this chapter presents an analysis of the cost of virtualization solutions in terms of energy consumption. It also explores the benefits and drawbacks that Clouds could face by deploying advanced functionalities such as Virtual Machine (VM) live migration, CPU throttling and virtual CPU pinning. This analysis is important for users and administrators who want to devise resource allocation schemes that endeavor to reduce the CO_2 footprint of Cloud infrastructures.

As we attempt to use recent technologies (e.g. VM live migration) to improve the energy efficiency of Clouds, this work can be positioned in the context of future Clouds. We adapt ERIDIS and propose an energy-efficient reservation framework termed as Green Open Cloud (GOC) [LO10] to manage Cloud resources. Benefiting from workload consolidation [SKZ08] enabled by virtualizing resources, the goal of this framework is to curb the energy consumption of Clouds without sacrificing the quality of service (in terms of performance, responsiveness and availability) of user applications. All components of the GOC architecture – Green policies, prediction solutions and network presence support – are presented and discussed. We demonstrate that under a typical virtualized scenario GOC can reduce the energy used by Clouds by up to 25% compared to basic Cloud resource management.

This chapter is organized as follows. Section 6.2 investigates the energy consumption of virtual machines. The architecture of GOC is presented in Section 6.5 and validation results are presented in Section 6.4. We conclude in Section 6.5.

6.2 Energy consumption of virtual machines

6.2.1 Context

Cloud computing appears as a promising model to serve the increasing demands of today's complex applications for computing power. Virtualization is generally promoted by research work as a means to decrease the energy consumption of large-scale distributed systems. However, the studies often lack real data on the electric consumption of virtualized infrastructures.

Our goal is to determine the energy cost of the life-cycle of a VM by measuring the power consumption of its host machine when the VM runs a job, when it is idle (no job), and when the VM is migrated. To identify this cost, we have performed some experiments whose results are later used for calibrating our energy-aware models.

Our experimental Cloud consists of HP Proliant 85 G2 Servers (2.2 GHz, 2 dual core CPUs per node) with XenServer 5.0[1] installed on them. Each Cloud node is linked to an external wattmeter that logs its power consumption every second. These energy logs are sent to an energy data collector for storage and are then made available to users via a Cloud portal. The measurement precision of our experimental platform is 0.125 watts and the maximum frequency is one measure per second, which is accurate enough to have a good idea about the consumption of virtual machines.

6.2.2 Power consumption of a virtual machine

Virtual machine technology is an essential building block for Cloud infrastructures. VMs, which can be created by users, are spawned on demand to meet the needs of applications.

We use XenServer and create our VMs by using the provided pre-configured templates of Debian Etch 4.0. Figure 6.1 shows the power consumption of the host node during the instantiation of a VM. The VM starts at $t = 10$, before which time the node is idle and has no VMs. The power consumption during the idle time is termed as P_{idle} and is expressed in watts. For this Cloud node, $P_{idle} = 207.75$ watts on average.

The instantiation phase ends at $t = 165$, by which time the VM is ready with a working network interface (that allows *ssh* commands, for example) with its own IP address (different from the hypervisor's). During the instantiation, the power consumption varies a lot, and so does the processor load. The peak consumption is 218 watts. At $t = 140$ the user is asked the new password of the VM and the network interface is initialized just after it.

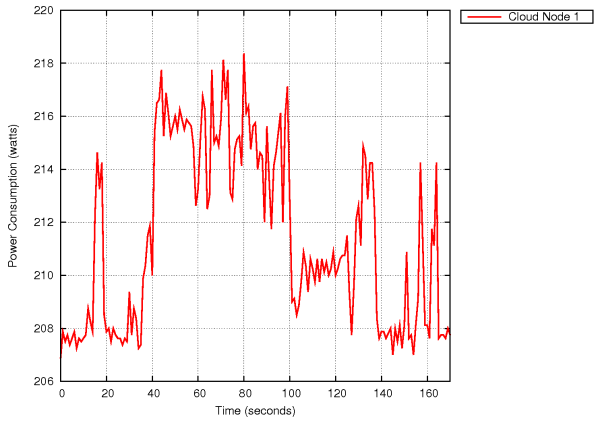

Figure 6.1: Virtual machine Instantiation.

Once the VM is instantiated, it can perform computing tasks, it can be stopped (shut down) and it can then be rebooted just like a real physical machine. Figure 6.2 shows the boot of a VM from $t = 10$ to $t = 30$, the execution of *cpuburn*[2] from $t = 40$ to $t = 100$ and then the shutting down of the VM from $t = 110$ to $t = 122$.

There are two notable aspects. First, the average power consumption from $t = 30$ to $t = 40$ is equal to the host's idle consumption (P_{idle}), which implies that an idle VM (with nothing running on it) does not make the host consume additional energy. This is counter-intuitive,

[1] XenServer is a cloud-proven virtualization platform that delivers the critical features of live migration and centralized multi-server management (`http://citrix.com/English/ps2/products/product.asp?contentID=683148`).

[2] cpuburn is a software designed to apply a high load to the processor (`http://pages.sbcglobal.net/redelm/`).

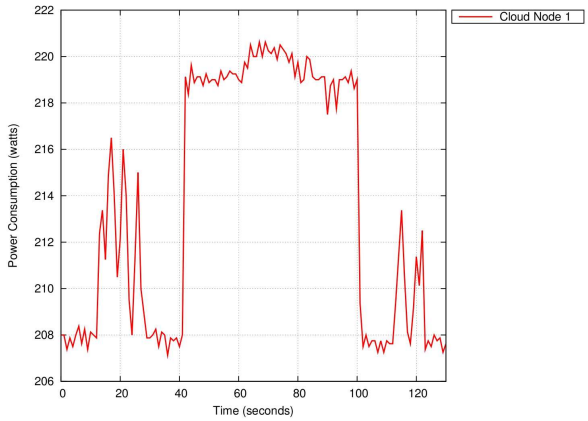

Figure 6.2: Life of a Virtual Machine: boot, run, and halt.

since the hypervisor consumes as much energy with 7 idle VMs as without any VM. Second, the boot and the shutdown consume really less energy than a *cpuburn*. For this reason, in future experiments and to simplify the interpretation of results, we disregard boot's and shutdown's. We consider simply the periods with jobs. Also for clarity, the jobs will all be *cpuburn*, and in this way we can compare the power consumption of different Cloud nodes when running jobs.

6.2.3 Competition

As there is only one hypervisor per node, VMs with high utilization can compete for resources. Indeed, if a VM process fully uses the hypervisor capacity, other VMs should wait for their turn.

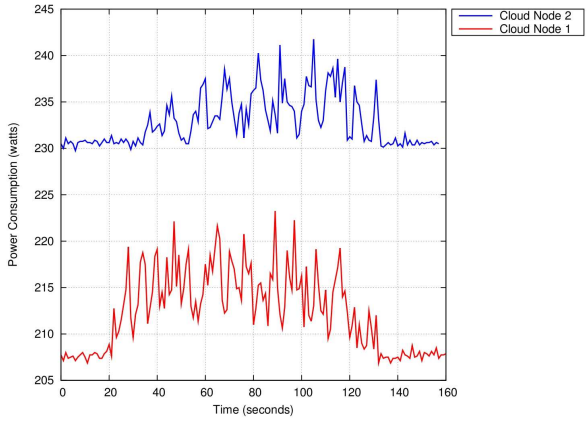

Figure 6.3: Booting 7 VMs simultaneously on two Cloud nodes.

Figure 6.3 presents the simultaneous boot of 7 VMs on two different Cloud nodes.[3] At $t = 20$ we boot the 7 VMs on Cloud node 1, one at a time (2 seconds between boot's) and at $t = 35$ we do the same on Cloud node 2. The simultaneity of the boots impacts their duration: they finish at $t = 130$ on Cloud node 1 and at $t = 135$ on Cloud node 2. In our case, a VM boot usually lasts 20 seconds (Figure 6.2). This example shows that the VMs are competing for hypervisor resources.

[3]The nodes do not have the same consumption because node 2 has high performance network cards (consuming additional 22 watts).

6.2.4 Migration

Migrating a VM on Xen consists of transferring its memory image from one Cloud host to another. If the VM is running and modifying its memory, it is stopped for a short period towards the end of the migration to copy the last dirty memory pages (those that are modified often). Hence, when a migration occurs, the completion of the job running on the migrated VM is delayed by a certain amount of time (the period during which the VM is stopped). This time, denoted by T_m, does not include the whole duration of the migration process. Moreover, if multiple migrations are required at the same time on the same node, they are queued and processed one at a time by the hypervisor (this can also be influenced by the network bandwidth).

In the experiment whose results are depicted in Figure 6.4 we have launched six *cpuburn's* one in each of six different VMs on Cloud node 1. The first starts at $t = 10$, when we see that the consumption rises to 209 watts. Then the second starts and the consumption reaches 230 watts. The third *cpuburn* starts and the node then consumes 242 watts. The fourth leads to 253 watts. Starting the fifth and the sixth jobs does not increase the consumption. The reason is that, as the jobs are CPU intensive (*cpuburn* uses 100 % of a CPU capacity) and as there are only four cores on the node (2 dual core CPUs), they are fully used with the first four VMs. The fifth VM appears as "free" in terms of energy cost because it shares already fully used resources. Obviously, these "energy-free" VMs have a cost in terms of performance.

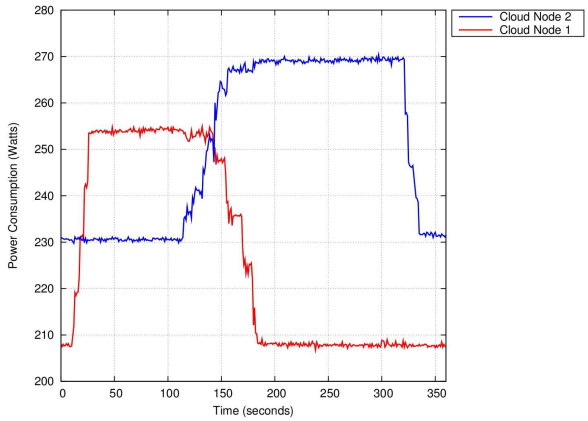

Figure 6.4: Migration of virtual machines.

Each *cpuburn* job lasts 300 seconds (Figure 6.4). At $t = 110$, we launch the migration of the 6 VMs from Cloud node 1 to Cloud node 2. The migration requires sustained attention from the hypervisor, which should copy the memory pages and send them to the new host node. Since it cannot handle 6 migrations at a time, they are performed one by one.

Competition occurs and we observe in power consumption of Cloud node 2 that the VMs arrived one at a time. The consumption of Cloud node 1 begins to decrease during the migration of the third VM, at which time, only three VMs remain running on the node. Each job completes 5 seconds late, this is T_m. The competition during the migration request does not affect much the jobs running on the last migrated VMs since they are still running during their wait for a migration.

6.2.5 Discussion

Migration is an important feature of our infrastructure. However, this technology has limitations [VBVB09], since it performs poorly with jobs that use intensely the network and the storage. The other main issue is to deal with the IP domain change when you move the virtual machine [TDG+06]. Moreover, this process is time consuming and costly in terms of network and storage usage. Although it is a process that continually uses the network and the disks of

the two host nodes, we will see that this technique is useful for our infrastructure and allows for great energy savings.

Among the components of a Cloud architecture, we have decided to focus on virtualization, which appears as the main technology used in these architectures. In the future, we plan to include algorithms employing other Cloud components such as accounting, pricing, admission control, and scheduling.

6.3 GOC architecture

6.3.1 Architectural components

The energy footprint of current data centers has become a critical issue, and it has been shown that servers consume a great amount of electrical power even when they are idle [OLG08b]. Existing Cloud architectures do not take full advantage of recent techniques for power management, such as VM live migration, advance reservations, CPU idling, and CPU throttling [NS07].

In the proposed architecture based on ERIDIS (Chapter 4), users submit their reservation requests via a Cloud portal (e.g. a Web server) (Figure 6.5). A reservation request submitted by a user to the portal contains the required number of VMs, its duration and the wished start time. GOC manages an agenda with resource reservations; all reservations are strict, and once approved by GOC, their start times cannot change. This is like a green Service Level Agreement (SLA) where the Cloud provider commits to give access to required resources (VMs) during the entire period that is booked in the agenda. This planning provides a great flexibility to the provider, who can have a better control on how the resources are provisioned.

Following the philosophy of Clouds, the GOC framework delivers resources in a pay-as-you-use manner to the provider: only utilized resources are powered on and consume electricity. The first step to reduce electricity wastage is to shut down physical nodes during idle periods. However, this approach is not trivial as a simple on/off policy can be more energy consuming than doing nothing because powering nodes off and on again consumes electricity and takes time. Hence, GOC uses prediction algorithms to avoid frequent on/off cycles. VM migration is used to consolidate the load into fewer resources, thus allowing the remaining resources to be switch off. VCPU pinning and capping are used as well for workload consolidation. GOC can also consolidate workloads considering time by aggregating resource reservations. When a user submits a reservation request, the GOC suggests alternative start times for the reservation request along with the predicted amount of energy it will consume under the different scenarios. In this way, the user can make an informed choice and favor workload aggregation over the time scale, hence avoiding excessive on/off cycles. On/off algorithms also raise problems for resource management systems, which can interpret a switched-off node as a failure. To address this issue, GOC uses a trusted proxy that ensures the nodes' network presence; the Cloud Resource Management System (RMS) communicates with this proxy instead of the switched-off nodes (Figure 6.5).

The key functionalities of the GOC framework adapted from ERIDIS are to:

- monitor Cloud resources with energy sensors to take efficient management decisions;
- provide energy usage information to users;
- switch off unused resources to save energy;
- use a proxy to ensure network presence of switched-off resources and thus provide interoperability with different RMSs;
- use VM migration to consolidate the load of applications in fewer resources;
- predict the resource usage to ensure responsiveness; and

- give "green" advice to users in order to aggregate resource requests.

The GOC framework works as an overlay atop existing Cloud resource managers, such that it can be used with all types of resource managers without impacting on their workings, such as their scheduling policies. This modular architecture allows for great flexibility and adaptivity to any type of future Cloud RMS architecture. The GOC framework relies on energy sensors (wattmeters) to monitor the electricity consumed by the Cloud resources. These sensors provide direct and accurate assessment of GOC policies, helping it use the appropriate power management solutions.

The GOC architecture, as described in Figure 6.5, comprises:

- a set of energy sensors providing dynamic and precise measurements of power consumption;
- an energy data collector that stores and provides the energy logs through the Cloud Web portal to users (to increase their energy-awareness);
- a trusted proxy for supporting the network presence of switched-off Cloud resources; and
- an energy-aware resource manager and scheduler which applies the green policies and gives green advice to users.

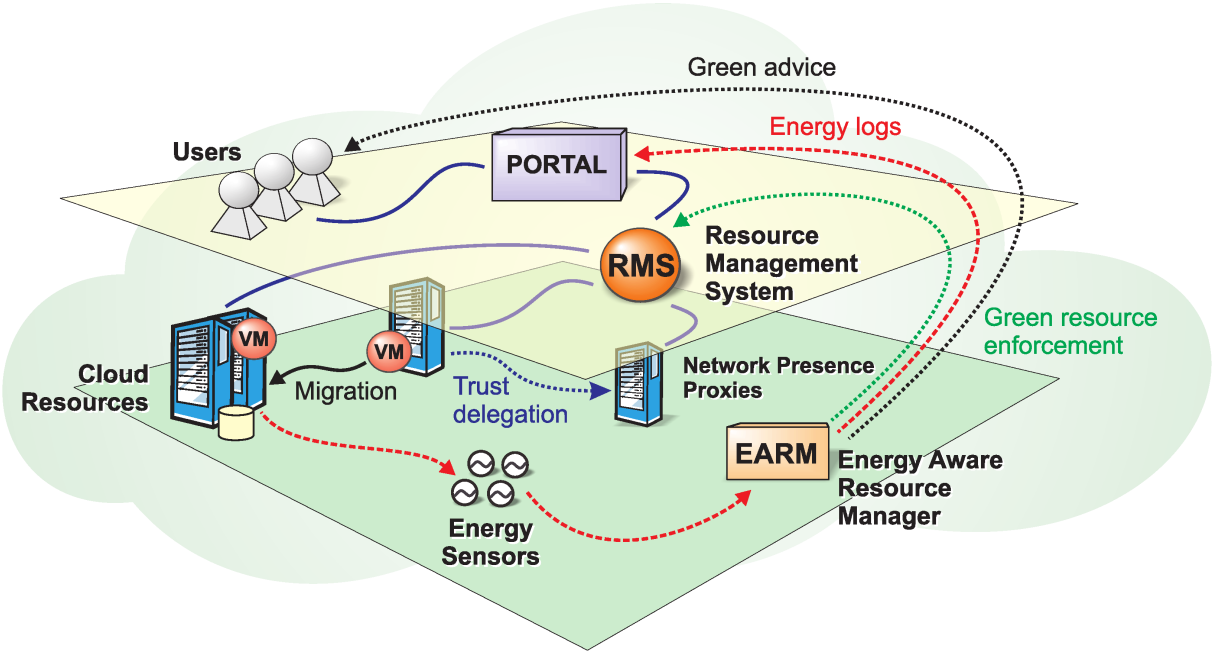

Figure 6.5: The GOC architecture.

The Cloud RMS manages the requests in coordination with the energy-aware resource manager which is permanently linked to the energy sensors. The energy-aware resource manager of GOC can be implemented either as an overlay on the existing Cloud RMS, or as a separate service. Figure 6.6 depicts a scenario where GOC is implemented as an overlay on the existing Cloud RMS. In the resource manager, the beige boxes represent the usual components of a future Cloud RMS (with agenda) whereas the green boxes depict the GOC functionalities. These add-ons are connected to the RMS modules and have access to the data provided by users (i.e. submitted requests) and the data in the reservation agenda.

When a user submits a request, the *admission control* module checks whether it can be admitted into the system by attempting to answer questions such as: Is the user allowed to use this Cloud? Is the request valid? Is the request compliant with the Cloud's usage chart?

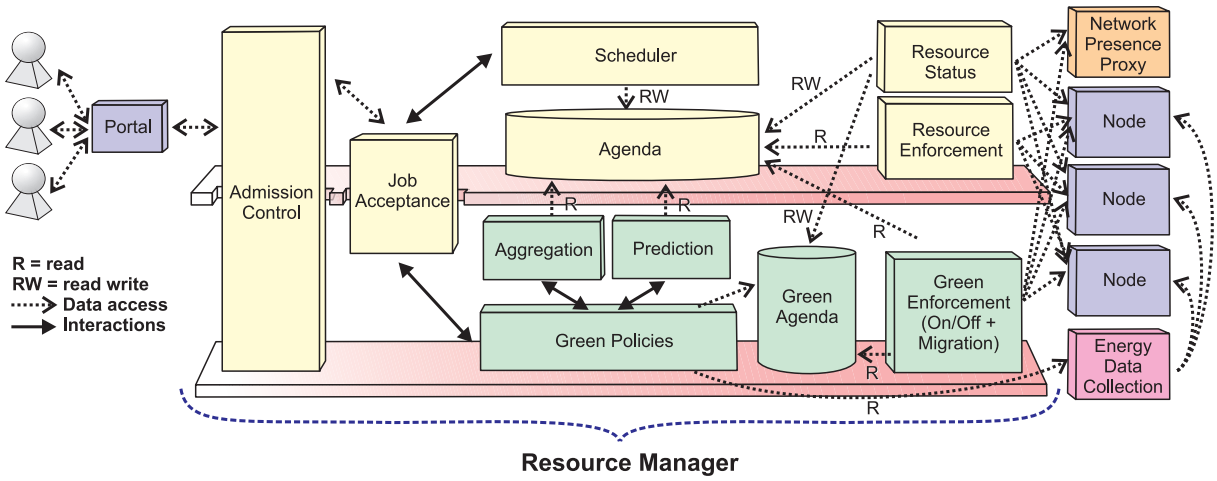

Figure 6.6: The GOC resource manager.

Then, the *job acceptance* module transfers the request to the *scheduler* and to the *green resource manager* (also called energy-aware manager and labeled "*Green Policies*" in Figure 6.6). The *scheduler* queries the *agenda* to see whether the requested reservation can be placed into it whilst the *green resource manager* uses its *aggregation* and *prediction* modules to find other less energy-consuming possibilities to schedule this reservation in accordance with its green policies. All the possible schedules (from the Cloud's *scheduler* and from the *green resource manager*) are then transferred to the *job acceptance* module which presents them to the user and prompts her for a reply.

The *agenda* is a database containing all the future reservations and the recent history (used by the prediction module). The *green agenda* contains all the decisions of the green resource manager: on/off and VM migrations (when to switch on and off the nodes and when to migrate VMs). These decisions are applied by the *green enforcement* module. The *resource enforcement* module manages the reservations (gives the VMs to the users) in accordance with the *agenda*. The *resource status* component periodically polls the nodes to know whether they have hardware failures. If the nodes are off, the component queries the *network presence proxy*, so the nodes are not woken up unnecessarily. The *energy data collector* module monitors the energy usage of the nodes and gives access to this information to the *green resource manager*. This data is also put on the Cloud Web portal, so that users can see their energy impact on the nodes and increase their energy-awareness.

6.3.2 Network presence through proxying

Switched-off nodes do not reply to queries made by the Cloud RMS, which can be interpreted as a resource failure. This issue is solved by using a trusted proxy to ensure the network presence of Cloud resources. When a Cloud node is switched off, all of its basic services (such as ping or heartbeat) are migrated to the proxy, which will then answer on behalf of the node when asked by the resource manager. The key challenge is to ensure the security of the infrastructure and avoid the intrusion of malicious nodes. Our trust-delegation model, described in detail in [DCGG+09], allows for great adaptivity of the GOC framework to any Cloud RMS.

Similar proxy mechanisms are described in [NC10] and [Wer08]. They are essential not to wake up nodes too frequently with messages that cannot even be meant for this node (e.g. broadcast messages such as DHCP requests) [GCN05]. Our proxy should also respond to specific monitoring systems such as Ganglia[4]. Hence, when the node is going to sleep, its proxy starts a

[4]Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids `http://ganglia.sourceforge.net/`.

Ganglia daemon – instead of the sleeping node – to answer to the Ganglia server that provides a monitoring interface for users and administrators. The proxy is able to wake up the node. If the node wakes up by itself (due to a scheduled job for example), it informs the proxy.

One of the main issue with proxying techniques is to deal with IP addressing and routing: the proxy should usurp the IP address of the sleeping nodes and should so be able to receive the IP packets sent to it. This issue can be solved by placing the proxy within the same IP domain as the sleeping node. That is why, we deploy one network presence proxy by cluster (which it is in the same IP domain as all the cluster nodes). Before going to sleep, the node sends a message to its proxy specifying information about its state (for the Ganglia daemon and the other monitoring tools) and its IP address, so the proxy knows it should answer to packets sent to this address.

6.3.3 Prediction algorithms

Similarly to T_s, defined in Section 4.6, we specify temporal parameters based on the energy consumption of the resources to know in what cases it would be more energy efficient to use VM migration. T_m is the bound on the remaining reservation's duration beyond which migration is more energy efficient: if the reservation is still running for more than $T - m$ seconds, the VMs should be migrated, otherwise they stay on their respective physical hosts. As migration takes time, migrating small jobs is not useful (with a duration lower than 20 seconds in the example). This lower-time bound is denoted by T_a. If a migration is allowed, the *green enforcement* module also waits T_a seconds after the beginning of a job before migrating it. It indeed allows initializing the job and the VM, so that the migration is simplified and we avoid the migration of small jobs, crashed jobs or VMs (in case of technical failure or bad configuration of the VM).

For the next reservation, the predicted arrival time is the average of the inter-submission time of the previous jobs plus a feedback. The feedback is computed with the previous predictions and consists of the average of the errors made by computing the n previous predictions (n is an integer). An error is the difference between an observation and a predicted value. For estimating other features of the reservation (size in number of resources and length), the same kind of algorithm is used (average of the previous values at a given time).

We have seen in Section 5.5 that even with a small n (6 for example) we can obtain good results (70% of good predictions on experimental Grid traces). This prediction model is simple, but does not require many memory accesses and is fast to compute; all crucial features of real-time infrastructures. The prediction algorithm has several advantages: it gives acceptable results, it requires a small history (no need to store large amounts of data) and it is fast and responsive to request bursts.

6.3.4 Green policies

Among the components of a Cloud architecture, we have focused so far on virtualization, which appears as the main technology used in these architectures. GOC also uses migration to dynamically unbalance the load between the Cloud nodes in order to shut down unused nodes, and thus save energy.

GOC algorithms can employ other Cloud components, including accounting, pricing, admission control, and scheduling. The role of the green policies is to implement such strong administrator decisions at the Cloud level. For example, the administrator can establish a power budget per day or per user. The *green policies* module will reject any reservation request that exceeds the power budget. This mechanism is similar to a green SLA between user and provider.

Green accounting will give to "green" users (the users who accept to delay their reservation in order to aggregate it with others to save energy) credits that are used to give priority to

their requests when a burst of reservation requests occurs. Business models can also employ this accounting system to encourage users to be energy aware.

6.4 Experimental results

6.4.1 Experimental scenario

This section describes results obtained with a prototype of GOC. Our experimental platform, as discussed earlier, consists of HP Proliant 85 G2 Servers (2.2 GHz, 2 dual core CPUs per node) with XenServer 5.0 [BDF+03a] on each node.

The following experiments aim to illustrate the working of GOC infrastructure as well as to highlight and compare the energy consumption induced by a Cloud infrastructure under different management schemes. Our experimental platform consists of two identical Cloud nodes, one resource manager that is also the scheduler, and an energy data collector. All these machines are connected to an Ethernet router. Hereafter a 'job' is a reservation made by a user to have the resources at the earliest possible time. When a user submits a reservation, she specifies the length in time and the number of required resources. Although this reservation mechanism may look unusual, it is likely to be used by next generation Clouds where frameworks would support these features to help cloud providers avoid over-provisioning resources. Scheduling reservations with limits, such as a time duration, will help Cloud managers to manage their resources in a more energy-efficient way. This is for instance, the case of a user with budget constraints and whose reservations will have a defined time-frame that reflects how much she is willing to pay for using the resources. In addition, this reflects a scenario where service clouds with long-lived applications have their resource allotments adapted to changes in cost conditions (e.g. cost changes in electricity) and scheduled maintenances.

Our job arrival scenario is as follows:

- $t = 10$: 3 jobs of length equals to 120 seconds each and 3 jobs of length 20 seconds each;
- $t = 130$: 1 job of length 180 seconds;
- $t = 310$: 8 jobs of length 60 seconds each;
- $t = 370$: 5 jobs of length 120 seconds each, 3 jobs of length 20 seconds each and 1 job of length 120 seconds, in that order.

The reservations' length is short in order to keep the experiment and the graph representation readable and understandable. Each reservation is for a VM running *cpuburn*. As discussed beforehand, the graphs show only the *cpuburn* burn phase in order to reduce their length and improve their readability (VMs are booted before the experiment start and halted once the experiment completes).

These experiments, although in a small scale for clarity sake, represent the least favorable case in terms of energy consumption as *cpuburn* fully uses the CPU; one of the most energy consuming component of physical nodes as seen in Chapter 2. Moreover, *cpuburn* jobs are accurately visible on the power consumption curves, with clear steps for each added VM (as previously shown in Figure 6.2). Yet, we are aware that real applications do not have exactly the same behavior. In particular, they are more affected by their physical location (if two computing tasks that should exchange results are on the same physical node, their completion is faster) and they are more sensible to live migration (impact on the execution time).

On our testbed, each node can host up to seven VMs, identical in terms of memory and CPU configuration, and with the *debian etch* distribution. As our infrastructure does not depend on any particular resource manager, we do not change the scheduling of the reservations and the assignment of VMs to physical machines. The only exception is when a physical node is off,

when we then attribute its jobs to the awoken nodes if they can afford it; otherwise we switch it on.

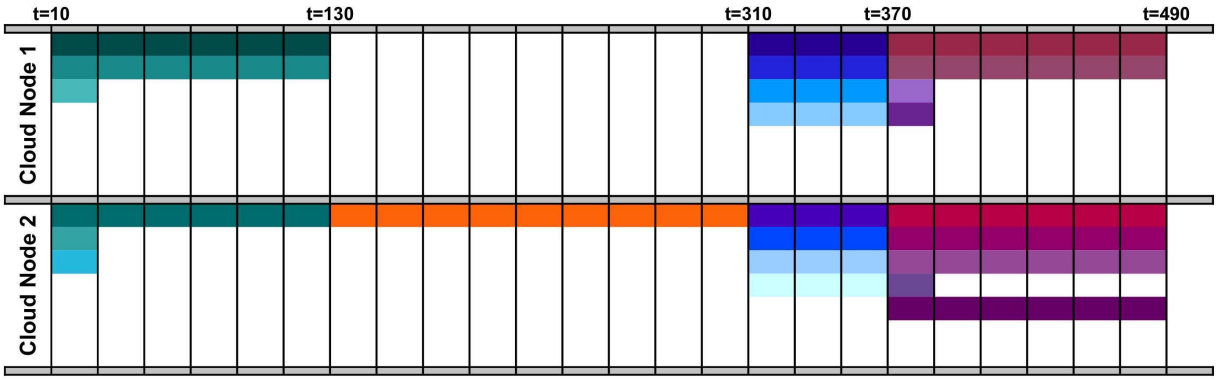

Figure 6.7: Gantt chart for the round-robin scheduling.

In order to validate our framework and to prove that it saves energy regardless the underlying scheduler, we have studied two different scheduling techniques:

- *round-robin*: a typical distributed-system scheduling algorithm where the first job is assigned to the first Cloud node, the second job to the second node, and so on. When all the nodes are idle, the scheduler changes their order (we do not always attribute the first job in the queue to the first node). The behavior of this scheduling technique with the previously defined job arrival scenario is shown in Figure 6.7.
- *unbalanced*: algorithm broadly used for load consolidation, in which the scheduler puts as many jobs as possible on the first Cloud node and, if there are still jobs left in the queue, it uses the second node, and so on (as before, when all the nodes are idle, we change the order to balance the roles). We can see this scheduling with the previously defined job arrival scenario in Figure 6.8.

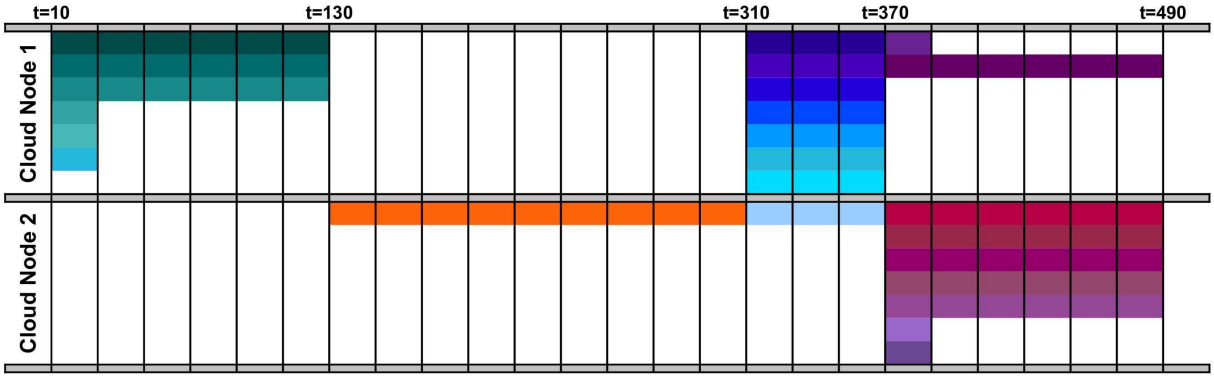

Figure 6.8: Gantt chart for the unbalanced scheduling.

These two scheduling algorithms are well-known and widely used in large-scale distributed system schedulers. For each of these algorithms, we use four scenarios to compare our VM management scheme against a basic one. These four scenarios are as follows:

- *basic*: no changes are made, this scenario represents the Cloud with no power management.
- *balancing*: migration is used to balance the load between the Cloud nodes. This scenario presents the case of a typical load balancer which aims to limit node failures and heat production.

- *on/off*: unused nodes are switched off. This is the scenario with a basic power management;
- *green*: we switch off the unused nodes and we use migration to unbalance the load between Cloud nodes. This allows us to aggregate the load on some nodes and switch off the other ones. This is the scenario that corresponds to GOC.

6.4.2 Results

For each scheduling technique, we have run the four scenarios, one at a time, and logged the energy consumed by the experimental testbed [LO10].

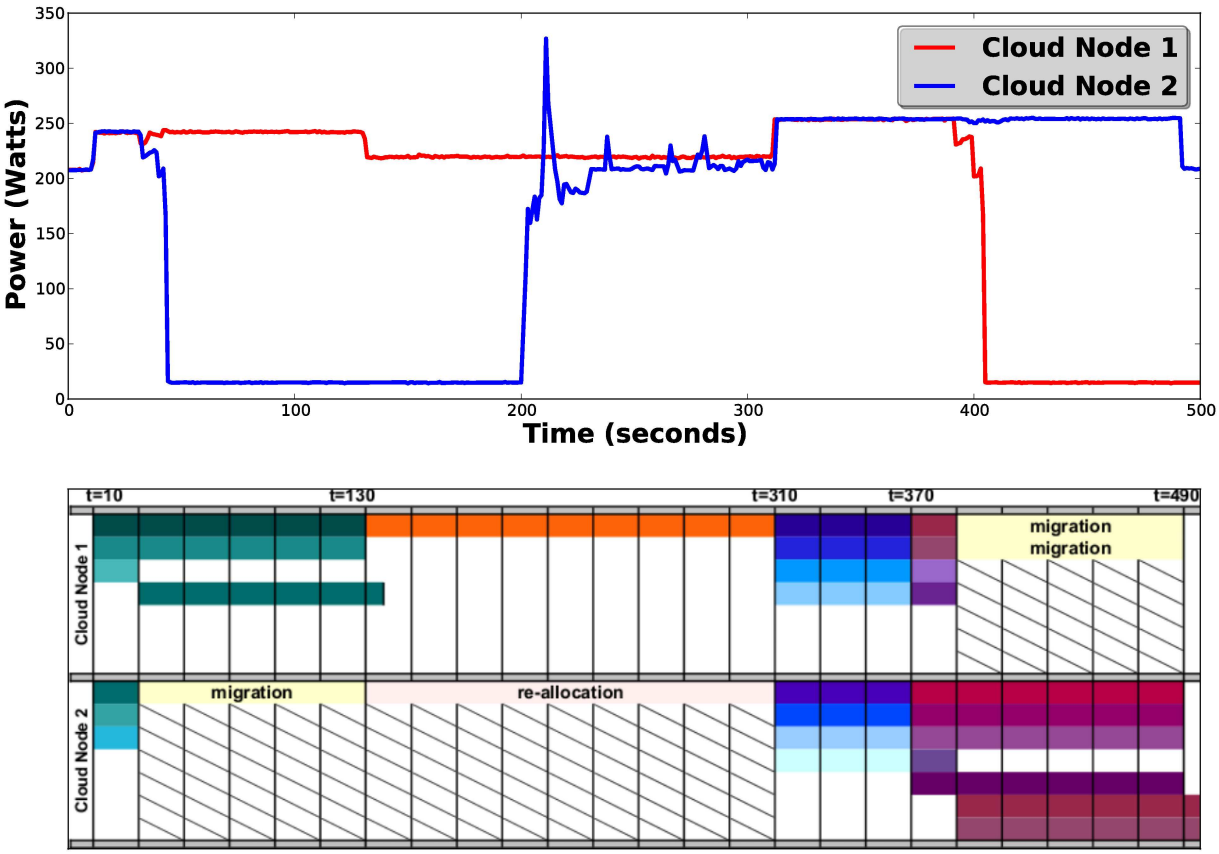

Figure 6.9: Green scenario with round-robin scheduling.

Figure 6.9 shows the course of the experiment for the two nodes with the round-robin scheduling applied to the green scenario. The upper part of the figure shows the energy consumption of the two nodes while the lower part presents the Gantt chart with reservations (time is in seconds). At time $t = 30$, the second job (first VM on Cloud node 2) is migrated to free Cloud node 1 to then switch node 2 off. This migration does not occur before $T_a = 20$ seconds. Small power peaks are noticeable during the migration, which confirms the results of Section 6.2 stating that migration is not really costly if it does take too long.

This migration leads to the re-allocation of the job starting at $t = 130$ on Cloud node 1 since Cloud node 2 has been switched off and Cloud node 1 is available. At $t = 200$, Cloud node 2 is booted to be ready to receive the jobs starting at $t = 310$. During the boot, an impressive peak consumption occurs. It corresponds to the physical start of all the fans and the initialization of all the node components. Yet, this peak is more than compensated by the halting of the node ($P_{OFF} = 20$ Watts) during the time period just before the boot since the node inactivity time was greater than T_s (defined in Section 6.3.3).

During the seventh job, one can notice that a VM running *cpuburn* consumes about 10 Watts which represents about 5% of the idle consumption of the Cloud node. At time $t = 390$, two new VM migrations are performed, and they draw small peaks on the power consumption curves. From that time, Cloud node 1 is switched off and Cloud node 2 has 6 running VMs. We observe that the fifth and the sixth VMs cost nothing (no additional energy consumption compared to the period with only four VMs) as explained in Section 6.2 as this Cloud node has four cores. This experiment shows that GOC greatly takes advantage of the idle periods by using its green enforcement solutions: VM migration and shut down. Job re-allocation can avoid on/off cycles. Significant amounts of energy are saved.

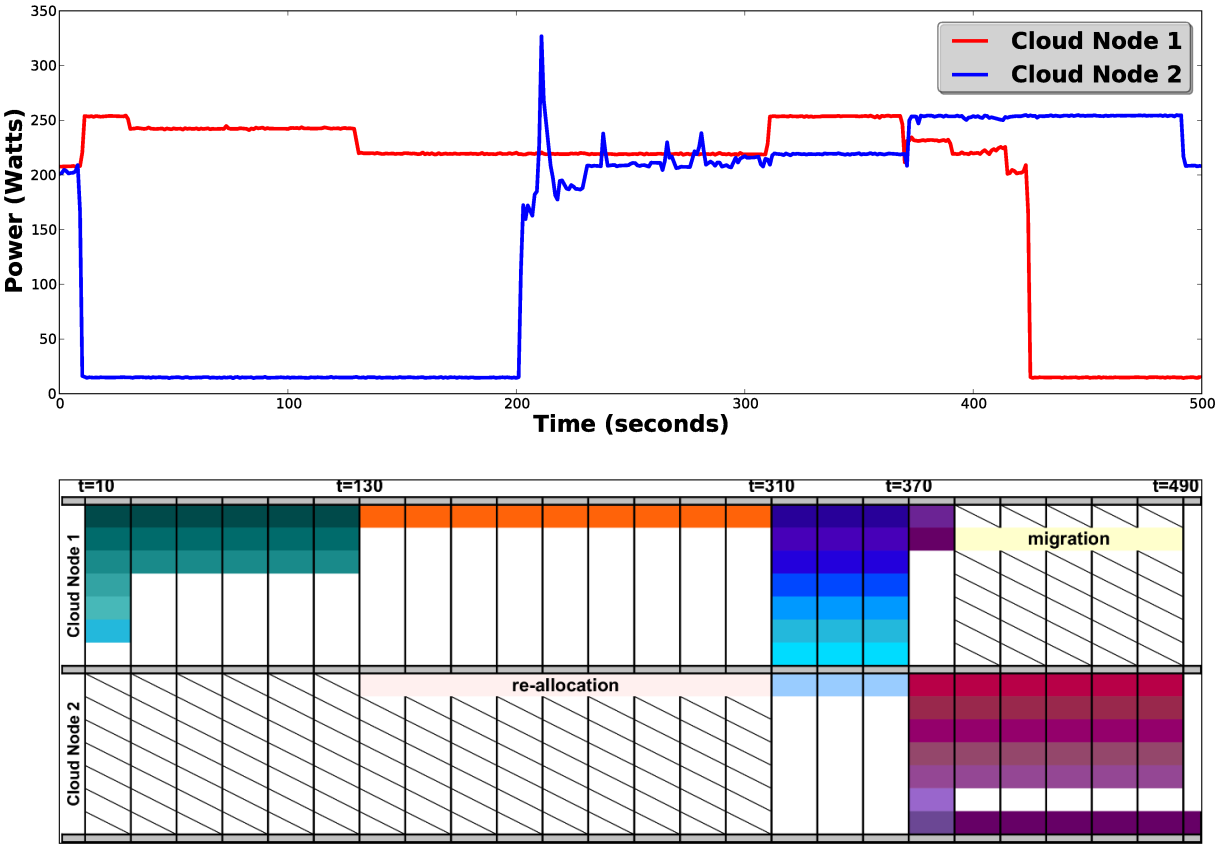

Figure 6.10: Green scenario with unbalanced scheduling.

The green scenario with unbalanced scheduling is presented in Figure 6.10. For the green scenario, the unbalanced scheduling saves more energy, since two migrations less are needed compared to the round-robin scheduling, and all the first burst jobs are allocated to Cloud node 1 allowing Cloud node 2 to stay off. In fact, this is the general case for all the scenarios (Figure 6.11); the unbalanced scheduling is more energy-efficient since it naturally achieves a better consolidation on fewer nodes.

Summarized results of all the experiments are shown in Figure 6.11. As expected, the green scenario, which illustrates GOC behavior, consumes less energy under both scheduling scenarios. The balancing scenario is more energy consuming than the basic scheme for unbalanced scheduling; a phenomenon caused by numerous migrations for the balancing scenario that cancels out the benefits of these migrations.

The noticeable figure is that the green scenario with unbalanced scheduling consumes 25% less electricity than the basic scenario representing the current Cloud administration. This figure obviously depends on the Cloud usage. However, it shows that great energy savings are achievable with minor impacts on the utilization: small delays occur due to migrations (less than 5 seconds for these experiments) and to physical node boots (2 minutes for our

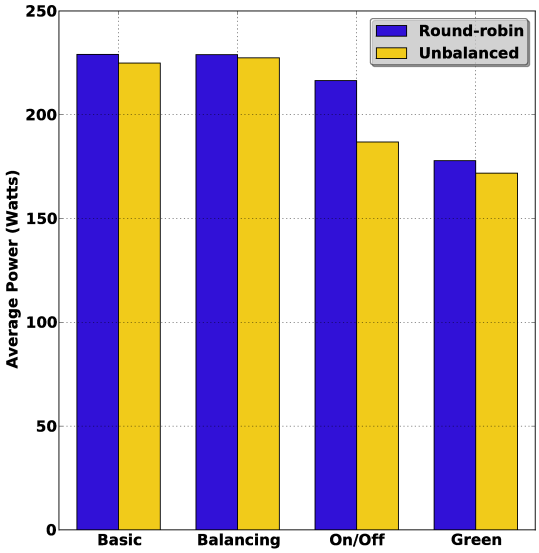

Figure 6.11: Comparison results.

experimental platform nodes). These delays can be improved by using better VM live-migration techniques [HNO+09], and faster booting schemes (like suspend to disk and suspend to RAM).

6.5 Conclusion

As Clouds are more and more broadly used, the Cloud computing ecosystem becomes a key challenge with strong repercussions. It is urgent to analyze and to encompass all the stakeholders of a Cloud's energy usage to design adapted framework solutions. It is also essential to increase the awareness of users in order to reduce their CO_2 footprint caused by their use of Cloud infrastructures. This matter has led us to measure and to study the electrical cost of basic operations concerning the VM management in Clouds, such as the boot, the halt, the inactivity and the launching of a sample *cpuburn* application. These observations show that VMs do not consume energy when they are idle and that booting and halting VMs produce small power peaks, but are not really costly in terms of time and energy.

As Clouds are on the way to becoming an essential element of future Internet, new technologies have emerged for increasing their capacity to provide on-demand XaaS (everything as a service) resources in a pay-as-you-use fashion. Among these new technologies, live migration seems to be a really promising approach allowing on-demand resource provisioning and consolidation. We have studied the contribution that this technique could constitute in terms of energy efficiency. VM live migration – whose performance can be improved – is reliable, fast, and requires little energy compared to the amount it can save.

These first measurements have allowed us to propose an original software framework: the Green Open Cloud (GOC) which aims at controlling and optimizing the energy consumed by the overall Cloud infrastructure. This energy management, based on the ERIDIS infrastructure, uses different techniques:

- energy monitoring of the Cloud resources with energy sensor for taking efficient management decisions;
- displaying the energy logs on the Cloud portal to increase the energy-awareness;
- switching off unused resources to save energy;
- utilization of a proxy to carry out network presence of switched-off resources and thus provide interoperability with any kind of RMS;

- use of VM migration to consolidate the load in fewer resources;
- prediction of the next resource usage to ensure responsiveness; and
- giving green advice to users in order to aggregate resources' reservations.

The GOC framework embeds features that, in our opinion, will be part of the next-generation Clouds, such as advance reservations, which enable a more flexible and planned resource management; and VM live-migration which allows for great workload consolidation.

Furthermore, the GOC framework was the subject of an experimental validation made with two different scheduling algorithms to prove GOC's adaptivity to any kind of Resource Management System (RMS). Four scenarios were studied to compare the GOC behavior with the basic behavior of Cloud RMS's. Energy measurements on these scenarios show that GOC can save up to 25% of the energy consumed by a basic Cloud resource management; savings that are also reflected in the operational costs of the Cloud infrastructures. However, the validation results were obtained on a small experimental platform using *cpuburn* applications. We plan to conduct experiments on a larger scale with representative applications of Cloud environments.

This work shows that important energy savings and thus CO_2 footprint reductions are achievable in future Clouds at the cost of minor performance degradation. There is still room for improvement to increase the users' energy-awareness, the main leverage to trigger a real involvement from the Cloud providers and designers to reduce the electricity demand of Clouds and contribute to a more sustainable ICT industry.

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing

7

High-level Energy-aware Reservation Model for End-to-end networkS

7.1 Introduction

Scientists are increasingly relying on networks for high-speed data transfers, results dissemination and collaborations. Networks are thus becoming the critical component of ICT infrastructures. In 2007, to distribute the entire collection of the Hubble telescope's data (about 120 terabytes of data) to various research institutions, scientists chose to copy this data on hard disks and send them via mail, because it was faster than using a network [Far07]. To circumvent similar issues, dedicated networks are built to transfer large amounts of scientific data – the case of the LHC (Large Hadron Collider), which produces 15 million gigabytes of data every year [LHC].

With network infrastructure accounting for a considerable share of the electricity consumed by current data centers, reducing the energy consumption of wired networks has become a key concern for equipment manufacturers and providers. When designing networks, energy efficiency should be considered alongside important goals such as scalability, reliability, response time, low overhead, interoperability and easiness of use. Moreover, the design of power management policies should be adapted to the network characteristics, including the topology, traffic and usage scenario (e.g. peer-to-peer, web servers, VoIP).

This chapter focuses on improving the energy efficiency of dedicated networks – such as those deployed in data centers, enterprises, across banks [SBA+07], and research networks (e.g. UltraScience Net[1]). Unlike the Internet, these networks present controlled traffic conditions and less intricate topologies. In addition, most traffic is concentrated on a few links [SBA+07, BAM10] and consists of relatively large data transfers such as bulk transfers, backup operations and file transfers [PAB+05]. As these networks often aim at transporting large data volumes at high-speed rates, one can plan and schedule these transfers in more energy-efficient ways using a reservation system [OL11a].

By adapting ERIDIS, we propose a High-level Energy-aware Reservation Model for End-to-end networkS called HERMES. Towards reducing energy consumption, we first design an Energy Consumption mOdel For End-to-end Networks (ECOFEN) that is used to compute the energy estimates required by the scheduling algorithm of ERIDIS. The architecture and functionalities of HERMES are adapted from ERIDIS.

For HERMES, the resource management of ERIDIS is adapted to the particular context of networks. While ERIDIS' management was hierarchical, HERMES' is distributed because, in contrast to data centers, Grids and Clouds, often there is no managing node in wired networks. However, one can imagine such a management system for future dedicated networks. Hence, to know which solution is the best among the centralized and the decentralized approaches, we evaluate both for different networks.

This chapter is organized as follows. Section 7.2 describes the energy consumption model.

[1] `http://www.csm.ornl.gov/ultranet/`

The network model used to describe reservation-based dedicated networks is introduced in Section 7.3. Section 7.4 presents the overall architecture of HERMES. The reservation process is described in 7.5, while the resource management is detailed in 7.6. A simulation-based evaluation of HERMES is proposed in Section 7.7. The centralized and decentralized approaches are compared in Section 7.8. Finally, we conclude in Section 7.9.

7.2 ECOFEN: Energy Consumption mOdel For End-to-end Networks

As seen on Chapter 2, several energy models have been proposed to express the electrical consumption of specific network devices (e.g. routers [WPM02] and switches [AK08, HCP09]) and sections of networks (e.g. backbones [CMN09], optical networks [Tuc10a]). Taking into account these models and energy values from literature, we design an Energy Consumption mOdel For End-to-end Networks, called ECOFEN. It estimates the energy consumption of end-to-end transfers (taking into account all the crossed devices between the source and the destination). Afterwards, this model is used to compute the estimates required by the scheduling algorithm (Section 4.5). So, it should be able to evaluate the energy consumption for a given transfer at different dates and using different paths.

Our model allows for the use of energy saving techniques such as on/off algorithms that switch off unused links and ports; and Adaptive Link Rate (ALR) [GCS06], which can adapt the link rate to the actual traffic.

The framework presented here can be applied to any network topology embedding various types of devices and supporting any traffic. Contrary to other approaches, the proposed framework determines the consumption of end-to-end communications while accounting for background traffic.

7.2.1 Energy consumption per network device

This model considers the consumption of devices that are connected to a power socket, thus excluding the links. The energy "cost" of links is factored in the consumption of the devices they connect. For a router with energy saving features, the power used for transferring data depends on the length of the link, where long links require more power. As of writing, however, varying the power used for transferring data was not a feature enabled by default in routers; they often use the maximum power for all ports.

For a device (e.g. router, repeater or Ethernet card), the energy consumed by a transfer is given by:

$$E = E_{boot} + E_{work} + E_{halt} \tag{7.1}$$

where E_{boot} and E_{halt} can be zero if there is no need to boot and halt the device (e.g. due to transfer aggregation or if the device is always on). Manufacturers and designers can influence the energy consumed during boot and halt by working on hardware components to shorten these periods. Moreover, protocol designers also have a role to play in the quest for energy efficiency. For example, the protocol used to re-synchronize linked nodes must be optimized to make the on/off technique usable [GCS06, SRH05]. In spite of these factors, optimizing E_{boot} and E_{halt} is out of the scope of this thesis.

Every device has a fixed energy cost (E_{idle}) that reflects its idle energy consumption (i.e. when no transfer is taking place), and a variable cost that depends on the traffic. The energy E_{work} consumed during a transfer comprises these two costs and depends on:

- BD the used bandwidth,
- L the duration of the transfer,

- the cross traffic on the device[2],
- the device type (router, NIC, ...).

The fixed cost (E_{idle}) depends only on the device type, whereas the first three items of the above list vary with time [MSBR09a]. The interplay between network usage and energy consumption is explored by several power-cost models. Classical models do not consider rate adaptation and thus present an almost constant consumption (the line labeled *Fully powered On*). Yet, if we allow rate adaptation to the load, the corresponding consumption model is linear by steps as shown by the line labeled *Adaptive Link Rate* in Figure 7.1[3].

Another classical model, but less realistic for current devices, is the proportional power cost. Some researchers advocate that efforts should be made to reach this model [BH07].

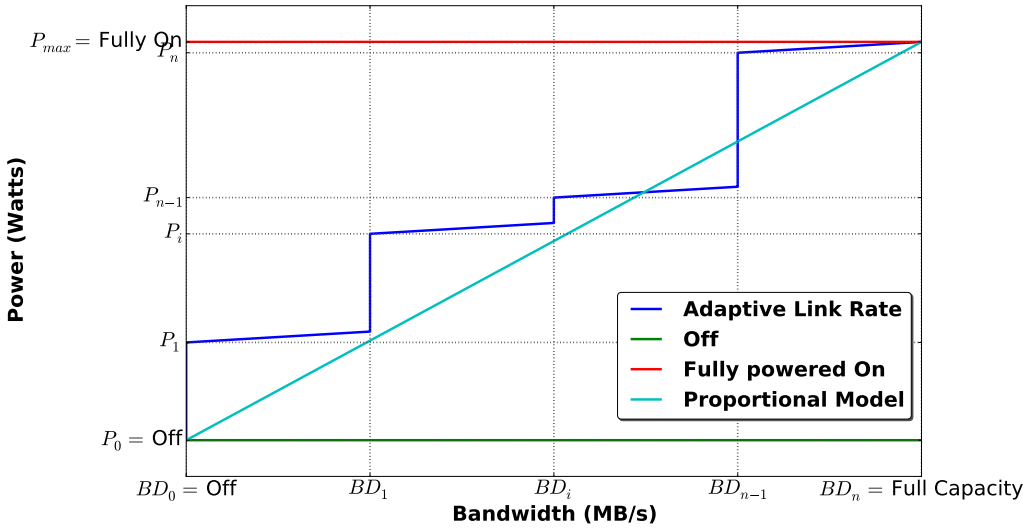

Figure 7.1: Models of power cost as a function of bandwidth on a router port during a transfer.

For a router, we also consider two distinct consumptions:

- a variable consumption incurred by the ports during transfers;
- a fixed consumption that corresponds to the energy required to run the router even when it is idle.

Figure 7.1 represents the theoretical power consumption of a port during a data transfer. *Adaptive Link Rate* (ALR) shows how the port power consumption is influenced by the transmission rate. Currently, ALR permits only a few discrete link rates. For example, for a 10 Gb/s port, the possible link rates are 10 Mb/s, 100 Mb/s, 1 Gb/s and 10 Gb/s. These rates, represented by BD_i on the x-axis of Figure 7.1, are discrete values selected to support the current traffic. Hence, when a port is transmitting at 20 Mb/s, it consumes the energy to operate at 100 Mb/s (i.e. 10 Mb/s $<$ 20 Mb/s $<$ 100 Mb/s) and that resulting from the actual traffic (with a linear dependence), as represented in Figure 7.1.

Hereafter, our model considers that the power function P_{work} of ports is a piecewise affine function as presented by *Adaptive Link Rate* in Figure 7.1. This power function presents a strong dependence on the link rate and a small dependence on the traffic. Using the notations

[2]We avoid counting the consumption of a router several times for multiple simultaneous transfers. The cost of each transfer crossing a router comprises a part of the static consumption of the router. This part is computing depending on the number of transfers and the bandwidth used by each transfer. For example, in the case of two transfers with the same bandwidth, each of them is considered to consume 50% of the static consumption of the router.

[3]This graph does not show the energy and the time required to alternate between link rates.

of Figure 7.1, the power function of a port P_{work} can be written as a function of the bandwidth BD:

$$P_{work} = \begin{cases} P_0 & \text{if } BD = 0 \\ \alpha_1 BD + P_1 & \text{if } BD \in]0; BD_1] \\ \vdots & \\ \alpha_i BD + (P_i - \alpha_i BD_{i-1}) & \text{if } BD \in]BD_{i-1}; BD_i] \\ \vdots & \\ \alpha_n BD + (P_n - \alpha_n BD_{n-1}) & \text{if } BD \in]BD_{n-1}; BD_n] \end{cases} \tag{7.2}$$

where α_i are the slopes ($\alpha_i > 0$) of the different linear portions (power levels) delimited by the BD_i (the different transmission rate levels) and P_i defines the start power of each different level.

7.2.2 Energy consumption per network transfer

Based on the energy model per device, the total energy used by a transfer from Node A to Node B in the example of Figure 7.2 through Router 1 and 2, if there is no cross traffic, is:

$$\begin{aligned} E_{transfer} &= E_{EthernetCard}(NodeA, BD, L) + E_{Router}(Router1) \\ &+ E_{Port}(In, Router1, BD, L) + E_{Port}(Out, Router1, BD, L) \\ &+ E_{Port}(In, Router2, BD, L) + E_{Port}(Out, Router2, BD, L) \\ &+ E_{Router}(Router2) + E_{EthernetCard}(NodeB, BD, L) \end{aligned} \tag{7.3}$$

The functions E_{router} and E_{port} are different for each router. The energy consumed by an Ethernet card ($E_{EthernetCard}$) depends on:

- its model (capacity, manufacturer, hosting node, etc.), representing a fixed cost;
- BD the bandwidth used by the transfer;
- and L the duration of the transfer.

In the same way, the energy consumed by a router (E_{router}) for a given transfer depends only on the router type (size, manufacturer). Although it is a fixed cost, if multiple transfers are using the same routers (cross traffic for example), this cost is split among the transfers according to their duration. The energy consumed by a router port during a transfer (E_{port}) depends on:

- the router's model;
- if the traffic is coming in or out;
- BD the bandwidth used by the transfer;
- and L the length in time of the transfer.

7.2.3 Model calibration

The ECOFEN model gives the energy consumption of a given network (topology, type of devices, routing protocol) for a given traffic (bandwidth utilization). It is a generic model that can be used for any kind of wired networks and traffic, and with energy saving techniques such as on/off policies and rate adaptation.

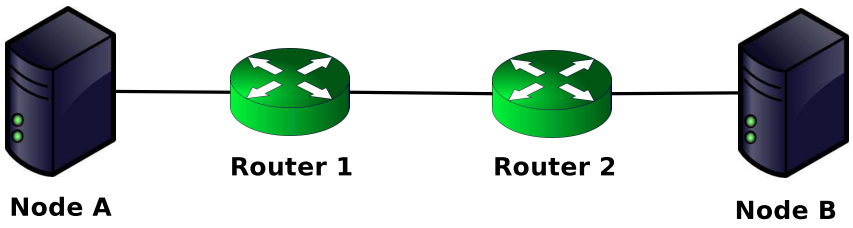

Figure 7.2: Scenario example.

ECOFEN assumes that the power function of every device (P_{work}) is known. Hence, α_i, BD_i and P_i are known for every device whose power consumption depends on the bandwidth. The fixed costs are also known for all devices.

With this information, the overall energy consumption per device and per data transfer can be computed. This requires a preliminary calibration phase with wattmeters for each type of device in order to obtain their power profiles (power functions depending on the usage like in the examples of Figure 7.1).

For example, in [MSBR09b], the authors provide power consumption values for a rack switch, where the power consumed by the chassis is 146 Watts and a port consumes 0.12 Watts, 0.18 Watts and 0.87 Watts when working at 10 Mbps, 100 Mbps and 1 Gbps, respectively. Hence, the dynamic power is really small compared to the static power. Typically, it represents between 3 and 25% depending on the device [MSBR09a].

In the future, the information described above should be included in the specifications of every network device. A simulator based on this model has been implemented, it is presented in Appendix A. For calibrating the consumption of different router types, we have used the values provided in [MSBR09a, MSBR09b, Idz09, Cis, CSB+08, BAH+09].

7.3 Network model: reservations on dedicated networks

In dedicated networks, the coordination among the network elements induced by the reservation process can be used to achieve common energy- and performance-driven goals, rather than selfish per-user goals.

The targeted applications [Bur05] include large-scale distributed applications, data-intensive peer-to-peer computing, cloud computing with virtual machine and data migrations, data-intensive applications (e.g., the CERN-LHC experiments), media-streaming services [HHX+05], etc. Advance reservations are especially useful for applications that require strong quality-of-service (QoS), as the case of Content Delivery Networks (CDN) [BCMR04]. The respect of QoS requirements demands a tight coordination among network elements.

In a first step, we will assume that we have a dedicated network where we can control and implement management features in each router and network device. Interoperability issues will be discussed later.

7.3.1 Reservation model

Some basic notions should be defined in order to understand the model.

Reservation: the user (end host) submits a data-transfer reservation to transfer a data volume (10 GB for example) between a source and a destination before a deadline (for example, "in two hours"). This is the only information required for simple data-transfer requests. These transfers are **malleable**, they are flexible enough to use any transmission rate, to have variable transmission rates over time, or to be split into several parts. We call a **path** a set of undirected links that connect a source and a destination.

Additional features can be specified, such as maximal and minimal bandwidths (e.g., for video streaming or if the receiver is limited by its storage capacities), transfer profiles (step functions that express variable bandwidth requirements over time). These transfers are called **rigid** in contrast to malleable transfers.

Here, reservations have the same meaning as in the ERIDIS description (Chapter 4). Yet, important differences occur between provisioning computing and networking resources. First, computing resources are independent while networking resources are not.[4] This dependence must be taken into account and a path should be selected for each request. Consequently, networking resources are not interchangeable.

Agenda: each network device (router, switch, bridge, repeater, hub, transmitter) has two agendas per port (per outgoing link) for both traffic directions (in and out). An agenda stores all the future reservations concerning its one-way link; an information that is sometimes called the book-ahead interval [Bur05]. This definition is similar to the definition given for ERIDIS in Chapter 4.

Furthermore, each network device also has an agenda stating the on and off periods and the switching stages between on and off. This global agenda is in fact the combination of all the device's per-port agendas: when no port is used for a certain amount of time (not too small), the network device can be switched off. Usage-prediction algorithms are used to avoid switching the device off if it is going to be used in near future. These prediction algorithms are described later.

Rather than using discrete time with fixed time slots during which the resource allocations are similar [Bur05, RRX09], the model uses continuous time. Indeed, as explained in [JLRS08], the storage of agendas is more flexible and less space-consuming with a continuous time model. Hence, we store the per-port agendas using a time-bandwidth list structure used in previous works [SRR+07, LWRZ08, LW08]. Each port maintains its reservation status using a **time-bandwidth list** (TB list) formed by $(t[i], b[i])$ tuples, where $t[i]$ is a time and $b[i]$ is a bandwidth. In these tuples, sorted in increasing order of $t[i]$, $b[i]$ denotes the available bandwidth of the concerned port during the time period $[t[i], t[i+1]]$. If $(t[i], b[i])$ is the last tuple, then it means that a bandwidth of $b[i]$ is available from $t[i]$ to ∞. Each $t[i]$ is called an **event** in the agenda.

Figure 4.5 presents a typical sequence of events for the reservation process:

1. a user submits a reservation request (specifying at least the data volume and the required deadline) to the network-management system (which will be detailed later);

2. the advance-reservation environment launches the negotiation phase including admission control, reservation scheduling and optimization policies;

3. the notification is sent to the user when his request is accepted or rejected, and when it is scheduled;

4. the reservation starts at the scheduled start time and ends at the scheduled end time, which occurs before the user-submitted deadline.

The network devices crossed during a transfer from node A to node B are called the **path** from A to B (set of vertices). During the reservation, the whole path and network devices required by the transferred data are reserved, but not necessarily in full (some bandwidth might still be available over some links along the path). This mechanism guarantees a high level of QoS and ensures that the deadline is respected. This mechanism assumes that the network elements are at least roughly synchronized.

[4]The resources make a path between a source and a destination, so links and nodes are not independent as servers are in Grid and Cloud environments.

7.3.2 Protocol model

Data transfers with advance bandwidth reservations require a transport protocol adapted to their specific network characteristics, which include high bandwidth, often low latency, and no congestion (not overloaded). As such, TCP is neither appropriate nor efficient enough, mainly because of its slow-start and congestion-window mechanisms. To increase the energy efficiency, the used protocol should be: reliable, fast, generate little overhead, be able to work with fixed bandwidth, and require few ACKs. Protocols that are more aggressive than TCP, such as XCP [KHR02] (eXplicit Control Protocol), can be used, and so can protocols relying on UDP for data transfer and TCP for control.

We assume that the used routing protocol is symmetric, which means that the path used from a node A to a node B and from B back to the node A are symmetric. This can be done by using, for example, the explicit routing functionality of MPLS [RVC01].

7.3.3 Router model

For the sake of clarity, every network device is called a router, apart from the end-host's Ethernet cards and the links. Firewalls, transmitters, and switches are considered as specific routers with different functionalities. This language simplification, made to ease the discussions, does not compromise the generality of the model. The global network thus comprises three categories of components – routers, links and end-hosts – where each category has its own energy-consumption function depending on its parameters, as stated in Section 7.2. Each router also stores its energy-cost function depending on its characteristics and utilization (i.e., its agendas).

As mentioned earlier, each router stores an agenda for each of its ports, and requires extended memory and computing capacities to maintain these agendas up-to-date. The reservations are thus stored in a decentralized manner in each concerned router. Routers should also have the capacity to use energy-efficient techniques such as ALR (Adaptive Link Rate) [BCN06] to adapt the transmission rate to the usage, and thus decrease the energy consumption when they are not working at full capacity.

7.3.4 Problem formulation

Up to now, in this chapter, we have proposed: 1) an end-to-end energy-cost model for general networks; 2) a network model for bulk data transfers with advance bandwidth reservations.

The scenario we are considering here is the following: a large number of files have to be transferred from multiple senders to multiple receivers. Each transmission corresponds to a single file and a single pair of nodes (sender-receiver), which are called end-users. Each end-user is directly connected to a *gateway* (also called bandwidth broker in such networks [ZDH01, Bur05] under centralized management). End-users submit reservation requests to gateways in order to reserve bandwidth for a given time to transfer a file to their receivers. Only the data transfer service provider knows the network, its topology and its state (traffic, energy consumption, on- or off-state). For a given transfer, we call the *sender's gateway* the gateway to which the sender is linked and the *receiver's gateway*, the gateway of the transfer receiver.

In such a context, the objective is to find a good trade-off between the considered network's performance (number of granted reservations) and energy consumption.

7.4 HERMES architecture

To improve availability, networks are often over-provisioned, which typically leads to a very low overall utilization [CGNG04]. Exploring periods of network under-utilization could result in significant energy savings, and with that goal, we have proposed the High-level Energy-awaRe Model for bandwidth reservation in End-to-end networks (HERMES). HERMES ensures

energy-efficiency by scheduling advance bandwidth reservations using request aggregation, on/off mechanisms and usage predictions.

Although organizations can generally feel compelled to reduce their energy consumption, they might be reluctant to follow approaches that increase the burden of managing the network infrastructure. Overlay networks are an attractive solution as they provide effective and reliable services, dealing with complexity without exposing it to end-users [DBB09]. HERMES can be used as an overlay atop existing networks, hiding the complexity of energy-efficient resource management from users.

Based on ERIDIS, HERMES combines several techniques:

- putting unused network components in sleep mode;
- energy optimization of the reservation's scheduling by aggregating reservations;
- minimizing the number of control messages required by the infrastructure;
- usage of Disruptive Tolerant Network (DTN) to manage the infrastructure;
- network-usage prediction to avoid too-frequent on/off cycles.

The following section will describe the usage of these techniques and how they are combined to form the proposed energy-efficient solution for global network-management. We assume that the routing protocol determines only one path between each pair of network nodes (the routing is unique and symmetric).

7.5 Reservation process

Assuming that the complete energy-cost functions are available for all the network nodes, the key problem is to use this information to schedule the bulk data transfers in an energy-efficient way.

On the basic scenario of Figure 7.3, node A wants to send data to node B. The user submits a reservation, with a minimum data volume V to be transferred and a deadline d, to a gateway (or bandwidth broker).

We use the ERIDIS scheduling algorithm (Algorithm 4.2) to aggregate as many reservations as possible, reducing the number of switching on and off and thus saving energy. To take an energy-efficient placement decision, the scheduler should know the agenda of all the network devices along the path (routers and ports) between A and B[5]. After the agenda's collection and merge, the **availability agenda**, which contains all the residual bandwidths of the path, is obtained. It is stored as a normal agenda with a time-bandwidth list.

Then, the reservation is scheduled at the least energy-consuming place. A **place** is a time period during which the reservation can be put without conflicting with others and without passing the deadline. As a reservation should be aggregated with others whenever possible, each event is tested to serve as the reservation start time or end time.

The energy consumption of each scheduling solution is computed using ECOFEN, employing the energy-cost function for each network component along the path (routers, Ethernet cards, ports, etc.) as explained in Section 7.2. The consumption also includes the cost to switch on and off the required network resources (ports, routers, etc.). It does not consider cases where placing a reservation at a given place in the agenda delays the switching off or moves the switching on forward.

When finding a place for a reservation, the algorithm strives to use as much bandwidth as possible to minimize the length of the reservation and hence the energy it consumes. This

[5]The following subsections explain how this collection is done.

algorithm has a worst-case complexity of $O(n^2)$ where n is the number of events in the availability agenda of the path.

In the scenario of Figure 7.3, no transmission from A to B can use more than 1 Gb/s because it is the maximum residual bandwidth available from A to B. Moreover, if a reservation R using 9.5 Gb/s of bandwidth has already been made between other end-users (not presented in the figure), another reservation from A to B can use only 500 Mb/s while reservation R takes place. In fact, it will use only 490 Mb/s because a **free bandwidth portion** – which can be either a fixed amount of bandwidth or a fraction of the link's capacity – is always kept on each link for transferring management messages and ACKs.

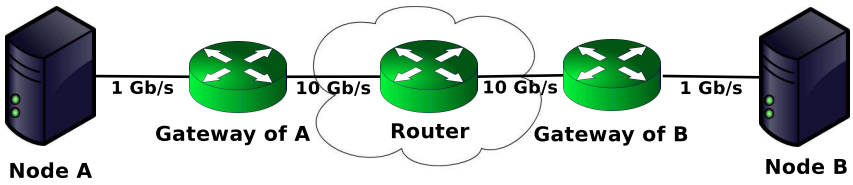

Figure 7.3: Simplified network example.

7.6 Request and resource management

As explained in Chapter 4, ERIDIS performs semi-decentralized resource management. HERMES, on the other hand, uses totally decentralized management – discussed later in Section 7.8 – where each resource stores its own agenda.

7.6.1 Request processing and agenda collection: usage of DTN

When a gateway receives a reservation request, it firstly performs admission control where the validity of the request is checked. Then, each request asks to collect the agendas of all the devices (ports and routers) along the network path between the source and the destination. In fact, the agenda of a link is stored twice by each router in its extremities[6]. This mechanism allows the two concerned routers to have the right energy-cost function (depending on the used bandwidth, and thus on the agenda) without having to ask its neighbors. Indeed, both sending and receiving data is energy consuming, thus both should be taken into account to compute the energy consumption.

To do this agenda collection, all the agendas of the path will be sent to the gateway of the receiver. The sender gateway will send a specific management message on the path containing the required agendas that it has (agenda of the receiving port linked to the sender and agenda of the transmitting port linked to the next hop on the path). Then, each router along the path will add to this message its own agenda and the agenda of the transmitting port linked to the next hop. Each router forwards this message to the next hop until it reaches the receiver gateway. The receiver gateway adds the agenda of its transmitting port linked to the receiver and asks the receiver its own agenda (containing on, off and transition periods between on and off). In the same way, this message also contains all the energy-cost functions of the network devices crossed (for each port and router).

In the end, the receiver gateway has all the agendas required to compute the availability agenda for the whole path as well as the energy the reservation will consume[7]. This process works only if all the ports and routers are on when the agenda collection is done, but in fact the network devices are put into sleep mode when they are not used. Disruption-Tolerant

[6] A link does not have storage capacity.

[7] This cannot be computed individually by each router because they required the agendas from other routers.

Networking (DTN) [FCG+06] technologies are used to solve this issue as they are adapted for this type of scenario where parts of the network are not always available thus impeding any guarantee of end-to-end connectivity at any time.

The rationale is to add a Time-To-Live (TTL) in seconds to each end-user request. If the request has neither reached the receiver gateway nor returned to the sender when the TTL expires, then all the sleeping nodes of the path are awaken to collect their agendas. While the TTL has not expired, the agenda collect message moves forward along the path until it reaches a sleeping node. Then, as long as the TTL is not expired, the message waits in the previous node for the sleeping node to wake up. When it wakes up (wake-up detection managed by the DTN protocol), the message is sent and continues its way. Thus, hop by hop, the agenda-collection message moves towards the receiver gateway.

However, DTN requires being able to wake up a node from a neighboring one, an issue that can be solved in different ways:

1) Using techniques inspired by sensor networks where each node is periodically awaken. The problem is to find a period that addresses both energy and performance concerns. This is not an on-demand waking service.

2) With wireless network cards with satellite connections on every node. This card remains always on (or is awaken frequently) and can awake the whole node upon reception of a special packet. This solution requires an operational wireless network.

3) Employing a low-power dedicated network for control, which is a spanning tree of the dedicated network. Each node of the actual network keeps a special low-power interface connected to the control network, able to wake up the whole node. This solution – similar to the previous one, but with a wired control network – implies duplication of all nodes, but not all the links.

All these techniques have an impact on energy consumption and must be studied under different scenarios. The TTL is determined by the user. A TTL equal to 0 means that all the paths should be awaken, if it is not already the case, and that the request should be processed as soon as possible.

An option frequently considered in the literature [SP03, SVP+10, BE09a, CCLM09] is to always keep a path between any two nodes and switch off just some links. However, under these solutions all nodes remain always on, even if idle, and they do not provide mechanisms to wake nodes up. Some research efforts handle this issue with periodical wake-up [NPI+08, BC08, BE09b]. A wake-on-arrival technique is described in [GS03] where routers are awaken automatically when detecting incoming traffic on their ports. While this is a highly desirable technique to save energy, it is not yet realistic since interfaces and routers do not yet provide this kind of hardware support. In fact, detecting traffic on all ports requires having a component always powered on for each port. In addition, after a wake-up a synchronization mechanism is required for the router to be fully working again [WVY+09].

7.6.2 Agenda merge

Once all agendas have been collected by the receiver gateway, they are merged to create the path's availability agenda used by the scheduling algorithm described in Section 7.5. This merging process is detailed in Algorithm 7.1. An example is depicted in Figure 7.4 with two agendas to merge. The first step is to build the time list by sorting all the events and deleting the duplicates. Then the bandwidth list is computed by looking at each agenda on the concerned date. One should remember to delete useless dates at the end of the merging process.

The complexity of this algorithm is $O(n \ \log n + n \times m)$ where n is the total number of events (length of the list t) and m is the number of merged agendas. Indeed, the sorting complexity is

$O(n \log n)$ and then, there is a for loop of size m in a for loop of size n which result in a $n \times m$ complexity.

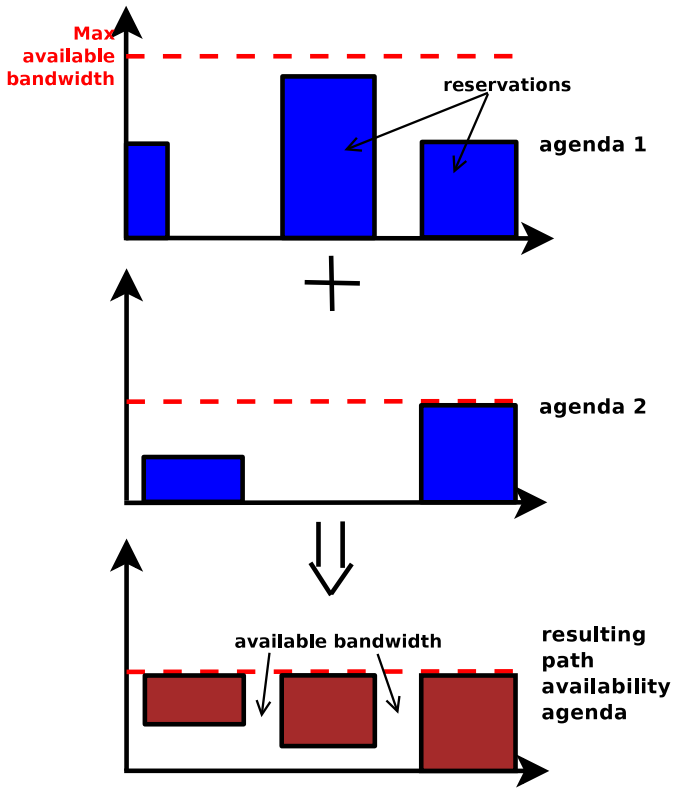

Figure 7.4: The agenda-merge process provides a path's **availability** agenda.

Algorithm 7.1 Agenda merge.

Sort in increasing order the list t of all the events from all the agendas and remove the duplicates.
ForEach event $t[i]$ in this ordered list **Do**
  ForEach agenda **Do**
    Compute the residual bandwidth at $t[i]$.
  Take the minimum of the residual bandwidths. It's $b[i]$.
  If $b[i] = b[i-1]$ **Then**
    Delete $t[i]$ from the final agenda.

7.6.3 Reservation granting

The receiver gateway schedules the reservation request because, in this way, only two end-to-end messages are required to schedule a reservation. The first message aggregates all the agendas and energy-cost functions from the sender to the receiver, whereas the second is sent by the receiver to the sender to inform about the schedule and to update all the agendas along the path. If a reservation cannot be granted, the sender gateway proposes another solution to the end-user with a less restrictive deadline. If the user accepts this solution, the sender gateway will then send the message to update all the agendas.

Figure 7.5 summarizes the steps involved in a reservation request. The sender gateway is also in charge of ensuring that the sender respects the network configuration of its reservation (bandwidth, destination, duration). It checks that the SLA is well respected by both parts.

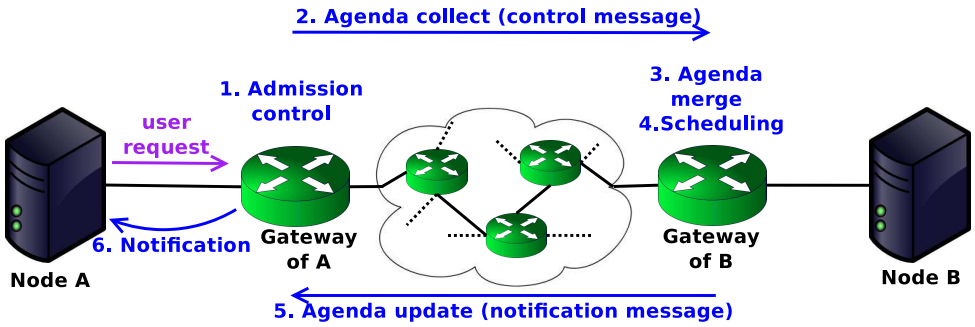

Figure 7.5: Steps to grant a reservation.

In the reservation granting process, a locking capability is required to ensure that different reservation requests do not interfere with one another.

7.6.4 Locking mechanism

It is mandatory to guarantee that requests are not scheduled to use the same time slot/bandwidth in one or more agendas. Hence, a locking mechanism is used to ensure that reservations do not overlap. This mechanism was not required in the case of data centers and Grids (Chapter 5) and Clouds (Chapter 6) since reservation management was hierarchical in that case and not distributed.

When the message collecting the agendas crosses a node, the node is informed of the part of its agendas required by the reservation. Then, this part is locked until the control message returns with updates. For fault-tolerance reasons, a timer is used on each node where the *default lock time* is denoted by τ. This timer is activated when an agenda is locked, and once it expires, the lock is released.

Only the ports' agendas are concerned with this locking mechanism. As the routers' agendas are used only during scheduling to find the most energy-efficient reservations, they are not affected by overlapping of concurrent requests. On a router, the overlapping requests are those that are being scheduled at the same time using the same interfaces.

When part of an agenda is locked, no other incoming request can reserve it. Instead, the incoming request tries to place a new lock to be activated once the current lock is released. When the current lock expires, the node sends an *update message* to the gateway scheduling the reservation that required the following lock. This message contains the updates made on the agenda during the previous lock.

To minimize the number of messages, if a lock is released less than θ_τ before it expires and there is no update on the agenda, no message is sent to the gateway concerned by the next lock. A message is unnecessary since the agenda has not been updated and the gateway already knows when its lock period begins.

The values of τ and θ_τ are common to the entire network and are based on the average latency between any two nodes. The rationale is that an update message is not required if it will arrive at a gateway after the start of its lock period.

Algorithm 7.2 Management of the locks on a router.

ForEach lock on an agenda **Do**
  If the lock timer initiated for a time period τ expires **Then**
    Release the lock;
    Inform the gateway concerned by the next lock on this agenda and start this new lock and its timer;
ForEach notification message (after the request scheduling stage) **Do**
  ForEach concerned agenda **Do**
    Update the agenda if necessary;
    Stop the concerned timer, the remaining time is denoted t_τ;
    Release the lock concerning this request;
    If there is a lock after the released one **Then**
      If t_τ is greater than θ_τ and the agenda has been updated **Then**
        Inform the gateway scheduling the request concerned by this lock (update message);
      Put this new lock on the concerned part of the agenda and start its timer;
ForEach incoming request (control message) **Do**
  ForEach concerned agenda **Do**
    If there is no lock on the concerned part of the agenda **Then**
      Put a lock on this part of the agenda;
    Else
      Put a lock on this part of the agenda after the last lock;
      Remember the gateway to contact at the release of the lock that precedes this new lock;

Algorithm 7.2, which runs as an infinite loop, shows how a router deals with agenda locks and timers. It also presents the process used to handle concurrent requests.

The request scheduling process is slightly impacted by this locking mechanism as shown in Algorithm 7.3, which describes the modified scheduling process at the receiver gateway.

Algorithm 7.3 Modification of the scheduling algorithm to include the locking mechanism.

If None of the concerned agendas has a lock **Then**
  Use the normal scheduling algorithm;
Else
  Start the scheduling algorithm by estimating the energy consumption at each possible place;
  ForEach lock **Do**
    Wait for the update message (sent after the lock release) until the beginning of its lock time slot;
  Estimate the energy consumption at new places created by the updates;
  Delete non-possible places due to the updates;
  Pick the least energy-consuming place;
  Send back the notification message to notify the user, to update the concerned agendas and to release the corresponding locks;

The algorithms described above respect the principle of fairness amongst requests since they are treated and scheduled ordered by submission time.

7.6.5 Prediction models and management of sleeping network devices

As outlined in [ZSGRG08], rate switching consumes time and energy. As in our framework every device knows its utilization, it can precisely anticipate and adjust its transmission rate to fit the demand without losing packets. The ALR buffer-threshold policy described in [GCS06] is useless

in this case. Switching network devices on and off also consumes energy and time [GS07b]. Since our framework aims to switch unused resources off to save energy, these switching stages should be accurately studied to determine whether it is more energy efficient to switch a device off and on again or to let it run idle.

As reservations complete, HERMES executes the ERIDIS algorithm to switch off nodes if they are idle for more than T_s seconds (Section 4.6).

This algorithm is used in a distributed manner and executed at the end of a transfer for each port independently of one another. The prediction algorithms, which rely on the recent history (past agenda) of a port, are based on average values of past inactivity periods and feedbacks. The latter are average values of differences between past predictions and the corresponding events in the agenda.

7.6.6 Re-planning capacity

Thus far we have presented algorithms that work in a static way, which means that a decision cannot be changed once it has been taken. Our re-planning capacity allows reservations to move dynamically after their registration in the path agendas, still guaranteeing the same QoS and respecting the user deadline. It is based on the re-planning capacity of ERIDIS presented in Section 4.9. Off-line algorithms can often lead to optimal solutions in terms of energy conservation because all the reservation requests are known from the start. However, it is not our case since we use on-line algorithms.

Algorithm 7.4 Re-planning of reservations to save more energy.

$E \leftarrow \{R_b, R_e\}$;
While $M \neq$ **Do**
  ForEach event t in E which is smaller than $M[0]$ **Do**
    Try to place $M[0]$ before and after t;
  Estimate the energy consumption for each possible place (including the current one);
  Pick the least consuming one;
  If the picked place is different from the current one **Then**
    Store this update;
    Add the beginning and end of this moved reservation in E;
  Remove $M[0]$ from M;
Send all the updates in the notification message;

In our distributed scenario, each node has partial information about the agendas of all the other nodes. Moreover, as agendas are inter-dependent, if you change a reservation in an agenda, all the other agendas, which may be on different nodes, need to change. Hence, off-line algorithms cannot be applied directly in this case.

To design the re-planning capacity, we use the path agenda generated by the gateways merging operation before the scheduling process. When the reservation is scheduled, the gateway has all the reservations that start on the first node of the path and end on the last node (or reverse path). As the gateway has all the agendas, it can move them.

After scheduling a reservation R, if there are other reservations using the exact same path and start after R begins, they could be re-scheduled and aggregated with R granted that doing so saves more energy than the current schedule. The re-planning algorithm is detailed in Algorithm 7.4, where E is the list of new events, R_b is the start date of reservation R, R_e is the end date of reservation R, and M is the set of reservations occurring after the start of R on the exact same path.

7.6.7 Discussion

The proposed network management optimizes the energy consumption of the overall architecture at any point in time. However, we have not optimized the energy required by transfers themselves. We have assumed that at any time, the most energy-efficient behavior is to use as much bandwidth as possible, but we have not proved that this algorithm minimizes the energy consumption.

Let us consider an example where node A wants to send 200 Mb of data to node B, and node A and B are directly connected by a 1 Gb/s link. Our algorithm will schedule the transfer and set the bandwidth at 1 Gb/s (excluding the free bandwidth portion). If we assume that the free bandwidth portion is negligible, it takes 0.2 seconds to transmit 200 Mb of data at 1 Gb/s. Therefore, this transfer will consume $E_{transfer}$ with $P_{EthernetCard}(NodeA, 1\ Gb/s)$ denoting the power consumed by node A when it transmits data at 1 Gb/s:

$$\begin{aligned} E_{transfer} &= E_{EthernetCard}(NodeA, 1\ Gb/s, 0.2\ s) + E_{EthernetCard}(NodeB, 1\ Gb/s, 0.2\ s) \\ &= P_{EthernetCard}(NodeA, 1\ Gb/s) \times 0.2 + P_{EthernetCard}(NodeB, 1\ Gb/s) \times 0.2 \end{aligned}$$

However, an alternative would be to adjust the Ethernet card to work at 100 Mb/s. It would not use the full capacity, the transfer would take longer, and consume:

$$\begin{aligned} E'_{transfer} &= E_{EthernetCard}(NodeA, 100\ Mb/s, 2\ s) + E_{EthernetCard}(NodeB, 100\ Mb/s, 2\ s) \\ &= P_{EthernetCard}(NodeA, 100\ Mb/s) \times 2 + P_{EthernetCard}(NodeB, 100\ Mb/s) \times 2 \end{aligned}$$

If we assume identical NICs with the same power consumption $P_{EthernetCard}(100Mb/s)$ and $P_{EthernetCard}(1Gb/s)$ depending on the rate, then the second solution uses less energy to transfer the data if and only if: $P_{EthernetCard}(1Gb/s) > 10 \times P_{EthernetCard}(100Mb/s)$.

If we use the figures provided in [ZSGRG08] for a NIC, we have $P_{EthernetCard}(100Mb/s) = 0.4$ Watts, and $P_{EthernetCard}(1Gb/s) = 3.6$ Watts. In that case, our scenario is the most energy efficient with a consumption equal to 0.72 Joules (and 0.8 Joules for the second scenario). However, here we only considered the energy used to transfer data and not the overall energy consumed by the infrastructure. Hence, these two energy consumptions do not represent the same period of time (0.2 and 2 seconds). To compare them over an identical time period, we should add to $E_{transfer}$ the cost of staying off during 1.8 seconds.

We have not taken into account the energy required to switch the NICs on at the start and to switch them off at the end of a transfer, since these energy costs are identical in both scenarios. This discussion shows that our algorithm should be compared to other solutions and that the optimal solution is hard to find even in scenarios with fixed routing. This situation results from the non-proportionality between energy and usage: cost functions are linear by steps and not just linear.

7.7 Experimental evaluation

Private networks have been the preferred network infrastructure for large enterprises and banks due to the level of isolation and security that they provide. Over the years, however, these networks have grown larger, become more complex and, though little attention has been given to it, consumed large amounts of electricity [MSBR09b].

7.7.1 BoNeS: Bookable Network Simulator

To validate our model, we have designed the Bookable Network Simulator (BoNeS) in Python. BoNeS, with more than 6000 code lines, takes as input a network-description file (topology

and router and link capacities) and network-traffic characteristics (e.g., statistical distribution of inter-arrival submissions, distribution of the reservation durations, source and destination nodes, distribution of the deadlines and TTLs). It then generates the network and bandwidth reservation traffic, and simulates different scheduling algorithms and compares their performance and energy consumption.

Dijkstra's algorithm is used at the beginning of the simulation to compute the shortest route between any pair of source and destination nodes. Then, another routing algorithm is used to compute a second shortest path, which is the most different from the shortest path in terms of used links. Hence, for each request, we first check if the first route (shortest path) can be used. Otherwise, we use the second route when it exists. This mechanism allow us to accept requests even if the primary route between sources and destinations is not available due to failures or other reservations, ensuring fault tolerance in the system.

The simulator runs five different scheduling algorithms on the generated traffic and network: 1) *first*: the reservation is scheduled at the earliest possible place in the agenda; 2) *first green*: the reservation is aggregated with the first possible reservation already accepted (before deadline), or scheduled to start as early as possible; 3) *last*: the reservation is scheduled at the latest possible place (respecting the deadline); 4) *last green*: the reservation is aggregated with the latest possible reservation already accepted (before deadline); 5) *green*: our model, where the energy consumption is estimated for each possible allocation, and the least consuming is chosen.

Our simulator provides the energy consumption for these five scheduling algorithms combined with our on/off technique where resources are switched off when they are not in use. The simulator also computes the energy consumption of the *first* scheduling without any on/off or rate adaptation algorithm (the state of current network technology), a case called *no off*. The generated network traffic consists of requests with:

- submission times distributed according to a log-normal distribution;
- data volumes generated with a negative exponential distribution;
- sources and destinations chosen randomly (equiprobability) among the end-hosts;
- deadlines generated with a Poisson distribution.

The probability distributions of the different traffic characteristics are customizable parameters. The distributions described here, henceforth used in the experiments, have been inspired by the results presented in [EYD07, BAM10].

The energy consumption of a networking device depends on the type of device, the number of ports, the port transmission rates (with rate adaptation), and the employed cabling solutions [MSBR09a]. Therefore, for each router, the energy consumption is modeled with two values corresponding to the chassis power ($P_{chassis}$) depending on whether it is on or off (around 150 W and 10 W respectively for a 1Gbps Ethernet switch [MSBR09b]). We take several values for the port power P_{port}: one for when it is off, another for when it is idle (working at the lower transmission rate), and one for each possible transmission rate. The time to boot and to shut down ports and routers is also taken into account.

For a 1 Gbps router, as an example, the values used in the simulations are presented in Table 7.1[8].

The overall energy consumption of servers is not taken into account, only the energy consumed by their Ethernet cards. Each 1 Gbps Ethernet card is assumed to consume 10 Watts when idle, and 15 Watts at full capacity.

[8]The values, in Watts, are per chassis and port.

Component	State	Power
Chassis	ON	150 W
	OFF	10 W
Port	1 Gbps	5 W
	100 Mbps	3 W
	idle, 10 Mbps	1 W

Table 7.1: Power parameters used for a 1 Gbps per-link router.

7.7.2 Evaluation 1: interbank networks

First, we have simulated the core of Fedwire Interbank Payment Network presented in [SBA+07] which represents 75% of daily transfers (Figure 7.6). This network comprises 66 nodes and 181 links. 25 nodes form a densely connected network, to which the remaining nodes connect. Our results for the entire network are provided in Table 7.2. Each router is assumed to consume 4 kW at full load [BAHT11]. Each data value is the result of 80 experiments of 4 hours of simulated time launched with 80 randomly generated requests that simulate 20% of load on each node.

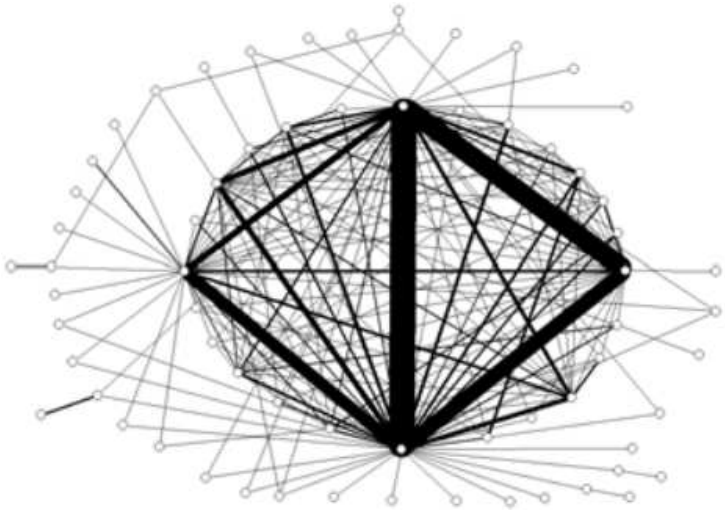

Figure 7.6: Core of Fedwire Interbank Payment Network.

Our model (*green*) provides the best results in terms of raw energy consumption, percentage of accepted requests and cost (in Wh per Tb sent in this network for this traffic). Compared to the current management (*no off*), it can save 42% of the energy currently consumed. One can see that the most relevant value is the cost, since the consumption depends on the duration of accepted requests.

Scheduling	First	First green	Last	Last green	Green	No off
Average	606 542	604 531	606 819	605 199	605 743	1 050 544
Standard deviation	7 106	6 994	7 012	7 010	7 090	1 511
Accepted requests	98.6%	98.3%	97.9%	97.5%	98.6%	98.6%
Cost in Wh per Tb	4 113	4 119	4 163	4 176	4 106	7 123

Table 7.2: Energy consumption for the core of Fedwire Interbank Payment Network.

7.7.3 Evaluation 2: enterprise networks

Figure 7.7 presents the simulated enterprise network whose topology comprises 21 routers, 360 end-hosts and 404 links. The hierarchical topology has been inspired by [Opp10]. The data center hosts computing resources as well as databases and storage systems. The two company servers represent the machines hosting typical enterprise services (i.e. DNS, DHCP or LDAP servers). The routers can also have advanced functionalities such as firewalls and packet filters.

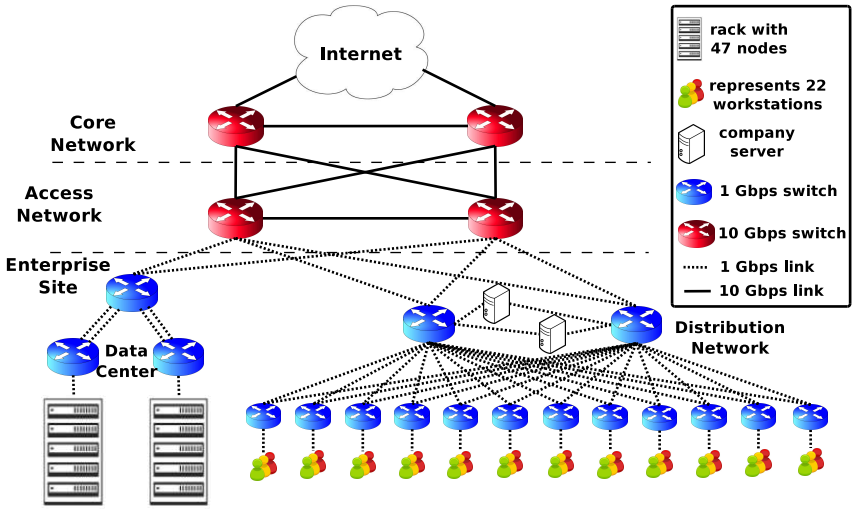

Figure 7.7: Example of an enterprise network's topology.

The results for the enterprise network are provided in Table 7.3 with an average load of 25% on the links. Our model still provides the best results and could save 22% of the energy. These results are based on future devices that will be able to perform rate adaptation and switch ports and entire routers on and off.

Scheduling	First	First green	Last	Last green	Green	No off
Average	4 174	4 195	4 202	4 429	4 152	5 333

Table 7.3: Cost in Wh per Tb for the enterprise network.

7.7.4 Evaluation 3: data center networks

The topology used to evaluate HERMES – comprising 482 servers (including 2 data servers), 24 routers and 552 links – is described in Figure 7.8. It represents a typical three-tier fat-tree architecture [AFRR+10, HSM+10] where the core tier comprises 4 layer-3 switches and the aggregation tier, responsible for routing, has 8 switches. Finally, each access tier switch is directly connected to 40 servers with 1 Gbps links. Two data servers, which contain for example the images or virtual machines to deploy on the nodes, are directly connected to the core network.

Two types of traffic are simulated on this network: the transfers among the nodes, a traffic induced by the user applications; and the transfers between the data servers and the computing nodes. The simulation has been launched 80 times for each experiment with requests generated as explained previously. Each simulation represents the behavior of the network during one hour. Table 7.4 shows the results obtained with a workload of 20% utilization of the links (i.e. links are used at their full capacity both ways during 20% of the time). This workload has been obtained by simulating 10,000 requests between the computing servers among themselves and 16000 requests between the data servers and the computing servers.

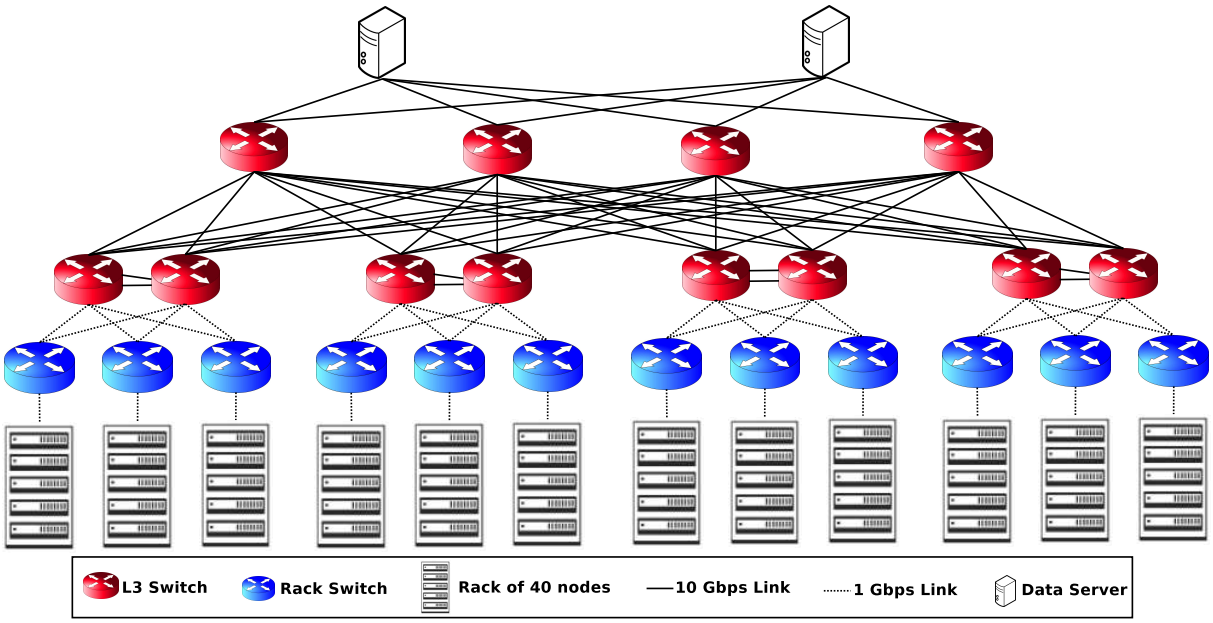

Figure 7.8: Typical three-tier architecture.

The *last green* scheduling is the least energy consuming, but it accepts a data volume 20% smaller than that of the *green* scheduling. The *first* scheduling has almost the same percentage of accepted volume, but it consumes 95 Wh more than the *green* scheduling. Therefore, the *green* scheduling presents the best trade-off between energy savings and request acceptance rate as shown by the cost in Wh per Tb.

Scheduling	First	First green	Last	Last green	Green	No off
Average (Wh)	6 111	6 039	5 684	5 625	5 944	21920
Standard deviation	97	93	76	70	84	371
Accepted volume (Tb)	141.98	141.54	120.24	113.70	141.97	141.98
Cost in Wh per Tb	43.04	42.66	47.27	49.47	41.87	154.39

Table 7.4: Energy consumption in Wh for 20% workload.

Table 7.5 shows the results obtained with a workload of 60% utilization of the links. We only present the cost in Wh per Tb since it is the meaningful value to compare the scheduling algorithms. This workload has been obtained by simulating 30,000 requests among the computing servers and 48000 requests between the data servers and the computing servers.

Scheduling	First	First green	Last	Last green	Green	No off
Average (Wh)	7 111	6 973	6 300	6 285	6 590	20 463
Standard deviation	362	335	100	106	305	809
Cost in Wh per Tb	42.18	41.37	40.21	41.25	39.09	121.37

Table 7.5: Cost in Wh per Tb for 60% workload.

By comparing Tables 7.4 and 7.5, one can notice that increasing the link's workload does not affect the cost in Wh/Tb of the scheduling algorithms. For a 60% workload, the *green* scheduling is still the best option in terms of both energy consumption and accepted volume.

Table 7.6 presents the energy saved under the two workloads with the green scheduling compared to the energy used when no switch off is allowed (*no off* case). In both cases, the energy savings that HERMES (*green*) can achieve are more than two-thirds of the energy consumed in

current infrastructures (*no off*): 68% of energy savings with 60% workload and 73% with 20% workload.

Load	20%	60%
Energy saved	73%	68%

Table 7.6: Results summary comparing *green* and *no off* results.

Until now, for HERMES, we have used a completely distributed resource management. Different architectural approaches can be applied to building management systems, including centralized, decentralized and clustered (where cluster-heads are responsible for the resources of clusters). The clustered scheme is often used for communication in ad-hoc sensor networks [Bas99], and so far it has not been applied to wired networks with the goal of minimizing the energy consumed by a management system. We advocate that this approach should be investigated to determine whether it saves energy when applied to management systems like it does in sensor networks.

7.8 Decentralized, clustering and centralized resource management

7.8.1 Network management approaches

In this section, we compare the three management approaches (decentralized, clustering and centralized) by applying them to HERMES. We present a simulation-based evaluation of these approaches for different networks and workloads, and show how much energy they consume when managing the reservation system. We did not made this study for data center, Grid and Cloud environments since they are naturally hierarchical and end-users do not have a direct access to resources[9] contrary to the case of networking resources.

Gathering the required agendas for request scheduling depends completely on where the agendas are stored. In the decentralized approach (followed up to now), each device gets its own agendas (per port and the global agenda). When a request is submitted, it should go on the path from sender to receiver collecting the required agendas along the way. In fact, the mechanism collects the agendas as follows. The sender gateway sends a specific management message to the receiver. The first node to receive it, adds its own availability agenda to the message and forwards it to the two next nodes that are the nearest to the destination. If the network topology is, for example, a simple tree with no redundant link, only one path is available and therefore, the message is sent only to the next node. The agendas of the transmitting ports are also included in the message. Each node includes the required agenda to the message and passes it to the next nodes. At the end, the destination gateway re-builds the end-to-end paths. In the centralized approach, a device stores all agendas and energy profiles. When a request is submitted, a control message is sent to the central node, it proceeds to agenda merging and scheduling and then sends the response to the sender.

The organization in clusters is similar to a hierarchical structure: selected nodes are responsible for other neighboring nodes. The selected nodes are called *cluster-heads*. We call *k-hop clustering* the network organization into clusters where the cluster-head is at maximum at k-hops from the nodes for which it is responsible. The control message in this approach should go to all the cluster-heads responsible for the nodes of the path between the source and the destination. The cluster-head election is done when the network is first initialized, following the algorithm based on node degrees described in [Bas99]. It should be performed each time the network changes (node additions or removals). This requirement is not constraining since dedicated networks do not change frequently. Considering the NSF network as an example,

[9]Users communicate with the resource management system via a portal which is the single access point to the resources it manages.

Figure 7.9 presents this election for all clustered cases: decentralized (0-hop), 1-hop clustering, 2-hop clustering and centralized (3-hop).

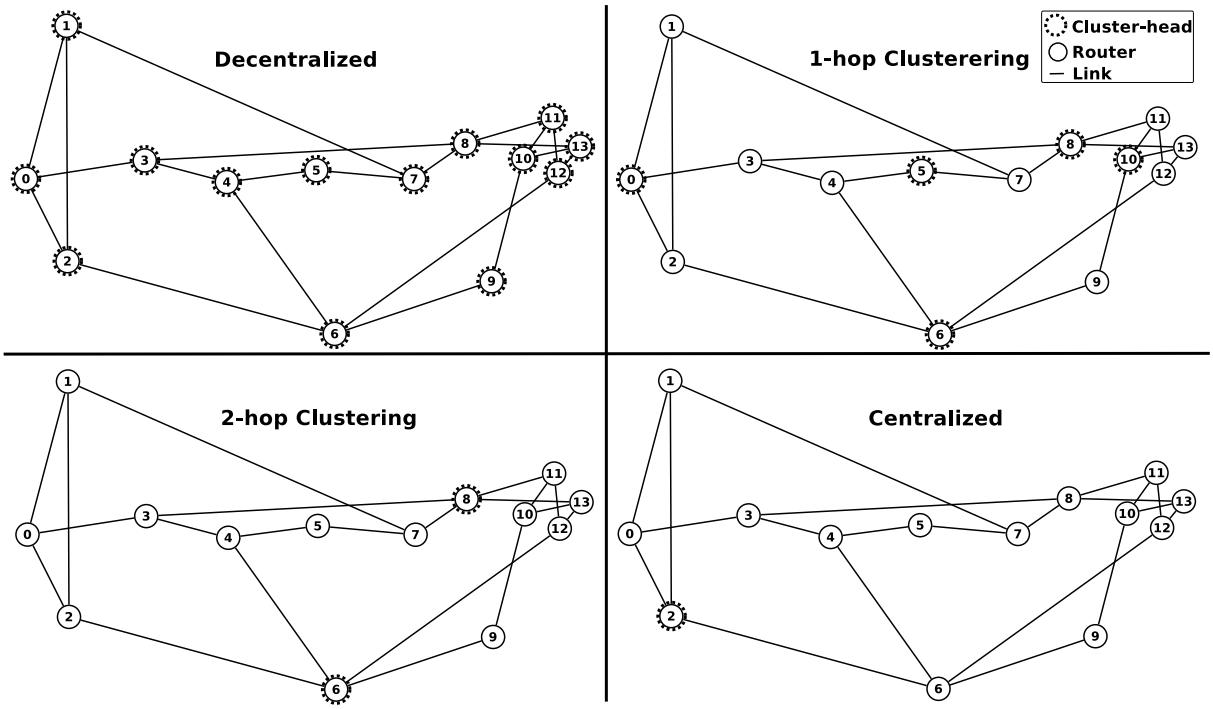

Figure 7.9: Examples of cluster-head election on the NSF network.

Similarly to the centralized scheme, in the clustering approach the path of the request is not the same as the path of the reservation. In fact, the reservation path goes from the sender to the receiver whereas the request path goes from the sender to the nodes storing the agendas of the nodes along the reservation path (i.e. the cluster-heads).

For example, using the decentralized approach in the network presented in Figure 7.9, to go from node 4 to node 13, the request uses nodes 4, 6, 12 and 13. In the 1-hop clustering, the request uses nodes 4, 5 (cluster-head of 4), 4, 6 (cluster-head of 6 and 12), 9 and 10 (cluster-head of 13). Hence, the control message ends with all the required agendas. The request path is longer than in the decentralized path. In the 2-hop clustering case, the request uses nodes 4, 6 (cluster-head of nodes 4, 6 and 12), 12, 11 and 8 (cluster-head of node 13). In the centralized approach, the request uses nodes 4, 6 and 2 (cluster-head of all the nodes). In the last two cases (2-hop and centralized), the alternate path that uses nodes 4, 3, 8 and 13, has the same request path as the first path (nodes 4, 6, 12, 13). Therefore, the control messages are aggregated on the same links and no other links need to be woken up. This is not the case for the 1-hop and the decentralized approaches, which require to wake up two distinct paths to get the agendas of the two possible reservation paths.

Because of the on/off capabilities of the network devices considered in HERMES, these different approaches lead to different overall energy consumptions. While the decentralized approach is more fault-tolerant and scalable, the centralized approach seems more energy-efficient (in terms of sent control messages) and simple to manage. Coordination is necessary to obtain system-wide energy-savings. Hence, the clustering approach could be a good trade-off between advantages and disadvantages of both approaches. In fact, in this approach several requests from different sources can be aggregated along the same path, and thus save energy.

7.8.2 Evaluation

The generated network traffic consists of requests with:

- submission times distributed according to a log-normal distribution;
- data volumes generated with a negative exponential distribution;
- sources and destinations chosen randomly (equiprobability);
- intervals between submission times and deadlines generated following a Poisson distribution;
- TTLs generated following a Poisson distribution.

We propose three networks randomly generated using the Molloy and Reed method [MR95]:

- *Net1* has 100 nodes and 259 links, and its diameter is 5.
- *Net2* has 100 nodes and 1029 links, and its diameter is 3.
- *Net3* has 200 nodes and 524 links, and its diameter is 6.

Table 7.7: Energy consumption for 10% workload on *Net1*.

Approach	centr.	1-hop	2-hop	3-hop	decentr.
Average (Wh)	13 970	14 091	13 948	13 794	13 897
Standard deviation	70	120	81	45	49
Messages	47 837	114 880	116 521	36 591	44 574

Each experiment, representing one hour of simulated time, has been launched 40 times on 40 different traffic files generated with the same characteristics to obtain representative average values. For each experiment, we evaluate the energy consumption of the whole network for the duration of the experiment taking into account the on/off and working consumptions of each device, and the number of 1-hop messages needed by all the reservations (i.e. if a message needs to traverse three links, it counts as three messages).

Table 7.8: Energy consumption for 10% workload on *Net2*.

Approach	centr.	1-hop	decentr.
Average (Wh)	15 467	15 578	15 565
Standard deviation	11	16	14
Messages	145 465	258 814	130 924

Table 7.7 presents the results for *Net1* with a 10% workload on each node. For this network with this traffic, there is a difference of almost 400 Wh between 1-hop and 3-hop (about 3% of the total consumption) for a 1-hour experiment. The 3-hop clustering presents the best results in terms of energy consumption and number of messages, beating the centralized and the decentralized approaches. As for the NSF Network example, the worst case in terms of number of messages is the 1-hop approach.

Table 7.9: Energy consumption for 10% workload on *Net3*.

Approach	centr.	1-hop	2-hop	3-hop	4-hop	decentr.
Average (Wh)	28 470	28 458	28 228	28 041	28 103	28 081
Deviation	80	148	125	98	63	65
Messages	107 999	265 648	310 038	290 247	90 401	87410

Table 7.8 presents the results for *Net2* for a 10% workload. In this case, with a highly connected network (node degree of 20 on average), the best solution is the decentralized approach. With this traffic, it can save 100 Wh per hour compared to the centralized approach (2-hop) and 110 Wh compared to the 1-hop clustering approach. The number of messages in the 1-hop case is almost twice the number in the 2-hop case. Yet, their energy consumptions are really close. As the network is really concentrated and the traffic is not intensive, most links are not necessary. In the decentralized approach, less links are used since the reservation path and the request path are the same. Thus, control messages can be aggregated with reservations, whereas in the decentralized approach (2-hop), the request path is shorter on average but different from the reservation path. Thus, more links are used and the traffic is less aggregated.

Table 7.10: Energy consumption for 40% workload on *Net1*.

Approach	centr.	1-hop	2-hop	3-hop	decentr.
Average (Wh)	15 213	14 975	15 498	15 839	15 813
Deviation	119	147	104	37	30
Messages	173 070	419 753	418 856	131 721	160 333

Table 7.9 presents results similar to Table 7.7: the 3-hop approach consumes the least power followed by the centralized approach, and decentralized approaches are the most energy consuming. Thus, increasing the network size does not affect the ranking of the approaches for a low usage (10%).

Table 7.11: Energy consumption for 75% workload on *Net1*.

Approach	centr.	1-hop	2-hop	3-hop	decentr.
Average (Wh)	16 578	16 003	16 344	16 655	16 607
Deviation	55	121	75	17	30
Messages	336 615	804 771	808 912	256 100	311 872

Table 7.10 presents the results for *Net1* for a 40% workload. The most energy efficient approach here is by far the 1-hop contrary to what we observed with a low workload (Table 7.7). When the workload increases further (see Table 7.11), the ranking is almost the same and the best solution is still the 1-hop approach, whereas the worst solution is the 3-hop approach. With medium and high workloads, the control messages do not require nodes to wake up as most of the network is already powered on due to the high traffic. These results show that clustering approaches should be considered alongside classical approaches to save energy.

Summary

The network management system has a non-negligible influence on the overall energy consumed by the network. We have identified network density and workload as factors that have an important impact on the energy consumption in reservation-based networks. These two network characteristics should be used to determine the management system to apply to the targeted network in order for it to be energy-efficient. Decentralized management performs better on concentrated networks with low traffic, whereas big clustering (i.e. with a large cluster size) approaches are more energy-efficient on lightly-used networks which are not concentrated. However, as the network traffic increases, small clustering techniques become more energy-efficient.

We are currently studying mechanisms to adapt the network management system configuration to the workload in an autonomic way. Generally, the density of wired networks does not vary a lot, while the workload can present daily or weekly patterns with peak traffic and slack periods.

7.9 Conclusion

Bandwidth provisioning has been made feasible to network operators over several years thanks to protocols such as MultiProtocol Label Switching (MPLS) [RVC01] and Resource ReSerVation Protocol (RSVP) [ZDE+93]. However, for end-users with no network-traffic knowledge, this task is impossible when there is no coordination between nodes.

On the other hand, as networks become increasingly essential, their electric consumption reaches unprecedented peaks [BO09]. Up to now, the main concern when designing network devices and protocols has been performance. Energy consumption has not been taken into account, but with the growth in electricity demand and its associated costs, it is time to consider energy as a major factor in network design.

The increasing number of large-scale applications requiring high-speed data transfer has encouraged the development of high-performance dedicated networks [LW08]. They are particularly favorable to the adoption of a reservation system since they carry out big data transfers requiring guaranteed QoS and they are managed by a single entity, hence nodes can easily exchange management information.

To reduce energy consumption of reservation-based dedicated networks, we have proposed: i) an end-to-end energy-cost model called ECOFEN (Energy Consumption mOdel For End-to-end Networks) that considers the topology and the traffic to estimate the energy consumed by a given network with energy-efficient components; ii) a network model which is adapted to bandwidth reservations for data transfers; and iii) a new complete and energy-efficient data transfer framework including scheduling algorithms that provide an adaptive and predictive management of the reservations.

Simulations have shown that, under given loads and topologies, the proposed framework can achieve important energy savings compared to the current approach with no energy-management system. For a typical three-tier architecture with a 60% load on average on the links, using HERMES can save 68% of energy compared to a current approach without energy management. However, our propositions rely on technology that is not yet readily available, such as low-power modes, small booting times, rate adaptation to the traffic and autonomic smart management.

We also show that the network management system (e.g. centralized, decentralized) has a non-negligible influence on the overall energy consumed by reservation-based networks. We have identified network density and workload as influencing factors. For example, decentralized management performs better on concentrated networks with low traffic. Network characteristics should be used to determine the management system to apply to be the most energy-efficient.

Our next step is to adapt HERMES to several types of networks in order to deal with small flows as well (mice), since HERMES is more suitable for bulk transfers or big flows (elephants). The idea is to aggregate small flows to obtain medium-sized flows (dogs). We also plan to deal with urgent flows either by provisioning bandwidth by anticipation using prediction algorithms or by always let on a low-power spanning tree network (sub-network of the initial network) which will be used to carry urgent messages. The problem of urgent message transport occurs only when parts of the network are in sleep-mode.

A scientist in his laboratory is not a mere technician: he is also a child confronting natural phenomena that impress him as though they were fairy tales.

Marie Curie

8 Conclusions and perspectives

Energy efficiency is now a central concern in large-scale distributed systems. In this thesis, we focused on designing an energy-efficient reservation framework for large-scale distributed systems. Dealing with data centers, computing Grids, Clouds and dedicated network resources, this framework, called ERIDIS, enables energy-efficient resource allocation via reservation scheduling, prediction algorithms, on/off resource management and green policies.

8.1 Conclusions

In this thesis, we reviewed and classified existing solutions to save energy in large-scale distributed systems. While not extensive, this study highlighted the main contributions and revealed that green computing and networking has been a really active research field for several years.

Attempting to check the applicability of a variety of energy consumption models for computing and networking resources, we monitored and measured the power consumed by our experimental platform. This adventurous path led us to design a novel instrumenting platform that measures every second the power consumption of monitored nodes – the nodes constitute a site of our experimental Grid. This infrastructure allowed us to investigate the power consumption of computing resources; an unavoidable step that provided us with a better understanding of resource wastage in large-scale distributed systems.

Boosted by this experience, we proposed an energy-efficient reservation infrastructure, called ERIDIS, for large-scale distributed systems. Combining state-of-the-art approaches (e.g. workload consolidation, sleeping modes and adaptation to the load) with prediction algorithms and reservation aggregation, this framework can be deployed in Grid, Cloud and dedicated network management systems. The design of such a unified framework was a challenging task due to the intrinsic difference among the target systems. However, common energy-management principles can be applied to these reservation-based systems in general.

As a proof of concept, we adapted ERIDIS to reservation-based Grid environments; an adaptation motivated by the usage analysis of an experimental Grid (i.e. Grid'5000, a French experimental testbed). This analysis revealed a heterogeneous usage made up with bursts and gaps, thus indicating that on/off techniques can be efficient. We hence designed an Energy-Aware Reservation Infrastructure (EARI), based on ERIDIS architecture, whose validation has been made using the Grid'5000 traces previously collected and studied. A replay of these traces with different green policies showed the efficiency of our prediction algorithms and of the whole infrastructure by comparing it with commonly used power management. We showed that for an one-year period, EARI could have reduced by a half the energy consumption of Grid'5000 computing resources – an amount that represents the energy consumption of a French village of 600 inhabitants.

In a Cloud context, we examined the power consumption of virtual machines when performing typical tasks (booting, running applications, switching off). We went a step further by analyzing the power consumption of a host performing live migration of virtual machines. These observations permitted us to adapt ERIDIS to Cloud environments using resource reservations

and live migration. Then, we implemented a prototype of this new framework, termed as the Green Open Cloud (GOC), on our local platform to measure and compare its energy efficiency with classical Cloud power management. Energy measurements on our Cloud scenarios showed reductions of 25% in energy consumption compared to basic resource management.

The remaining studies on large-scale distributed systems was on dedicated wired networks. As of writing, network devices that allow shutdown and slowdown techniques were not available, we designed a simulator called Energy Consumption mOdel For End-to-end Networks (ECOFEN), which builds on the well-known Network Simulator (NS2). ECOFEN embeds energy-efficient functionalities from the literature and can simulate the power consumption of large-scale networks using various scenarios in terms of traffic, protocols and topologies. By using the energy model incorporated in ECOFEN and adapting ERIDIS to the wired network case, we proposed a High-level Energy-aware Reservation Model for End-to-end networkS (HERMES). To compare this model with other scheduling approaches for reservation-based networks, we wrote a simulator called Bookable Network Simulator (BoNeS) using traffic laws and power consumptions of networking equipment found in the literature. These simulations on a typical three-tier fat-tree architecture with a 60% workload on average on the links showed a consumption reduction of 68% compared to classical resource management.

In each context, we used validation techniques as close to the reality as possible:

- replay of real usage traces using measured consumption values on an experimental platform for data centers and computing Grids since we did not have access to logs and energy consumption of a data center or Grid;
- prototype implementation on our testbed comprising wattmeters for Clouds because the energy consumption of resources was not available for accessible Cloud infrastructures;
- simulations using realistic topologies and traffic laws and energy consumptions taken from the literature for dedicated networks since neither a dedicated platform nor energy-aware networking equipment were available.

The adaptation of ERIDIS to these three contexts enhanced the framework and underlined the common principles that can be applied to these reservation based systems: on/off algorithms, workload consolidation, usage predictions and speed scaling mechanisms. The validation of these three frameworks, namely EARI, GOC and HERMES, which are based on ERIDIS, constitutes the proof of concept that we were looking for. It also proves that equipment manufacturers should add green functionalities to their products, and infrastructure managers should deploy energy-aware infrastructure to reduce their electricity bill.

8.2 Future directions

Some relevant issues of our approach have not been addressed in this thesis, of which we identify and constitute a few for future work.

8.2.1 Proposition of a complete resource reservation framework

To the best of our knowledge, no current Grid or Cloud infrastructure proposes reservation mechanisms for computing, storage and network resources at the same time. Mixing our three propositions – EARI, GOC and HERMES – to enable simultaneous resource reservation on Grids and Clouds seems a promising future work. It requires a more complex user interface to describe users requirements in terms of these types of resources, and the user should exactly know the timeline of its application to determine when he or she will require the resources to optimize the energy consumed by its reservation. The complexity and computation time of the scheduling algorithm also increases, since it adds correlations between the resources (e.g. the

reservation of computing resources depends also on the availability of storage and networking resources).

8.2.2 Validation using real applications on real testbeds

EARI has been validated using a replay mechanism of Grid'5000 traces. As we have access only to the reservation logs and not always to the CPU usage logs for example, we do not have the real energy consumption of user applications. Therefore, we use the maximal power consumption of the nodes to model the consumption of reserved nodes. Although this solution represents the worst case in terms of energy consumption, it does not faithfully reproduce the variations of the dynamic energy consumption observed during the analysis of the Grid'5000 energy logs. The main problem here is to find representative applications for this kind of system. The same problem occurred with the validation of GOC. We used a *cpuburn* application (which reaches the maximal power consumption) to model user's applications. This application is not impacted by resource sharing among the different virtual machines and the resulting lengthening of the application's duration. However, as in our case users reserve time periods of resources, this issue does not limit the scope of our validation. This application indeed represents the maximal power consumption for the whole duration of the reservations, but it prevents us from using dynamic voltage and frequency scaling techniques. For this reason we plan to reproduce the validations with representative applications for large-scale data centers, Grids and Clouds. In the case of dedicated networks, representative applications must be devised since few network management systems are based on reservations.

The validation of EARI on Grid'5000 traces has led to the implementation of its on/off capabilities into the resource management system of Grid'5000. GOC was validated through a prototype implementation over a small local testbed. We aim at implementing it on a bigger Cloud infrastructure, but this infrastructure should respect two constraints: having a reservation-based resource management and using live-migration of virtual machines. To our knowledge, no public or private cloud supports both of these features. Therefore, an implementation at a larger-scale can only be done on experimental platforms such as Grid'5000 where we can deploy our own middleware. For the validation of HERMES on real platforms, the problem is even more complex since reservation-based networks are essentially private and the kind of network equipment that we use to design HERMES does not exist on the market. However, we would like to explore programmable router capabilities to implement the HERMES framework on a real router as a proof of concept. A validation at a wider scale seems hard to realize due to budget constraints.

8.2.3 Fault-tolerance

The fault-tolerance issue is a main drawback in all the reservation-based infrastructures. Identical computing resources may be interchangeable. Yet, this is not the case for links or routers. If the interface or router encountering a failure has an empty agenda (no current reservation in the future) and if this element is not essential to the network (it does not disconnect the network), then HERMES behaves well since it uses the second path for each request and reservation concerned by this resource before re-launching the route computation algorithms. However, if some reservations are scheduled into the agenda of the dead resource, no mechanism is proposed to reschedule these reservations on others resources. Fault-tolerance mechanisms are easier in a centralized resource management where a single entity has an entire view of the system. We intend to develop mechanisms to deal with fault-tolerance in our reservation context for both centralized and decentralized management systems.

8.2.4 Managing best-effort jobs and traffic

Mixing best-effort jobs and resource reservations, like mixing best effort traffic and scheduled transfers, require mechanisms (like backfilling) beyond the scope of this thesis and is thus left

for future work. The main purpose for focusing on reservation-based systems was to place our work in a more controlled, and hence predictable environment to allow greater energy savings. Future work should relax this reservation constraint and apply our framework to more common infrastructures. This can also lead one to compare the energy-efficiency of management systems with and without reservations. We plan to adapt EARI and GOC to allow best-effort jobs. One solution to ensure the great reactivity of the system is to make fake resource reservations when load peaks are predicted, so powered-on resources are available for urgent jobs.

We also intend to adapt HERMES to all types of networks to deal with small flows (mice), since HERMES is currently more suitable for bulk transfers or big flows (elephants). The idea is to aggregate small flows to obtain medium-sized flows (dogs). We also plan to deal with urgent flows either by provisioning bandwidth in advance using prediction algorithms or by always letting a low-power spanning tree network (sub-network of the initial network or duplicated low-power network linking together the same end-hosts) powered on to transmit urgent messages. The problem of transmitting urgent messages occurs only when parts of the network are in sleep-mode.

A

The ECOFEN simulator

The problem when evaluating new network architectures and protocols is that large testbed platforms are really expensive and difficult to manage. That is why we have designed ECOFEN whose user's entries are the network topology and traffic. Based on configurable measurements of different network component (routers, switches, NICs, etc.), it provides the power consumption of the overall network including the end-hosts as well as the power consumption of each equipment over time.

In [Len10], the author aims at simulating the power and energy consumption of computer networks. But, his simulation models are based on measurements of PC-based equipment which differs from real routers in terms of architecture and energy consumption. An energy-aware Cloud simulator based on NS2 is presented in [KBK10]. The network model used is close to ours, but it assumes a linear relation between energy consumption and load and does not include energy-efficient capabilities such as Adaptive Link Rate and on/off since their goal is to provide a simulation environment for energy-aware Cloud computing data centers.

The framework presented here is the first, to our knowledge, that can be applied to any network topology with various kind of equipment and supporting energy-efficient techniques such switching off nodes and ALR.

The aim of this simulator is to compute the energy consumed by a network under a given traffic. A module for NS2, one of the most used simulation tools in the networking research community, was developed to offer this functionality. It is implemented as an NS2 module and uses the Energy Consumption mOdel For End-to-end Networks (ECOFEN) model described in Section 7.2. This simulator has been made in collaboration with Dino Lopez-Pacheco [OLGLLP11].

A.1 Architecture of the simulator

A.1.1 The Network Simulator NS2

NS2 is an open source network simulator for evaluating networks, protocols and topologies. It provides discrete event-based simulations including, among others, transport protocols, routing protocols, and multicast protocols over wired and wireless (local and satellite) networks[1]. The core is written in C++ and the provided simulation interface is in OTcl (an object-oriented extension of Tcl). A user describes a network topology and traffic using OTcl scripts.

Our module is integrated within NS2, and is thus written in C++. It defines energy properties for each network equipment and the model is implemented with the aim to obtain the energy consumption for each equipment and the overall consumption at every second.

[1] *http://www.isi.edu/nsnam/ns/*

A.1.2 The energy module

The NS2 module takes the following input parameters:

- a network topology (with link capacities and equipment types),
- network traffic (sources, destinations, type of traffic, rate profile, protocols),
- a scenario (beginning and end of the different traffic, failures, switching on and off of the network devices and ports),
- real energy consumption values for the network devices used in the topology, values that are included in the model detailed in the previous section to provide the overall consumption.

For example, for a router, the energy consumption values include the fix cost (power consumption of the chassis, the route processor module, etc.) and the variable costs (energy profile for each port depending on the link rate). From the variable costs, we compute the cost in Joules for a byte received or sent by an interface. This cost is used by the simulator to compute the overall consumption of the equipment.

The simulator is able to compute the energy consumed from the smallest possible network (2 end-host machines connected together) up to networks of thousands of nodes. This also implies the ability to choose the physical characteristics of hardware, and the bandwidth of a link as well as its latency.

The traffic between the network nodes is configurable: the user can choose where and when to send the packets, the packet sizes and characteristics of the used protocols. Any network usage can be simulated, such as voice over IP, Peer-to-Peer networking or browsing the Web. In addition, we can simulate different technologies proposed to save energy in wired networks (e.g. it is able to change the link state (on and off) and rate, and to extinguish nodes and ports).

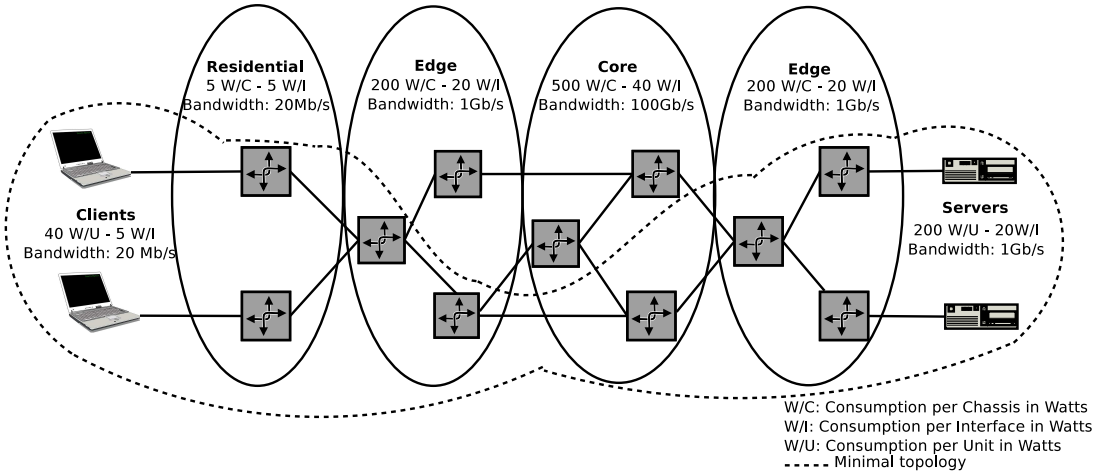

Figure A.1: Full and minimal topologies for the client/server scenario.

For a given experiment the simulator's outputs show the power consumption per node and per second, providing the ability to compare algorithms (routing or energy savings algorithms for example), network architecture designs and network components (different routers, switches, NICs, etc.) in terms of both energy consumption and performance.

A.2 Scenario and simulations

We evaluated the ECOFEN simulator under several scenarios.

A.2.1 Comparison of network architectures

The first scenario uses the network depicted in Figure A.1 considers a case with all the nodes (referred to as *full topology*) and another without three router nodes (referred to as *minimal topology*). For both cases, each of the two clients downloads a big file from one of the servers with a MTU of 1500 bytes. The power consumption values used for this experiment are given in Figure A.1. For each networked equipment, the simulator uses one value for the chassis and another per interface (port or NIC) when working at full speed. The cost per octet is computed from this latter value.

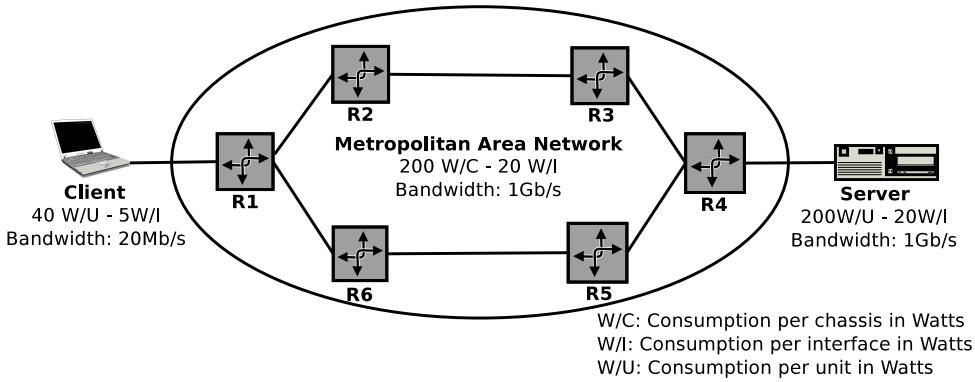

Figure A.2: Topology of the metropolitan network with redundant routing devices.

The goal of this example is to compare the energy consumption of two different network architectures for a given application. Table A.1 presents the total energy consumed by the network by evaluating the two proposed topologies.

As expected, the minimal topology consumes less, since idle nodes in the full topology consume a lot of energy. Thus, energy management algorithms for networks can take advantage of this to switch off unused resources.

Scenario	Consumption (Watts)
Full topology	3971.5 W
Minimal Topology	2351.5 W

Table A.1: Power consumption for the minimal and full topologies on a big file download scenario between clients and servers.

A.2.2 Evaluation of the dynamic energy cost of traffic

As a second scenario, the ECOFEN simulator allows to directly evaluate the energy cost of different devices, accounting for traffic changes and consequent variations in power consumption.

Figure A.3 presents the consumption of the Ethernet Card of node A on the topology presented in Figure 7.2. The generated traffic is a basic TCP flow sent by node A to node B using TCP NewReno, which is widely deployed on the Internet. This graph clearly illustrates the behavior of the window congestion of TCP.

A.2.3 Evaluation of energy-efficient strategies

The topology used for the third scenario is presented in Figure A.2. The two upper nodes (R2 and R3) are turned off at the 300th second of the simulation along with their interfaces and the interfaces with which they are linked. Thus, from the 300th second, the overall system consumes less energy as shown in Figure A.4. A rerouting algorithm is needed if the turned-off path is switched on again.

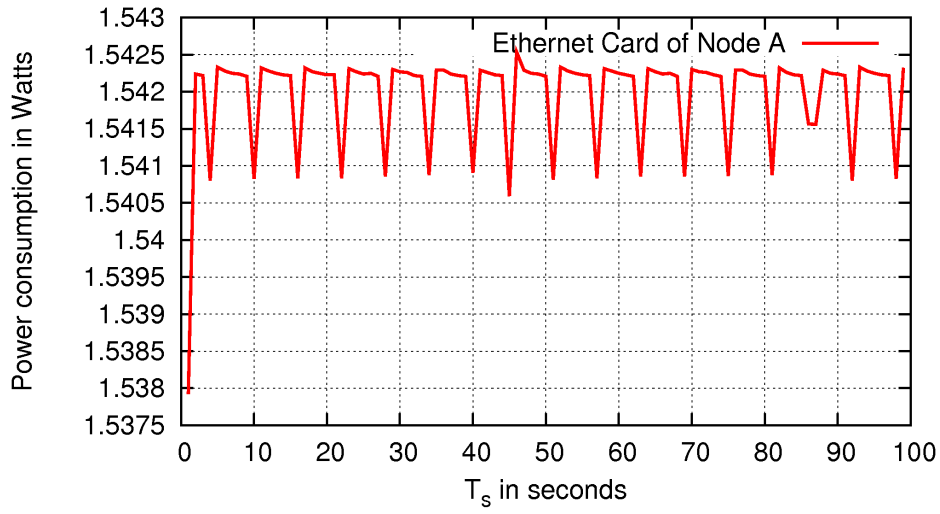

Figure A.3: Power consumption of a 100Mbps Ethernet Card with a TCP flow.

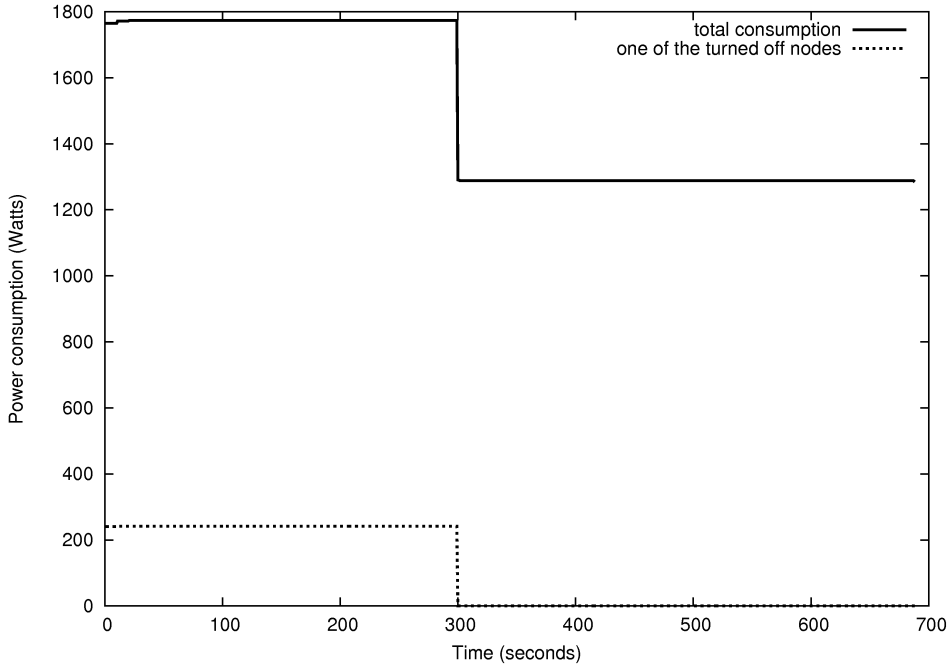

Figure A.4: Power consumption of the network for the simulation of an energy management algorithm that turns off two nodes.

This second example shows that this simulator, including basic energy saving techniques (on/off and rate adaptation), can be used to validate and compare new network energy management algorithms built on these basic techniques.

A.2.4 Evaluations on large-scale networks

To show that the simulator can be used for computing the energy consumption of large-scale networks, we simulate a hierarchical network consisting of 8 core routers, 52 edge routers, 52 access routers, 260 residential switches and 260 end-hosts. In this network, inspired by [CML+10], the nodes are interconnected by 1056 links. During the experiment that lasts 100 seconds, we simulate 130 TCP transfers between end-host pairs (randomly chosen). The flows are chosen randomly and launched at different times to create heterogeneous traffic: from 0 to 25 seconds, 30 TCP flows; from 25 to 50 seconds, 90 TCP flows; from 50 to 75 seconds, 30 TCP flows, and from 75 to 100 seconds, 130 TCP flows.

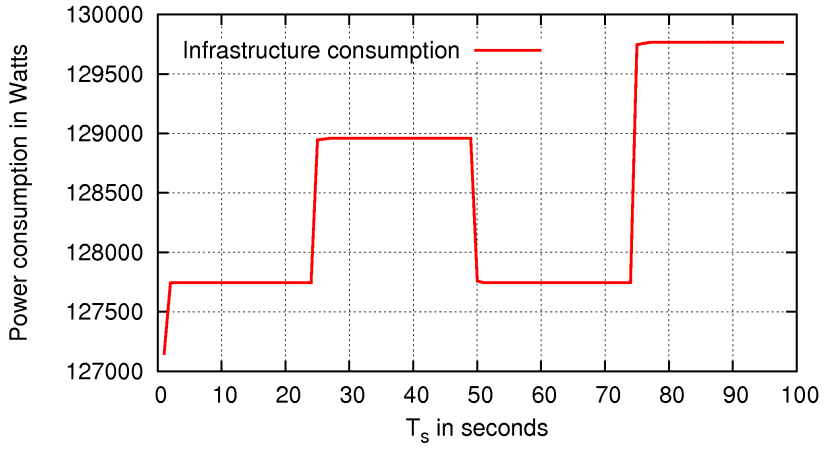

Figure A.5: Power consumption of a network including 632 nodes and 1056 links.

The total power consumption of this network is plotted in Figure A.5, which also shows the correlation between energy consumption and traffic. The dynamic or variable consumption is small compared to the fixed one, as discussed earlier.

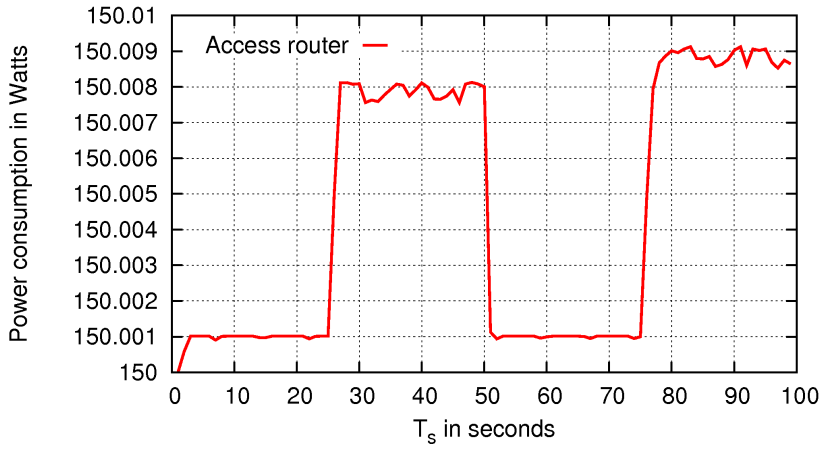

Figure A.6: Power consumption of an access router.

The simulator can provide the power consumption of each equipment at every second. To give an example, Figure A.6 shows the consumption of one of the access routers.

A.3 Conclusions and Future Works

This appendix presents a simulator based on the Energy Consumption mOdel For End-to-end Networks (ECOFEN). ECOFEN allows users to evaluate the power consumption of large scale networks by defining topologies, network devices, traffic type and network protocols. ECOFEN has been designed to support energy leverages based on some advanced functionalities not yet largely available in networks (like dynamically switching on and off some devices, adaptive link rate). This simulator should help network designers to simulate their proposed strategies on energy consumption reduction before deploying them at large scale.

The designed simulator is currently under finalization and will be made publicly available for the research community exploring energy issues in large scale networks. Some traffic plug-ins will be implemented in order to give more realism to observed traffic (like packet loss and jitter) and some real calibration values will be collected and injected in the simulator in order to include a collection of network equipment and devices (routers, interfaces, network cards) with realistic measured values. Our final goal is to provide user basic energy management functions such as on/off capabilities, rate adaptation, network virtualization and low-power idle mode (these techniques are described in Chapter 2). We believe that such functionalities will allow users to design their own high-level energy management on large-scale wired networks.

B
Publications

International journals

[1] Anne-Cécile Orgerie and Laurent Lefèvre, **"ERIDIS: Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems"**, *Parallel Processing Letters*, 21(2):133-154, June 2011.

[2] Anne-Cécile Orgerie, Laurent Lefèvre and Isabelle Guérin-Lassous, **"Energy-Efficient Bandwidth Reservation for Bulk Data Transfers in Dedicated Wired Networks"**, *Journal of SuperComputing, Special issue on Green Networks*, March 2011.

[3] Laurent Lefèvre and Anne-Cécile Orgerie, **"Designing and Evaluating an Energy Efficient Cloud"**, *Journal of SuperComputing, Special issue on Emerging Research in Parallel and Distributed Computing*, 51:352-373, March 2010.

[4] Laurent Lefèvre and Anne-Cécile Orgerie, **"Towards Energy Aware Reservation Infrastructure for Large-Scale Experimental Distributed Systems"**, *Parallel Processing Letters*, 19(3):419-433, September 2009.

Book chapters

[5] Anne-Cécile Orgerie and Laurent Lefèvre, **Energy-Efficient Reservation Infrastructure for Grids, Clouds and Networks**, *chapter in Energy Aware Distributed Computing Systems*, Wiley Series on Parallel and Distributed Computing, John Wiley & Sons, to appear in 2011.

[6] Anne-Cécile Orgerie and Laurent Lefèvre, **Energy-efficient data transfers in large-scale distributed systems**, *chapter in Handbook of Energy-Aware and Green Computing*, Chapman & Hall (ISBN: 978-1-46650-116-4), 2011.

[7] Anne-Cécile Orgerie, Marcos Dias de Assunção and Laurent Lefèvre, **Energy Aware Clouds**, *chapter in Grids, Clouds and Virtualization*, pages 145-170. Springer Book (ISBN: 978-0-85729-048-9), 2010.

International conferences

[8] Anne-Cécile Orgerie, Laurent Lefèvre and Isabelle Guérin-Lassous, **"On the Energy Efficiency of Centralized and Decentralized Management for Reservation-Based Net-**

works", *IEEE Global Communications Conference (GLOBECOM 2011)*, Houston, USA, December 2011.

[9] Anne-Cécile Orgerie and Laurent Lefèvre, **"Energy-Efficient Overlay for Data Transfers in Private Networks"**, *IEEE International Conference on Networks (ICON 2011)*, Singapore, December 2011.

[10] Anne-Cécile Orgerie and Laurent Lefèvre, **"Energy-Efficient Framework for Networks of Large-Scale Distributed Systems"**, *IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2011)*, Busan, Korea, pages 250-255, May 2011.

[11] Marcos Dias de Assunção, Anne-Cécile Orgerie and Laurent Lefèvre, **"An Analysis of Power Consumption Logs from a Monitored Grid Site"**, *IEEE/ACM International Conference on Green Computing and Communications (GreenCom-2010)*, Hangzhou, China, pages 61-68, December 2010.

[12] Georges Da Costa, Marcos Dias de Assunção, Jean-Patrick Gelas, Yiannis Georgiou, Laurent Lefèvre, Anne-Cécile Orgerie, Jean-Marc Pierson, Olivier Richard and Amal Sayah, **"Multi-Facet Approach to Reduce Energy Consumption in Clouds and Grids: The GREEN-NET Framework"**, *e-Energy 2010: International Conference on Energy-Efficient Computing and Networking*, Passau, Germany, pages 95-104, April 2010.

[13] Anne-Cécile Orgerie and Laurent Lefèvre, **"When Clouds become Green: the Green Open Cloud Architecture"**, *Parco 2009: International Conference on Parallel Computing*, Lyon, France, pages 228-237, September 2009.

[14] Anne-Cécile Orgerie, Laurent Lefèvre and Jean-Patrick Gelas, **"Save Watts in your Grid: Green Strategies for Energy-Aware Framework in Large Scale Distributed Systems"**, *ICPADS 2008: The 14th IEEE International Conference on Parallel and Distributed Systems*, Melbourne, Australia, pages 171-178, December 2008.

[15] Anne-Cécile Orgerie, Laurent Lefèvre and Jean-Patrick Gelas, **"Chasing Gaps between Bursts: Towards Energy Efficient Large Scale Experimental Grids"**, *PDCAT 2008: The Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, Dunedin, New-Zealand, pages 381-389, December 2008.

International workshops

[16] Anne-Cécile Orgerie, Laurent Lefèvre and Isabelle Guérin-Lassous, **"ECOFEN: an End-to-end energy Cost mOdel and simulator For Evaluating power consumption in large-scale Networks"**, *Sustainet: First International Workshop on Sustainable Internet and Internet for Sustainability, in conjunction with WoWMoM: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Lucca, Italy, June 2011.

[17] Anne-Cécile Orgerie, Laurent Lefèvre and Jean-Patrick Gelas, **"Demystifying Energy Consumption in Grids and Clouds"**, *Work in Progress in Green Computing (WIPGC) Workshop, in conjunction with the first International Green Computing Conference (IGCC)*, Chicago, USA, pages 335-342, August 2010.

[18] Marcos Dias de Assunção, Jean-Patrick Gelas, Laurent Lefèvre and Anne-Cécile Orgerie, **"The Green Grid'5000: Instrumenting a Grid with Energy Sensors"**, *INGRID Workshop: Instrumenting the Grid*, Poznan, Poland, May 2010.

[19] Georges Da Costa, Jean-Patrick Gelas, Yiannis Georgiou, Kamal Sharma, Laurent Lefèvre, Anne-Cécile Orgerie, Jean-Marc Pierson and Olivier Richard, **"The GREEN-NET Framework: Energy Efficiency in Large Scale Distributed Systems"**, *HPPAC 2009: High Performance Power Aware Computing Workshop in conjunction with IPDPS 2009*, Rome, Italy, May 2009.

National journal

[20] Anne-Cécile Orgerie, Laurent Lefèvre and Jean-Patrick Gelas, **"Étudier l'usage pour économiser l'énergie dans les systèmes distribués à grande échelle : l'approche EARI"**, *TSI : Technique et Science Informatiques*, volume 30 - n 5/2011, pages 515-538, May 2011.

National workshop

[21] Anne-Cécile Orgerie, Laurent Lefèvre and Jean-Patrick Gelas, **"Économies d'énergie dans les systèmes distribués à grande échelle : l'approche EARI"**, *JDIR 2009 : Journées Doctorales en Informatique et Réseaux*, Belfort, France, february 2009.

Research reports

[22] Anne-Cécile Orgerie and Laurent Lefèvre, **"Energy-Efficient Advance Bandwidth Reservation for Bulk Data Transfers in Dedicated Networks"**, *Proceedings of the COST IC0804 action - second year*, pages 72-76, 2011.

[23] Anne-Cécile Orgerie and Laurent Lefèvre, **"A year in the life of a large-scale experimental distributed system: the Grid'5000 platform in 2008"**, *INRIA research report RR-7481*, 2010.

[24] Anne-Cécile Orgerie and Laurent Lefèvre, **"Greening the Clouds!"**, *Proceedings of the COST IC0804 action - first year*, pages 59-62, 2010.

[25] Anne-Cécile Orgerie and Laurent Lefèvre, **"A year in the life of a large-scale experimental distributed system: usage of the Grid'5000 platform in 2007"**, *INRIA research report RR-6965, 2009.*

References

[AAF+09] M. Allalouf, Y. Arbitman, M. Factor, R. Kat, K. Meth, and D. Naor. Storage modeling for power estimation. In *ACM Israeli Experimental Systems Conference (SYSTOR)*, 2009. 8, 42

[ACGP09] G. Anastasi, M. Conti, I. Giannetti, and A. Passarella. Design and Evaluation of a BitTorrent Proxy for Energy Saving. In *IEEE Symposium On Computers And Communications (ISCC)*, pages 116–121, 2009. 28, 30

[ADBF+05] S. Andreozzi, N. De Bortoli, S. Fantinel, A. Ghiselli, G. Rubini, G. Tortone, and M. Vistoli. GridICE: a monitoring service for Grid systems. *Future Generation Computer Systems*, 21(4):559–571, 2005. 31

[AFRR+10] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2010. 112

[AGP10] G. Anastasi, I. Giannetti, and A. Passarella. A BitTorrent Proxy for Green Internet File Sharing: Design and Experimental Evaluation. *IEEE Computer Communications*, 33(7):794–802, 2010. 28, 30

[AHC+09] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta. Somniloquy: Augmenting Network Interfaces To Reduce PC Energy Usage. In *USENIX Symposium On Networked Systems Design And Implementation (NSDI)*, pages 365–380, 2009. 26, 30

[AK08] G. Ananthanarayanan and R. Katz. Greening the switch. Technical report, EECS Department, University of California, Berkeley, 2008. 24, 30, 96

[AMAMM01] H. Aydi, P. Mejia-Alvarez, D. Mossé, and R. Melhem. Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 95–105, 2001. 14, 18

[AMSL04] M. Alonso, J. Miguel Martinez, V. Santonja, and P. Lopez. Reducing power consumption in interconnection networks by dynamically adjusting link width. In *International European Conference on Parallel and Distributed Computing (Euro-Par)*, pages 882–890, 2004. 27, 30

[ASG10] Y. Agarwal, S. Savage, and R. Gupta. SleepServer: A software-only approach for reducing the energy consumption of PCs within enterprise environments. In *USENIX Annual Technical Conference (ATC)*, 2010. 26, 30

[BAH+09] J. Baliga, R. Ayre, K. Hinton, W.V. Sorin, and R.S. Tucker. Energy Consumption in Optical IP Networks. *Journal of Lightwave Technology*, 27(13):2391–2403, 2009. 21, 24, 99

[BAHT09] J. Baliga, R. Ayre, K. Hinton, and R.S. Tucker. Architectures for energy-efficient IPTV networks. In *Conference on Optical Fiber Communication (OFC)*, 2009. 29, 30

[BAHT11] J. Baliga, R. Ayre, K. Hinton, and R. Tucker. Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport. *Proceedings of the IEEE*, 99(1):149–167, 2011. 111

[BAM10] T. Benson, A. Akella, and D. Maltz. Network traffic characteristics of data centers in the wild. In *Conference on Internet measurement (IMC)*, pages 267–280, 2010. 95, 110

[Bar05] L. A. Barroso. The Price of Performance. *ACM Press, Queue*, 3:48–53, September 2005. 5

[Bas99] S. Basagni. Distributed clustering for ad hoc networks. In *International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pages 310–315, 1999. 114

[BAS+08] J. Baliga, R. Ayre, W. Sorin, K. Hinton, and R. Tucker. Energy Consumption In Access Networks. In *Conference On Optical Fiber Communication/National Fiber Optic Engineers Conference (OFC/NFOEC)*, pages 1–3, 2008. 21, 24

[BBC+11] R. Bolla, R Bruschi, K. Christensen, F. Cucchietti, F. Davoli, and S. Singh. The Potential Impact of Green Technologies in Next Generation Wireline Networks - Is There Room for Energy Savings Optimization? *IEEE Communications*, 2011. xv, 20

[BBCL11] R. Bolla, R. Bruschi, A. Cianfrani, and M. Listanti. Enabling backbone networks to sleep. *IEEE Network*, 25(2):26–31, 2011. 25, 30

[BBDC11] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti. Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures. *IEEE Communications Surveys and Tutorials*, 13(2):223–244, 2011. xiii, 18, 19, 27, 28

[BC08] F. Blanquicet and K. Christensen. PAUSE Power Cycle: A New Backwards Compatible Method to Reduce Energy Use of Ethernet Switches. Technical report, Ethernet Alliance White Paper, 2008. 104

[BC09] J. Blackburn and K. Christensen. A Simulation Study of a New Green BitTorrent. In *International Workshop On Green Communications (GreenCom, IEEE ICC Workshop)*, pages 1–6, 2009. 27, 28, 30

[BCMR04] J.W. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. *IEEE/ACM Transactions on Networking*, 12(5):767–780, 2004. 48, 99

[BCN06] M. Bennett, K. Christensen, and B. Nordman. Improving the Energy Efficiency of Ethernet: Adaptive Link Rate Proposal. Ethernet Alliance White Paper, 2006. 101

[BCSS10] M. Balman, E. Chaniotakisy, A. Shoshani, and A. Sim. A Flexible Reservation Algorithm for Advance Network Provisioning. In *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–11, 2010. 47

[BDF+03a] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *ACM Symposium on Operating Systems Principles (SOSP)*, pages 164–177, 2003. 16, 18, 89

[BDF03b] L.-O. Burchard and M. Droste-Franke. Fault Tolerance in Networks with an Advance Reservation Service. In *International Workshop on Quality of Service (IWQoS)*, pages 215–228, 2003. 49

[BE09a] B. Bathula and J. Elmirghani. Energy Efficient Optical Burst Switched (OBS) Networks. In *GreenComm: International Workshop On Green Communications*, 2009. 20, 104

[BE09b] B. Bathula and J. Elmirghani. Green networks: Energy efficient design for optical networks. In *IFIP International Conference on Wireless and Optical Communications Networks*, 2009. 20, 104

[BH07] L.A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37, 2007. 11, 97

[BHT07] J. Baliga, K. Hinton, and R. Tucker. Energy Consumption of the Internet. In *Joint International Conference On Optical Internet And Australian Conference On Optical Fibre Technology (COIN-ACOFT)*, pages 1–3, 2007. 20, 24

[BJR94] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice-Hall International, Inc., 3rd edition, 1994. 13

[BKWW03] F. Bellosa, S. Kellner, M. Waitz, and A. Weissel. Event-Driven Energy Accounting for Dynamic Thermal Management. In *Workshop on Compilers and Operating Systems for Low Power (COLP)*, 2003. 40

[BLS05] L.-O. Burchard, B. Linnert, and J. Schneider. Rerouting strategies for networks with advance reservations. In *International Conference on e-Science and Grid Computing (E-Science)*, 2005. 49

[BMQ+07] G. Boss, P. Malladi, D. Quan, L. Legregni, and H. Hall. Cloud Computing. Technical report, IBM, 2007. 14

[BO09] M. Baldi and Y. Ofek. Time For A "Greener" Internet. In *International Workshop On Green Communications (GreenCom, IEEE ICC Workshop)*, 2009. 18, 24, 29, 30, 118

[Bur05] L.-O. Burchard. Networks with Advance Reservations: Applications, Architecture, and Performance. *Journal of Network and Systems Management*, 13(4):429–449, 2005. 48, 49, 99, 100, 101

[BYV+09] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, 25(6):599–616, 2009. 14

[CAT+01] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing energy and server resources in hosting centers. In *ACM symposium on Operating systems principles (SOSP)*, pages 103–116, 2001. 9, 12, 13, 14, 18

[CBBA10] A. Castagnetti, C. Belleudy, S. Bilavarn, and M. Auguin. Power Consumption Modeling for DVFS Exploitation. In *Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD)*, pages 579–586, 2010. 7

[CBK10] K. Chard, K. Bubendorfer, and P. Komisarczuk. High Occupancy Resource Allocation for Grid and Cloud systems, a Study with DRIVE. In *ACM International Symposium on High Performance Distributed Computing (HPDC)*, pages 73–84, 2010. 46, 47

[CCD+05] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In *IEEE/ACM International Workshop on Grid Computing (Grid)*, 2005. 32

[CCLM09] L. Chiaraviglio, D. Ciullo, E. Leonardi, and M. Mellia. How Much Can the Internet Be Greened? In *GreenComm: International Workshop On Green Communications*, 2009. 22, 24, 30, 104

[CD01] J. Chase and R. Doyle. Balance of Power: Energy Management for Server Clusters. In *Workshop on Hot Topics in Operating Systems (HotOS)*, 2001. 12, 14, 18

[CDCG+05] N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounie, P. Neyron, and O. Richard. A batch scheduler with high level components. In *Conference on Cluster computing and Grid (CCGrid)*, pages 776–783, 2005. 32, 61

[CEL+10] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, and E. Vittorini. An Energy Saving Routing Algorithm for a Green OSPF Protocol. In *IEEE Conference on Computer Communications (INFOCOM) Workshops*, 2010. 28, 30

[CFH+05] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Symposium on Networked Systems Design & Implementation (NSDI)*, pages 273–286, 2005. 16, 18

[CG05] L. Cherkasova and R. Gardner. Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. In *USENIX Annual Technical Conference (ATEC)*, 2005. 15, 18, 44

[CGNG04] K. Christensen, C. Gunaratne, B. Nordman, and A. George. The next frontier for communications networks: power management. *Computer Communications*, 27(18):1758–1770, 2004. 18, 19, 25, 26, 30, 101

[CHL+08] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 337–350, 2008. 14, 18, 40, 41

[Cis] Cisco data sheets. http://www.cisco.com. 99

[CLN11] S. Chaisiri, B. Lee, and D. Niyato. Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 2011. 47

[CM10] L. Chiaraviglio and I. Matta. GreenCoop: Cooperative Green Routing with Energy-efficient Servers. In *International Conference on Energy-Efficient Computing and Networking (e-Energy)*, 2010. 29, 30

[CMKR05] G. Chen, K. Malkowski, M. Kandemir, and P. Raghavan. Reducing Power with Performance Constraints for Parallel Sparse Applications. In *International Parallel and Distributed Processing Symposium (IPDPS)*, 2005. 14, 18

[CML+10] L. Chiaraviglio, M. Mellia, A. Lombardo, C. Panarello, and G. Schembra. Energy Saving and Network Performance: A Trade-Off Approach. In *International Conference On Energy-Efficient Computing And Networking*, 2010. 20, 127

[CMN08] L. Chiaraviglio, M. Mellia, and F. Neri. Energy-Aware Networks: Reducing Power Consumption by Switching Off Network Elements. Technical report, FEDERICA-Phosphorus tutorial and workshop (TNC), 2008. 25, 26, 30

[CMN09] L. Chiaraviglio, M. Mellia, and F. Neri. Energy-Aware Backbone Networks: A Case Study. In *International Workshop On Green Communications (GreenCom, IEEE ICC Workshop)*, 2009. 21, 24, 26, 30, 96

[CP11] X. Chen and C. Phillips. Dynamic Energy Management for IP-over-WDM Networks. In *PostGraduate Symposium on the Convergence of Telecommunications Networking and Broadcasting (PGNet)*, 2011. 29, 30

[CPB03] E. Carrera, E. Pinheiro, and R. Bianchini. Conserving disk energy in network servers. In *International conference on Supercomputing (ICS)*, pages 86–97, 2003. 11, 18

[CRH08] C. Castillo, G.N. Rouskas, and K. Harfoush. Efficient resource management using advance reservations for heterogeneous Grids. In *International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–12, 2008. 1, 45, 47

[CRN+10] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J.A. Maestro. IEEE 802.3az: the road to Energy Efficient Ethernet. *IEEE Communications Magazine*, 48(11):50–56, 2010. 26, 30

[CSB+08] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power Awareness in Network Design and Routing. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 457–465, 2008. 22, 27, 28, 30, 99

[DBB09] J. Ding, I. Balasingham, and P. Bouvry. Management of Overlay Networks: A Survey. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, pages 249–255, 2009. 102

[DCA+03] R. Doyle, J. Chase, O. Asad, W. Jin, and A. Vahdat. Model-based resource provisioning in a Web service utility. In *USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003. 12, 18

[DCDdAG+10] G. Da Costa, M. Dias de Assunção, J.-P. Gelas, Y. Georgiou, L. Lefèvre, A.-C. Orgerie, J.-M. Pierson, O. Richard, and A. Sayah. Multi-facet approach to reduce energy consumption in Clouds and Grids: the GREEN-NET framework. In *ACM International Conference on Energy-Efficient Computing and Networking (e-Energy)*, pages 95–104, 2010. 79

[DCGG+09] G. Da-Costa, J.-P. Gelas, Y. Georgiou, L. Lefèvre, A.-C. Orgerie, J.-M. Pierson, O. Richard, and K. Sharma. The GREEN-NET Framework: Energy Efficiency in Large Scale Distributed Systems. In *Workshop on High-Performance, Power-Aware Computing (HPPAC, IPDPS workshop)*, 2009. 42, 87

[DD06] H.G. Dietz and W.R. Dieter. Compiler and runtime support for predictive control of power and cooling. In *Parallel and Distributed Processing Symposium (IPDPS)*, 2006. 8

[DdAGLO10] M. Dias de Assunção, J.-P. Gelas, L. Lefèvre, and A.-C. Orgerie. The Green Grid'5000: Instrumenting a Grid with Energy Sensors. In *International Workshop on Distributed Cooperative Laboratories: Instrumenting the Grid (INGRID)*, 2010. 35, 44

[DdAOL10] M. Dias de Assunção, A.-C. Orgerie, and L. Lefèvre. An Analysis of Power Consumption Logs from a Monitored Grid Site. In *IEEE/ACM International Conference on Green Computing and Communications (GreenCom)*, pages 61–68, 2010. 64

[dLJ06] P. de Langen and B. Juurlink. Leakage-aware multiprocessor scheduling for low power. In *International conference on Parallel and Distributed Processing (IPDPS)*, 2006. 13, 18

[Eco11] From Windows 95 to Windows 7. EcoInfo report, 2011. xiii, 10, 18

[EHW08] B. Eckart, X. He, and Q. Wu. Performance Adaptive UDP for High-Speed Bulk Data Transfer over Dedicated Links. In *IEEE International Symposium On Parallel And Distributed Processing (IPDPS)*, pages 1–10, 2008. 48

[ET05] E. Elmroth and J. Tordsson. A grid resource broker supporting advance reservations and benchmark-based resource selection. In *Workshop on Applied Parallel Computing (PARA)*, pages 1061–1070, 2005. 46, 47

[EYD07] D. Ersoz, M.S. Yousif, and C.R. Das. Characterizing Network Traffic in a Cluster-based, Multi-tier Data Center. In *International Conference on Distributed Computing Systems (ICDCS)*, 2007. 110

[Far07] C. Farivar. Google's Next-Gen of Sneakernet, 2007. 95

[FCG+06] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, and P. McDonald. When TCP Breaks: Delay- and Disruption- Tolerant Networking. *IEEE Internet Computing*, 10(4):72–78, 2006. 104

[FK98] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., 1998. 1

[FKL+99] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *International Workshop on Quality of Service (IWQoS)*, pages 27–36, 1999. 46

[FLR+09] S. Figuerola, M. Lemay, V. Reijs, M. Savoie, and B. St. Arnaud. Converged Optical Network Infrastructures in Support of Future Internet and Grid Services Using IaaS to Reduce GHG Emissions. *Journal of Lightwave Technology*, 27(12):1941–1946, 2009. 12, 18

[FPK+05] V. Freeh, F. Pan, N. Kappiah, D. Lowenthal, and R. Springer. Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster. In *IEEE International Parallel and Distributed Processing Symposium, (IPDPS)*, 2005. 13, 18

[FRM+10] E. Feller, L. Rilling, C. Morin, R. Lottiaux, and D. Leprince. Snooze: A Scalable, Fault-Tolerant and Distributed Consolidation Manager for Large-Scale Clusters. In *IEEE/ACM International Conference on Green Computing and Communications (GreenCom)*, pages 125–132, 2010. 16, 18

[FS09] W.-C. Feng and T. Scogland. The Green500 List: Year One. In *Workshop on High-Performance, Power-Aware Computing (HPPAC, IPDPS workshop)*, 2009. 6, 8

[FVG+08] J. Fontan, T. Vazquez, L. Gonzalez, Ruben S. Montero, and I. Llorente. OpenNebula: The Open Source Virtual Machine Manager for Cluster Computing. In *Open Source Grid and Cluster Software Conference – Book of Abstracts*, 2008. 15, 18

[FWB07] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ACM International symposium on Computer architecture (ISCA)*, pages 13–23, 2007. xv, 7, 9, 14, 16, 18, 40, 41

[GCN05] C. Gunaratne, K. Christensen, and B. Nordman. Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed. *International Journal of Network Management*, 15(5):297–310, 2005. 8, 9, 18, 19, 26, 27, 30, 42, 87

[GCS06] C. Gunaratne, K. Christensen, and S. Suen. Ethernet Adaptive Link Rate (ALR): Analysis Of A Buffer Threshold Policy. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, 2006. 27, 30, 42, 96, 107

[GFC05] R. Ge, X. Feng, and K. Cameron. Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters. In *ACM/IEEE Conference on Supercomputing (SC)*, 2005. 7

[GG03] Y. Gu and R. Grossman. SABUL: A Transport Protocol for Grid Computing. *Journal of Grid Computing*, 1(4):377–386, 2003. 48

[GGS04] M. Gupta, S. Grover, and S. Singh. A Feasibility Study for Power Management in LAN Switches. In *IEEE International Conference On Network Protocols (ICNP)*, pages 361–371, 2004. 26, 30

[GHMP08] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel. The Cost of a Cloud: Research Problems in Data Center Networks. *ACM SIGCOMM Computer Communication Review*, 39:68–73, 2008. 12

[Gin05] G. Ginis. Low-Power Modes for ADSL2 and ADSL2+. White paper - Broadband Communications Group, Texas Instruments, 2005. 27

[GIYC06] L. Grit, D. Irwin, A. Yumerefendi, and J. Chase. Virtual Machine Hosting for Networked Clusters: Building the Foundations for "Autonomic" Orchestration. In *International Workshop on Virtualization Technology in Distributed Computing (VTDC)*, 2006. 16

[GLB08] C. Guok, J. Lee, and K. Berket. Improving the bulk data transfer experience. *International Journal of Internet Protocol Technology*, 3(1):46–53, 2008. 48

[GNMP09] M. Guenach, C. Nuzman, J. Maes, and M. Peeters. Trading off rate and power consumption in DSL systems. In *International Workshop On Green Communications (GreenCom, IEEE GLOBECOM Workshop)*, 2009. 23

[Gre07] The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE. Green Grid report, 2007. 7

[Gre10a] Unused Servers Survey Results Analysis. Green Grid report, 2010. 5

[Gre10b] Make IT Green: Cloud Computing and its Contribution to Climate Change. Greenpeace report, 2010. 2, 81

[Gre11] How dirty is your data? Greenpeace report, 2011. xiii, 2, 5, 12, 18

[GS03] M. Gupta and S. Singh. Greening of the Internet. In *SIGCOM Conference on Applications, technologies, architectures, and protocols for computer communications*, pages 19–26, 2003. 18, 24, 25, 26, 30, 42, 104

[GS07a] M. Gupta and S. Singh. Dynamic Ethernet Link Shutdown for Energy Conservation on Ethernet Links. In *IEEE International Conference On Communications (ICC)*, pages 6156–6161, 2007. 25, 26, 30

[GS07b] M. Gupta and S. Singh. Using Low-Power Modes for Energy Conservation in Ethernet LANs. In *IEEE International Conference On Computer Communications (INFOCOM)*, pages 2451–2455, 2007. 24, 25, 30, 108

[GS09] E. Gelenbe and S. Silvestri. *Optimisation of Power Consumption in Wired Packet Networks*, volume 22, pages 717–729. Springer, 2009. 28, 29, 30

[Ham09] S. Hamm. With Sun, IBM Aims for Cloud Computing Heights, 2009. 14, 17

[Hay08] B. Hayes. Cloud computing. *Communication of the ACM*, 51(7):9–11, 2008. 16

[HCP09] H. Hlavacs, G. Da Costa, and J.-M. Pierson. Energy Consumption of Residential and Professional Switches. In *IEEE International Conference on Computational Science and Engineering (CSE)*, pages 240–246, 2009. 23, 42, 96

[HFPJ10] M. Hasan, F. Farahmand, A. Patel, and J. Jue. Traffic Grooming in Green Optical Networks. In *IEEE International Conference on Communications (ICC)*, 2010. 22

[HHX+05] M. Hefeeda, A. Habib, D. Xu, B. Bhargava, and B. Botev. CollectCast: A peer-to-peer service for media streaming. *Multimedia Systems*, 11(1):68–81, 2005. 48, 99

[HJ08] X. He and Y. Jia. Procrastination Scheduling for Fixed-Priority Tasks with Preemption Thresholds. In *IFIP International Conference on Network and Parallel Computing (NPC)*, pages 255–265, 2008. 14, 18

[HJL09] M. Hayenga, N.E. Jerger, and M. Lipasti. SCARAB: A single cycle adaptive routing and bufferless network. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 244–254, 2009. 29, 30

[HLM06] F. Hermenier, N. Loriant, and J.-M. Menaud. Power Management in Grid Computing with Xen. In *Frontiers of High Performance Computing and Networking ISPA 2006 Workshops*, number 4331 in LNCS, pages 407–416, 2006. 15, 18

[HLYD02] E. He, J. Leigh, O. Yu, and T. Defanti. Reliable Blast UDP: Predictable High Performance Bulk Data Transfer. In *IEEE International Conference On Cluster Computing*, pages 317–324, 2002. 48

[HNO+09] T. Hirofuchi, H. Nakada, H. Ogawa, S. Itoh, and S. Sekiguchi. A live storage migration mechanism over WAN and its performance evaluation. In *ACM International workshop on Virtualization technologies in distributed computing (VTDC)*, pages 67–74, 2009. 93

[HSL02] X. He, X. Sun, and G. Von Laszewski. A QoS Guided Scheduling Algorithm. In *International Workshop on Grid and Cooperative Computing (GCC)*, pages 442–450, 2002. 47

[HSM+10] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: saving energy in data center networks. In *USENIX conference on Networked systems design and implementation (NSDI)*, 2010. 112

[HSMR09] S. Harizopoulos, M. Shah, J. Meza, and P. Ranganathan. Energy Efficiency: The New Holy Grail of Data Management Systems Research. In *Conference on Innovative Data Systems Research (CIDR)*, 2009. 16

[HSRJ08] A. Hylick, R. Sohan, A. Rice, and B. Jones. An Analysis of Hard Drive Energy Consumption. In *IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems (MASCOTS)*, 2008. 42

[HWX+11] C. Hu, C. Wu, W. Xiong, B. Wang, J. Wu, and M. Jiang. On the design of green reconfigurable router toward energy efficient Internet. *IEEE Communications Magazine*, 49(6):83–87, 2011. 25, 30

[IC98] L. Irish, , and K. Christensen. A "Green TCP/IP" to Reduce Electricity Consumed by Computers. In *IEEE Southeast Conference*, pages 302–305, 1998. 26, 27, 28, 30

[IDE+06] A. Iosup, C. Dumitrescu, D. Epema, Hui Li, and L. Wolters. How are Real Grids Used? The Analysis of Four Grid Traces and Its Implications. In *IEEE/ACM International Conference on Grid Computing*, 2006. 16

[Idz09] F. Idzikowski. Power consumption of network elements in IP over WDM networks. Technical report, TKN Group, 2009. 99

[IOR+10] F. Idzikowski, S. Orlowski, C. Raack, H. Woesner, and A. Wolisz. Saving energy in IP-over-WDM networks by switching off line cards in low-demand scenarios. In *Optical Network Design and Modelling (ONDM)*, 2010. 27, 29

[ITU] ITU. ITU Recommendation G.992.3: Asymmetric digital subscriber line transceivers 2 (ADSL2). 27

[JBB92] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323, 1992. 48

[JCN08] M. Jimeno, K. Christensen, and B. Nordman. A Network Connection Proxy to Enable Hosts to Sleep and Save Energy. In *IEEE International Performance, Computing and Communications Conference (IPCCC)*, pages 101–110, 2008. 26, 30, 42

[JG06] R. Jejurikar and R. Gupta. Energy aware task scheduling with task synchronization for embedded real-time systems. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1024–1037, 2006. 14, 18

[JJH+09] G. Jung, K. Joshi, M. Hiltunen, R. Schlichting, and C. Pu. A Cost-Sensitive Adaptation Engine for Server Consolidation of Multitier Applications. In *ACM/IFIP/USENIX International Conference on Middleware (Middleware)*, pages 1–20, 2009. 12, 17, 18

[JLRS08] E.-S. Jung, Y. Li, S. Ranka, and S. Sahni. An Evaluation of In-Advance Bandwidth Scheduling Algorithms for Connection-Oriented Networks. In *International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pages 133–138, 2008. 48, 49, 100

[Kal60] R. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME – Journal of Basic Engineering*, 82(Series D):35–45, 1960. 13

[KBK07] K. Kim, R. Buyya, and J. Kim. Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters. In *IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, pages 541–548, 2007. 39

[KBK10] D. Kliazovich, P. Bouvry, and S. Khan. GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 2010. 123

[KCH09] E. Kalyvianaki, T. Charalambous, and S. Hand. Self-Adaptive and Self-Configured CPU Resource Provisioning for Virtualized Servers Using Kalman Filters. In *ACM International Conference on Autonomic Computing (ICAC)*, pages 117–126, 2009. 12, 13

[KFL05] N. Kappiah, V. Freeh, and D. Lowenthal. Just In Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs. In *ACM/IEEE Conference on Supercomputing (SC)*, 2005. 13, 18

[KHR02] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *ACM SIGCOMM conference on Data communication*, pages 89–102, 2002. 101

[KKH+08] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang. Power and Performance Management of Virtualized Computing Environments via Lookahead Control. In *IEEE International Conference on Autonomic Computing (ICAC)*, pages 3–12, 2008. 12, 13, 18

[KLY+05] E. Kim, G. Link, K. Yum, N. Vijaykrishnan, M. Kandemir, M. Irwin, and C. Das. A holistic approach to designing energy-efficient cluster interconnects. *IEEE Transactions on Computers*, 54(6):660–671, 2005. 21

[KN01] M. Kim and B. Noble. Mobile Network Estimation. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 298–309, 2001. 13

[Kno08] K. Knopper. Got the Power: Reality and myth in the quest for green computing. *Linux Pro Magazine*, pages 31–35, 2008. 43

[Koo08] J. Koomey. Worldwide electricity used in data centers. *Environmental Research Letters*, 3(3), 2008. 1

[KR10] J. Kang and S. Ranka. Dynamic slack allocation algorithms for energy minimization on parallel machines. *Journal of Parallel and Distributed Computing*, 70:417–430, 2010. 14

[Lea06] C. Leangsuksun et al. IPMI-based Efficient Notification Framework for Large Scale Cluster Computing. In *International Symposium on Cluster Computing and the Grid Workshops (CCGrid)*, 2006. 9, 18, 56

[Len10] R. Lent. Simulating the power consumption of computer networks. In *Workshop on Computer Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD)*, pages 96–100, 2010. 123

[Les10] Server power measurements. LessWatts report, 2010. xiii, 10, 11, 18

[LFL06] M. Lim, V. Freeh, and D. Lowenthal. Adaptive, Transparent Frequency and Voltage Scaling of Communication Phases in MPI Programs. In *ACM/IEEE Conference on Supercomputing (SC)*, 2006. 14, 18

[LHC] http://lcg.web.cern.ch/lcg/public/default.htm. 95

[LKWG11] C. Lange, D. Kosiankowski, R. Weidmann, and A. Gladisch. Energy Consumption of Telecommunication Networks and Related Improvement Options. *IEEE Journal of Selected Topics in Quantum Electronics*, 17(2):285–295, 2011. 21

[LO09] L. Lefèvre and A.-C. Orgerie. Towards Energy Aware Reservation Infrastructure for Large-Scale Experimental Distributed Systems. *Parallel Processing Letters*, 19(3):419–433, 2009. 61

[LO10] L. Lefèvre and A.-C. Orgerie. Designing and Evaluating an Energy Efficient Cloud. *The Journal of SuperComputing*, 51(3):352–373, 2010. 81, 91

[LRS09] Y. Li, S. Ranka, and S. Sahni. In-advance path reservation for file transfers In e-Science applications. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 176–181, 2009. 49

[LSRS09] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram. Delay tolerant bulk data transfers on the Internet. In *International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 229–238, 2009. 48

[LW08] Y. Lin and Q. Wu. On Design of Bandwidth Scheduling Algorithms for Multiple Data Transfers in Dedicated Networks. In *Symposium On Architectures For Networking And Communications Systems (ANCS)*, pages 151–160, 2008. 1, 48, 100, 118

[LW09] Y. Lin and Q. Wu. Path Computation with Variable Bandwidth for Bulk Data Transfer in High-Performance Networks. In *High-Speed Networks Workshop (HSN)*, 2009. 48

[LWRZ08] Y. Lin, Q. Wu, N. Rao, and M. Zhu. On Design of Scheduling Algorithms for Advance Bandwidth Reservation in Dedicated Networks. In *GreenCom (IEEE INFOCOM Workshop)*, pages 1–6, 2008. 100

[LZLH09] J. Liu, F. Zhao, X. Liu, and W. He. Challenges Towards Elastic Power Management in Internet Data Centers. In *IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 65–72, 2009. 17, 18

[MAD+05] A. Mcgough, A. Afzal, J. Darlington, N. Furmento, A. Mayer, and L. Young. Making the Grid Predictable through Reservations and Performance Modelling. *The Computer Journal*, 48:358–368, 2005. 46

[Maj09] S. Majumdar. The "Any-Schedulability" Criterion for Providing QoS Guarantees through Advance Reservation Requests. In *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 490–495, 2009. 46, 47

[MB06] A. Merkel and F. Bellosa. Balancing power consumption in multiprocessor systems. *ACM SIGOPS Operating Systems Review*, 40(4):403–414, 2006. 7, 12, 14, 18

[McK09] McKinsey & Company. Revolutionizing Data Center Efficiency. Technical report, http://www.mckinsey.com/clientservice/bto/pointofview/Revolutionizing.asp, 2009. 61

[MCRS05] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making Scheduling "cool": Temperature-Aware Workload Placement in Data Centers. In *USENIX Annual Technical Conference (ATEC)*, 2005. 16

[Mei08] T. Meinl. Advance Reservation of Grid Resources via Real Options. In *IEEE Joint Conference on E-Commerce Technology (CEC) and Enterprise Computing, E-Commerce and E-Services (EEE)*, pages 3–10, 2008. 47

[MLVH+02] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar. Critical power slope: understanding the runtime effects of frequency scaling. In *ACM International conference on Supercomputing (ICS)*, pages 35–44, 2002. 16, 18

[Moo98] G. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86:82–85, 1998. 5

[MR95] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6:161–179, 1995. 116

[MRZ+03] R. Mishra, N. Rastogi, D. Zhu, D. Mossé, and R. Melhem. Energy aware scheduling for distributed real-time systems. In *International Symposium on Parallel and Distributed Processing (IPDPS)*, 2003. 14, 18

[MSBR09a] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. A Power Benchmarking Framework for Network Devices. *Networking*, 5550:795–808, 2009. 23, 24, 97, 99, 110

[MSBR09b] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. Energy Aware Network Operations. In *INFOCOM Workshops*, 2009. 99, 109, 110

[NB07] T. Nguyen and A. Black. Preliminary Study on Power Consumption of Typical Home Network Devices. Technical report, CAIA, 2007. 23

[NC10] B. Nordman and K. Christensen. Proxying: The Next Step in Reducing IT Energy Use. *IEEE Computer*, 43(1):91–93, 2010. 25, 87

[NPI+08] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing Network Energy Consumption via Sleeping qnd Rate-Adaptation. In *USENIX Symposium On Networked Systems Design & Implementation (NSDI)*, pages 323–336, 2008. 26, 27, 28, 30, 104

[NS07] R. Nathuji and K. Schwan. VirtualPower: coordinated power management in virtualized enterprise systems. In *ACM SIGOPS symposium on Operating systems principles (SOSP)*, pages 265–278, 2007. 14, 15, 18, 43, 85

[NWC+08] D. Nurmi, R. Wolski, C. Crzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems. Technical report 2008-10, Department of Computer Science, University of California, Santa Barbara, 2008. 15, 18

[Odl03] A. Odlyzko. Data Networks are Lightly Utilized, and will Stay that Way. *Review of Network Economics*, 2, 2003. 19, 25

[OL10] A.-C. Orgerie and L. Lefèvre. A year in the life of a large-scale experimental distributed system: usage of the Grid'5000 platform in 2008. Research Report 7481, INRIA, 2010. 61

[OL11a] A.-C. Orgerie and L. Lefèvre. Energy-Efficient Bandwidth Reservation for Bulk Data Transfers in Dedicated Wired Networks. *Journal of Supercomputing, Special Issue on Green Networks*, 2011. 95

[OL11b] A.-C. Orgerie and L. Lefèvre. *Energy-Efficient Reservation Infrastructure for Grids, Clouds and Networks*, chapter Energy Aware Distributed Computing Systems, Wiley Series on Parallel and Distributed Computing. John Wiley & Sons, 2011. 45

[OL11c] A.-C. Orgerie and L. Lefèvre. ERIDIS: Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems. *Parallel Processing Letters*, 21:133–154, 2011. 45

[OLG08a] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas. Chasing Gaps between Bursts : Towards Energy Efficient Large Scale Experimental Grids. In *International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pages 381–389, 2008. 61

[OLG08b] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas. Save Watts in your Grid: Green Strategies for Energy-Aware Framework in Large Scale Distributed Systems. In *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 171–178, 2008. 61, 85

[OLG10] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas. Demystifying Energy Consumption in Grids and Clouds. In *Work in Progress in Green Computing, IGCC Workshop*, 2010. 15, 56

[OLGLLP11] A.-C. Orgerie, L. Lefèvre, I. Guérin-Lassous, and D. Lopez Pacheco. ECOFEN: an End-to-end energy Cost mOdel and simulator For Evaluating power consumption in large-scale Networks. In *Sustainet: International Workshop on Sustainable Internet and Internet for Sustainability (in conjunction with WoWMoM)*, 2011. 123

[Opp10] P. Oppenheimer. *Top-Down Network Design*. Cisco Press, 2010. 112

[PAB+05] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *Conference on Internet Measurement (IMC)*, 2005. 95

[PBK+06] C. Palansuriya, M. Buchli, K. Kavoussanakis, A. Patil, C. Tziouvaras, A. Trew, A. Simpson, and R. Baxter. End-to-End Bandwidth Allocation and Reservation for Grid applications. In *Conference on Broadband Communications, Networks and Systems (BROADNETS)*, pages 1–9, 2006. 1, 45

[PCGL07] M. Patterson, D. Costello, P. Grimm, and M. Loeffler. Data center TCO: a comparison of high-density and low-density spaces. In *Thermal Challenges in Next Generation Electronic Systems (THERMES)*, 2007. 16

[PSA97] D.-T. Peng, K. Shin, and T. Abdelzaher. Assignment and scheduling communicating periodic tasks in distributed real-time systems. *IEEE Transactions on Software Engineering*, 23(12):745–758, 1997. 47

[PSBG02] C. Patel, R. Sharma, C. Bash, and S. Graupner. Energy Aware Grid: Global Workload Placement based on Energy Efficiency. Technical report, HP Laboratories, 2002. xiii, 12, 13, 14, 16, 18

[PV10] M. Pawlish and A.S. Varde. Free cooling: A paradigm shift in data centers. In *International Conference on Information and Automation for Sustainability (ICIAFS)*, pages 347–352, 2010. 12

[PW10] R. Prodan and M. Wieczorek. Negotiation-Based Scheduling of Scientific Grid Workflows through Advance Reservations. *Journal of Grid Computing*, 8:493–510, 2010. 46

[PZSJ09] A. Patel, Y. Zhu, Q. She, and J. Jue. Routing and Scheduling for Time-Shift Advance Reservation. In *Conference on Computer Communications and Networks (ICCCN)*, pages 1–6, 2009. 48, 49

[RCRM11] P. Reviriego, K. Christensen, J. Rabanillo, and J.A. Maestro. An Initial Evaluation of Energy Efficient Ethernet. *IEEE Communications Letters*, 15(5):578–580, 2011. 26

[Rei94] W. Reinhardt. Advance Reservation of Network Resources for Multimedia Applications. In *International Workshop on Multimedia (IWAKA)*, pages 23–33, 1994. 49

[RGM09] J.C Restrepo, C. Gruber, and C. Machuca. Energy Profile Aware Routing. In *IEEE International Conference on Communications (ICC Workshops)*, pages 1–5, 2009. 24, 26, 27, 30

[RL03] K. Rajamani and C. Lefurgy. On evaluating request-distribution schemes for saving energy in server clusters. In *IEEE International Symposium on Performance Analysis of Systems and Software*, pages 111–122, 2003. 43

[RMHL10] P. Reviriego, J. Maestro, J. Hernandndez, and D. Larrabeiti. Burst Transmission for Energy-Efficient Ethernet. *IEEE Internet Computing*, 14(4):50–57, 2010. xiii, 26

[Rob09] L. Roberts. A radical new router. *IEEE Spectrum*, 46(7):34–39, 2009. 25, 29, 30

[RRX09] K. Rajah, S. Ranka, and Ye Xia. Advance Reservations and Scheduling for Bulk Transfers in Research Networks. *Transactions on Parallel and Distributed Systems*, 20(11):1682–1697, 2009. 49, 100

[RSRK07] S. Rivoire, M. Shah, P. Ranganathan, and C. Kozyrakis. JouleSort: a balanced energy-efficiency benchmark. In *ACM SIGMOD international conference on Management of data*, pages 365–376, 2007. 8

[RV11] Z. Raza and D. Vidyarthi. A Computational Grid Scheduling Model to Maximize Reliability Using Modified GA. *International Journal of Grid and High Performance Computing (IJGHPC)*, 3:1–20, 2011. 47

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, 2001. 49, 101, 118

[SBA+07] K. Soramki, M. Bech, J. Arnold, R. Glass, and W. Beyeler. The Topology of Interbank Payment Flows. *Physica A: Statistical Mechanics and Its Applications*, 379(1):317–333, 2007. 95, 111

[SBB04] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: a cooperative bulk data transfer protocol. In *IEEE International Conference On Computer Communications (INFOCOM)*, pages 941–951, 2004. 48

[SBP+05] R. Sharma, C. Bash, C. Patel, R. Friedrich, and J. Chase. Balance of Power: Dynamic Thermal Management for Internet Data Centers. *IEEE Internet Computing*, 9(1):42–49, 2005. 12, 14, 18

[SEP05] V. Soteriou, N. Eisley, and L.-S. Peh. Software-Directed Power-Aware Interconnection Networks. In *International Conference On Compilers, Architectures And Synthesis For Embedded Systems (CASES)*, pages 274–285, 2005. 21

[SF10a] M. Siddiqui and T. Fahringer, editors. *Grid Resource Management - On-demand Provisioning, Advance Reservation, and Capacity Planning of Grid Resources.* Springer, 2010. 1, 46

[SF10b] B. Subramaniam and W.-C. Feng. Statistical Power and Performance Modeling for Optimizing the Energy Efficiency of Scientific Computing. In *IEEE/ACM International Conference on Green Computing and Communications (GreenCom)*, pages 139–146, 2010. 7

[SGRO+08] K. Sabhanatarajan, A. Gordon-Ross, M. Oden, M. Navada, and A. George. Smart-NICs: Power Proxying For Reduced Power Consumption In Network Edge Devices. In *IEEE Computer Society Annual Symposium On VLSI (ISVLSI)*, pages 75–80, 2008. 26, 30

[Sil08] Data Center Energy Forecast. Silicon Valley Leadership Group White Paper, 2008. 16

[Sim10] K. M. Sim. Grid Resource Negotiation: Survey and New Directions. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(3):245–257, 2010. 47

[SK04] D. Shin and J. Kim. Dynamic voltage scaling of periodic and aperiodic tasks in priority-driven systems. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 653–658, 2004. 14, 18

[SKB98] A. Schill, S. Kühn, and F. Breiter. Design and evaluation of an advance reservation protocol on top of RSVP. In *International Conference on Broadband Communications (BC)*, pages 23–40, 1998. 48, 49

[SKD07] G. Singh, C. Kesselman, and E Deelman. A provisioning model and its comparison with best-effort for performance-cost optimization in Grids. In *International symposium on High performance distributed computing (HPDC)*, pages 117–126, 2007. 46

[SKWM01] S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel. An accurate and fine grain instruction-level energy model supporting software optimizations. In *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2001. 40

[SKZ08] S. Srikantaiah, A. Kansal, and F. Zhao. Energy Aware Consolidation for Cloud Computing. In *Workshop on Power Aware Computing and Systems (HotPower)*, 2008. 12, 15, 16, 18, 43, 81

[SLB07] J. Stoess, C. Lang, and F. Bellosa. Energy management for hypervisor-based virtual machines. In *USENIX Annual Technical Conference (ATC)*, pages 1–14, 2007. 14, 18, 43

[Sma08] SMART 2020: Enabling the low carbon economy in the Information Age. report by The Climate Group, 2008. 2

[SMLF09] B. Sotomayor, R.S. Montero, I.M. Llorente, and I. Foster. Resource Leasing and the Art of Suspending Virtual Machines. In *Conference on High Performance Computing and Communications (HPCC)*, pages 59–68, 2009. 1, 45, 47

[SP03] V. Soteriou and L.-S. Peh. Dynamic Power Management for Power Optimization of Interconnection Networks Using On/Off Links. In *Symposium On High Performance Interconnects*, 2003. 26, 30, 104

[SPJ06] L. Shang, L.-S. Peh, and N. Jha. PowerHerd: a distributed scheme for dynamically satisfying peak-power constraints in interconnection networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(1):92–110, 2006. 21, 22, 24, 27, 29, 30

[SRH05] D. Snowdon, S. Ruocco, and G. Heiser. Power Management and Dynamic Voltage Scaling: Myths and Facts. In *Workshop on Power Aware Real-time Computing*, 2005. 8, 9, 14, 16, 18, 96

[SRR+07] S. Sahni, N. Rao, S. Ranka, Y. Li, E.-S. Jung, and N. Kamath. Bandwidth Scheduling and Path Computation Algorithms for Connection-Oriented Networks. In *International Conference on Networking (ICN)*, 2007. 48, 100

[SSvdBZ09] S. Subramanyam, R. Smith, P. van den Bogaard, and A. Zhang. Deploying Web 2.0 Applications on Sun Servers and the Opensolaris Operating System. Sun BluePrints 820-7729-10, Sun Microsystems, 2009. 16

[Ste98] J. Steele. ACPI thermal sensing and control in the PC. In *Wescon*, 1998. 10, 18

[SV09] T. Singh and P. Vara. Smart Metering the Clouds. *IEEE International Workshops on Enabling Technologies*, pages 66–71, 2009. 17, 31

[SVF06] M. Siddiqui, A. Villazon, and T. Fahringer. Grid capacity planning with negotiation-based advance reservation for optimized QoS. In *ACM/IEEE conference on Supercomputing (SC)*, 2006. 46, 47

[SVP+10] A. Silvestri, A. Valenti, S. Pompei, F. Matera, A. Cianfrani, and A. Coiro. Energy saving in optical transport networks exploiting transmission properties and wavelength path optimization. *Optical Switching and Networking*, 2010. 104

[SWHK08] M. Steinder, I. Whalley, J.E. Hanson, and J.O. Kephart. Coordinated management of power usage and runtime performance. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 387–394, 2008. 29, 30, 39, 40, 41

[TAG+11] A. Tzanakaki, M. Anastasopoulos, K. Georgakilas, J. Buysse, M. De Leenheer, C. Develder, S. Peng, R. Nejabati, E. Escalona, D. Simeonidou, N. Ciulli, G. Landi, M. Brogle, A. Manfredi, E. Lopez, J. Riera, J. Garcia-Espin, P. Donaldio, G. Parladori, and J. Jimenez. Energy Efficiency in integrated IT and Optical Network Infrastructures: The GEYSERS approach. In *Workshop on Green*

Communications and Networking (in conjunction with INFOCOM), 2011. 29, 30

[TBL09] R. Talaber, T. Brey, and L. Lamers. Using Virtualization to Improve Data Center Efficiency. Technical report, The Green Grid, 2009. 14, 15, 16, 18, 43

[TCH+08] J. Torres, D. Carrera, K. Hogan, R. Gavalda, V. Beltran, and N. Poggi. Reducing wasted resources to help achieve green data centers. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–8, 2008. 15, 18, 44

[TDG+06] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P. Y. Wang. Seamless live migration of virtual machines over the MAN/WAN. *Future Generation Computer Systems*, 22(8):901–907, 2006. 15, 16, 18, 84

[TF04] H.-L. Truong and T. Fahringer. SCALEA-G: A unified monitoring and performance analysis system for the grid. *Scientific Programming*, 12(4):225–237, 2004. 31

[TGV07] Q. Tang, S. Gupta, and G. Varsamopoulos. Thermal-Aware Task Scheduling for Data Centers through Minimizing Heat Recirculation. In *IEEE International Conference on Cluster Computing (Cluster)*, pages 129–138, 2007. 67

[THK+10] Y. Tanimura, K. Hidetaka, T. Kudoh, I. Kojima, and Y. Tanaka. A distributed storage system allowing application users to reserve I/O performance in advance for achieving SLA. In *IEEE/ACM International Conference on Grid Computing (GRID)*, pages 193–200, 2010. 47

[TMM06] M. Tatezono, N. Maruyama, and S. Matsuoka. Making Wide-Area, Multi-site MPI Feasible Using Xen VM. In *Workshop on Frontiers of High Performance Computing and Networking (in conjunction with ISPA)*, volume 4331 of *LNCS*, pages 387–396. Springer, 2006. 15

[Tuc10a] R. Tucker. Green Optical Communications - Part I: Energy Limitations in Transport. *IEEE Journal of Selected Topics in Quantum Electronics, Special Issue on Green Photonics*, 2010. 96

[Tuc10b] R. Tucker. Green Optical Communications - Part II: Energy Limitations in Networks. *IEEE Journal of Selected Topics in Quantum Electronics, Special Issue on Green Photonics*, 2010. 23, 29, 30

[USC+08] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood. Agile Dynamic Provisioning of Multi-tier Internet Applications. *ACM Transactions on Autonomous and Adaptive Systems*, 3(1):1–39, 2008. 12, 18

[VAN08a] A. Verma, P. Ahuja, and A. Neogi. pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems. In *ACM/IFIP/USENIX International Middleware Conference (Middleware)*, pages 243–264, 2008. 12, 17, 18

[VAN08b] A. Verma, P. Ahuja, and A. Neogi. Power-aware dynamic placement of HPC applications. In *ACM International conference on Supercomputing (ICS)*, pages 175–184, 2008. 42

[VBG09] G. Varsamopoulos, A. Banerjee, and S. Gupta. Energy Efficiency of Thermal-Aware Job Scheduling Algorithms under Various Cooling Models. In *Contemporary Computing*, volume 40 of *Communications in Computer and Information Science*, pages 568–580, 2009. 67

[VBVB09] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya. Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation. Technical report, Clouds Laboratory, University of Melbourne, Australia, 2009. 16, 18, 84

[VHDGV+10] W. Van Heddeghem, M. De Groote, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester. Energy-efficiency in telecommunications networks: Link-by-link versus end-to-end grooming. In *Conference on Optical Network Design and Modeling (ONDM)*, 2010. 29, 30

[Wer08] P. Werstein. An Experimental Network Proxy for Power Managed End Nodes. In *International Conference onAdvanced Information Networking and Applications Workshops*, pages 668–674, 2008. 87

[WKB+08] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford. Virtual routers on the move: live router migration as a network-management primitive. *ACM SIGCOMM Computer Communication Review*, 38(4):231242, 2008. 29, 30

[WPM02] H.-S. Wang, L.-S. Peh, and S. Malik. A Power Model for Routers: Modeling Alpha 21364 and InfiniBand Routers. *Symposium on High-Performance Interconnects*, page 21, 2002. 22, 23, 24, 96

[WS04] B. Wang and S. Singh. Computational Energy Cost of TCP. In *IEEE International Conference On Computer Communications (INFOCOM)*, pages 206–216, 2004. 28, 30

[WTK+08] L. Wang, J. Tao, M. Kunze, A. Castellanos, D. Kramer, and W. Karl. Scientific Cloud Computing: Early Definition and Experience. In *IEEE International Conference on High Performance Computing and Communications (HPCC)*, pages 825–830, 2008. 1

[WvLDW10] L. Wang, G. von Laszewski, J. Dayal, and F. Wang. Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS. In *Conference on Cluster, Cloud and Grid Computing (CCGrid)*, pages 368–377, 2010. 7, 14, 18

[WVY+09] S.-W. Wong, L. Valcarenghi, S.-H. Yen, D. Campelo, S. Yamashita, and L. Kazovsky. Sleep Mode for Energy Saving PONs: Advantages and Drawbacks. In *GreenComm: International Workshop On Green Communications*, pages 1–6, 2009. 104

[XXG+10] C. Xie, F. Xu, N. Ghani, E. Chaniotakis, C. Guok, and T. Lehman. Load-Balancing for Advance Reservation Connection Rerouting. *IEEE Communications Letters*, 14(6), 2010. 49

[YCKT09] C.-Y. Yang, J.-J. Chen, T.-W. Kuo, and L. Thiele. An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 694–699, 2009. 14, 18

[YDS95] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 374–382, 1995. 14, 18

[YL11] S. Yeo and H. Lee. Mathematical Modeling of Performance and Utility Consumption for a Heterogeneous Cloud Computing Environment. *Computer*, PP(99), 2011. 8

[YMB02] T. T. Ye, G. Micheli, and L. Benini. Analysis of power consumption on switch fabrics in network routers. In *ACM Design Automation Conference (DAC)*, pages 524–529, 2002. 23, 42

[YSLQ10] G. Yang, Y. Shen, K. Li, and W. Qu. A Comprehensive Task Scheduling Algorithm in Grid. In *Annual ChinaGrid Conference (ChinaGrid)*, pages 3–10, 2010. 47

[YSS10] N. Yamanaka, S. Shimizu, and Gao Shan. Energy efficient network design tool for green IP/Ethernet networks. In *Conference on Optical Network Design and Modeling (ONDM)*, 2010. 25, 30

[YZJ05] L. Yan, L. Zhong, and N. Jha. User-perceived latency driven voltage scaling for interactive applications. In *Design Automation Conference (DAC)*, pages 624–627, 2005. 14, 18

[YZL+10] Y. Yang, Y. Zhou, L. Liang, D. He, and Z. Sun. A Sevice-Oriented Broker for Bulk Data Transfer in Cloud Computing. In *International Conference on Grid and Cooperative Computing (GCC)*, pages 264–269, 2010. 47

[ZC08] J. Zhuo and C. Chakrabarti. Energy-efficient dynamic task scheduling algorithms for DVS systems. *ACM Transactions on Embedded Computing Systems*, 7:17:1–17:25, 2008. 14, 18

[ZCTM10] Y. Zhang, P. Chowdhury, M. Tornatore, and B. Mukherjee. Energy Efficiency in Telecom Optical Networks. *IEEE Communications Surveys Tutorials*, 12(4):441–458, 2010. 21

[ZDD+04] Q. Zhu, F. David, C. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. *IEEE International Symposium on High-Performance Computer Architecture*, 2004. 42

[ZDE+93] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7:8–18, 1993. 49, 118

[ZDH01] Z.-L. Zhang, Z. Duan, and Y. Hou. On Scalable Design of Bandwidth Brokers. *IEEE Transactions on Communications*, 8:2011–2025, 2001. 48, 101

[ZMV04] X. Zheng, A. Mudambi, and M. Veeraraghavan. FRTP: Fixed Rate Transport Protocol - A modified version of SABUL for end-to-end circuits. In *Workshop on Provisioning and Transport for Hybrid Networks (PATHNets, BroadNets workshop)*, 2004. 48

[ZS05] S. Zanikolas and R. Sakellariou. A taxonomy of grid monitoring systems. *Future Generation Computer Systems*, 21(1):163–188, 2005. 31

[ZSC10] S. Zhang, D. Shen, and C.-K. Chan. Energy efficient time-aware traffic grooming in wavelength routing networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, 2010. 29, 30

[ZSG+03] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang. Modeling Hard-Disk Power Consumption. In *USENIX Conference on File and Storage Technologies (FAST)*, pages 217–230, 2003. 42

[ZSGRG08] B. Zhang, K. Sabhanatarajan, A. Gordon-Ross, and A. George. Real-Time Performance Analysis of Adaptive Link Rate. In *IEEE Conference On Local Computer Networks (LCN)*, pages 282–288, 2008. 27, 30, 107, 109

Astract

Over the past few years, the energy consumption of Information and Communication Technologies (ICT) has become a major issue. Nowadays, ICT accounts for 2% of the global CO2 emissions, an amount similar to that produced by the aviation industry.

Large-scale distributed systems (e.g. Grids, Clouds and high-performance networks) are often heavy electricity consumers because – for high-availability requirements – their resources are always powered on even when they are not in use.

Reservation-based systems guarantee quality of service, allow for respect of user constraints and enable fine-grained resource management. For these reasons, we propose an energy-efficient reservation framework to reduce the electric consumption of distributed systems and dedicated networks.

The framework, called ERIDIS, is adapted to three different systems: data centers and grids, cloud environments and dedicated wired networks. By validating each derived infrastructure, we show that significant amounts of energy can be saved using ERIDIS in current and future large-scale distributed systems.

Résumé

Depuis quelques années, économiser l'énergie est devenu un enjeu majeur dans les technologies de l'information et de la communication (TIC). Celles-ci représentent en effet 2% des émissions de CO2 de la planète, soit autant que l'aviation.

Les systèmes distribués (grilles, clouds, réseaux haute performance) constituent de gros consommateurs d'électricité. En effet, pour des besoins de haute disponibilité, leurs ressources sont allumées en permanence et notamment lorsqu'elles ne sont pas utilisées.

Les systèmes de réservation garantissent qualité de service et respect des contraintes de l'utilisateur. Ils permettent également une gestion plus fine des ressources. Pour limiter la consommation électrique des systèmes distribués et des réseaux dédiés, nous avons proposé un système de réservation de ressources efficace en énergie.

Ce système de réservation, appelé ERIDIS, a été adapté à trois infrastructures distribuées différentes : les centres de calcul et les grilles, les environnements de cloud et les réseaux filaires dédiés. Dans les trois cas, des validations ont été menées et elles ont montré que des économies d'énergie significatives pouvaient être réalisées en utilisant ERIDIS dans les systèmes distribués actuels et futurs.