Saving Energy in QoS Networked Data Centers

by

Mohammad Shojafar

A thesis submitted
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Information and Communication Engineering
at
Sapienza University of Rome

Certified by:
Enzo Baccarelli
Scientist
Thesis Supervisor

May 2016
XXVIII

I would like to dedicate this thesis to my loving parents and my lovely wife Nasim …

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 150,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 250 figures.

Mohammad Shojafar
May 2016
XXVIII

Acknowledgements

I owe my deepest gratitude to my adviser - a very humble and kind person - Professor Enzo Baccarelli. Without his advice, this work would have not shaped up the way it has. Throughout my stay in Sapienza University, he has been the most supportive and understanding person, and has helped me achieve whatever I have craved for. It is only through his views, that I have learned mostly about a researcher, a professor and most importantly - an adviser. I cannot forget the support given by my parents (the gods of my life) for always being there beside me and also my lovely wife - Nasim, from whom I have a lot to learn in the area of life, philosophy and about myself. Life without friends is incomplete, like music to a song. I would like to thank my close friends - Hamid and Hossein, who have made me realize their importance. I will never forget my endless discussions with Professors Mukesh Singhal and Jemal Abawajy who have taught me how to question everything.

Abstract

One of the major challenges that cloud providers face is minimizing power consumption of their data centers. To this point, majority of current research focuses on energy efficient management of resources in the Infrastructure as a Service model using virtualization and through virtual machine consolidation. However, current virtualized data centers are not designed for supporting communication–computing intensive real-time applications, such as, info-mobility applications, real-time video co-decoding. In fact, imposing hard-limits on the overall per-job delay forces the overall networked computing infrastructure to adapt quickly its resource utilization to the (possibly, unpredictable and abrupt) time fluctuations of the offered workload. Jointly, a promising approach for making networked data centers more energy-efficient is the use of traffic engineering-based method to dynamically adapt the number of active servers to match the current workload. Therefore, it is desirable to develop a flexible and robust resource allocation algorithm that automatically adapts to time-varying workload and pays close attention to the consumed energy in computing and communication in virtualized networked data centers (VNetDCs).

In this thesis, we propose three new dynamic and adaptive energy-aware algorithms scheduling policies that model and manage VNetDCs. Our focuses are to propose *i)* admission control of the offered input traffic; *ii)* balanced control and dispatching of the admitted workload; *iii)* dynamic reconfiguration and consolidation of the Dynamic Voltage and Frequency Scaling (DVFS)-enabled Virtual Machines (VMs) instantiated onto the parallel computing platform; and, *iv)* rate control of the traffic injected into the TCP/IP mobile connection. Necessary and sufficient conditions for the feasibility and optimality of the proposed schedulers are also provided in closed-form. Specifically, the first approach, called VNetDC, the optimal minimum-energy scheduler for the joint adaptive load balancing and provisioning of the computing-plus-communication resources. VNetDC platforms have been considered which operate under hard real-time constraints. VNetDC has capability to adapt to the time-varying statistical features of the offered workload without requiring any a priori assumption and/or knowledge about the statistics of the processed data. GreenNetDC is the second scheduling policy that is a flexible and robust resource allocation algorithm that automatically adapts to time-varying workload and pays close attention to

the consumed energy in computing and communication in VNetDCs. GreenNetDC not only ensures users the Quality of Service (through Service Level Agreements) but also achieves maximum energy saving and attains green cloud computing goals in a fully distributed fashion by utilizing the DVFS-based CPU frequencies. Finally, the last algorithm tested an efficient dynamic resource provisioning scheduler which applied in Networked Data Centers (NetDCs). This method is connected to (possibly, mobile) clients through TCP/IP-based vehicular backbones The salient features of this algorithm is that: *i)* It is adaptive and admits distributed scalable implementation; *ii)* It is capable to provide hard QoS guarantees, in terms of minimum/maximum *instantaneous* rate of the traffic delivered to the client, *instantaneous* goodput and total processing delay; and, *iii)* It explicitly accounts for the dynamic interaction between computing and networking resources, in order to maximize the resulting energy efficiency. Actual performance of the proposed scheduler in the presence of :*i)* client mobility; *ii)*wireless fading; *iii)*reconfiguration and two-thresholds consolidation costs of the underlying networked computing platform; and, *iv)*abrupt changes of the transport quality of the available TCP/IP mobile connection, is numerically tested and compared against the corresponding ones of some state-of-the-art static schedulers, under both synthetically generated and measured real-world workload traces.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this chapter is to introduce some basic concepts concerning the world of Data Centers (DCs) in order to know in what context we are working and so that they can then better understand the purpose of our work and justify its value and its applications. It will make first a short description of DC structures as technological paradigm and then describe some challenges and the proposed objectives to respond them. Cloud computing is an innovative distributed computing paradigm that is widely accepted by public and private organizations. The main objective of providers is to obtain maximum profits and guarantee QoS requirements of customers. One of the main concerns is energy expenditure in this environment. Therefore, Energy consumption has become a significant concern for cloud service providers due to financial as well as environmental factors. As a result, cloud service providers are seeking innovative ways that allow them to reduce the amount of energy that their data centers consume. This chapter will be shifted attention to the concept of green Cloud and the problem of current interest in energy conservation with the aim of presenting the technological environment in the DCs.

1.1 Data Center Structure

Data Centers (DCs) are large scale, mission-critical computing infrastructures that are operating around the clock [13, 12, 75] to propel the fast growth of IT industry and transform the economy at large. The criticality of data centers has been fueled mainly by two phenomenon. First, the ever increasing growth in the demand for data computing, processing and storage by a variety of large scale cloud services, such as Google and Facebook, by telecommunication operators such as British Telecom [57], by banks and others, resulted in the proliferation of large DCs with thousands of servers (sometimes with millions of servers). Second, the requirement for supporting a vast variety of applications ranging from those that run for a few seconds to those that run persistently on shared hardware platforms [13] has promoted building large scale computing infrastructures. As a result, DCs have been touted as one of the key enabling technologies for the fast growing IT industry and at the same time, resulting in a global market size of 152 billion US dollars by 2016 [101]. Data centers being large scale computing infrastructures have huge energy budgets, which have given rise to various energy efficiency issues.

A general approach to manage data center energy consumption consists of four main steps (see Fig. 1.1): feature extraction, model construction, model validation or finding the solution, and apply the result in the practice as an application of the model to a task such as prediction. In Fig. 1.1 phases:

i) **Feature extraction:** In order to reduce the energy consumption of a data center, we first need to measure the energy consumption of its components [11, 7] and identify where most of the energy is spent. This is the task of the feature extraction phase.

ii) **Model construction**: The selected input features are used to build an energy consumption model using analysis techniques such as multi-gradient iteration methods, regression, machine learning etc. One of the key problems we face in this step is that certain important system parameters such as the power consumption of a particular component in a data center

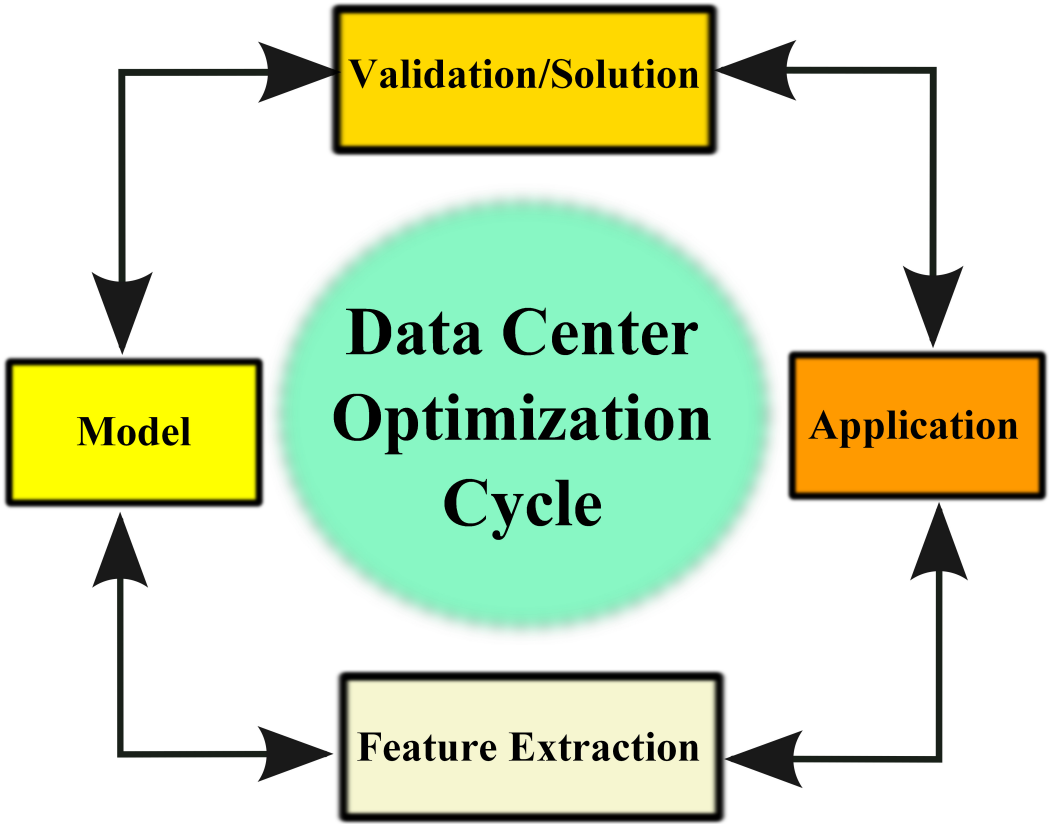


Fig. 1.1 A systematic view of the DC energy consumption modeling and solution. The data center system optimization cycle consists of four main steps: feature extraction, model construction, model validation/solution, and application.

cannot be measured directly. The outcome of this step is a power model. A model is a formal abstraction of a real system. Models for computer systems can be represented as equations, graphical models, rules, decision trees, sets of representative examples, neural networks, etc. The choice of representation affects the accuracy of the models, as well as their interoperability by people [11].

iii) **Model validation and solution**: The model needs to be validated for its fitness for its intended purposes. After validating, it needs to be resolved using online/offline techniques.

iv) **Application**: The validated model and solution can be used as the basis for predicting the component or system's energy consumption. Such predictions can then be used to improve the energy efficiency of the data center, for example by incorporating the model

into techniques such as dynamic voltage frequency scaling (DVFS) [48, 46, 98], resource virtualization [64], improving the algorithms used by the applications [9] or even completely shutting down unused servers [68][62], etc. In detail, we can use the solution for forecasting the trends in energy efficiency. It means, in daily operations of computer systems, users and data center operators need to understand the power usage patterns of computer systems in order to maximize their energy efficiency. Experimental verification using real test data is generally expensive and inflexible. Energy models on the other hand are much cheaper and more adaptive to changes in operating parameters [52]. Many different power consumption optimization schemes have been developed on top of power consumption models which are represented as mathematical functions [103]. Power modeling is an active area of research, studying both linear and nonlinear correlations between the system utilization and power consumption [13, 12, 57, 101, 32]. One main reason why DC energy consumption is very high is because servers that are ON but idle do consume significant amounts of energy, even when they are doing nothing [26]. Therefore, prediction techniques can be used to estimate future cloud workloads so as to appropriately decide whether and when physical machines (PMs) need to be put to sleep and when they need to be awakened to accommodate new VM requests. However, predicting cloud workloads can be very challenging due to the diversity as well as the sporadic arrivals of client requests, each coming at a different time and requesting different amounts of various resources (CPU, memory, bandwidth, etc.). The fact that there are infinite possibilities for the combinations of the requested amounts of resources associated with these requests requires classifying requests into multiple categories, based on their resource demands [47]. For each category, a separate predictor is then needed to estimate the number of requests of that category, which allows to estimate the number of PMs that are to be needed. Using these predictions, efficient power management decisions can be made, where an idle PM is switched to sleep only if it is predicted that it will not be

needed for a period long enough to compensate the overhead to be incurred due to switching it back ON later when needed.

1.2 Data Center Issues and Objectives

This subsection tackles the most important issues related to the energy-efficient for each DC. Energy efficiency of DCs has attained a key importance in recent years due to its *(i)* high economic, *(ii)* environmental, and *(iii)* performance impact. First, data centers have high economic impact due to multiple reasons. A typical data center (DC) may consume as much energy as 25,000 households. Data center spaces may consume up to 100 to 200 times as much electricity as standard office space [101]. Furthermore, the energy costs of powering a typical data center doubles every five years [13]. Therefore, with such steep increase in electricity use and rising electricity costs, power bills have become a significant expense for today's DCs [101, 32]. Second, data center energy usage creates a number of environmental problems [10, 100]. For example, in 2005, the total data center power consumption was 1% of the total US power consumption, and created as much emissions as a mid-sized nation like Argentina [10]. In 2010 the global electricity usage by DCs was estimated to be between 1.1% and 1.5% of the total worldwide electricity usage [4], while in the US the DCs consumed 1.7% to 2.2% of all US electrical usage [56]. Finally, running in the idle mode servers consume a significant amount of energy. Large savings can be made by turning off these servers. This and other measures such as workload consolidation need to be taken to reduce DC electricity usage. At the same time, these power saving techniques reduce system performance, pointing to a complex balance between energy savings and high performance. The energy consumed by a DC can be broadly categorized into two parts [51]: energy use by IT equipment (e.g., servers, networks, storage, etc.) and usage by infrastructure facilities (e.g., cooling and power conditioning systems). The amount of energy consumed by these two subcomponents depend on the design of the DC as well as the efficiency of

the equipment. However, modeling the exact energy consumption behavior of a data center, either at the whole system level or the individual component level, is not straightforward.

In summary, to deal with the challenges highlighted with the above research problems, the following objectives have been extracted:

- Explore, analyze, and evaluate the research in the area of energy-efficient DCs techniques to gain a clear understanding of the DC solutions.
- Propose two adaptive distributed job schedulers in order to support real/synthetic workloads to preserve energy over the joint of computing and communication part inside networked data centers (NetDCs) which highlight computing, communication and reconfiguration costs.
- Develop an offline/online algorithm for energy-efficient distributed dynamic VM consolidation for IaaS environment satisfy QoS constraints.

1.3 Contributions

The main contributions of this thesis can be simplified into 4 categories: *classification and analysis of the DC energy provisioning*, *competitive analysis of dynamic and adaptive energy-aware algorithms*, *a novel adaptive energy-aware algorithm which stresses on the joint of the TCP/IP-aware QoS mobile connection and computing and communication cost of the NetDCs*, and *implementation of problem solution* of the aforementioned algorithms.

The key contributions for the competitive analysis of dynamic and adaptive energy-aware algorithms consist of *i)* few modeling efforts targeted at power consumption of the entire data center *ii)* many state-of-the-art power models are based on a few CPU or server metrics, and *iii)* the effectiveness and accuracy of these power models remain open questions.

The second contribution is the competitive analysis of dynamic and adaptive energy-aware algorithms. In the first type of the proposed method, in order to attain energy saving

in such kind harsh computing scenario, a joint balanced provisioning and adaptive scaling of the networking-plus-computing resources is demanded. To be summarized: first, the contrasting objectives of low consumptions of both networking and computing energies in delay and bandwidth-constrained Virtualized NetDCs (VNetDCs) are cast in the form of a suitable constrained optimization problem, namely, the Computing and Communication Optimization Problem (CCOP). Second, due to the nonlinear behavior of the rate-vs.-power-vs.-delay relationship, the CCOP is not a convex optimization problem and neither guaranteed-convergence adaptive algorithms nor closed-form formulate are, to date, available for its solution. Hence, in order to solve the CCOP in exact and closed-form, we prove that it admits a loss-free (e.g., optimality preserving) decomposition into two simpler loosely coupled sub-problems, namely, the CoMmunication Optimization Problem (CMOP) and the ComPuting Optimization Problem (CPOP). Third, we develop a fully adaptive version of the proposed resource scheduler that is capable to quickly adapt to the a priori unknown time-variations of the offered workload and converges to the optimal resource allocation without the need to be restarted. In the second type of the proposed method, we introduce a new approach to decrease energy consumption in computing, communication and switching cost in online independent workloads running on virtualized cloud with parallel channels which are assigned for each servers in data centers. We defines SLA as the minimum computing and communication time, maximal response time in the deployed system. In other words, each incoming service which is same as the incoming workload has to be processed within a deadline determined by the SLA which will be defined as a hard deadline parameter for the system. To clarify the discussion and better place individual contributions in context, we outlined a framework for cloud resource management, which lays the basis for the core of the paper, the state-of-the-art energy provisioning approach that particularly considers optimum VM frequency, optimum bandwidth rate required for each VM, and proper workload quota for each VM in each incoming workload.

The third contribution is TCP/IP-aware adaptive energy-aware pursuing some directions in this thesis which are listed below:

- The global introduction of a novel distributed architecture to energy and performance efficient dynamic VM consolidation in the VNetDC.

- A novel model that allows a derivation of a randomized control policy that optimally solves the problem of maximizing the TCP/IP mobile communication goodput and minimizing the expected energy consumption for the joint computing and communication in VNetDC under an explicitly specified QoS goal for any known stationary workload and a given state configuration in the online setting.

- An optimal nonconvex optimization algorithm and proof of its optimality using multi-gradient stochastic approximation with fast convergence rate.

- A heuristically adapted solution for handling unknown non-stationary workloads using the time-correlated sliding window while providing the advantage of explicit specification of QoS goals.

And the ultimate contribution of the thesis is implementing the problem solutions using CVX [35] (i.e., using MOSEK sub-package), Cloudsim [38], and Matlab optimization packages. CVX is a Matlab-based modeling system for convex optimization. CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard Matlab expression syntax. In its default mode, CVX supports a particular approach to convex optimization that we call disciplined convex programming. Under this approach, convex functions and sets are built up from a small set of rules from convex analysis, starting from a base library of convex functions and sets. Constraints and objectives that are expressed using these rules are automatically transformed to a canonical form and solved. We use Cloudsim (i.e., CloudSim is developed in the Cloud Computing and Distributed Systems (CLOUDS)

Laboratory, at the Computer Science and Software Engineering Department of the University of Melbourne) to model and simulate the solutions in a simulated based environment.

1.4 Thesis Structure

The remainder of the thesis is organized as follows:

- Chapter 2 highlights the most recent related works and discuss research issues related to conflicting requirements of maximizing quality of services (QoSs) (availability, reliability, etc.) delivered by the cloud services while minimizing energy consumption of the data center resources.

- Chapter 3 proposes two novel algorithms for distributed dynamic VM consolidation which are derived from [23], [21], [85], [86], and [22]:

  ◇ Nicola Cordeschi, **Mohammad Shojafar**, Enzo Baccarelli, "Energy-saving self-configuring networked data centers", *Computer Networks*, ISSN: 1389-1286, Volume 57, Issue 17, Pages: 3479–3491, *Elsevier Science*, The Netherlands, 2013.

  ◇ Nicola Cordeschi, **Mohammad Shojafar**, Danilo Amendola, Enzo Baccarelli, "Energy-efficient adaptive networked datacenters for the QoS support of real-time applications", ISSN: 0920-8542, Volume 71, Issue 2, Pages: 448-478, Springer, *The Journal of Supercomputing (SUPE)*, 2014.

  ◇ **Mohammad Shojafar**, Nicola Cordeschi, Enzo Baccarelli, "Resource Scheduling for Saving Energy in Reconfigurable Internet Data Centers", *Handbook of Research on Next-Generation High Performance Computing*, IGI Global, to be appear 2016.

◇ Nicola Cordeschi, **Mohammad Shojafar**, Danilo Amendola, Enzo Baccarelli, "Energy-Saving QoS Resource Management of Virtualized Networked Data Centers for Big Data Stream Computing", Handbook in *Emerging Research in Cloud Distributed Computing Systems*, IGI Global, Pages: 122-155, 2015.

- Chapter 4 proposes novel algorithm for distributed dynamic VM consolidation derived from [83], [84] [82], [18], and [19]:

  ◇ **Mohammad Shojafar**, Nicola Cordeschi, Danilo Amendola, Enzo Baccarelli, "Energy-saving adaptive computing and traffic engineering for real-time-service data centers", *Communication Workshop (ICCW), 2015 IEEE International Conference on*, London, UK, Page: 1800-1806, 2015.

  ◇ **Mohammad Shojafar**, Nicola Cordeschi, Enzo Baccarelli,"Energy-efficient Adaptive Resource Management for Real-time Vehicular Cloud Services", *IEEE Transactions on Cloud Computing (TCC)*, Volume PP, Issue 99, Pages:1-14, 2016.

  ◇ **Mohammad Shojafar**, Nicola Cordeschi, Jemal H. Abawajy, Enzo Baccarelli, "Adaptive Energy-Efficient QoS-Aware Scheduling Algorithm for TCP/IP Mobile Cloud", *Global Communication Workshop (GLOBECOM), 2015 IEEE International Conference on*, San Diego, USA, Pages: 1-6, 2015.

  ◇ Nicola Cordeschi, Danilo Amendola, **Mohammad Shojafar**, Enzo Baccarelli, "Distributed and Adaptive Resource Management in Cloud-assisted Cognitive Radio Vehicular Networks with Hard Reliability Guarantees", *Vehicular Communications*, Volume 2, Issue 1, Page: 1–12, *Elsevier Science*, The Netherlands, 2015.

  ◇ Nicola Cordeschi, Danilo Amendola, **Mohammad Shojafar**, Enzo Baccarelli, "Performance evaluation of primary-secondary reliable resource-management in vehicular networks", *The 25th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications(IEEE PIMRC 2014)*, Page: 959-964, 2014.

- Chapter 5 concludes the thesis with a summary of the main findings, discussion of future research directions, and final remarks.

In the next chapter we make a description on the state of work strictly related.

Chapter 2

State-of-the-art related Works

Regarding the paradigm of *Green Cloud* (from a technological standpoint and also taking into account the problem of communication) recent studies that have already taken steps to make the point about the state of the art has been made in several researches. Large volume of research work has been done in the area of power and energy-efficient resource management in Cloud/DCs. As power and energy management techniques are closely connected, from this point we will refer to them as power management. The goal is to give a appropriate overview of the works related to the topics addressed in this thesis, underline the motivations behind the latter and details their limitations compared to the proposed methods.

2.1 Data Center Energy Consumption: A System Perspective

Therefore, we conduct an in-depth study of the existing work in data center power models, and to organize the models using a coherent layer-wise abstraction as shown in Figure 2.1.

In general we can categorize the constituents of a data center as belonging to one of two layers, software and hardware. The software layer can be further divided into two subcategories, the OS/virtualization layer, and the application layer. In this chapter we describe the power consumption modeling work in the software layer. Throughout this

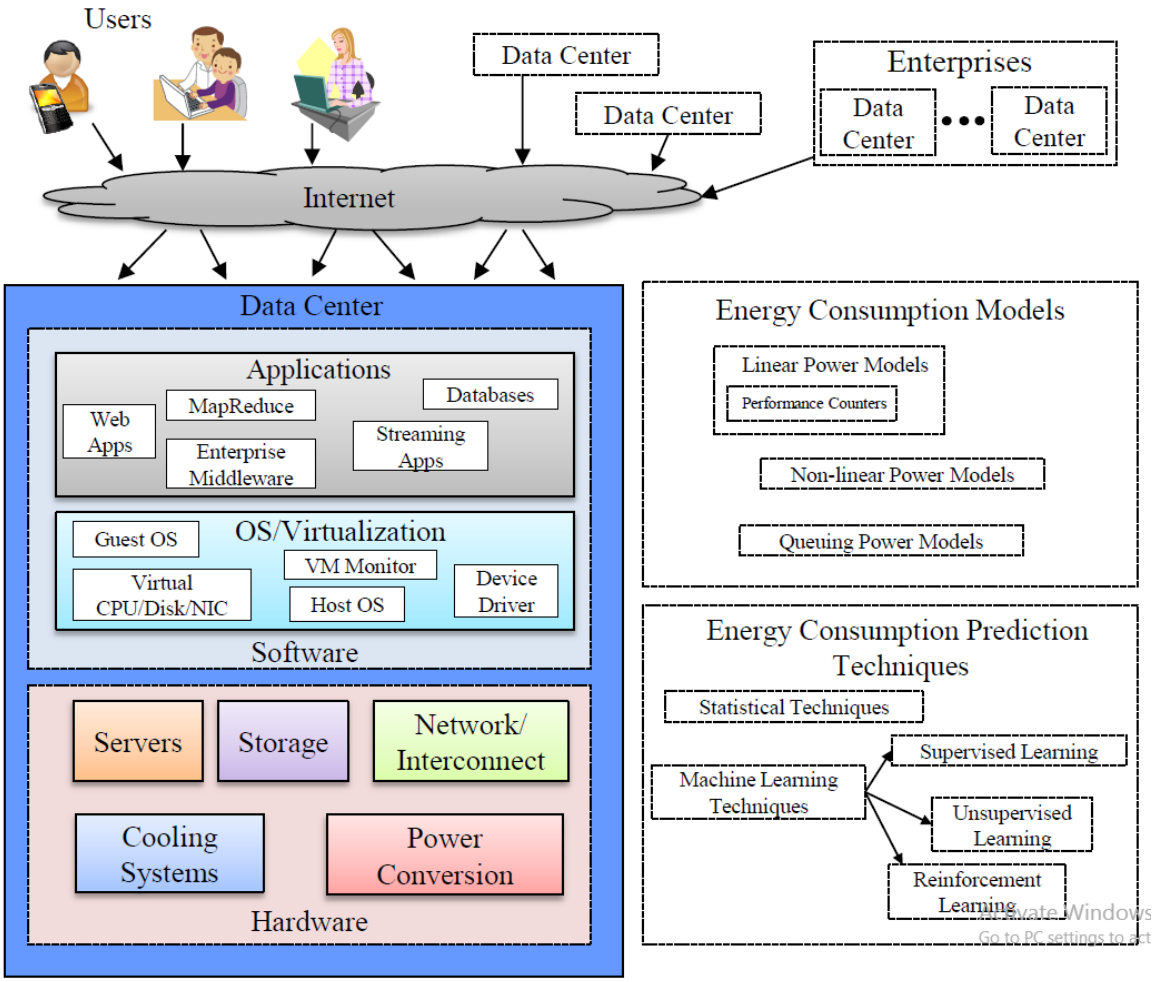


Fig. 2.1 A holistic view of the context for energy consumption modeling and prediction in data centers

process, we highlight various energy consumption modeling and prediction techniques during this process which are applied at various different levels of the data center systems of systems. Power models play a fundamental role in energy-efficiency research of which the goal is to improve the components' and systems' design or to efficiently use the existing hardware[25]. Some energy-provisioning models use non-linear power models [94, 103] and some use queuing power models [42, 28, 91, 83]. For brevity, authors in certain works [28] the server's CPU usage and operation frequency are used for modeling a server's power

consumption. Whilst the approach is promising, but power consumption of each core must be known beforehand which could be resolved in the proposed methods in this thesis.

2.2 Related Works

The main challenge in DCs is the minimization of the energy usage, while still meeting the QoS requirements of the supported applications. For this purpose, adaptive and scalable energy-aware scheduling algorithms are required that jointly perform the allocation of the networking and computing resources on the Cloud and over the TCP/IP mobile connections which link the DCs to the mobile clients. There have been numerous works in the area of the Internet DCs which aim at providing various models and techniques to seamlessly integrate the management of computing-communication virtualized platforms, in order to provide QoS, robustness and reduced energy consumption [24]. In this respect, several works focused on the energy-efficiency, by exploiting customized reconfigurable hardware, such as, Dynamic Voltage and Frequency Scaling (DVFS) [69, 6], and/or the virtualization of the networking-computing resources [5]. DVFS technique uses in DCs for energy provisioning by dynamically adjusted processors. DVFS is applied in most of the modern computing units, such as cluster computing and supercomputing, to reduce power consumption and achieve high reliability and availability. Resource virtualization refers to instantiate several VMs on a same physical server, in order to reduce the numbers of hardware, while improving the utilization of resources. Therefore, we should take into account both virtualization and DVFS techniques, which are energy-aware tools in cloud data-centers to manage resource provisioning and energies.

The methods categorizes to save energy in DC are right-sizing and VMs consolidation. Right-sizing of data centers—dynamically resizing the active servers as load varies under the control of a market-based resource allocation algorithm. But, VM consolidation algorithms

that balance the level of consolidation with the performance degradation arising from resource contention.

Furthermore, one of the important requirements for a Cloud computing environment is the provision of reliable Service Level Agreements (SLAs). SLAs can be managed *globally* or *locally* by the cloud providers by adding some policies for each VM in each server or in the data centers which include several servers. At the local level, the system leverages the power of each VM based on the SLA management policies locally, while, at global level, this policy can be handled by general SLA policies among servers or data centers.

Another issue of growing concern in cloud environments is to evenly distribute huge amount of workloads over various servers, which is called *load balancing*. Load balancing algorithms seek to distribute workloads across a number of servers, so that the average executions are minimized [97]. Load balancing schemes can be classified as *static* or *dynamic*. In static schemes, the current states of the servers are not considered when dispatching the workloads; examples of such schemes include Random Selection of servers and Round Robin policies. Dynamic schemes involve direct notification or indirect inference of server states by the load balancer [97].

Updated surveys of current technologies and open communication challenges about energy-efficient DCs have been recently presented in [45, 21]. Specifically, power management schemes that exploit DVFS techniques for performing resource provisioning are the focus of [6, 5, 78]. Although these contributions consider hard deadline constraints, they do not account, indeed, for the performance penalty and energy-vs.-delay tradeoff stemming from the finite transmission rates of the utilized network infrastructures. Energy-saving dynamic provisioning of the computing resources in virtualized green data centers is the topic of [104, 91]. Specifically, the authors of [104] formulate the optimization problem as a feedback control problem that must converge to an a priori known target performance

level. While this approach is suitable for tracking problems, it cannot be employed for energy-minimization purposes, where the target values are a priori unknown.

Roughly speaking, the common approach pursued by [68] is to formulate the afforded minimum-cost resource allocated problem as sequential optimization problem and, then, solve it by using limited look-ahead control. Hence, the effectiveness of this approach relies on the ability to accurately predict the future workload and the performance degrades when the workload exhibits almost unpredictable time fluctuations. In order to avoid the prediction of future workload, [91] resorts to a Lyapunov-based technique, that dynamically optimizes the provisioning of the computing resources by exploiting the available queue information. Although the pursued approach is of interest, it relies on an inherent delay-vs.-utility tradeoff, that does not allow us to account for hard deadline constraints.

The suitable exploitation of some peculiar features of the network topology of current Networked Data Centers (NetDCs) is at the basis of the capacity-planning approach recently proposed in [96]. For this purpose, a novel traffic engineering-based framework is developed, which aims at reducing the number of active switches, while simultaneously balancing the resulting communication flows. Although the attained reductions of the energy consumed by the networking infrastructures are, indeed, noticeable, the capacity-planning approach in [95] does not consider, by design, the corresponding energy consumed by the computing servers and, which is the most, it subsumes delay-tolerant application scenarios. The joint analysis of the computing-plus-communication energy consumption in virtualized NetDCs is, indeed, the focus of [69, 68], where delay-tolerant Internet-based applications are considered. Interestingly, the main lesson stemming from these contributions is that the energy consumption due to data communication may represent a large part of the overall energy demand, especially when the utilized network is bandwidth-limited. Overall, these works numerically analyze and test the energy performance of some state-of-the-art schedulers for NetDCs, but do not attempt to optimize it through the dynamic joint scaling of the

available communication-plus-computing resources. Passing to consider the research area on the mobile communication, a first research line focuses on the cross-layer analysis and optimization of TCP/IP traffic control mechanisms for single-antenna and multi-antenna mobile connections [65, 30]. These contributions support the conclusion that an optimal control of the energy employed by the wireless transmission is an effective means to improve the resulting TCP goodput. However, this conclusion is partially offset by the fact that [65, 30] neglect the computing aspects. Analogous conclusion holds for the works in [70] and [84], in which optimized schedulers are derived by exploiting nonlinear optimization and queuing theory. Specifically, the scheduler developed in [70] does not present adaptive capability, while, the scheduler in [84] does not account for the limitation on the energy budget of the underlying mobile TCP/IP connection.

The forecast development of adaptive ubiquitous applications (such as, for example, iCloud) for highly parallel mobile (possibly, vehicular) processing platforms demands for a novel design approach that integrates both computing and communication aspects, while it is capable to effectively cope with the inherently stochastic and time-varying nature of the mobile domain. This approach should be characterized by a tight interaction between two still distinct engineering fields, e.g., Parallel Computing [12] and Vehicular Communication [43]. Roughly speaking, both these paradigms cover a set of formal tools and methodologies capable to lead to the design of technological platforms for the parallel execution and ubiquitous delivering of QoS-demanding applications, but, up to date, this shared target is accomplished by pursuing non-integrated approaches. Specifically, Parallel Computing focuses on the execution of compute-intensive applications by using remote computing resources, that may be acceded through the (possibly, wireless) Internet [12, 6]. In this framework, tools for dealing with the dynamic behavior of the computing and communication resources mainly focus on the achievement of target levels of computing efficiency [12]. However, Mobile Communication mainly deals with design of systems equipped with several

communication and (possibly) computing resources, whose combined utilization aims at providing seamless ubiquitous services to mobile (possibly, vehicular) clients [43].

By integrating these two (still distinct) paradigms, Vehicular Networking is in the progress of merging with the Mobile Internet and Mobile Cloud Computing (MCC), so as to constitute an integrated communication/computing information platform [99]. While safe navigation has always been the prime motivation behind vehicular communications, it is believed that vehicular networks will provide communication infrastructures for a much broader range of large-scale high-mobile applications. Several solutions have been recently proposed to address the expected mission of next-generation of vehicular networks [49], and Vehicular Cloud Computing (VCC) is emerging as the most appealing solving paradigm. VCC is, indeed, a composite paradigm, which combines multiple networking technologies (such as, mobile networking, WLAN technology, wireless sensor networking) together with MCC. Industry and academia have identified three main classes of potential applications to be supported by networked VCC-based infrastructures, namely, safety-related services, transport-related services and infotainment services [99]. These services are provided by Clouds to the requiring Vehicular Clients (VCs) mainly through the Vehicle-to-Infrastructure (V2I) communication mode. In the V2I mode, each served VC downloads information from the serving (generally, static) Roadside Unit (RSU). Furthermore, multiple RSUs may be also interconnected, in order to form a wireless backbone, in order to provide the connection to remote Clouds through the mobile Internet. Each RSU may also acts as a proximate cloud (that is, a CloudLet (CL)) [49], in order to provide local services which do not require the access to remote Clouds. Vice versa, in order to deliver more computation-intensive infotainment services (such as audio/video streaming, big-data streaming, podcasting, vehicular mapping), the RSU only acts as an access point (AP) and relay information from the remote Cloud. Since smartphones are usually constrained by limited resources, including energy, computation and storage, the joint optimized adaptive management of the Cloud-generated

vehicular TCP/IP traffic and the computational workload to be handled by the remote Cloud is one of the main challenges which hampers the rapid development of VCC infrastructures [29].

In the next chapters, firstly, we propose a new adaptive virtualized networked data center, VNetDC; a joint balanced provisioning and adaptive scaling of the networking-plus-computing resources, in order to attain energy saving in such kind harsh computing scenario (data center environment). This is the focus of the first part of the next chapter, whose main contributions may be so summarized. First, the contrasting objectives of low consumptions of both networking and computing energies in delay and bandwidth-constrained VNetDCs are cast in the form of a suitable constrained optimization problem, namely, the Computing and Communication Optimization Problem (CCOP). Second, due to the nonlinear behavior of the rate-vs.-power-vs.-delay relationship, the CCOP is not a convex optimization problem and neither guaranteed-convergence adaptive algorithms nor closed-form formulate are, to date, available for its solution. Hence, in order to solve the CCOP in exact and closed-form, we prove that it admits a loss-free (e.g., optimality preserving) decomposition into two simpler loosely coupled sub-problems, namely, the CoMmunication Optimization Problem (CMOP) and the ComPuting Optimization Problem (CPOP). Third, we develop a fully adaptive version of the proposed resource scheduler that is capable to quickly adapt to the a priori unknown time-variations of the offered workload and converges to the optimal resource allocation without the need to be restarted. Secondly, we propose a new traffic engineering-based approach, called *GreenNetDC*, to reduce energy consumptions in computing, communication and reconfiguration in LAN-equipped virtualized Clouds. We introduce SLA as a hard time-constraint on the computing and communication processing in *GreenNetDC*: *GreenNetDC* changes the VMs status and the allocated resources on a per-job basis to maintain the SLA and saves energy. In a nutshell, the main objective of this scheduler is to introduce a joint computing-plus-communication framework and develop an efficient scheduler for virtualized

data centers that takes into account the allowed discrete processing frequencies for VMs hosted by DVFS-enabled CPU cores. It is important to note that this feature has an internal effect in each CPU while facing online workload. Specifically, *GreenNetDC* aims to:

- define an architectural framework and principles for energy-efficient VNetDC;
- develop an efficient energy-aware resource allocation and provisioning algorithm in a way that improves the energy efficiency of a data center under hard SLA constraints;
- develop an adaptive version of the scheduling algorithm for energy-efficient mapping of workload quota to the available VMs.

Notable features of the resulting *GreenNetDC* scheduler are its ease of implementation, and the ability to manage time-varying workload at the a minimal reconfiguration cost.

Lastly, in this thesis, we develop and test a new scheduler for minimizing the energy consumption induced by computing, communication and reconfiguration costs in Internet-based virtualized DCs which utilize end-to-end TCP/IP mobile energy-constrained connections under hard limits on the per-job total processing time. Our scheduler performs dynamic load balancing and uses online job decomposition for adaptive resource management. It leads to the optimum processing speeds and bandwidth rates on a per-VM basis, as well as the proper workload quota for each VM on a per-job basis. Furthermore, it also performs admission control and adaptive management of the transmission rate of the Cloud-to-Vehicular TCP/IP mobile connections of Fig.4.1, in order to meet QoS constraints at the minimum energy wasting.

Chapter 3

QoS-aware Energy-efficient Schedulers for the NetDC

In this chapter, we present two proposed schedulers that allows us to dynamically allocate tasks size, rate of computing, communication rate, in a Virtualized Networked Data Center (VNetDC) operating under tight constraints of delay per-job. The First one, we present a jointly scheduler which is adaptively accomplish load balancing and provisioning resources using input buffer admission control. The later, a new adaptive real-based resource provisioning model introduced which is focused on the exploitation of the virtualization technology to maximize energy saving according to the considered discrete CPU frequency ranges of each server in a VNetDC.

3.1 VNetDC Scheduler

In this section of this chapter, we develop the optimal minimum-energy scheduler for the adaptive joint allocation of the task sizes, computing rates, communication rates and communication powers in VNetDCs that operate under hard per-job delay-constraints.

3.1.1 An Adaptive Energy-efficient VNetDC

The considered VNetDC platform works at the Middleware layer of the underlying protocol stack. Our objective is the minimization of the overall computing-plus-communication

energy consumption. The main new contributions of the paper are the following ones: *i)* the computing-plus-communication resources are *jointly* alloted in an *adaptive* fashion by accounting in *real-time* for both the (possibly, unpredictable) time-fluctuations of the offered workload and the reconfiguration costs of the considered VNetDC platform; *ii) hard* per-job delay constraints on the overall allowed computing-plus-communication latencies are enforced; and, *iii)* in order to deal with the inherently *nonconvex* nature of the resulting resource optimization problem, a novel solving approach is developed, that leads to the *lossless* decomposition of the afforded problem into the cascade of two simpler sub-problems. The sensitivity of the energy consumption of the proposed scheduler on the allowed processing latency, as well as the Peak-to-Mean Ratio (PMR) and the correlation coefficient (i.e., the smoothness) of the offered workload is numerically tested under both synthetically generated and real-world workload traces. Finally, as an index of the attained energy-efficiency, we compare the energy consumption of the proposed scheduler with the corresponding ones of some benchmark static, hybrid and sequential schedulers and numerically evaluate the resulting percent energy gaps.

3.1.2 The considered VNetDC model

The considered VNetDC platform is sketched in Fig.3.1. Our objective is to minimize the computing-plus-communication energy which is wasted by the VNetDC for processing the input traffic. In order to cope with the aforementioned challenges (i.e., data volume, uncontrolled arrival rates and non stationary), we enforce three main constraints. First, the overall per-job storage-plus-computing-plus-communication delay must be limited in a *hard* way. Second, energy-saving *adaptive* reconfigurations of the available computing-plus-communication resources must be performed in *real-time* (i.e., on a *per-job* basis). Third, a priori assumptions about the time-fluctuations and/or statistical behavior of the input traffics *cannot* be enforced.

A VNetDC platform for the parallel real-time processing of data traffics is composed by multiple reconfigurable Virtual Machines (VMs), which are interconnected by a switched rate-adaptive Virtual LAN (VLAN) and are managed by a central controller [5, 79]. The central controller dynamically performs the admission control of the input flows and the allocation of the virtual resources. Hence, emerging VNetDCs are composed by four main components, i.e., the input/output gateway routers, the working storage, the Virtual Machine Manager (VMM) and the switched VLAN (see Fig.3.1). According to an emerging trend [24, 58], the platform of Fig.3.1 implements the SaaS model and operates at the Middleware layer. Inspired by recent contributions on VNetDCs [63], we assume a discrete time-slotted system, where the time slot length $T_t(s)$ (s means second) can range from hundreds of milliseconds to few minutes. The m-th slot spans the (semi-open) interval $[mT_t, (m+1)T_t)$, where $m \geq 0$ is the integer-valued slot index. According to Fig.3.1, (possibly) multiple Internet flows are multiplexed by the input gateway router and (part of the) multiplexed data traffic is temporarily buffered by the working storage of Fig.3.1. At the beginning of each slot m, all the current backlog (of size $L_{tot}(m)(bit)$) [1] of the working storage is cleared (e.g., the working storage is emptied). The backlog is passed to the VNetDC platform of the Fig.3.1 and constitutes the current input job of size $L_{tot}(bit)$. The VNetDC platform processes, in turn, the currently submitted input job within a time interval which is limited up to T_t seconds. Finally, at the end of the current slot, the processed job (e.g., the output job) is passed to the output gateway router of Fig.3.1.

Before proceeding, some remarks about the considered VNetDC platform of Fig.3.1 are in order. First, the working storage of Fig.3.1 acts as a $G/G/1$ queue, so that assumptions on the statistics of the input data traffic are not required. Second, since the current backlog of the working storage is cleared at the beginning of each slot and the per-job processing time of the VNetDC platform is limited up to T_t seconds, the overall per-job queue-plus-computing-plus-communication delay is *hard-limited* up to $2T_t$ seconds. Third, since, in

[1] In the sequel, we understand the dependence on the slot index m when it is not strictly demanded.

slotted systems, the queue backlog is sampled on a per-slot basis, the slot length T_t is the minimum hard upper bound on the queue delay.

As a consequence, in order to avoid the overflow of the working storage of Fig.3.1, it suffices that its buffering capacity equates T_t times to the maximum expected rate (in (bit/s)) of the multiplexed input data traffic.

3.1.2.1 Input jobs and offered workload

According to the described framework, at the beginning of each time-slot, a new job of size $L_{tot}(bit)$ arrives at the input of the scheduler of Fig.3.1. The input job is characterized by: *i)* the size L_{tot} of the workload to be processed; *ii)* the maximum tolerated processing delay T_t; and, *iii)* the job granularity, that is, the (integer-valued) maximum number $M_T \geq 1$ of independent parallel tasks embedded into the submitted job. Let $M_V \geq 1$ be the maximum number of VMs that are available in Fig.3.1. In principle, each VM may be modeled as a virtual server, that is capable to process f_c bits per second [71]. Depending on the size L (bit) of the task to be currently processed by the VM, the corresponding processing rate f_c may be adaptively scaled at run-time, and it may assume values over the interval $[0, f_c^{\max}]$, where $f_c^{\max}$ (bit/s) is the maximum per-job allowed processing rate[2]. Furthermore, due to the real-time nature of the considered application scenario, the time allowed the VM to fully process each submitted task is fixed in advance at Δ (s), *regardless* of the actual size L of the task currently assigned to the VM. In addition to the currently assigned task (of size L), the VM may also process a background workload of size $L_b(bit)$, that accounts for the programs of the guest Operating System [71]. Hence, by definition, the *utilization factor* η of the VM equates [71]: $\eta \triangleq f_c/f_c^{max} \in [0,1]$. Then, as in [53, 79, 55], let $\mathscr{E}_c = \mathscr{E}_c(f_c)$ $(Joule)$ be the overall energy consumed by the VM to process a single task of duration Δ at the processing

[2]Since L_{tot} is expressed in $(bits)$, we express f_c in (bit/s). However, all the presented developments and formal properties still hold verbatim when L_{tot} is measured in *Jobs* and, then, f_c is measured in $(Jobs/cycle)$. Depending on the considered application scenario, a job may be a bit, frame, datagram, segment, or an overall file record.

rate f_c, and let $\mathscr{E}_c^{\max} = \mathscr{E}_c(f_c^{\max})$ $(Joule)$ be the corresponding maximum energy when the VM operates at the maximum processing rate f_c^{max}. Hence, by definition, the (dimensionless) ratio

$$\Phi(\eta) \triangleq \frac{\mathscr{E}_c(f_c)}{\mathscr{E}_c^{\max}} = \Phi\left(\frac{f_c}{f_c^{\max}}\right), \tag{3.1}$$

is the so-called Normalized Energy Consumption *(NEC)* of the considered VM [97]. From an analytical point of view, $\Phi(\eta) : [0,1] \rightarrow [0,1]$ is a function of the actual value η of the utilization factor of the VM. Its analytical behavior depends on the specific features of the resource provisioning policy actually implemented by the VMM of Fig.3.1 [53, 107]. Fig.3.1 operates at the Middleware layer and atop the Virtualization Layer. Black boxes indicate Virtual Network Interface Cards (VNICs), which terminate end-to-end TCP-based connections [5]. However, at least for CMOS-based physical CPUs, the following two (mild) assumptions on $\Phi(\eta)$ are typically met [97, 55]; *i)* $\Phi(\eta)$ is not decreasing in η and it attains its minimum (possibly, positive) value b at $\eta = 0$; and, *ii)* $\Phi(\eta)$ is convex in η. Just as a practical example, the analytical form assumed by $\Phi(\eta)$ for CMOS-based CPUs is recognized to be well described by the following c-powered one [97, 107, 55]:

$$\Phi(\eta) = (1-b)\,\eta^c + b, \quad 0 \leq \eta \leq 1, \tag{3.2}$$

where $c \geq 1$, while $b \in [0,1)$ represents the fraction of the full-loaded energy consumption $\mathscr{E}_c^{max}$ which is wasted by the VM in the idle state, that is, $\mathscr{E}_c^{idle} \triangleq b\,\mathscr{E}_c^{max}$.

3.1.2.2 Power-limited virtualized communication infrastructure

Let $M \triangleq \min\{M_V, M_T\}$ be the degree of concurrency of the submitted job, let L_{tot} be the overall size of the job currently submitted to the VNetDC, and let $L_i \geq 0$, $i = 1,\ldots,M$, be the size of the task that the Scheduler of Fig.3.1 assigns to the i-th VM, i.e., $VM(i)$. Hence, the following constraint: $\sum_{i=1}^{M} L_i = L_{tot}$ guarantees that the current input job of size L_{tot} is partitioned into (at the most) M parallel tasks. In order to keep at the minimum the

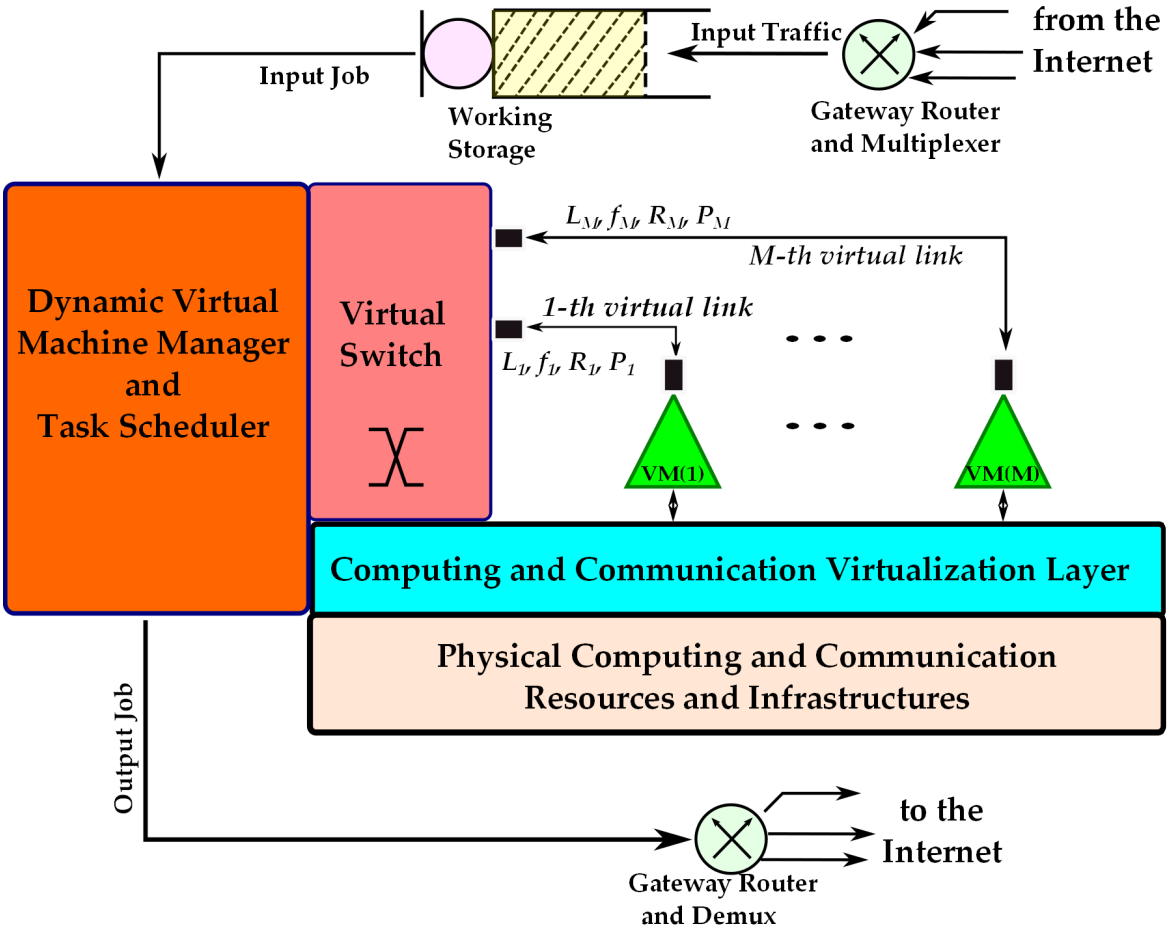


Fig. 3.1 The Proposed VNetDC architecture.

transmission delays from (to) the Scheduler to (from) the VMs of Fig.3.1, as in [6],[5], we assume that each $VM(i)$ communicates to the Scheduler through a dedicated reliable virtual link, that operates at the transmission rate of R_i (bit/s), $i = 1,\ldots,M$ and it is equipped with suitable Virtual Network Interface Cards (VNICs) (see Fig.3.1). The one-way transmission-plus-switching operation over the i-th virtual link drains a (variable) power of P_i^{net} $(Watt)$, where P_i^{net} is the summation: $P_i^{net} \triangleq P_T^{net}(i) + P_R^{net}(i)$ of the power $P_T^{net}(i)$ consumed by the transmit VNIC and the corresponding power $P_R^{net}(i)$ wasted by the receive VNIC (see Fig.3.1).

Passing to consider the actual value of P_i^{net}, we observe that, in order to limit the implementation cost, current data centers utilize off-the-shelf rackmount physical servers,

which are interconnected by commodity Fast/Giga Ethernet switches [5]. Furthermore, they implement TCP suites (mainly, the TCPNewReno one) for performing congestion control and attaining end-to-end (typically, multi-hop) reliable communication [69]. At this regard, we note that the data center-oriented versions of the legacy TCPNewReno suite in [3, 27] assure the same steady-state reliable throughput of the legacy TCPNewReno protocol. This means, in turn, that the steady-state average throughput $R_i(bit/s)$ of the i-th virtual link of Fig.3.1 (i.e., the i-th end-to-end transport connection) is given by [59, section 3.7]:

$$R_i = \sqrt{\frac{3}{2v}} \frac{MSS}{\overline{RTT_i}\sqrt{\overline{P_i}^{Loss}}}, \qquad i = 1,\ldots,M. \tag{3.3}$$

As it is known [59], MSS (bit) in (3.3) is the maximum segment size, $v \in \{1,2\}$ is the number of per-ACK acknowledged segments, $\overline{RTT_i}$ is the average round-trip-time of the i-th end-to-end connection, and $\overline{P_i}^{Loss}$ is the average segment loss probability of the i-th connection. At this regard, several studies point out that $\overline{P_i}^{Loss}$ scales down for increasing P_i^{net} as in [20]

$$\overline{P_i}^{Loss} = (g_i P_i^{net})^{-d}, \qquad i = 1,\ldots,M, \tag{3.4}$$

where g_i (W^{-1}) is the coding gain-to-receive noise power ratio of the i-th connection, while the positive exponent d measures the diversity gain of the frequency/time interleavers implemented at the Physical layer of the underlying protocol stack. Explicit closed-form expressions for g_i and d may be found, for example, in [20]. Hence, after introducing (3.4) into (3.3), we obtain

$$P_i^{net} = \Omega_i \left(\overline{RTT_i} R_i\right)^{\alpha}, \qquad i = 1,\ldots,M, \tag{3.5}$$

with $\alpha \triangleq (2/d) \geq 1$ and $\Omega_i \triangleq \frac{1}{g_i}\left(\frac{1}{MSS}\sqrt{\frac{2v}{3}}\right)^{\alpha}$, $i = 1,\ldots,M$. Therefore, since the corresponding one-way transmission delay equates: $D_i = L_i/R_i$, the resulting one-way communication energy $\mathscr{E}^{net}(i)$ which is needed for sustaining the i-th virtual link of Fig.3.1 is: $\mathscr{E}^{net}(i) = P_i^{net}(L_i/R_i)$.

Before proceeding, we point out that the α-powered (convex) formula in (3.5) holds *regardless* of the actual (possibly, multi-hop) topology of the adopted physical network (e.g., Fat-tree, BCube, DCell [1]). In fact, the validity of (3.5) relies on the (minimal) assumption that the TCP-based transport connections work in the steady-state (e.g., in the Congestion Avoidance state).

Remark 1 *Other applicable communication formulas*

Other communication infrastructure is available which are listed below. The actual value assumed by P_i^{net} depends on the corresponding transmission rate R_i, noise spectral power density $\mathscr{N}_0(i)$ (W/Hz), bandwidth W_i (Hz) and (nonnegative) gain g_i of the i-th link [50]. The Shannon-Hartley exponential formula:

$$P_i^{net} \equiv P_i^{net}(R_i) = \zeta_i \left(2^{R_i/W_i} - 1\right), \tag{3.6}$$

with $\zeta_i \triangleq \frac{\mathscr{N}_0^{(i)} W_i}{g_i}$, $i = 1,\ldots,M$, and the α-powered formula:

$$P_i^{net} \equiv P_i^{net}(R_i) = \Omega_i \left(\frac{R_i}{W_i}\right)^{1/\alpha}, \tag{3.7}$$

with $\alpha \in]0,1[$, and $\Omega_i \triangleq \frac{\mathscr{N}_0^{(i)} W_i}{(\ln(2))^{1/\alpha} g_i}$, $i = 1,\ldots,M$, are examples of power-rate functions of practical interest [50, Chap.3]. □

3.1.2.3 Reconfiguration costs

Under the per-job hard delay constraints which are typical of streaming services [24], the VMM of Fig.3.1 must carry out two main operations at run-time, namely, virtual machine management and load balancing. Specifically, goal of the virtual machine management is to adaptively control the Virtualization Layer of Fig.3.1. In particular, the set of the

(aforementioned) VM's attributes:

$$\{\Delta, f_c^{max}(i), \Phi_i(\eta_i), \mathscr{E}_c^{\max}(i), L_b(i),\ i = 1,\ldots,M\}, \tag{3.8}$$

are dictated by the Virtualization Layer and, then, they are passed to the VMM of Fig.3.1. It is in charge of the VMM to implement a suitable frequency-scaling policy, in order to allow the VMs to scale up/down in real-time their processing rates f_c's at the minimum cost [97]. At this regard, we note that switching from the processing frequency f_1 to the processing frequency f_2 entails an energy cost of $\varepsilon(f_1;f_2)$ $(Joule)$. Although the actual behavior of the function $\varepsilon(f_1;f_2)$ may depend on the adopted Dynamic Voltage and Frequency Scaling (DVFS) technique, any practical $\varepsilon(f_1;f_2)$ function typically retains the following general properties [71]: *i)* it depends on the absolute frequency gap $|f_1 - f_2|$; *ii)* it vanishes at $f_1 = f_2$ and is not decreasing in $|f_1 - f_2|$; and, *iii)* it is jointly convex in f_1, f_2. A quite common practical model which retains the aforementioned formal properties is the following one:

$$\varepsilon(f_1;f_2) = k_e\,(f_1 - f_2)^2\ (Joule), \tag{3.9}$$

where k_e $(Joule/(Hz)^2)$ dictates the per-VM reconfiguration cost induced by an unit-size frequency switching. Typical values of k_e for current reconfigurable virtualized computing platforms are limited up to few hundreds of $\mu J's$ per $(MHz)^2$ [71]. For sake of concreteness, in the analytical developments of the following section 3.1.2, we directly subsume the quadratic model in (3.9). The generalization to the case of $\varepsilon(.;.)$ functions that meet the aforementioned (more general) analytical properties is, indeed, direct.

3.1.2.4 On the Virtual-to-Physical QoS resource mapping in VNetDCs

Due to the hard delay-sensitive feature of the considered stream services, the Virtualization Layer of Fig.3.1 must guarantee that the demands for the computing $\{f_i\}$ and communication

$\{R_i\}$ resources done by the VLAN are mapped onto adequate (i.e., large enough) computing (e.g., CPU cycles) and communication (i.e., link bandwidths) physical supplies.

In our setting, efficient QoS mapping of the virtual demands $\{f_i\}$ for the computing resources may be actually implemented by equipping the Virtualization Layer of Fig.3.1 with a per-VM queueing system, that implements the (recently proposed) *mClock* scheduling discipline [39, section 3]. Interestingly enough, Table 1 of [39] points out that the *mClock* scheduler works on a per-VM basis and provides: *i)* resource isolation; *ii)* proportionally fair resource allocation; and, *iii) hard* (i.e., *absolute*) resource reservation by adaptively managing the computing power of the underlying multi-rate physical CPUs (see the *Algorithm 1* of [39] for a code of the *mClock* scheduler).

About the TCP-based networking virtualization, several (quite recent) contributions [7, 36, 41, 102] point out that the most appealing property of the emerging data centers for the support of delay-sensitive services is the *agility*, i.e., the capability to assign arbitrary physical server to *any* service *without* experiencing performance degradation. To this end, it is recognized that the virtual network atop the Virtualization Layer should provide a *flat* networking abstraction (see Fig.1 of [5]). The Middleware-layer architecture in Fig.3.1 of the considered VNetDC is aligned, indeed, with this requirement and, then, it is general enough to allow the implementation of agile data centers.

Specifically, according to [5], the VNetDC of Fig.3.1 may work in tandem with *any* Network Virtualization Layer that is capable to map the rate-demands $\{R_i\}$ onto bandwidth-guaranteed end-to-end (possibly, multi-hop) connections over the actually available underlying physical network. Just as examples of practical state-of-the-art Networking Virtualization tools, *Oktopous* [7, Fig.5] provides a switched LAN abstraction atop tree-shaped physical network topologies, while *VL2* [36, section 4] works atop physical Clos' networks. Furthermore, *SecondNet* [41, section 3] and *VNET/P* [102, section 4.2] provide bandwidth-guaranteed virtualized Ethernet-type LAN environments atop TCP-based end-to-end connections. For

this purpose, *SeconNet* implements Port-Switching based Source Routing (PSSR) [41, section 4], while *VNET/P* relies on suitable Layer 2 Tunneling Protocols (L2TPs) [102, section 4.3]. An updated survey and comparison of emerging virtual networking technologies is provided by Table 2 of [5]. Overall, several recent studies support the conclusion that these virtual networking technologies allow the (aforementioned) data center-oriented versions of TCP-like transport protocols to operate in the Congestion Avoidance state (i.e., in the steady-state) during about 99.9% of the overall working time [3, 27]. As a consequence, since the current data centers utilize physical network topologies with high bisection bandwidths and limited over-subscription ratios [1], the corresponding average round-trip-times in (3.3) are very low and less than *1(ms)* [3, 27]. The net effect is that the resulting virtual links of Fig.3.1 typically work under contention-free conditions [69, 5].

Before proceeding, we anticipate that the solving approach developed in sub-section 3.1.5 still holds when the summation: $\sum_{i=1}^{M} \mathscr{E}^{net}(i)$ in (3.10) is replaced by a single energy function $\chi(.)$ which is jointly convex and not decreasing in the variables $\{L_i\}$ (see (3.18.1)). This generalization could be employed for modeling flow coupling effects that may be (possibly) induced by the *imperfect* isolation provided by the Networking Virtualization Layer.

Remark 2 *Discrete computing rates*

Actual VMs are instantiated atop physical CPUs that offer, only a finite *set:* $\mathscr{A} \triangleq \left\{\widehat{f}^{(0)} \triangleq 0, \widehat{f}^{(1)}, \ldots, \widehat{f}^{(Q-1)} \triangleq f_c^{max}\right\}$ *of Q discrete computing rates. Hence, in order to deal with both continue and discrete reconfigurable computing units, we borrow the approach formerly developed, for example, in [72, 96]. Specifically, after indicating by* $\mathscr{B} \triangleq \left\{\widehat{\eta}^{(0)} \triangleq 0, \widehat{\eta}^{(1)}, \ldots, \widehat{\eta}^{(Q-1)} \triangleq 1\right\}$ *the discrete values of* η *that correspond to the frequency set* $\mathscr{A}$*, we build up a* Virtual Energy Consumption *curve* $\widetilde{\Phi}(\eta)$ *by resorting to a piecewise linear interpolation of the allowed Q operating points:* $\left\{\left(\widehat{\eta}^{(j)}, \Phi(\widehat{\eta}^{(j)})\right), j = 0, \ldots, (Q-1)\right\}$. *Obviously, such virtual curve retains the (aforementioned) formal properties and, then, we may use it as the true energy consumption curve for virtual resource provisioning [72]. Unfortunately, being the virtual curve of continuous type, it is no longer guaranteed that*

the resulting optimally scheduled computing rates are still discrete valued. However, as also explicitly noted in [72, 96], any point $(\eta^*, \Phi(\eta^*))$*, with* $\widehat{\eta}^{(j)} < \eta^* < \widehat{\eta}^{(j+1)}$*, on the virtual curve may be actually attained by time-averaging over* Δ *secs the corresponding surrounding vertex points:* $\left(\widehat{\eta}^{(j)}, \Phi(\widehat{\eta}^{(j)})\right)$ *and* $\left(\widehat{\eta}^{(j+1)}, \Phi(\widehat{\eta}^{(j+1)})\right)$*. Due to the piecewise linear behavior of the virtual curve, as in [72, 96], it is* guaranteed *that the average energy cost of the discrete-rate system equates that of the corresponding virtual one over* each *time interval of duration* Δ *(i.e., on a* per-job *basis).* □

3.1.3 Optimal allocation of the virtual resources

In this sub-section, we deal with the second functionality of the VMM of Fig.3.1, namely, the dynamic load balancing and provisioning of the virtualized communication-plus-computing resources. Specifically, this functionality aims at properly tuning the task sizes $\{L_i, i = 1,\ldots,M\}$, communication rates $\{R_i, i = 1,\ldots,M\}$ and computing rates $\{f_i, i = 1,\ldots,M\}$ of the networked VMs of Fig.3.1. The goal is to minimize (on a per-slot basis) the overall resulting communication-plus-computing energy:

$$\mathscr{E}_{tot} \triangleq \sum_{i=1}^{M} \mathscr{E}_c(i) + \sum_{i=1}^{M} \mathscr{E}^{net}(i) \ \ (Joule), \tag{3.10}$$

under the (aforementioned) *hard* constraint T_t on the allowed per-job processing time. This last depends, in turn, on the (one-way) delays $\{D_i, i = 1\ldots M\}$ introduced by the VLAN and the allowed per-task processing time Δ. Specifically, since the M virtual connections of Fig.3.1 are typically activated by the Switch Unit in a parallel fashion [45], the overall two-way communication-plus-computing delay induced by the i-th connection of Fig.3.1 equates: $2D_i + \Delta$, so that the hard constraint on the overall per-job execution time reads as in:

$$\max_{1 \le i \le M} \{2D_i\} + \Delta \le T_t. \tag{3.11}$$

Thus, the overall *Computing and Communication Optimization Problem (CCOP)* assumes the following form:

$$\min_{\{R_i,f_i,L_i\}} \sum_{i=1}^{M} \left\{ \Phi_i \left(\frac{f_i}{f_i^{\max}} \right) \mathscr{E}_i^{\max} + k_e \left(f_i - f_i^0 \right)^2 + 2P_i^{net}(R_i) \left(\frac{L_i}{R_i} \right) \right\}, \tag{3.12.1}$$

$$\text{s.t.: } (L_i + L_b(i)) \leq f_i \Delta, \ \ i = 1, \ldots, M, \tag{3.12.2}$$

$$\sum_{i=1}^{M} L_i = L_{tot}, \tag{3.12.3}$$

$$0 \leq f_i \leq f_i^{max}, \ \ i = 1, \ldots, M, \tag{3.12.4}$$

$$L_i \geq 0, \ i = 1, \ldots, M, \tag{3.12.5}$$

$$\frac{2L_i}{R_i} + \Delta \leq T_t, \ \ i = 1, \ldots, M, \tag{3.12.6}$$

$$\sum_{i=1}^{M} R_i \leq R_t, \tag{3.12.7}$$

$$R_i \geq 0, \ i = 1, \ldots, M. \tag{3.12.8}$$

About the stated problem, the first two terms in the summation in (3.12.1) account for the computing-plus-reconfiguration energy $\mathscr{E}_c(i)$ consumed by the $VM(i)$, while the third term in (3.12.1) is the communication energy $\mathscr{E}^{net}(i)$ requested by the corresponding point-to-point virtual link for conveying L_i bits at the transmission rate of R_i (bit/s). Furthermore, f_i^0 and f_i in (3.12.1) represent the current (i.e., already computed and consolidated) computing rate and the target one, respectively. Formally speaking, f_i is the variable to be optimized, while f_i^0 describes the *current* state of the $VM(i)$, and, then, it plays the role of a known constant. Hence, $k_e \left(f_i - f_i^0 \right)^2$ in (3.12.1) accounts for the resulting reconfiguration cost. The constraint in (3.12.2) guarantees that $VM(i)$ executes the assigned task within Δ secs, while the (global) constraint in (3.12.3) assures that the overall job is partitioned into M parallel tasks. According to (3.11), the set of constraints in (3.12.6) forces the VNetDC to process the overall job within the assigned hard deadline T_t. Hence, since the queue delay of the working

Table 3.1 VNetDC taxonomy

Symbol	Meaning/Role
$L_{tot}\ (bit)$	Job's size
$f_i\ (bit/s)$	Computing rate of $VM(i)$
$L_i\ (bit)$	Task's size of $VM(i)$
$R_i(bit/s)$	Communication rate of the i-th virtual link
$R_t(bit/s)$	Aggregate communication rate of the VLAN
$\Delta(s)$	Per-job maximum allowed computing time
$T_t(s)$	Per-job maximum allowed computing-plus-communication time
$\mathscr{E}_i^{\max}(Joule)$	Per-job maximum energy consumed by $VM(i)$
$f_i^{\max}(bit/s)$	Maximum computing rate of $VM(i)$
$\mathscr{E}_c(i)(Joule)$	Per-job maximum energy consumed by $VM(i)$
$\mathscr{E}^{net}(i)(Joule)$	Per-job communication energy consumed by the i-th virtual link
$\mathscr{E}_{tot}(Joule)$	Per-job total consumed energy

storage of Fig.3.1 is also limited up to T_t, the overall per-job queue-plus-communication-plus-computing delay is limited up to $2T_t$. Finally, the global constraint in (3.12.7) limits up to $R_t\ (bit/s)$ the aggregate transmission rate sustainable by the underlying VLAN of Fig.3.1, so that R_t is directly dictated by the actually considered VLAN standard [5, 45]. Table 3.1 summarizes the main taxonomy used in this sub-chapter.

Remark 3 *On the setup cost of turning OFF the idle servers.*

Due to the noticeable power drained by idle servers, turning the idle servers OFF is commonly considered an energy-effective policy. However, nothing comes for free, so that, although the above conclusion holds under delay-tolerant application scenarios, it must be carefully re-considered when hard limits on the allowed per-job service time T_t are present [53, 79], [55, 92]. In fact, the setup time I for transitioning a server from the OFF state to the ON one is currently larger than 200 seconds and, during the overall setup time, the server typically wastes power on the maximum state [92]. Hence, under real-time constrains, there are (at least) two main reasons to refrain to turn the idle servers OFF. First, the analysis recently reported in [92] points out that turning the idle servers OFF increases. *Second, in*

order to avoid outage-events, the setup time I must be limited up to a (negligible) fraction of the per-job service time T_t (see the hard constraint in (3.12.6)*). Hence, since the tolerated per-job service times of communication-oriented real-time applications are typically limited up to few seconds (see, for example, Tables 1,2 of [104]), the results of the aforementioned performance analysis induce to refrain to turn the idle servers OFF, at least when the setup time I is two orders of magnitude larger than the corresponding service times. However, as a counterbalancing aspect, the performance results reported in the (quite recent) contributions [53, section 3.4], [79, section 3.1], [55] unveil that, under real-time constraints, there is still large room for attaining energy savings by* adaptively *scaling up/down the set of the utilization factors in* (3.2)*. This is, indeed, the energy-management approach we pursue in the following sub-sections, where we focus on the topic of the energy-efficient adaptive configuration of the VNetDC of Fig.3.1.* □

The rest of this sub-chapter is organized as follows. First, we prove that the stated *CCOP* admits a *loss-free* (i.e., optimality preserving) decomposition into two simpler coupled sub-problems, namely, the *CoMunication Optimization Problem (CMOP)* and the *ComPuting Optimization Problem (CPOP)* (see the *Proposition 1* in the sequel). Afterwards, in *Proposition 2* and *Proposition 3*, we give the necessary and sufficient conditions for the feasibility of the *CCOP*. Finally, in *Proposition 4*, we present the solution of the *CCOP*.

3.1.3.1 Solving approach and optimal provisioning of the virtual resources

The *CCOP* in (3.12) is *not* a convex optimization problem. This due to the fact that, in general, each function: $P_i^{net}(R_i)(L_i/R_i)$ in (3.12.1) is not jointly convex in L_i, R_i, even in the simplest case when the power function $P_i^{net}(R_i)$ reduces to an assigned constant P_i^{net}. Therefore, neither guaranteed-convergence iterative algorithms nor closed-form expressions are, to date, available to compute the optimal solution $\left\{\widehat{L}_i, \widehat{R}_i, \widehat{f}_i, i = 1, \ldots, M\right\}$ of the *CCOP*.

However, we observe that the computing energy in (3.12.1) (i.e., the first two terms of (3.12.1)) depends on the variables $\{f_i\}$, while the corresponding network energy (i.e.,

the last term (3.12.1)) depends on $\{R_i\}$. Furthermore, the constraints in (3.12.2)-(3.12.5) depend on $\{f_i\}$ and $\{L_i\}$, while the constraints in (3.12.6)-(3.12.8) depend on $\{R_i\}$ and $\{L_i\}$. Hence, in the sequel, we proceed to the closed-form calculation of the optimal set $\{R_i^*\}$ of the communication rates which minimizes the network energy in (3.12.1) under the constraints (3.12.6)-(3.12.8). Therefore, after expressing $\{R_i^*\}$ as functions of $\{L_i\}$, we introduce the obtained expressions for $\{R_i^*\}$ into (3.12.1). Finally, we perform the minimization of the resulting (3.12.1) over the remaining sets of variables $\{f_i\}$, $\{L_i\}$ under the constraints (3.12.2)-(3.12.5). The feasibility of this approach proves, in turn, that the (aforementioned) *CMOP* and *CCOP* are two *loosely coupled* sub-problems (in the sense of [14, section III]) and that the equations (3.12.2),(3.12.6) are the corresponding coupling constraints. Specifically, for any assigned nonnegative vector $\overrightarrow{L}$ of the task's sizes, the *CMOP* is the (generally nonconvex) optimization problem in the communication rate variables $\{R_i, i = 1,\ldots,M\}$, so formulated:

$$\min_{\{R_i\}} \sum_{i=1}^{M} \frac{2P_i^{net}(R_i)}{R_i} L_i, \tag{3.13.1}$$

$$\text{s.t.: CCOP's constraints in (3.12.6)-(3.12.8).} \tag{3.13.2}$$

Let $\{R_i^*(\overrightarrow{L}), i = 1,\ldots,M\}$ be the optimal solution of the *CMOP* in (3.13), and let

$$\mathscr{S} \triangleq \left\{ \overrightarrow{L} \in \left(\mathbb{R}_0^+\right)^M : \left(\frac{L_i}{R_i^*(\overrightarrow{L})} \right) \leq \frac{(T_t - \Delta)}{2},\ i = 1,\ldots,M; \sum_{i=1}^{M} R_i^*(\overrightarrow{L}) \leq R_t \right\}, \tag{3.14}$$

be the region of the nonnegative M-dimensional Euclidean space which is constituted by all $\overrightarrow{L}$ vectors that meet the constraints in (3.12.6)-(3.12.8). Thus, after introducing the dummy function: $\mathscr{X}(L_1,\ldots,L_M) \triangleq \sum_{i=1}^{M} \frac{2P_i^{net}(R_i^*)}{R_i^*} L_i$, the *CPOP* is formally stated as in:

$$\min_{\{L_i, f_i\}} \sum_{i=1}^{M} \left(\Phi_i \left(\frac{f_i}{f_i^{\max}} \right) \mathscr{E}_i^{\max} + k_e \left(f_i - f_i^0 \right)^2 \right) + \mathscr{X}(\overrightarrow{L}), \tag{3.15.1}$$

$$\text{s.t.: CCOP's constraints in (3.12.2)-(3.12.5) and } \overrightarrow{L} \in \mathscr{S}. \tag{3.15.2}$$

Let $\{L_i^*, R_i^*, f_i^*, i = 1, \ldots, M\}$ indicate the (possibly, empty) set of the solutions of the cascade of the *CMOP* and *CPOP*. The following *Proposition 1* proves that the cascade of these sub-problems is equivalent to the *CCOP*.

Proposition 1 *The CCOP in* (3.12.1)-(3.12.8) *admits the* same *feasibility region and the* same *solution of the cascade of the CMOP and CPOP, that is,* $\{\widehat{L}_i, \widehat{R}_i, \widehat{f}_i, i = 1, \ldots, M\} = \{L_i^*, R_i^*, f_i^*, i = 1, \ldots, M\}$.

proof: By definition, the region $\mathscr{S}$ in (3.14) fully accounts for the set of the *CCOP* constraints in (3.12.6)-(3.12.8), so that the constraint $\overrightarrow{L} \in \mathscr{S}$ is equivalent to the set of constraints in (3.12.6)-(3.12.8). Since these constraints are also accounted for by the *CMOP*, this implies that $\mathscr{S}$ fully characterizes the coupling induced by the variables $\{L_i, i = 1, \ldots, M\}$ between the two sets of constraints in (3.12.2)-(3.12.5) and (3.12.6)-(3.12.8). Therefore, the claim of *Proposition 1* directly arises by noting that, by definition, $\mathscr{X}(L_1, \ldots, L_M)$ is the result of the constrained optimization of the objective function in (3.12.1) over the variables $\{R_i, i = 1, \ldots, M\}$, and $\mathscr{X}(L_1, \ldots, L_M)$ is part of the objective function of the resulting *CPOP* in (3.15.1).

Before proceeding, we point out that (3.14) depends on both $\{L_i\}$ and $\{R_i^*\}$ and, then, it formally captures the effects of the *CMOP* on the resulting *CPOP*. This confirms that the *CMOP* and *CPOP* are, indeed, two coupled (i.e., *not* separated) sub-problems. About the feasibility of the *CMOP*, the following result holds (see [23, section III] for the proof).

Proposition 2 *Let each function* $P_i^{net}(R_i)/R_i$ *in* (3.13.1) *be continue and not decreasing for* $R_i \geq 0$. *Hence, for any assigned vector* $\overrightarrow{L}$, *the following two properties hold:*

i) *the CMOP in* (3.13.1) *is feasible* if and only if *the vector* $\overrightarrow{L}$ *meets the following condition:*

$$\sum_{i=1}^{M} L_i \leq (R_t(T_t - \Delta))/2; \tag{3.16}$$

ii) *the solution of the CMOP is given by the following closed-form expression:*

$$R_i^*(\overrightarrow{L}) = R_i^*(L_i) = (2L_i/(T_t - \Delta)), i = 1, \ldots, M. \tag{3.17}$$

Being the condition in (3.16) necessary and sufficient for the feasibility of the *CMOP*, it fully characterizes the feasible region $\mathscr{S}$ in (3.14). This last property allows us to recast the *CPOP* in the following equivalent form:

$$\min_{\{L_i, f_i\}} \sum_{i=1}^{M} \left(\Phi_i \left(\frac{f_i}{f_i^{\max}} \right) \mathscr{E}_i^{\max} + k_e \left(f_i - f_i^0 \right)^2 \right) + \mathscr{X}(\overrightarrow{L}), \tag{3.18.1}$$

$$\text{s.t.: CCOP's constraints in (3.12.2)-(3.12.5) and (3.16)}, \tag{3.18.2}$$

where (see (3.17)):

$$\mathscr{X}(L_1, \ldots, L_M) = (T_t - \Delta) \sum_{i=1}^{M} P_i^{net} \left(\frac{2L_i}{T_t - \Delta} \right). \tag{3.19}$$

Since the constraint in (3.16) involves only the offered workload, it may be managed as a feasibility condition and this observation leads to the following formal result (proved in the Appendix A of [23]).

Proposition 3 *The CCOP in* (3.12) *is feasible if and only if the CPOP in* (3.18) *is feasible. Furthermore, the following set of* $(M+1)$ *conditions:*

$$L_b(i) \leq \Delta f_i^{\max}, \; i = 1, \ldots, M, \tag{3.20.1}$$

$$L_{tot} \leq \min \left\{ \sum_{i=1}^{M} \left(f_i^{\max} \Delta - L_b(i) \right); \frac{R_t}{2} \left(T_t - \Delta \right) \right\}, \tag{3.20.2}$$

is necessary and sufficient for the feasibility of the CPOP.

Passing to consider the solution of the *CPOP*, after denoting by $\pi_i(.)$ the following dummy function:

$$\pi_i(f_i) \triangleq \left(\frac{\mathscr{E}_i^{max}}{f_i^{max}} \right) \frac{d\Phi_i}{d\eta} \left(\frac{f_i}{f_i^{max}} \right) + 2k_e f_i, \;\; i = 1, \ldots, M, \tag{3.21}$$

and by $\pi_i^{-1}(.)$ its inverse, let $TH(i)$ be the nonnegative threshold so defined:

$$TH(i) \triangleq 2(\partial P_i^{net}(R_i)/\partial R_i)|_{R_i=0}, \; i = 1, \ldots, M. \tag{3.22}$$

Hence, after indicating by $(\partial P_i^{net}(R_i)/\partial R_i)^{-1}(y)$ the inverse function of $\partial P_i^{net}(R_i)/\partial R_i$, the following *Proposition 4* analytically characterizes the optimal scheduler (see the Appendix A for the proof).

Proposition 4 *Let the feasibility conditions in* (3.20) *be met. Let the* $\mathscr{X}(.)$ *function in* (3.19) *be strictly jointly convex and let each function:* $P_i^{net}(R_i)/R_i$, $i = 1,\ldots,M$, *be increasing for* $R_i \geq 0$. *Hence, the global optimal solution of the CPOP is unique and it is given by:*

$$f_i^* = \left[\pi_i^{-1}\left(2k_e f_i^0 + \nu_i^*\Delta\right)\right]_{f_i^{min}}^{f_i^{max}}, \tag{3.23.1}$$

$$L_i^* = \mathbf{1}_{[\nu_i^*>0]}\left(f_i^*\Delta - L_b(i)\right) + \mathbf{1}_{[\nu_i^*=0]}\left[\frac{(T_t-\Delta)}{2}\left(\frac{\partial P_i^{net}(R_i)}{\partial R_i}\right)^{-1}\left(\frac{\mu^*}{2}\right)\right]_+, \tag{3.23.2}$$

where $f_i^{min} \triangleq L_b(i)/\Delta$, *and the nonnegative scalar* ν_i^* *is defined as in*

$$\nu_i^* \triangleq \left[\mu^* - \frac{2\partial P_i^{net}}{\partial R_i}\left(R_i = \frac{2L_i^*}{(T_t-\Delta)}\right)\right]_+. \tag{3.24}$$

Finally, $\mu^* \in \mathbb{R}_0^+$ *in* (3.23.2) *is the unique nonnegative root of the following algebraic equation:*

$$\sum_{i=1}^{M} L_i^*(\mu) = L_{tot}, \tag{3.25}$$

where $L_i^*(.)$ *is given by the r.h.s. of* (3.23.2), *with* μ^* *replaced by the dummy variable* μ.

3.1.4 Adaptive online implementation of the optimal scheduler

About the main structural properties and implementation aspects of the optimal scheduler, the following considerations may be of interest.

3.1.4.1 Hibernation effects

The i-th VM is hibernated when $L_i^* = 0$ (i.e., no exogenous workload is assigned to the $VM(i)$) and the corresponding processing rate f_i^* is *strictly larger* than the minimum one:

$f_i^{min} \triangleq L_b(i)/\Delta$ which is requested for processing the background workload $L_b(i)$ (see (3.12.2)). In principle, we expect that the hibernation of $VM(i)$ may lead to energy savings when k_e , f_i^0's and the ratios $\{P_i^{net}/R_i\}$'s are large, while the offered workload L_{tot} is small. As proved in the Appendix B, this is, indeed, the behavior exhibited by the optimal scheduler, that hibernates $VM(i)$ at the processing frequency f_i^* in (3.23.1) when the following hibernation condition is met:

$$\mu^* \leq TH(i). \tag{3.26}$$

Interestingly, the i-th hibernation threshold in (3.22) is fully dictated by the power-rate behavior of the i-th virtual link of Fig.3.1, while the corresponding hibernated frequency in (3.23.1) depends on the computing system parameters (see (3.21)). This provides additional evidence that the (aforementioned) *CMOP* and *CPOP* are coupled sub-problems, that is, the optimal allocations of the computing and communication resources interact.

3.1.4.2 Adaptive implementation of the optimal scheduler

From an application point of view, remarkable features of the optimal scheduler of *Proposition 4* are that: *i)* it leads to the *parallel* computation (with respect to the i-index) of the $3M$ variables $\{f_i^*, L_i^*, R_i^*, i = 1, \ldots, M\}$; and, *ii)* its implementation complexity is *fully* independent from the (possibly, very large) size L_{tot} of the offered workload. Therefore, the Task scheduler of Fig.3.1: *i)* computes the optimal value μ^* of the Lagrange multiplier, that is, it solves on a per-slot basis the equation in (3.25) by implementing the iterates in (3.27); *ii)* passes the computed μ^* to the VMs; *iii)* gathers the resulting optimal task sizes $\{L_i^*\}$; *iv)* dispatches the corresponding tasks for the processing; and, *v)* gathers the processed tasks and builds up the processed job.

Moreover, in time-varying environments characterized by abrupt and unpredictable time-fluctuations of the input workload L_{tot}, the per-job evaluation of the solution of the nonlinear equations' system in (3.23.1)-(3.25) may be accomplished by running the iterates in (3.27)-

(3.28.3). They are a primal-dual version[3] of the gradient projection algorithm with adaptive step-size (see (3.29),(3.30)). These iterates aim at computing the global minimum of the (convex) Lagrangian function in (A.1). For this purpose, at the beginning of each slot, the iterates receive in input the size L_{tot} of the current job to be processed and, then, they begin to run by starting from the optimal resource allocation already computed at the previous slot. Afterwards, at the convergence, they return the optimal resource allocation in (3.17), (3.23.1), (3.23.2) for the current slot, together with the corresponding current optimal values in (3.24), (3.25) of the Lagrange multipliers. Since the gradient of the corresponding Lagrangian function in (A.1) with respect μ is: $\left(L_{tot} - \sum_{i=1}^{M} L_i\right)$ and the closed-form relationships in (3.23.1), (3.23.2), (3.24) hold, the gradient algorithm reduces to the following quasi-Newton iterates (see the Appendix A for the proof):

$$\mu^{(n)} = \left[\mu^{(n-1)} - \alpha^{(n-1)}\left(\sum_{i=1}^{M} L_i^{(n-1)} - L_t\right)\right]_+, \tag{3.27}$$

with $\mu^{(0)} \geq 0$, $L_i^{(0)} \geq 0$. In (3.27), $n \geq 1$ is an integer-valued iteration index, $\{\alpha^{(n-1)}\}$ is a (suitable) positive step-size sequence, and the following dummy iterates in the n-index also hold (see (3.23.1),(3.23.2) and (3.24)):

$$\nu_i^{(n)} = \left[\mu^{(n)} - \frac{2\partial P_i^{net}}{\partial R_i}\left(\frac{2L_i^{(n-1)}}{(T_t - \Delta)}\right)\right]_+, \tag{3.28.1}$$

$$f_i^{(n)} = \left[\pi_i^{-1}\left(2k_e f_i^0 + \nu_i^{(n)}\Delta\right)\right]_{f_i^{min}}^{f_i^{max}}, \tag{3.28.2}$$

$$L_i^{(n)} = \mathbf{1}_{[\nu_i^{(n)}>0]}\left(f_i^{(n)}\Delta - L_b(i)\right) + \mathbf{1}_{[\nu_i^{(n)}=0]}\left[\frac{(T_t-\Delta)}{2}\left(\frac{\partial P_i^{net}(R_i)}{\partial R_i}\right)^{-1}\left(\frac{\mu^{(n)}}{2}\right)\right]_+. \tag{3.28.3}$$

[3]Formally speaking, the primal-dual algorithm is an iterative procedure for solving convex optimization problems, which applies quasi-Newton methods for updating the primal-dual variables [8, pp.407-408]. See the Appendix A for the analytic details.

Regarding the asymptotic global convergence to the optimum of the iterations in (3.27), (3.28), the following result holds (see the Appendix B for the proof).

Proposition 5 *Let the feasibility condition in* (3.20) *be met and let* $\left\{\alpha^{(n-1)}\right\}$ *in* (3.27) *be positive and vanishing for* $n \to \infty$*, i.e.,* $\lim_{n\to\infty} \alpha^{(n-1)} = 0^+$*. Hence, the iterations in* (3.27)*,* (3.28) *converge to the global optimum for* $n \to \infty$*, regardless of the initial conditions* $\left\{L_i^{(0)} \geq 0\right\}$*,* $\mu^{(0)} \geq 0$*.*

Proposition 5 points out that the adaptive version in (3.27), (3.28) of the proposed scheduler attains the global convergence to the solving point of *Proposition 4*. The numerical plots of section 3.1.5.3 confirm the actual global convergence of the iterations in (3.27),(3.28). In principle, the actual choice of $\left\{\alpha^{(n-1)}\right\}$ in (3.27) also impacts on the rate of convergence and tracking capability of the iterations. At this regard, we note that an effective choice for coping with the unpredictable time-variations of the offered workload is provided by the gradient-descendant algorithm of [60] for the adaptive updating of the step-size in (3.27). In our framework, this updating reads as in [60, equation (2.4)]:

$$\alpha^{(n)} = \max\left\{0; \min\left\{\beta; \alpha^{(n-1)} - \gamma V^{(n-1)}\left(\sum_{i=1}^{M} L_i^{(n-1)} - L_{tot}\right)\right\}\right\}, \tag{3.29}$$

where β and γ are positive constants, while $V^{(n-1)}$ is updated as in [60, equation (2.5)]:

$$V^{(n)} = \left(1 - \alpha^{(n-1)}\right) V^{(n-1)} - \left(\sum_{i=1}^{M} L_i^{(n-1)} - L_{tot}\right), \tag{3.30}$$

with $V^{(0)} = 0$. In practice, the iteration index *n must run faster* than the slot-time T_t. Although the actual duration of each *n*-indexed iteration may depend on the considered application, it should be small enough to allow the iterates in (3.27), (3.28) to converge to the global optimum within a limited fraction of the slot-time T_t. In fact, at the beginning of each slot, the iterates in (3.27),(3.28) are performed by starting from the optimal resource allocation computed at the previous slot. Then, after attaining the convergence, the iterates halt and the

corresponding resource reconfiguration takes place. Just as an example, we anticipate that, in Fig.3.4, the iterates in (3.27),(3.28) run during the time intervals $n \in [0,15]$, $n \in [30,45]$ and $n \in [60,75]$, in order to compute the corresponding new optimal resource allocation. However, during these intervals, reconfiguration of computing-plus-communication resources is *not* performed and, then, no reconfiguration costs are incurred. At the end of these transient phases (e.g., at $n = 15$, $n = 75$ and $n = 75$ in Fig.3.4), a new resource configuration takes place and, then, reconfiguration costs are incurred. Afterwards, the processing of the input jobs by the VMs is performed during the remaining parts of the corresponding time-slots (e.g., during the intervals $n \in [16,30]$, $n \in [46,60]$ and $n \in [61,75]$ in Fig.3.4). Overall, a single reconfiguration action is performed during each slot time T_t, regardless of the actual behaviors of the plots of Fig.3.4 during the transient phases. At this regard, we anticipate that, on the average, about 10-15 iterations (in the n-index) are needed, in order to converge to the steady-state values with an accuracy within 1%.

3.1.5 Performance comparison and sensitivity

We evaluate and compare the per-job average communication-plus-computing energy $\overline{\mathscr{E}}^*_{tot}$ consumed by the proposed optimal scheduler under both synthetically generated and real-world workload traces.

3.1.5.1 Simulated stochastic setting

Specifically, in order to stress the effect of the reconfiguration costs, we begin to model the workload size as an independent and identically distributed (i.i.d.) random sequence $\{L_{tot}(mT_t), m = 0,1,\ldots\}$, whose samples are r.v.'s evenly distributed over the interval $[\overline{L}_{tot} - a, \overline{L}_{tot} + a]$, with $\overline{L}_{tot} = 8\ (Mbit)$. By setting the spread parameter a to 0 $(Mbit)$, 2 $(Mbit)$, 4 $(Mbit)$, 6 $(Mbit)$ and 8 $(Mbit)$, we obtain PMRs of 1 (i.e., the offered workload is of constant size), 1.25, 1.5, 1.75 and 2.0, respectively. About the dynamic setting of $\left\{f_i^0\right\}$ in (3.12.1), at the first round of each batch of the carried out simulations, all the frequencies f_i^0's are

reset. Afterwards, at the m-th round, each f_i^0 is set to the corresponding optimal value f_i^* computed at the previous $(m-1)$-th round. Since each round spans an overall slot-time T_t, all the reported numerical results properly *account* for the reconfiguration cost in (3.9). Furthermore, these results have been evaluated by implementing the adaptive version in (3.27)-(3.30) of the optimal scheduler, with the duration of each n-indexed iteration set to $(T_t/30)$ secs. Finally, the reported numerical results subsume the power-rate function (3.5) at $\alpha = 1.2$, together with the computing energy function in (3.2) at $c = 2.0$ and $b = 0.5$.

3.1.5.2 Impact of the hibernation phenomena and reconfiguration cost

An instance of hibernation of the instantiated VMs is exemplified by the plots of Fig.3.2. They refer to the already described application scenario with $\zeta_i = [0.5 + 0.5(i-1)]$ (mW), $f_i^0 = 0.2 f_i^{max}$, $i = 1, \ldots, M$. Specifically, the upper bars of Fig.3.2 report the (numerically evaluated) optimal average processing rates f_i^*'s, while the lower bars refer to the corresponding optimal average workloads L_i^*'s. An examination of the lower bars of Fig.3.2 points out that only the first nine VMs are permanently loaded, while the corresponding upper bars confirm, indeed, that *all* the available VMs constantly run at positive processing rates. This means that, in the considered application scenario, the last three VMs are hibernated by the optimal scheduler. About this last aspect, Fig.3.3 reports the effects of the reconfiguration costs on $\overline{\mathscr{E}}_{tot}^*$ at $k_e = 0.005$ $(Joule/(MHz)^2)$, $k_e = 0.05(Joule/(MHz)^2)$ and $k_e = 0.5(Joule/(MHz)^2)$. Interestingly, these plots show that $\overline{\mathscr{E}}_{tot}^*$ increases for growing k_e's only for small values of M, while the optimal number M^* of VMs to be instantiated (e.g., the right size of the *NetDC*) decreases for increasing k_e's.

3.1.5.3 Impact of the VLAN setup and tracking capability

Goal of a first set of numerical tests is to evaluate the effects on the per-job average energy $\overline{\mathscr{E}}_{tot}^*$ consumed by the optimal scheduler of the size M of the VNetDC and the setting of the TCP-based VLAN. For this purpose, we pose: $T_t = 5$ (s), $R_t = 100$ (Mb/s), $PMR = 1.25$, $k_e = 0.05$ $(J/(MHz)^2)$, $f_i^{max} = 105$ $(Mbit/s)$, $\mathscr{E}_i^{max} = 60$ $(Joule)$, $\Delta = 0.1$ (s), $\overline{RTT}_i = 700(\mu s)$ and

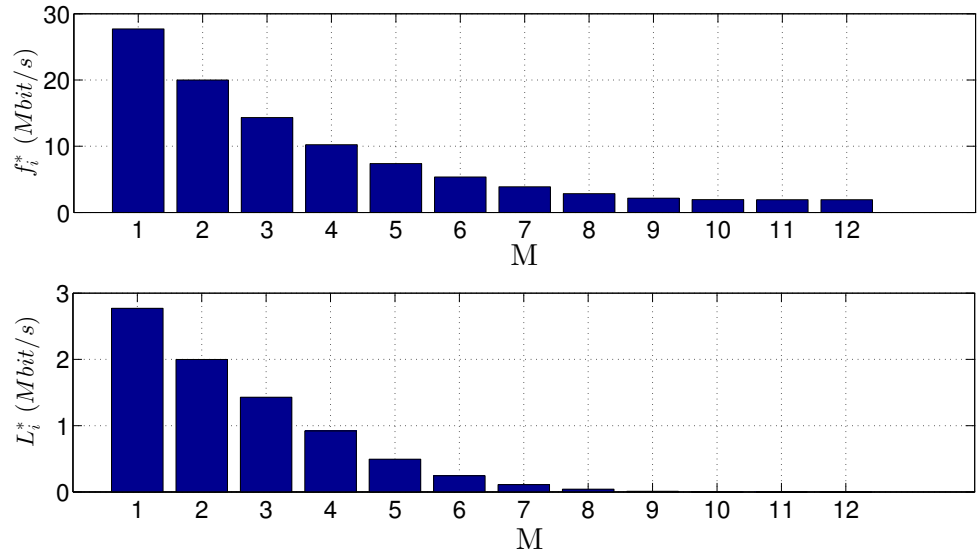


Fig. 3.2 Hibernation phenomena for the application scenario of Sec.3.1.5.3.

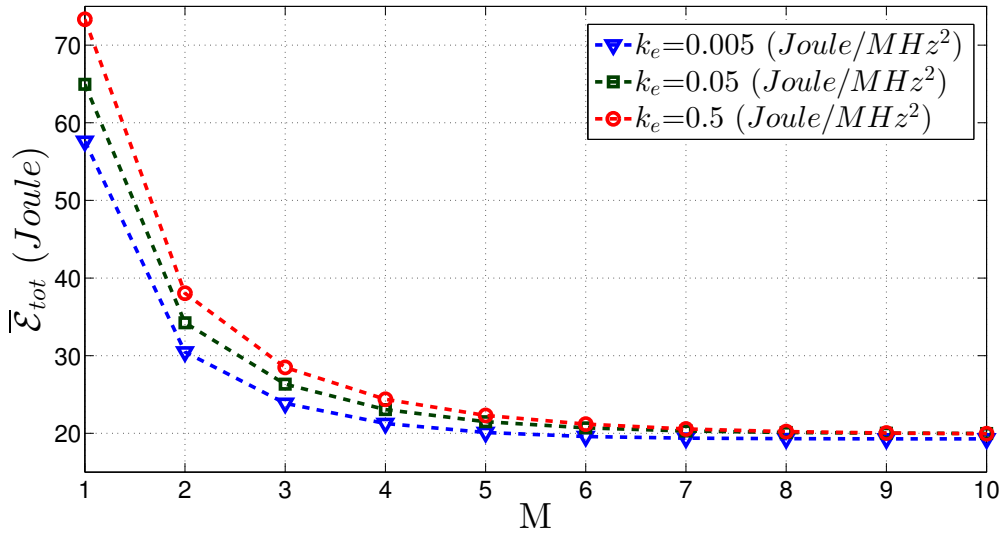


Fig. 3.3 $\overline{\mathscr{E}}_{tot}$ for the application scenario of Sec.3.1.5.3 with $k_e = 0.005$, 0.05 and 0.5 $(Joule/(MHz)^2)$.

$L_b(i) = 0$. Since the quality of the virtual links of Fig.3.1 is captured by the corresponding coefficients $\{\Omega_i\}$ in (3.5), we have numerically evaluated the average consumed energy $\overline{\mathscr{E}}^*_{tot}$ under the following settings: *i)* $\Omega_i = 0.5$ (mW); *ii)* $\Omega_i = [0.5 + 0.25(i-1)]$ (mW); and,

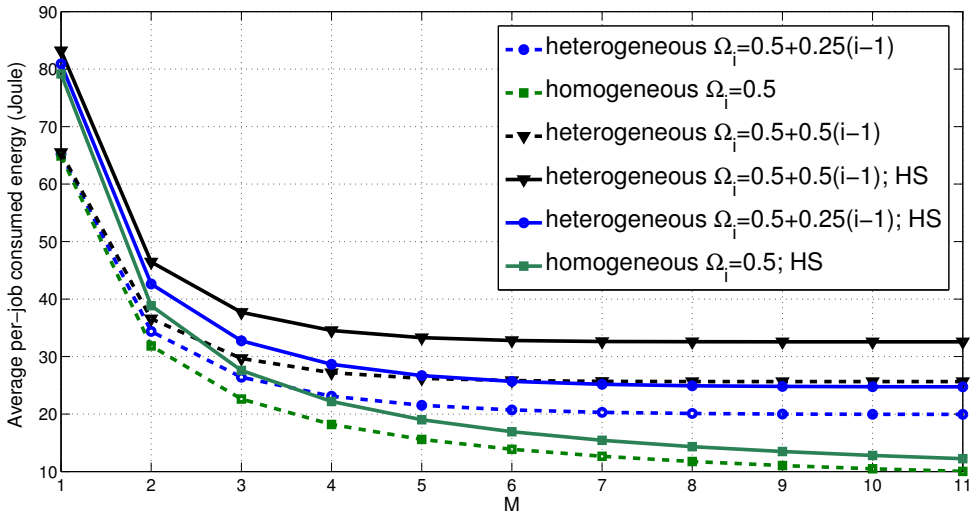


Fig. 3.4 Effects of the link quality on the energy consumptions of the proposed scheduler (dashed plots) and the benchmark hybrid scheduler (continue plots) for the application scenario of Sec.3.1.5.3.

iii) $\Omega_i = [0.5 + 0.5(i-1)]$ (mW), $i = 1,\ldots,M$. The obtained numerical plots are drawn in Fig.3.4. As it could be expected, larger Ω_i's penalize the overall energy performance of the emulated VNetDC. Interestingly, since $\overline{\mathscr{E}}^*_{tot}$ is, by definition, the *minimum* energy when up to M VMs may be turned ON, at fixed P_i^{net}'s, $\overline{\mathscr{E}}^*_{tot}$ decreases for increasing M and, then, it approaches a minimum value that does not vary when M is further increased (see the flat segments of the four uppermost plots of Fig.3.4). In order to evaluate the energy reduction due to the scaling up/down of the communication rates, we have also implemented a benchmark scheduler, namely, the *Hybrid Scheduler (HS)*. It performs, by design, the adaptive minimum-energy allocation of the task sizes $\{L_i\}$ and computing rates $\{f_i\}$, while it holds the communication rates $\{R_i\}$ fixed at the peak values $\{R_i^{max}\}$ which are required for transporting the peak workload with a communication latency limited up to $(T_t - \Delta)$ seconds (see (3.12.6)). From a formal point of view, *HS* implements the solution of the constrained minimization problem in (3.12.1)-(3.12.5) over the variables $\{L_i, f_i\}$ at $R_i = R_i^{max}$. We point out that such a kind of scheduler has been considered in the recent contributions [78, 58]

for copying with the abrupt time-fluctuations of big data streams. The continue plots of Fig.3.3 report the energy consumptions of the *HS*. A comparison of these plots with the corresponding ones of the proposed scheduler shows energy reductions which range from 22% (case of $\Omega_i = 0.5$) to 27% (case of $\Omega_i = 0.5 + 0.5(i-1)$). These results confirm the expectation [96] that noticeable energy savings may be attained by *jointly* scaling up/down the available computing-plus-communication resources.

Finally, in order to appreciate the sensitivity to the parameters β, γ of the adaptive version in (3.29), (3.30) of the proposed scheduler, Fig.3.5 reports the numerically measured time-behavior of $\mu^{(n)}$ in (3.27) when the offered workload abruptly passes from $L_{tot} = 8$ $(Mbit/s)$ to $L_{tot} = 10$ $(Mbit/s)$ at $n = 30$ and, then, it falls out to $L_{tot} = 6$ $(Mbit/s)$ at $n = 60$. The application scenario already described at the beginning of this sub-section has been emulated at $M = 10$ and $\Omega_i = 0.5$ (mW). The solid piece-wise linear plot of Fig.3.5 marks the steady-state optimal values of μ. These optimal values have been calculated by solving the nonlinear equations' system in (3.23.1),(3.23.2),(3.24) and (3.25) through offline centralized Matlab-based numerical methods. The main shortcomings of this centralized approach are that: *i)* it does not scale with the number of the involved variables, so that its computation complexity grows with the number $3M$ of the involved variables (approximately, as $O((3M)^4)$ in our tests); and, *ii)* it is not adaptive, so that the computation of the solution of the considered equations' system must be re-started from scratch at the beginning of each time-slot. Overall, an examination of the plots of Fig.3.5 supports two main conclusions. First, the adaptive version of the optimal scheduler quickly reacts to abrupt time-variations of the workload and it is capable to converge to the corresponding steady-state optimum within about 10-15 iterations. Second, virtually indistinguishable trajectories for $\mu^{(n)}$ are obtained for γ ranging over the interval $[0.1, 0.6]$, so that in Fig.3.5 we report the time-evolutions of $\mu^{(n)}$ at $\beta = 0.008, 0.01, 0.04$, and $\gamma = 0.4$. As already noted in [60], also in our framework, the sensitivity of the adaptive version of the optimal scheduler on β, γ is *negligible*, at

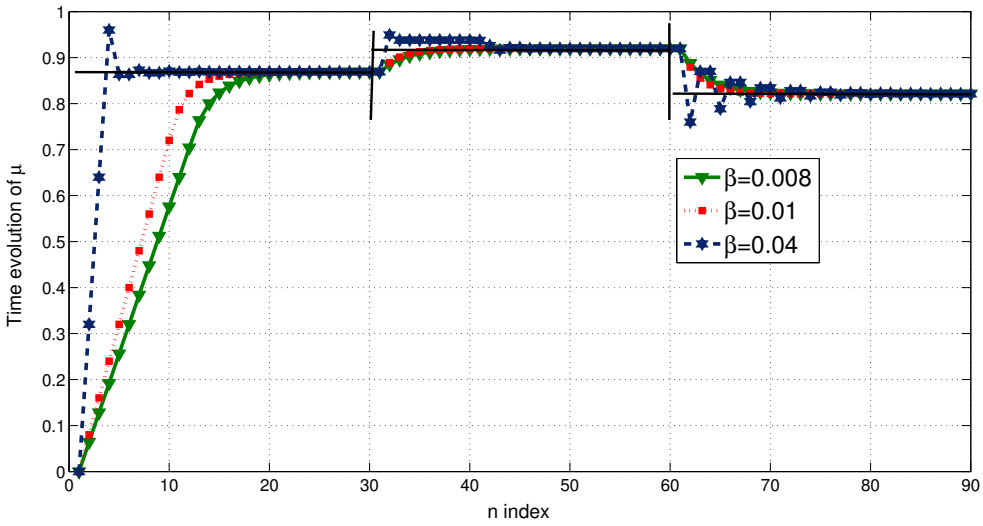


Fig. 3.5 Time evolution (in the n-index) of $\mu^{(n)}$ in (3.27) for the application scenario of section 3.1.5.3.

least for values of β and γ in (3.29) ranging over the intervals $[10^{-3}, 10^{-1}]$ and $[0.1, 0.6]$, respectively.

3.1.5.4 Computing-vs.-communication tradeoff

Intuitively, we expect that small Δ's values give rise to high per-VM computing frequencies (see (3.12.2)), while too large Δ's induce high end-to-end communication rates (see (3.12.6)). However, we also expect that, due to the adaptive power-rate control provided by the optimal scheduler (see (3.23.2),(3.24)), there exists a broad range of Δ's values that attains an optimized tradeoff. The plots of Fig.3.6 confirm, indeed, these expectations. They refer to the application scenario of section 3.1.5.3 at $M = 2, 10$, $\overline{L}_{tot} = 4, 12$ $(Mbit)$, $a = 1, 3$ $(Mbit)$ and $\Omega_i = [0.5 + 0.25(i-1)]$, $i = 1, \ldots, M$ (mW). An examination of these plots supports two main conclusions. First, the energy consumption of the optimal scheduler attains the minimum for values of the ratio (Δ / T_t) falling into the (quite broad) interval $[0.1, 0.8]$. Second, the effects of the ratio (Δ / T_t) on the energy performance of the scheduler are negligible when the considered VNetDC operates far from the boundary of the feasibility region dictated

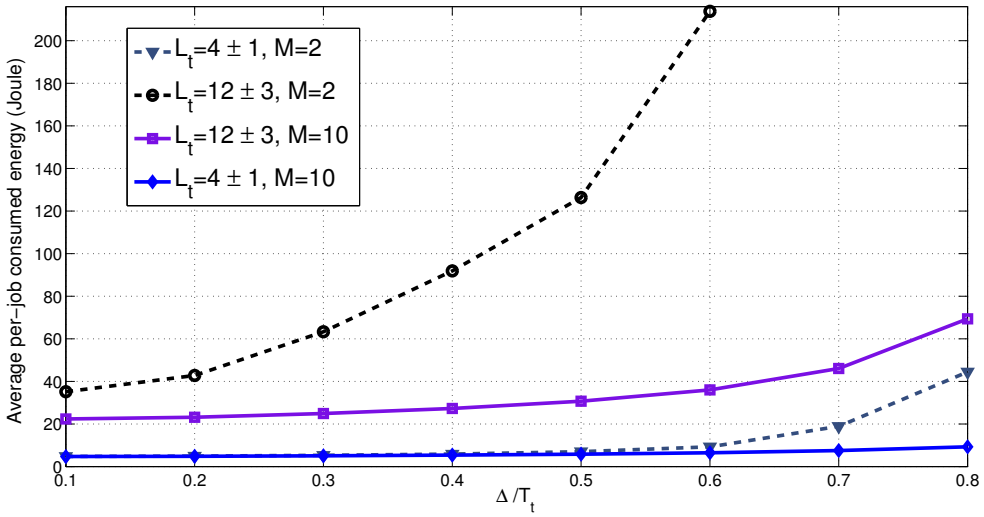


Fig. 3.6 Impact on $\overline{\mathscr{E}}_{tot}$ of the computing-vs.-communication delay tradeoff.

by (3.20.1)-(3.20.2) (see the two lowermost curves of Fig.3.6). Interestingly enough, the increasing behavior of the curves of Fig.3.6 gives practical evidence that the computing energy dominates the overall energy consumption at vanishing (Δ/T_t) (see (3.12.2)), while the network energy becomes *substantial* for $(\Delta/T_t) \to 1^-$ (see (3.12.6)).

3.1.5.5 Performance impact of discrete computing rates

Main goal of this set of numerical tests is to acquire insight into: *i)* the average energy reduction stemming from the exploitation of multi-rate computing techniques; and, *ii)* the energy penalty induced by the frequency-switching over a finite number Q of allowed per-VM processing rates (see (3.9)). For this purpose, the same operating scenario of the above section 3.1.5.3 has been considered at $k_e = 5 \times 10^{-4}$ $(Joule/(MHz)^2)$ and $\Omega_i = [0.5 + 0.25(i-1)]$, (mW) $i = 1,\ldots,M$. The energy curves obtained at: *i)* $Q = +\infty$; *ii)* $Q = 6$ (i.e., discrete computing rates with six computing rates evenly spaced over $[0, f_i^{max}]$); and, *iii)* $Q = 2$ (i.e., each VM may operate at $f_i = 0$ or $f_i = f_i^{max}$), are drawn in Fig.3.7. Interestingly, we have ascertained that the not monotonic behavior of the uppermost curve of Fig.3.7 is the result of two effects that are dominating at $Q = 2$. First, at $Q = 2$, each active VM is forced to

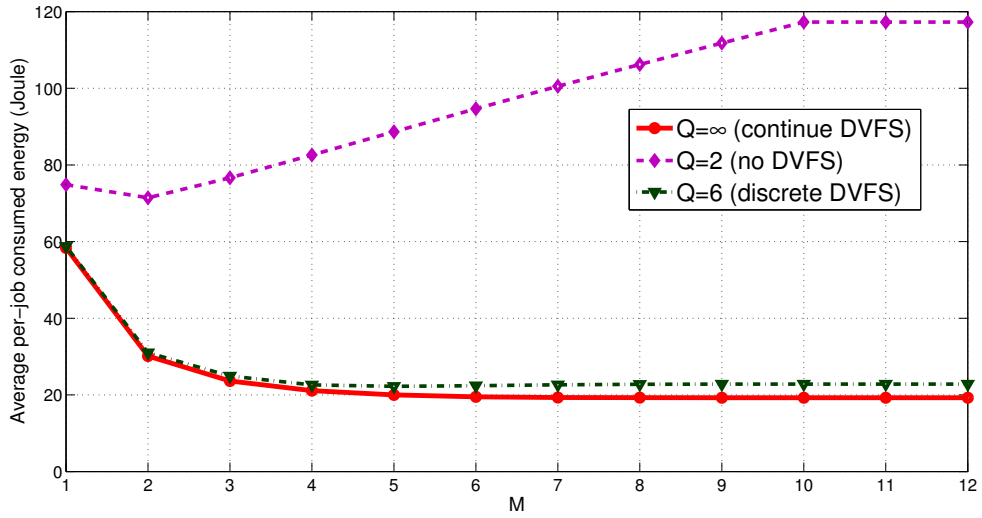


Fig. 3.7 Effects of the computing-rates on the energy performance of the platform of Fig.3.1. The frequency-switching energy penalty in (3.9) is considered.

operate at f_i^{max}, so that the increment in the computing energy induced by the activation of an additional VM scales up as $\mathscr{E}_i^{max}$. Second, at $Q = 2$, the energy overhead in (3.9) required for switching from $f_i = 0$ to $f_i = f_i^{max}$ (or vice versa) is maximum.

As a consequence, the plots of Fig.3.7 support the following three main conclusions. First, at $Q = 2$, the activation of only two VMs (if feasible) stems out as the most energy-saving solution. Second, the relative gap in Fig.3.7 between the uppermost curve (at $M = 2$) and the lowermost one (at $M = 9$) is very large. Third, the relative gap between the two lowermost curves of Fig.3.7 is limited up to 15%.

3.1.5.6 Performance comparison under synthetic workload traces

Testing the sensitivity of the performance of the proposed scheduler on the PMR of the offered workload is the goal of the numerical results of this sub-section. Specifically, they aim at unveiling the impact of the PMR on the average energy consumption of the proposed scheduler and comparing it against the corresponding ones of two state-of-the-art schedulers, namely, the STAtic Scheduler *(STAS)* and the SEquential Scheduler *(SES)*. Intuitively, we

expect that the energy savings attained by dynamic schedulers increase when multi-rate reconfigurable VMs are used, especially at large PMR values. However, we also expect that not negligible reconfiguration costs may reduce the attained energy savings and that the experienced reductions tend to increase for large PMRs. In order to validate these expectations, we have simulated the communication-computing platform of Section 3.1.5.3 at $\Omega_i = 0.9\ (mW)$, and $k_e = 0.005(Joule/(MHz)^2)$.

About the simulated *STAS*, we note that current virtualized data centers usually rely on static resource provisioning, where, by design, a *fixed* number M_s of VMs constantly runs at the maximum processing rate f^{max}[6]. The goal is to constantly provide the exact computing capacity needed for satisfying the peak workload $L_{tot}^{max} \triangleq (\overline{L}_{tot} + a)\ (Mb)$. About the simulated *SES*, it exploits (by design) *perfect* future workload information over a time-window of size $\mathscr{I} \geq 2$ (measured in multiple of the slot period T_t), in order to perform *off-line* resource provisioning at the minimum reconfiguration cost. By design, the *SES* simulates the solution of the following sequential minimization problem:

$$\min_{\{R_i(m),f_i(m),L_i(m)\}} \sum_{m=1}^{\mathscr{I}} \sum_{i=1}^{M} \left\{ \Phi_i \left(\frac{f_i(m)}{f_i^{\max}} \right) \mathscr{E}_i^{\max} + k_e \left[f_i(m) - f_i(m-1) \right]^2 + 2 \frac{P_i^{net}(R_i(m))}{R_i(m)} L_i(m) \right\}, \tag{3.31}$$

under the constraints in (3.12.2)-(3.12.8). We have numerically evaluated the solution of the above sequential problem by implementing in Matlab the dynamic programming tools of [89]. In order to put under the right perspective the following performance comparisons, three main explicative remarks are in order. First, although the *STAS* does not experience reconfiguration costs, it induces resource overbooking. Hence, the resulting per-job average communication-plus-computing energy consumption $\bar{\mathscr{E}}_{tot}^{(STAS)}(Joule)$ gives a worst-case benchmark for numerically evaluating the energy efficiency (i.e., the percent energy gaps) of dynamic schedulers. Second, the capacity planning studies in [6] refer, indeed, to static schedulers and this provides, indeed, an additional motivation for considering the energy

Table 3.2 Average energy reductions attained by proposed (*VNetDC*) and the sequential schedulers over the static one at $k_e = 0.005$ and $f_i^{\max} = 80$.

PMR	**VNetDC at** $\Delta = 0.05\ (s)$	**SES at** $\Delta = 0.05\ (s)$	**VNetDC at** $\Delta = 0.1\ (s)$	**SES at** $\Delta = 0.1\ (s)$
1	0%	0%	0%	0%
1.25	51%	54%	65%	68%
1.5	45%	48%	62%	65%
1.75	63%	67%	57%	62%
2	57%	62%	50%	56%

performance of the *STAS* as a benchmark. Third, since the *SES* ideally assumes perfect knowledge of future workload, it cannot be implemented in practice. However, its energy performance fixes, by design, a best-case benchmark, which gives insight about the room for further performance improvements. Table 3.2 reports the average energy savings (in percent) provided by the proposed scheduler and the sequential scheduler over the static one. Furthermore, in order to test the performance sensitivity of these schedulers on the allowed computing delay, the cases of $\Delta = 0.05\ (s)$ and $0.1\ (s)$ have been considered. In order to guarantee that the static, sequential and proposed schedulers retain the *same* energy performance at $PMR = 1$ (i.e., under *constant* offered workload), the numerical results of Table 3.2 have been evaluated by forcing the sequential and the proposed schedulers to utilize the *same* number of VMs which are activated by the static scheduler. Although this operating condition strongly penalizes the resulting performance of the sequential and proposed schedulers, nevertheless, an examination of the numerical results reported in Table 3.2 leads to three main conclusions. First, the average energy saving of the proposed scheduler over the static one approaches 65%, even when the VMs are equipped with a limited number $Q = 4$ of discrete processing rates and the reconfiguration energy overhead is accounted for. Second, the performance loss suffered by the proposed adaptive scheduler with respect to the sequential one tends to increase for growing PMRs, but it remains limited up to $3\% - 7\%$.

Third, the performance sensitivity of the proposed and sequential schedulers on the allowed computing delay Δ is generally not critical, at least for values of Δ corresponding to the flat segments of the curves of Fig.3.6. Finally, we have experienced that, when the proposed scheduler is also free to optimize the number of utilized VMs, the resulting average energy saving over the static scheduler approaches $90\% - 95\%$ [23, section V].

3.1.5.7 Performance comparison under real-world workload traces

These conclusions are confirmed by the numerical results of this sub-section, that refer to the real-world workload trace of Figs. 3.8,3.9. Specifically, Fig.3.8 reports the real-word traces of Fig.14.a of[90], which represents an 1-hour HTTP-type arrival process really measured at the front-end Web servers of the 1998 Soccer World Cup site. Fig.3.9 draws the recent real-world trace of Fig.2 of [106]. It refers to the I/O workload taken from four RAID volumes of an enterprise storage cluster in Microsoft [106, section IV.A]. The numerical tests carried out in this sub-section refer to the communication-computing infrastructure of section 3.1.5.6 at $k_e = 0.5\ (Joule/(MHz)^2)$ and $\Delta = 1.2\ (s)$. Furthermore, in order to maintain the peak workload still fixed at 16 $(Mbit/slot)$, we assume that each arrival of Figs.3.8,3.9 carries out a workload of 0.533 $(Mbit)$.

Since the (numerically evaluated) PMR of the workload trace of Fig.3.8 is *limited* up to 1.526 and the corresponding time-covariance coefficient ρ is *large* and approaches 0.966, the workload trace of Fig.3.8 is *smoother* than those previously considered in section 3.1.5.6. Hence, we expect that the corresponding performance gaps of the proposed and sequential schedulers over the static one are somewhat less than those reported in section 3.1.5.6 for the case of i.i.d. workload. However, we have tested that, even under the (strongly) correlated workload trace of Fig.3.8, the average energy reductions of the proposed scheduler over the static and hybrid ones still approaches 40% and 19%, respectively. The corresponding energy saving of the sequential scheduler over the proposed one remains limited up to $5\% - 5.5\%$.

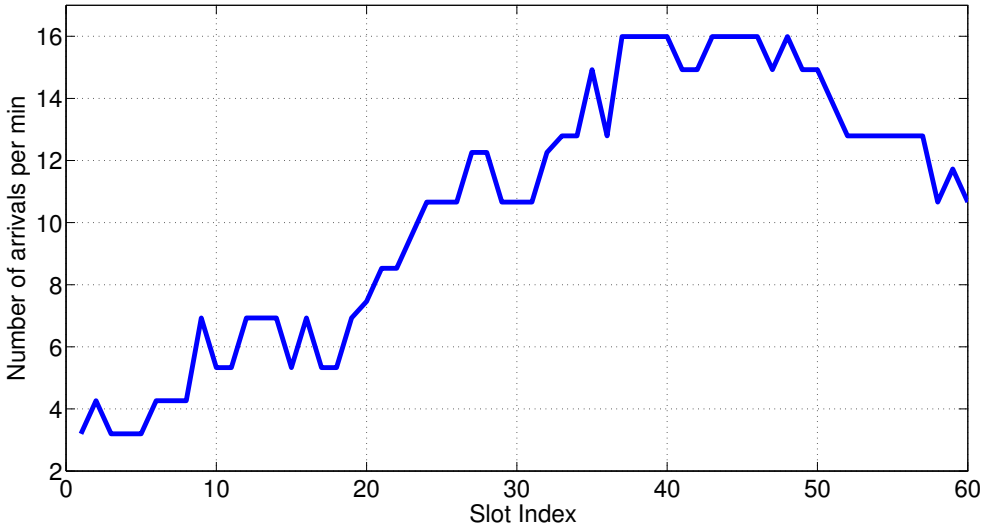


Fig. 3.8 Measured trace of the arrivals of HTTP sessions at the Web servers of the 1998 Soccer World Cup site [90]. The corresponding PMR and covariance coefficient ρ equate 1.526 and 0.966, respectively. The flat peaks reflect buffering delays.

Quite different performance gaps are expected under the workload trace of Fig.3.9. In fact, in this case, the (numerically evaluated) PMR is larger than those of the i.i.d. synthetic traces of section 3.1.5.1 and equates $PMR = 2.49$, while the corresponding time-covariance coefficient is low and approaches $\rho = 0.85$. In agreement with the quasi-random behavior of the trace of Fig.3.9, we have tested that the corresponding energy reduction of the proposed scheduler over the static and hybrid ones approach 68% and 28% respectively, while the corresponding average energy saving of the sequential scheduler over the proposed one is around $5.5\% - 6\%$.

3.2 GreenNetDC Scheduler

In this section, we present a joint-computation-and-communication data center resource provisioning model, called *GreenNetDC*. Special attention is given to the power management techniques that exploit the virtualization technology to save energy. The model not only ensures users the Quality of Service (through Service Level Agreements) but also achieves

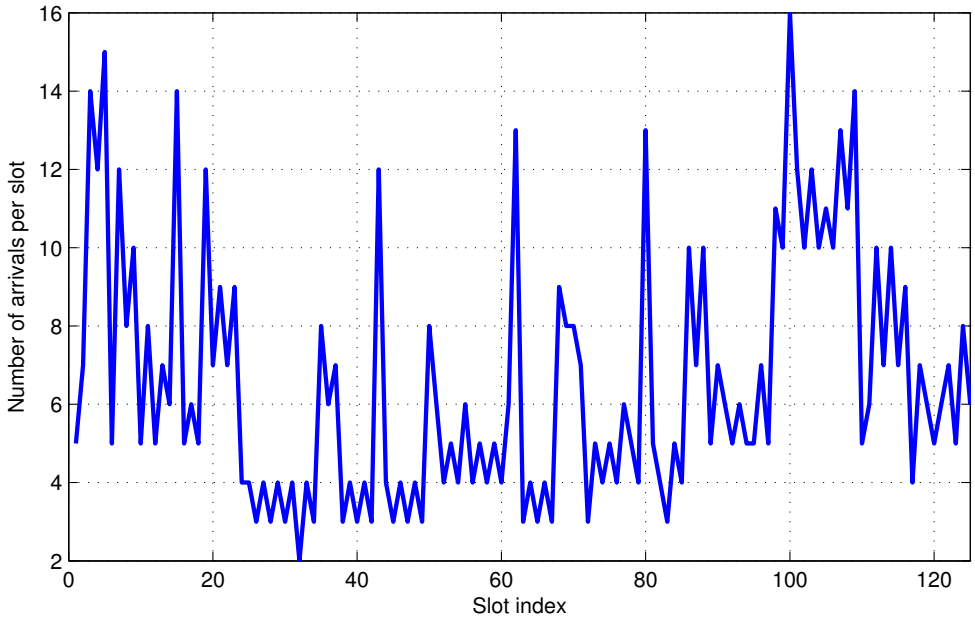


Fig. 3.9 Sampled trace of an I/O workload from an enterprise cluster in Microsoft [106, Fig.2]. The corresponding PMR and time-correlation coefficient are 2.49 and 0.85, respectively.

maximum energy saving and attains green cloud computing goals in a fully distributed fashion by utilizing the DVFS-based CPU frequencies.

In a nutshell, the main objective of this section is to introduce a joint computing-plus-communication framework and develop an efficient scheduler for virtualized data centers that takes into account the allowed discrete processing frequencies for VMs hosted by DVFS-enabled CPU cores. It is important to note that this feature has an internal effect in each CPU while facing online workload. The CPU manages its frequency (i.e., called *old*) according to its current workload and its performance constraints and the new frequency is called *optimum* frequency. The difference between old and optimum frequency is called *reconfiguration frequency* and the reconfiguration frequency incurs energy cost which is called *reconfiguration cost*. Specifically, our work aims to:

- define an architectural framework and principles for energy-efficient VNetDC;

- develop an efficient energy-aware resource allocation and provisioning algorithm in a way that improves the energy efficiency of a data center under hard SLA constraints;

- develop an adaptive version of the scheduling algorithm for energy-efficient mapping of workload quota to the available VMs.

3.2.1 System Model and Considered GreenNetDC Architecture

This sub-section introduces a model which concentrates on the discrete frequencies for each DVFS-based CPU of VMs (the primary version of the work is published in [81]). Specifically, we describe the components of the proposed architecture that is shown in Fig.3.10. In sub-section 3.2.1.1, we describe a general platform of the proposed architecture that minimizes the total energy of the joint-computing-plus-communication operations; sub-section 3.2.1.2 explains the fundamental parts of the proposed model; and, finally, sub-section 3.2.1.3 presents the energy-aware part of the architecture.

3.2.1.1 The GreenNetDC architecture

Motivated by the model considerations, the *GreenNetDC* architecture is composed of two components. The first one, which is shown in the orange box in Fig.3.10, consists of the LAN switch and load balancer. The second one which is shown in bottom of the Fig.3.10 comprises of physical machines or servers which is connected to the switch by a network. Each server consists of a virtualization layer and a physical layer. Virtualization layer enables accessing appropriate resources and deploy virtual machines (VMs) on a server hardware. The physical layer of the *GreenNetDC* provides an elevated view of the underlying physical machine to the virtualization layer. Each server connects to the switch component via an end-to-end communicating link as shown in the Fig.3.10. The intra-cluster communication is supported through message passing. Communication is bounded by the bandwidth and

the power which is related to the transmit rate of each end-to-end link (each bidirectional arrowed line drawn from the switch to the VM in the Fig.3.10).

After the gateway, we have a manager module which carries out two main operations, namely, Virtual Machine Management/Monitor (VMM) and task switching, which aims to perform job decomposition in M VMs with their channels (or end-to-end links). The goal of the VMM is to dynamically manage the VM Layer (i.e., resource dispatching central controller of Fig.3.10), so as to attain an optimal mapping of the available resources onto multiple (possibly heterogeneous) VMs (resource allocation). Computing resources are a collection of large number of servers, (e.g., physical machines (PMs)), each composing of one or more cores, memory, network interface and local I/O. The computing cost is calculated based on the energy consumed during the processing of tasks of the dispatched workload for each VM.

A new job is initiated by an event, that is constituted by the arrival of a file of size L_{tot} (bit) (see the upper part of the Fig.3.10). Due to the real-time nature of the considered applications, full processing of the input file must be carried out within a given (e.g., a priori assigned) deterministic time T_t (s) (s means seconds). Finally, at the end of job's processing, the results are passed to the gateway router of Fig.3.10 and sent back to the cloud applications.

3.2.1.2 Offered workload

Several components play crucial role in the proposed architecture such as compute nodes or VMs, communication channels and offered workload. We assume *GreenNetDC* is composed by M VMs with individual memories (i.e., it means each VM has an autonomous memory; also there is no migration policy considered for jobs/VMs). A server has three general modes: OFF, active and inactive. In OFF mode, the server power is turned OFF. In active mode, it executes tasks and in inactive mode it is ON but does not execute any task. Since the toggle duration for a VM into and out of a power-saving mode (OFF/ON) is relatively high (e.g.,

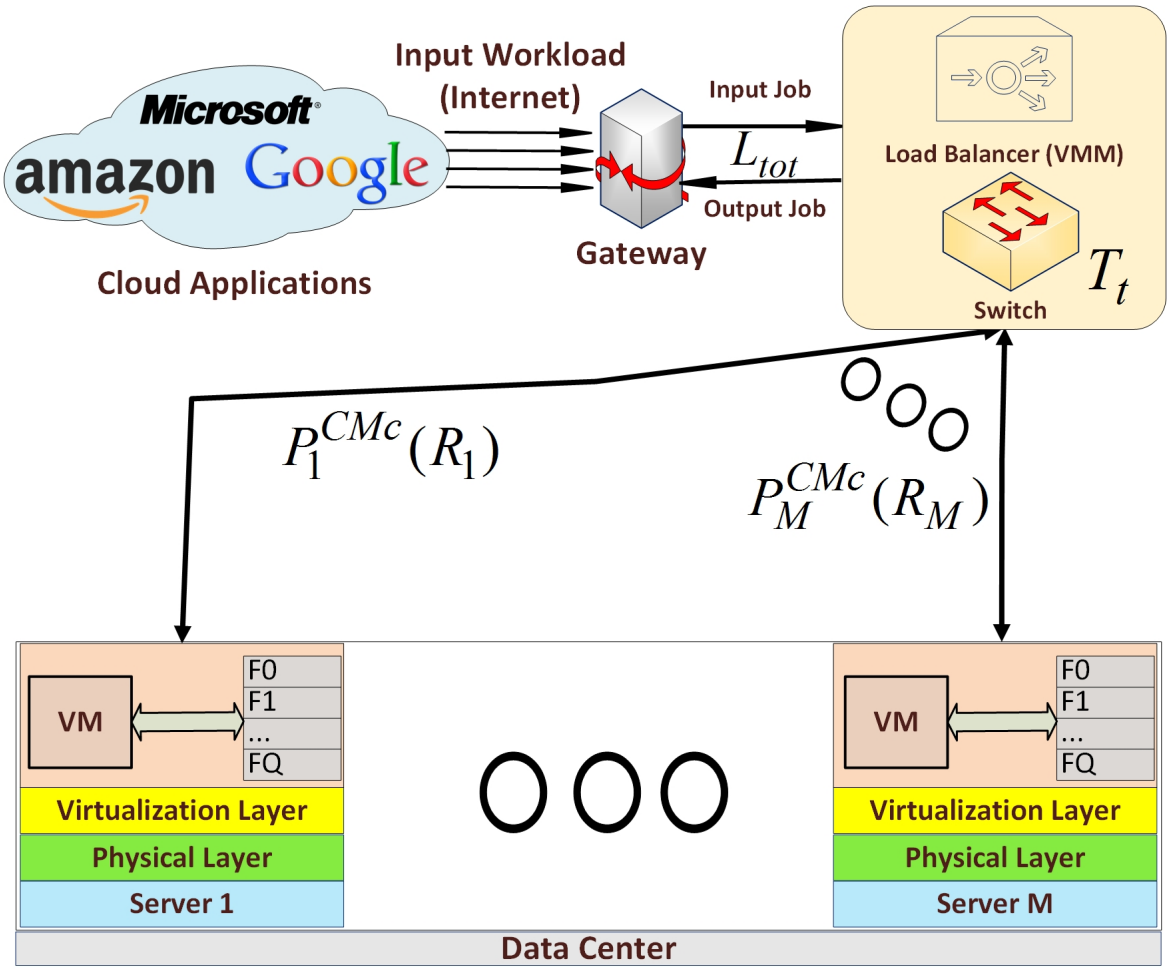


Fig. 3.10 The considered *GreenNetDC* architecture.

20ms in [9]), we use *idle state* (inactive mode) for each VM, which indicates that VM is done its task and now is in the hibernating mode, consuming less energy. In the model, we use this state for each VM which will be explained later.

3.2.1.3 Workload instances

The workload is the received amount of requests from the Internet which should be processed according to their *hard* deadlines T_t. We tested our approach with synthetic and real-world trace workloads which are explained in sub-section 3.2.4. We model the workload as independently identically distributed (i.i.d.). The incoming workload (i.e., L_{tot}) is split into

M quotas (fractions) that must be assigned to the active VMs and passed over contention-free parallel end-to-end links to reach each VM for processing/computing.

3.2.1.4 VM characterization

A VM needs to satisfy the QoS requirements, such as, required virtual processor speed, memory size, storage size, operating system and other hardware/software environmental parameters. In the context of energy-aware scheduling and resource management, we model a VM in terms of required processor frequency and required execution time. Thus, $VM(i)$'s attributes are defined as

$$\left\{T(i), F_i^{idle}, f_i^{max}, P_i^{idle}, A(i), C_{eff}(i), \omega(i)\right\}, \quad i = 1,2,\ldots,M, \tag{3.32}$$

where $T(i)$ is the maximum duration considered for execution time of $VM(i)$. For simplicity, we assume the homogeneous case with $T(i) \equiv T$ for all VMs, although our model can work under heterogeneous computational time for each VM; P_i^{idle} and F_i^{idle} are the idle power and idle frequency for i-th VM, respectively. We consider homogeneous idle powers $P_i^{idle} = P^{idle}$ and homogeneous idle frequencies $F_i^{idle} = F^{idle}$; f_i^{max} is the maximum processing speed of the i-th VM in (bit/s); $A(i)$ and $C_{eff}(i)$ represent the active percentage of gates and effective capacitance load of i-th VM CPU, which we assume to be constant in our model, e.g., $A(i) = A$ and $C_{eff}(i) = C_{eff}$; and, $\omega(i)$ is relative energy cost incurred by the instantiation of the considered VM for the execution of the planned task which is dimensionless (to simplify the model, we consider $\omega(i) = 1.$). From a more practical perspective in $\omega(i)$, larger values for ω make more energy-expensive the execution of the planned task on the considered VM, and $\omega = \infty$ forbids at all the execution of the planned task. Therefore, a suitable setting of the attributes $\omega(i), i = 1,\ldots,M$ may capture task preferences, heterogeneity in the functionalities of the available VMs, as well as underlying task-placement constrains [87]. More explanation on VM frequency is given in next subsection on computing energy model.

3.2.2 Energy consumptions in GreenNetDC

In this sub-section, we detail the energy model, which is categorized into three types of energies: ComPutational cost (denoted *CPc*), REconfiguration cost (denoted *REc*), and CoMmunication cost (denoted *CMc*) which is borrowed from the channel cost of the section 3.1.4.

3.2.2.1 ComPutational Cost in GreenNetDC

The proposed computational energy model is based on the VMs' CPUs characteristics and VM states which was described in sub-section 3.2.1. DVFS technique is applied in VMs processors to reduce the energy consumption by decreasing the VMs' frequencies. It is assumed that each VM can operate at multiple processing frequencies and each (discrete) frequency is active for a specific time [93]. From a more practical perspective in DVFS-based VMs' CPU, $VM(i)$ is able to work with frequency f_i duration t_i. Hence, $f_i t_i$ is the resulting processed workload in *bit*s.

In general, DVFS technology allows to work at various frequencies in its active mode. Q is the number of frequency segmentations between the (real) minimum/ maximum frequency for each VM processor that is able to work with DVFS technology. For instance, AMD Turion MT-34 can operate at six frequencies ranging which are from 800 to 1800 $(Mbit/s)$ [108] while Crusoe RTM-5800 makes available 5 discrete-frequencies falling into the interval 300 to 933 $(Mbit/s)$ [54]. Hence, we can write:

$$\{f_0 \triangleq F^{idle} < f_1 < f_2 < \ldots < f_Q \triangleq f_{max}\}. \tag{3.33}$$

Other components of a system such as memory, bus, etc, operate at a single frequency and consume the same power in both task's active and idle state [108]. According to [80, 33], the dynamic power consumption, P_{dyn} of each VM working at f is given by:

$$P_{dyn} = AC_{eff} f v^2, \tag{3.34}$$

where, A, C_{eff}, f, and v represent the active percentage of gates, effective capacitance load, the VM frequency, and supply voltage, respectively [80, 33]. On the other hand, $\mathscr{E}_{stat}$ is normally proportional to $\mathscr{E}_{dyn}$ and is related to the VM voltages, power for short Circuits of CMOS gates, and leakage power ($P_{leakage}$) which is described in eq. (3.35):

$$\begin{gathered} \mathscr{P}_{stat} = P_{leakage} + P_{Short-Circuit}, \\ \text{where} \\ P_{leakage} = vI_{leakage} \text{ and } P_{Short-Circuit} = AvI_{short}, \end{gathered} \tag{3.35}$$

where in $P_{Short-Circuit}$ the power expended by the 'short-circuit' current which momentarily flows between the supply voltage and ground when the CMOS gates switch. Estimates the duration of the flow, and I_{short} the amount of current and $P_{leakage}$ estimates the power due to current leakage, $I_{leakage}$ estimates this current. The P_{stat} power consumption is usually less than the dynamic power-consumption for high supply voltages and low cycles of tasks for the processor. Nowadays, in practice, the first term dominates so, we negligible compare with dynamic power. On the other hand, the frequency and voltage are correlated to each-other as eq. (3.36) [80, 33]:

$$f \propto \frac{(v - v_{th})^2}{v}, \tag{3.36}$$

where, v_{th} is threshold voltage which is much less than v [80, 33]. Since voltage enters quadratically in eq. (3.34), it would seem that reducing voltage would be the most attractive means of reducing power consumption. As mentioned before, each VM goes into the *idle* mode with P^{idle} power consumption which is a nonnegative, not decreasing, continuous and (strictly) convex function of $P^{idle} \geq 0$. If we consider that total computation time allocated for $VM(i)$ is T, the total computational cost equates

$$\mathscr{E}_{CPc}(i) \triangleq \sum_{j=0}^{Q} AC_{eff} F_{ij}^3 t_{ij}, \; i = 1, \ldots, M, \tag{3.37}$$

where t_{ij} is the time duration of the j-th active discrete range of frequency in $VM(i)$ and $F_{ij}, \forall i = 1,\ldots,M,\ j = 0,\ldots,Q$ denotes the discrete frequencies for each VM in their Q+1 discrete ranges. We assume that the T for the VM can be split and assigned to the discrete frequencies considered for each VM. Therefore, the fraction of the workload can be executed over various active discrete frequencies in time quota. Also, the duration that VM is in idle mode won't receive any workload to process but this time is included in T.

3.2.2.2 REconfiguration cost in GreenNetDC

The basic task of the VMM is to manage suitable frequency-scaling mechanisms, to allow the VMs to adjust in real-time their processing frequency f_i [9]. We note that switching from frequency f_1 to frequency f_2 incurs an energy cost $\mathscr{E}_{REc}(f_1;f_2)$ $(Joule)$ [93, 9]. Although the actual behavior of the switching energy-function $\mathscr{E}_{REc}(f_1;f_2)$ depends on the adopted DVFS technique and the underlying physical CPUs [77]. A common practical model that retains the aforementioned formal properties is borrowed from the (3.9).

Each VM according to the fraction of workload allocated to it, is able to work with some ranges of discrete frequencies that we called *active discrete frequencies*. The switching cost in *GreenNetDC* is split into *internal* and *external* costs. In *internal* cost, the reconfiguration cost of changing the internal-switching among active discrete frequencies of $VM(i)$ is calculated while *external* cost is the different frequencies of the first active discrete frequency for the next incoming workload and the ultimate active discrete frequency for the previous workload. For example, if we consider a VM that has 5 discrete frequencies and this VM is able to work with 3 lowest of its frequencies based on assigned fraction of workload, we consider these three frequencies as a list of active discrete frequencies for this workload fraction and calculate internal and external frequency differences and compute the reconfiguration energy according to eq. (3.9).

3.2.2.3 CoMmunication cost (CMc) in GreenNetDC

In *GreenNetDC* model, we assume that each VM communicates to the scheduler via a *dedicated* (i.e., contention free) reliable link that works at the transmission rate of R_i (bit/s), $i = 1,\ldots,M$ operates in the i-th bidirectional, symmetric, link drains a (fixed) power of $P^{CMc}(i)$ $(Watt)$.

About the actual value of P_i^{CMc}, we note that in order to limit the implementation cost, current data centers utilize off-the-shelf rackmount physical servers which are interconnected by commodity Fast/Giga Ethernet switches. Furthermore, they implement legacy TCP protocols (mainly, the TCP New Reno one) for attaining end-to-end (typically, multi-hop) reliable communication []. In this regard, we note that the data center-oriented versions of the legacy TCP New Reno protocol proposed in [3, 27] allow the managed end-to-end transport connections to operate in the Congestion Avoidance state during 99.9% of the working time, while assuring the same end-to-end reliable throughput of legacy TCP New Reno protocol. Therefore, the communication power cost of the proposed model can be simplified respect to (3.5) to

$$P_i^{CMc}(R_i) = \Omega_i \left(\overline{RTT_i} R_i\right)^2 + P_{Idle}, \qquad i = 1,\ldots,M, \tag{3.38}$$

where $\Omega_i \triangleq \frac{1}{g_i}\left(\frac{1}{MSS}\sqrt{\frac{2v}{3}}\right)^2$ $i = 1,\ldots,M$; MSS (bit) is the maximum segment size; $v \in \{1,2\}$ is the number of per-ACK acknowledged segments; g_i $(Watt^{-1})$ is the coding gain-to-receive noise power ratio of the i-th end-to-end connection; $\overline{RTT_i}$ is the average round-trip-time of the i-th end-to-end connection (e.g., $\overline{RTT_i}$ less than 1 (ms) in typical data centers [27]); and P_{Idle} is the idle power cost for each end-to-end link. Hence, the corresponding one-way transmission delay equates: $D(i) = \sum_{j=1}^{Q} F_{ij} t_{ij}/R_i$, so that the corresponding one-way communication energy $\mathscr{E}^{CMc}(i)$ is:

$$\mathscr{E}^{CMc}(i) \triangleq P_i^{CMc}(R_i)\left(\sum_{j=1}^{Q} \frac{F_{ij} t_{ij}}{R_i}\right) (Joule). \tag{3.39}$$

Table 3.3 *GreenNetDC* notation.

Symbol	Meaning/Role
$F_i^{idle}(bit/s)$	Idle fixed frequency for $VM(i)$
$F_{ij}\ (bit/s)$	Fixed computing rates of $VM(i)$
$F_{ij}^0(bit/s)$	Consolidated computing rate of the VM(i)
$R_i\ (bit/s)$	Communication rate of the i-th virtual link
$R_t(bit/s)$	Aggregate communication rate of the Virtual LAN
$T(s)$	Per-job maximum allowed computing time
$t_i(s)$	The optimum computing time for calculation of $VM(i)$
$t_{ij}(s)$	The optimum computing time for $VM(i)$ with F_{ij}
$T_t(s)$	Per-job maximum allowed total time
$\mathscr{E}_{CPc}(i)(Joule)$	Computing/CPU consumed energy for $VM(i)$
$\mathscr{E}_{REc}(i)(Joule)$	Reconfiguration cost for $VM(i)$
$\mathscr{E}^{CMc}(i)(Joule)$	Network energy consumed for i-th end-to-end link

Specifically, the energy consumption of end-to-end link does not effect the policy of computation, and is completely independent. Table 3.3 summarizes the notations used in the sub-section.

3.2.3 The *GreenNetDC* Optimization Problem and Solution

In this section, we introduce our dynamic load balancing and resource provisioning joint computing-plus-communication approach, respecting the DVFS-based active discrete frequencies and their time fractions for each VM. Specifically, this problem aims at properly tuning the workload fractions $\{f_i t_i, i = 1,\ldots,M\}$, the end-to-end link data transferring rates $\{R_i, i = 1,\ldots,M\}$ and the computing rates $\{f_i, i = 1,\ldots,M\}$ of the networked VMs. The goal is to minimize the overall resulting communication-plus-computing energy, formally defined as

$$\mathscr{E}_{tot} \triangleq \sum_{i=1}^{M} \mathscr{E}_{CPc}(i) + \sum_{i=1}^{M} \mathscr{E}_{REc}(i) + \sum_{i=1}^{M} \mathscr{E}^{CMc}(i)\ \ (Joule), \tag{3.40}$$

where $\mathscr{E}_{REc}(i)$ is the reconfiguration cost of $VM(i)$ under the *hard* constraint T_t on the allowed per-job execution time. The last term of (3.40) depends, in turn, on the (one-way) delays $\{D(i), i = 1 \ldots M\}$ introduced by the Virtual LAN (Fig.3.10's end-to-end virtual links) and the allowed per-task processing time T. Specifically, since the M virtual connections of Fig.3.10 are typically activated by the Switch in a parallel fashion [40], the overall two-way communication-plus-computing delay induced by the i-th connection equates to $2D(i)+T$ and the hard constraint on the per-job execution time reads as in: $\max_{1\leq i\leq M}\{2D(i)\}+T \leq T_t$. In our approach, we propose a (simple) model which accounts for the discrete frequencies for each VM. We know that in real VM processors which are based on DVFS technologies, there are limited ranges for discrete frequencies (i.e., the processors normally have 4-5 frequencies [80]). Therefore, f_i may define Q different discrete values. Generally, the working frequency of each VM is greater than zero or equal to zero (in other words, VM can be in hibernate mode when a request to serve is received). Moreover, the product of optimum frequency and its correlated time for achieving this optimum frequency can be divided into $Q+1$ fixed discrete CPU frequencies and the time can be split into $Q+1$ sub-durations for each discrete frequency.

The goal is to minimize $\mathscr{E}_{tot}$ under SLA induced constraints. Formally speaking, we consider a DVFS-based VM which has a specific range of available frequencies and resolve the nonconvexity problem in the following manner: each VM moves from one of its discrete frequencies to another for computing the assigned workload in its split time. The time breaks into Q discrete unknown time variables for VMs. Therefore, we can use the known vector of frequency for each VM. Also, we have a vector of inter-times related to each VM's frequency (i.e., t_{ij} for j-th time of $VM(i)$) . Each cell of this vector retains the duration VM work with its corresponded frequency for current incoming workload. After computation, each VM sends-back its processed information over the end-to-end link to the VMM which sends it to the cloud over the Internet. The system keeps the list of the active servers to decide for the

next incoming workload from the gateway. It helps to distribute workloads across a number of available servers, to minimize the average energy and execution time. From a more precise and practical perspective, the *Objective Problem (OP)* assumes the following form:

$$\min_{\{R_i,t_{ij}\}} \sum_{i=1}^{M}\sum_{j=0}^{Q}\left[AC_{eff}F_{ij}^{3}t_{ij}\right] + \sum_{i=1}^{M}\mathscr{E}_{REc}(i) + \sum_{i=1}^{M}\sum_{j=1}^{Q} 2P_i^{CMc}(R_i)\left(\frac{F_{ij}t_{ij}}{R_i}\right), \tag{3.41.1}$$

subject to:

$$\sum_{i=1}^{M}\sum_{j=1}^{Q} F_{ij}t_{ij} = L_{tot}, \tag{3.41.2}$$

$$\sum_{j=0}^{Q} t_{ij} = T, \quad i = 1,\ldots,M, \tag{3.41.3}$$

$$\sum_{j=1}^{Q}\frac{2F_{ij}t_{ij}}{R_i} + T \leq T_t, \;\; i = 1,\ldots,M, \tag{3.41.4}$$

$$\sum_{i=1}^{M} R_i \leq R_t. \tag{3.41.5}$$

The above optimization problem can be understood as follows. Eq. (3.41.1) represents the joint computing-plus-communication cost which accounts for the VMs' frequency switching cost for each incoming workload. The equality in (3.41.2) states that the summation of product of discrete frequency by duration for responses for all VMs should be equal to incoming workload L_{tot}. The total time T allowed for the computation is calculated according to the equation (3.41.3). From this equation, the summation of total time-quotas for each active VM is equal to the considered computation time for each incoming workload. The (global) constraint in (3.41.4) forces the Cloud to process the overall jobs within the assigned hard deadline T_t, and thus, it guarantees that the overall communication-computing platform operates in hard real-time. The inequality in (3.41.5) assures the bandwidth consistency and feasibility of the system. It means, VM's R_i must be less than the maximum considered

bandwidth. This equation works like a water-filling problem in order to control aggregate end-to-end link bandwidth load balancing and adjusts the bandwidth for each server according to their workload quotas. The duration of the computing interval for each discrete frequency range (t_{ij} for F_{ij}) is bounded by T.

The reported version of the problem is non-convex due to the non-convexity of the communication terms of the objective function in (3.41.1). Note that the rest of the constraints are affine or can be easily written in convex form in their considered range. It would be likelihood to split these three different activities (e.g., computation, managing the reconfiguration of frequencies and communication) and schedule them separately for an efficient execution. Put it simply, there are three tasks to be considered: *Computation-aware*, *Communication-aware*, and *Reconfiguration-aware* tasks.

From a *Computation-aware* point of view, we can *simply* write the *computation optimizing problem* as follows:

$$\min_{t_{ij}} AC_{eff} \sum_{i=1}^{M} \sum_{j=0}^{Q} F_{ij}^{3} t_{ij}, \tag{3.42}$$

$$\text{subject to (3.41.2), (3.41.3).} \tag{3.43}$$

where t_{ij} is the computational time of $VM(i)$ working at F_{ij}. On the basis of this observation, eq. (3.42) is linear in the control variable t_{ij} and can be easily solved based on two constraints (3.41.2), (3.41.3). We can solve this linear problem by the equation system reported in the *Appendix C*.

From a *Communication-aware* point of view, the third term in (3.41.1) in nonconvex in the variables R_i and $F_{ij}t_{ij}$. Formally speaking, for any assigned nonnegative vector $\overrightarrow{F_{ij}t_{ij}}$ of the workload fractions (job sizes), CMc is generally nonconvex in the communication rate variables $\{R_i, i = 1,\ldots,M\}$, and the resulting optimization problem reads as in:

$$\min_{R_i} \sum_{i=1}^{M} \sum_{j=1}^{Q} 2P_i^{CMc}(R_i) \left(\frac{F_{ij}t_{ij}}{R_i} \right), \tag{3.44}$$

subject to (3.41.4) and (3.41.5). (3.45)

It is proved in *Proposition 6* that this problem can be put in convex as below reported.

Proposition 6 The expression of $\mathscr{E}^{CMc}$ can be put in the following form (see the following proof):

$$\sum_{i=1}^{M}\sum_{j=1}^{Q} 2P_i^{CMc}(R_i)\left(\frac{F_{ij}t_{ij}}{R_i}\right) = (T_t - T)\sum_{i=1}^{M}\sum_{j=1}^{Q} P_i^{CMc}\left(\frac{2F_{ij}t_{ij}}{T_t - T}\right). \tag{3.46}$$

Proof: Let $\{R_i^*(\vec{F}_{ij}\vec{t}_{ij}), i = 1,\ldots,M\}$ be the optimal solution of the eq. (3.44), and let

$$\begin{aligned}\mathscr{C} \triangleq \Bigg\{ &\left(\overrightarrow{F_{ij}t_{ij}}\right) \in (\mathbb{R}_0^+)^M : \left(\sum_{j=1}^{Q} F_{ij}t_{ij}/R_i^*\left(\overrightarrow{F_{ij}t_{ij}}\right)\right) \leq \\ &(T_t - T)/2, i = \{1,2,\ldots,M\}, j = \{1,2,\ldots,Q\}; \\ &\sum_{i=1}^{M}\sum_{j=1}^{Q} R_i^*\left(\overrightarrow{F_{ij}t_{ij}}\right) \leq R_t \Bigg\},\end{aligned} \tag{3.47}$$

be the region of nonnegative M-dimensional Euclidean space constituted by all $\overrightarrow{F_{ij}t_{ij}}$ vectors meeting the constraints in (3.41.4) and (3.41.5). For feasibility and solution of (3.44) we have

i) The CMc in (3.44) is feasible *if and only if* the vector $\overrightarrow{F_{ij}t_{ij}}$ meets the following condition:

$$\sum_{i=1}^{M}\sum_{j=1}^{Q} F_{ij}t_{ij} \leq (R_t(T_t - T))/2 \tag{3.48}$$

ii) The solution of the CMc is given by the following closed-form expression:

$$\begin{aligned}R_i^*\left(\overrightarrow{F_{ij}t_{ij}}\right) \equiv R_i^*\left(\sum_{j=1}^{Q} F_{ij}t_{ij}\right) \equiv \\ \left(\sum_{j=1}^{Q} 2F_{ij}t_{ij}/(T_t - T)\right), i = 1,\ldots,M.\end{aligned} \tag{3.49}$$

For any assigned $\overrightarrow{F_{ij}t_{ij}}$, the objective function in (3.44) is the summation of $M(Q+1)$ nonnegative terms, where the ij-th term depends only on R_i for all j. Thus, being the

objective function in (3.44) separable and its minimization may be carried out component-wise. Since the ij-th term in (3.44) is increasing in R_i and the constraints in (3.41.4) and (3.41.5) must be met, the ij-th minimum is attained when the constraints in (3.41.4) and (3.41.5) are binding, and this proves the validity of (3.48). Finally, the set of rates in (3.49) is feasible for the CMc *if and only if* the constraint in (3.41.5) is met, and this proves the validity of the feasibility condition in (3.49).

Moreover, the end-to-end links' power cost $\sum_{j=1}^{Q} 2P_i^{CMc}(R_i)\left(F_{ij}t_{ij}/R_i\right)$ is the product of the end-to-end link formula which is based on TCP New Reno (i.e., a specific type of TCP congestion-avoidance algorithm which is applied for congestion control in the Internet that we modeled our one-way transmission-plus-switching operation over each end-to-end link or link in the proposed model) in (3.38) and is continuous, nonnegative and nondecreasing for $R_i > 0, \forall i \in \{1,\ldots,M\}$, with the multi-variable coefficient which can be feasible if only the following equation holds (i.e., we use "$\rightarrow$" which means *implies*):

$$\sum_{j=1}^{Q} \frac{2F_{ij}t_{ij}}{R_i} + T \leq T_t \rightarrow \left(\sum_{j=1}^{Q} \frac{F_{ij}t_{ij}}{R_i}\right) \leq \frac{(T_t - T)}{2}. \tag{3.50}$$

Equation (3.50) is obtained by manipulating equation (3.41.4). To make the optimization problem easier to solve, we recast the second control variable by rewriting R_i based on another control variable (t_{ij}) as follows:

$$\sum_{j=1}^{Q} \frac{2F_{ij}t_{ij}}{R_i} + T \leq T_t \rightarrow R_i \geq \sum_{j=1}^{Q} \left(\frac{2F_{ij}t_{ij}}{T_t - T}\right). \tag{3.51}$$

So, we can apply the result of equations (3.50) and (3.51) in the third term of the objective function, to achieve the following

$$\sum_{i=1}^{M} 2P_i^{CMc}(R_i)\left(\sum_{j=1}^{Q} \frac{F_{ij}t_{ij}}{R_i}\right) = (T_t - T)\sum_{i=1}^{M} P_i^{CMc}\left(\sum_{j=1}^{Q} \frac{2F_{ij}t_{ij}}{T_t - T}\right), \tag{3.52}$$

To recap, the end-to-end link function $\mathscr{E}^{CMc}(.)$ which is based on two control variables ($\mathscr{G}(R_i;t_{ij})$) can be written at the following result for changing the second control variable R_i to a function of other control variable in eq. (3.53):

$$\mathscr{E}^{CMc}(i) = \mathscr{G}(R_i;t_{ij}) \triangleq \mathscr{H}(t_{ij}). \tag{3.53}$$

The new formula for energy-aware communication end-to-end link just depends on the summation of time variables for each VM and the main function ($\mathscr{H}(.)$) can be written according to the equation (3.38). Thus, this proves the third term in (3.41.1) is *convex*. □

The following *Proposition 7* describes the feasibility conditions for the optimization problem in (3.40).

Proposition 7 The following set of conditions is *necessary and sufficient for the feasibility* of the optimization problem in (3.41.1)-(3.41.5) (see the proof):

$$L_{tot} \leq R_t \frac{(T_t - T)}{2}, \tag{3.54.1}$$

$$L_{tot} \leq \sum_{i=1}^{M} T f_i^{max}. \tag{3.54.2}$$

Proof: The proof of eq. (3.54.1) stems from the constraint in equation (3.41.4):

$$\begin{aligned} &\sum_{j=1}^{Q} \frac{2F_{ij}t_{ij}}{R_i} + T \leq T_t \overset{(a)}{\rightarrow} \sum_{j=1}^{Q} F_{ij}t_{ij} \leq \frac{(T_t - T)}{2} R_i \overset{(b)}{\rightarrow} \\ &\sum_{i=1}^{M}\sum_{j=1}^{Q} F_{ij}t_{ij} \leq \frac{(T_t - T)}{2} \sum_{i=1}^{M} R_i \overset{(c)}{\rightarrow} L_{tot} \leq R_t \frac{(T_t - T)}{2} \end{aligned} \tag{3.55}$$

where: (a) in (3.55) we swap the equation positions and calculate $\sum_{j=1}^{Q} F_{ij}t_{ij}$ based on R_i (i.e., we know that (3.41.4) represents a constraint that exists and is true); (b) the left term (which is positive) is less than the right term (which is positive too, because we respect the hard deadline T_t and T never accomplishes T_t and won't be larger than it), therefore, we derive summation for all discrete defined times fraction of VMs, and this equation can be

derived and expanded; finally, in (c) the left hand of the inequality: $\sum_{i=1}^{M}\sum_{j=1}^{Q} F_{ij}t_{ij}$ according to the second constraint of *OP* (i.e., (3.41.2)) is equal to L_{tot} and the right hand inequality, $\sum_{i=1}^{M} R_i$, stems from the equation (3.41.5) which is less than R_t and it is obvious that left hand of the equation is less than $R_t(T_t - T)/2$. We next prove the second part of (3.54).

To prove the equation (3.54.2), we start from the modified equation (3.41.3) in *OP*, in which F_{ij} and t_{ij} are positive, simultaneously. Thus, we have:

$$\sum_{j=0}^{Q} t_{ij} = T \overset{(d)}{\rightarrow} \sum_{j=1}^{Q} t_{ij} \leq T \overset{(e)}{\rightarrow} f_i^{max}\sum_{j=1}^{Q} t_{ij} \leq T f_i^{max} \overset{(f)}{\rightarrow} \sum_{i=1}^{M}\left(f_i^{max}\sum_{j=1}^{Q} t_{ij}\right) \leq \sum_{i=1}^{M}(T f_i^{max}) \overset{(g)}{\rightarrow}$$
$$\sum_{i=1}^{M}\sum_{j=1}^{Q}\left(F_{ij}t_{ij}\right) \leq \sum_{i=1}^{M}\left(f_i^{max}\sum_{j=1}^{Q} t_{ij}\right) \leq \sum_{i=1}^{M}(T f_i^{max}) \overset{(h)}{\rightarrow} \sum_{i=1}^{M}\sum_{j=1}^{Q}\left(F_{ij}t_{ij}\right)$$
$$\leq \sum_{i=1}^{M} T f_i^{max} \overset{(i)}{\rightarrow} L_{tot} \leq \sum_{i=1}^{M} T f_i^{max}. \tag{3.56}$$

where: (d) in (3.56) holds because the summation of the VM's time fractions without the considering the duration for the idle mode should be equal or less than total hard-limit assigned for each server; (e) and (f) in (3.56) represent the product of positive values f_i^{max} and summation over M VMs for the calculated inequality, simultaneously; (g) shows that the left hand side of the inequality achieved after (f) is higher than the constraint in (3.41.2) (i.e., it is clear $F_{ij} \leq f_i^{max}$ and as a result $\sum_{i=1}^{M} F_{ij} \leq \sum_{i=1}^{M} f_i^{max}$) and (h) indicates that this inequality can be simplified as the result of (i) and this proves the validity of the second feasibility condition of *OP*. □

From a *Reconfiguration-aware* point of view, the second term in (3.41.1) (i.e., $\mathscr{E}_{REc}(i)$) can be split into two reconfiguration costs. The first one is the cost of changing discrete frequencies of $VM(i)$ from F_{ij} to $F_{i(j+k)}$ (i.e., k steps movement to reach to the next *active discrete frequency*) and span $t_{i(j+k)}$ seconds. The second cost is the reconfiguration cost for the switching from the current final *active discrete frequency* (F_{ij}^0) of $VM(i)$ to the first

active discrete frequency of $VM(i)$ in the next slot time. Note that *active discrete frequencies* are found based on the their related times-quota variables. In other words, we track each *active discrete frequency* (F_{ij}) while the constraint ($t_{ij} > 0$) is met. Now, we have a list of *active discrete frequencies* for each VM and per-slot basis. Passing from current active discrete frequency to another active discrete frequency affects the reconfiguration cost. We use the FCFS (First Come, First Serve) technique for visiting each frequency in the active discrete frequency list of each $VM(i)$. It means that in $VM(i)$'s active discrete frequencies list, we start from the first active discrete frequency: F_{ik} and move to the second active discrete frequency in the list: $F_{i(k+1)}$. Therefore, we calculate the difference as follows: $\Delta F_{ik} \triangleq F_{i(k+1)} - F_{ik}$, and the reconfiguration cost is resulting $k_e \Delta F_{ik}^2$. We continue until end of $VM(i)$' list. If we consider homogeneous VMs, the total cost of internal-switching for all VMs is: $k_e \sum_{i=1}^{M} \sum_{k=0}^{K} (\Delta F_{ik}^2)$, where $k \in \{0, 1, \ldots, Q\}$, $K \leq Q$ is the number of active discrete frequencies for $VM(i)$ (*the first type* of reconfiguration cost).

On the other hand, the external-switching cost is calculated as multiplication of k_e with the quadratic differences between the last active discrete frequency of $VM(i)$ for the current workload and the primary active discrete frequency of $VM(i)$ in the next incoming workload, which is denotes as Ext_Cost. In a nutshell, the total reconfiguration energy can be written as: $\sum_{i=1}^{M} \mathscr{E}_{REc}(i) \equiv k_e \sum_{i=1}^{M} \sum_{k=0}^{K} (\Delta F_{ik})^2 + k_e \sum_{i=1}^{M} Ext\_Cost$. In the worst case, $K = Q$ and for external-switching, we need to move Q steps to F_0. Formally speaking, we need to visit all the possible active discrete frequencies of each $VM(i)$(internal-switching cost: $k_e M \sum_{k=0}^{Q} (\Delta F_k)^2$), and external-switching cost is: $k_e M (F_Q^t - F_0^{t-1})^2$, where t is the current time and $(t-1)$ is the previous time (refers to the previous incoming workload) and F_Q^t and F_0^{t-1} express the maximum discrete frequency of each VM (i.e., we assume VMs are homogeneous) and idle discrete frequency (the first frequency range of each VM) while it received the t-th workload and the $(t-1)$-th workload, simultaneously. Overall, the Algorithm 1 summarizes the steps of the proposed method for each incoming workload L_{tot}.

Algorithm 1 *GreenNetDC* algorithm

Set (M, T, T_t) ▷ General SLA parameters
Set $(Q, F_0, f^{max}, A, C_{eff})$ ▷ Comp. parameters of the M VMs
Set $(k_e, R_t, W_i, P^{idle}, \zeta$ or $\Omega)$ ▷ Chanl. parameters of the M VMs
Recieve L_{tot}
Test the feasibility conditions in (3.54.1), (3.54.2)
$b_1 \triangleq \left(L_{tot} \leq R_t \frac{Q(T_t - T)}{2}\right)$
$b_2 \triangleq \left(L_{tot} \leq \sum_{i=1}^{M} f^{max}(i) T\right)$
if $\sim (b_1 \& b_2)$ **then**
    error('Program is not feasible')
else
Specific optimization problem:
    Minimize $(\mathscr{E}_{tot}, \mathscr{E}_{CPU}, \mathscr{E}_{Reconf}, \mathscr{E}^{net})$
    Subject to:
    Constraints (eqs. (3.41.2)- (3.41.5))
end if
return $\mathscr{E}_{tot}, \mathscr{E}_{CPU}, \mathscr{E}_{Reconf}, \mathscr{E}^{net}$

Remark 4 *Implementing Complexity analysis of OP*

Let $\mathscr{W}$ be the size of the incoming workload, M be the number of engaged VMs in the problem, and Q be the number of discrete CPUs' frequencies. Time complexity of GreenNetDC algorithm for each server is $O(\mathscr{W} \times Q)$. In addition, since all servers are working in parallel, the overall time complexity of the proposed algorithm for processing the offered workload $\mathscr{W}$ is $O(MQ\mathscr{W})$. □

3.2.4 Simulation Results and Performance Comparisons

This section presents the tested performance of the *GreenNetDC* for a set of offered workloads and compares the simulation results with the IDEAL no-DVFS techniques our benchmark presented in [68], the Standard (or Real) available DVFS-enabled technique (currently, one of the methods that being used in the DVFS-enabled data centers) [54], the Networked data centers approach or *NetDC* [23], the *Lyapunov* method in [91] and the *HybridNetDC* [17] which take into account reconfiguration and communication costs. Note that we add the idle mode to processors and end-to-end links in these methods (see Appendix C for

details). Besides, *HybridNetDC* compared to *NetDC* has higher energy consumption in terms of energy provisioning due to fixed end-to-end link power for each VMs. We also tested *GreenNetDC* with a state-of-the-art scheduler (or load balancer): the STAtic Scheduler (STAS) (i.e., See Appendices B and D for details) [9]. Therefore, the proposed approach is able to apply in real data centers unless *NetDC* that works based on calculated proportional of real frequency, which cannot deploy in real environment. We should emphasize that *NetDC* [23] and IDEAL no-DVFS technique [68] are presented in order to work with the continues ranges of frequencies which is not feasible, unrealistic and impossible in reality, and also while the proposed scheduler could be one of the best practical solutions in networked data centers.

3.2.4.1 Experimental Setup

In order to evaluate the per-job average communication-plus-computing energy $\bar{\mathscr{E}}_{tot}$ consumed by the proposed scheduler, we implemented a prototype of the adaptive scheduler under paravirtualized Xen 3.3 as VMM and Linux 2.6.18 as guest OS kernel (see Fig.3.10). The adaptive scheduler is implemented in SW at the driver domain (i.e., Dom0) of the legacy Xen 3.3. Out of approximately 1100 lines of SW code needed for implementing the proposed scheduler, 45% is directly reused from existing Xen/Linux code. The reused code includes part of the Linux's TCP New Reno congestion control suite and Xen's I/O buffer management.

The implemented experimental setup comprises of four quad-core Dell PowerEdge servers equipped with 3.06 GHz Intel Xeon CPU and 4GB of RAM. All servers are connected through commodity Fast Ethernet NICs. In all carried out tests, we configure the VMs with 512MB of memory and utilize the TCP New Reno protocol for implementing the needed VM-to-VM transport connections. The simulation is done using a simulator we developed, named TEST-DVFS, which works by using CVX solver, the state-of-the-art Stanford optimizing solver over Matlab [35]. TEST-DVFS simulates the algorithm in DVFS-enabled data centers

by enabling DVFS functionalities not only for the components performance model but also for the offered workloads and energy models.

3.2.4.1.1 TEST-DVFS implementation The goal of the implemented testbed (e.g., the TEST-DVFS one) is to illustrate the efficiency of our model and the proposed algorithm (*GreenNetDC*) compared to the aforementioned techniques.

The TEST-DVFS testbed consistes of the following modules:

- *Workload module*: This module is developed to simulate various types of offered workloads to the system, which is related to the workload distributions and their (in)dependency (from) to each-other and the various peak-to-mean ratio (PMR) of the workload (i.e., synthetic and real traces).

- *Component module*: All the considered components of the system, VMs, channels, DVFS, are implemented in this module for each VM in a PM (see Fig.3.10). Therefore, each VM is able to work at a specific discrete frequency over a given interval.

- *Working module*: The working module focuses on the energy model, scheduling types and network topology.

Table 3.4 summarizes the test parameters for the first scenario and their default values in TEST-DVFS simulator, where F1=$\{0.15,1.867, 2.133, 2.533, 2.668\}$ $(Mbit/s)$, F2=$\{0.3, 0.533, 0.667, 0.8, 0.933\}$ $(Mbit/s)$ and F3=$\{5, 10, 20, 30, 100\}$ $(Mbit/s)$. The discrete frequency for the $F1$ is taken from Intel Nehalem Quad-core Processor [93], and for $F2$ is taken from power-scalable real cluster Crusoe with CPU type TM-5800 in [54] and $F3$ represents synthetic simulated discrete frequency. Scenario two and scenario three focus on multiple VMs, as detailed in Table 3.5 and Table 3.6, respectively. The main difference between the considered scenarios is the corresponding communication parameters. The second one uses eq. (3.4) and third scenario uses eq. (3.7) for the network cost, respectively.

Table 3.4 Default values for the first test scenario.

Parameter	**Parameter**
$VMs = M = \{1,\ldots,12\}$	$T_t = 5\ (s)$, $T = 0.1\ (s)$
$R_t = 100\ (Mbit/s)$	$C_{eff} = \{1, 10, 100\}$
$k_e = \{0.005, 0.05, 0.5\}\ (Joule/(Mbit/s)^2)$	
F={F1, F2, F3}	$A = 1$, $Q = 4$
$P_i^{idle} = \{5, 50\}\ (mW)$	$\overline{RTT}_i = 700(\mu s)$
$\Omega_i = \{5, 50\}\ (mW)$	$P_{Idle} = 5\ (W)$

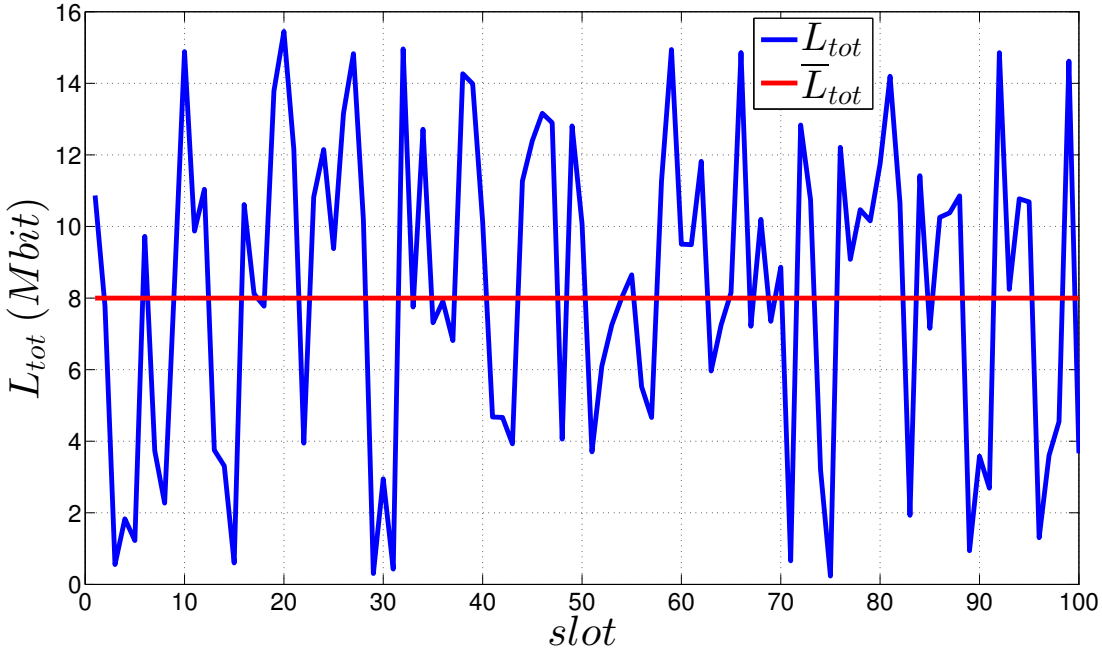
Table 3.5 Default values for the second test scenario.

Parameter	**Parameter**
VMs=M=$[1,\ldots,10]$	$T_t = 7$, $T = 5\ (s)$
$R_t = 100\ (Mbit/s)$	$C_{eff} = 1$
$k_e = 0.05\ (Joule/(MHz)^2)$	$w = 1$, $W_i = 1$
F=F2 (MHz)	$Q = 4$, $A = 1$
$P_{Idle} = 0.5\ (W)$	$\zeta_i = 0.5\ (mW)$

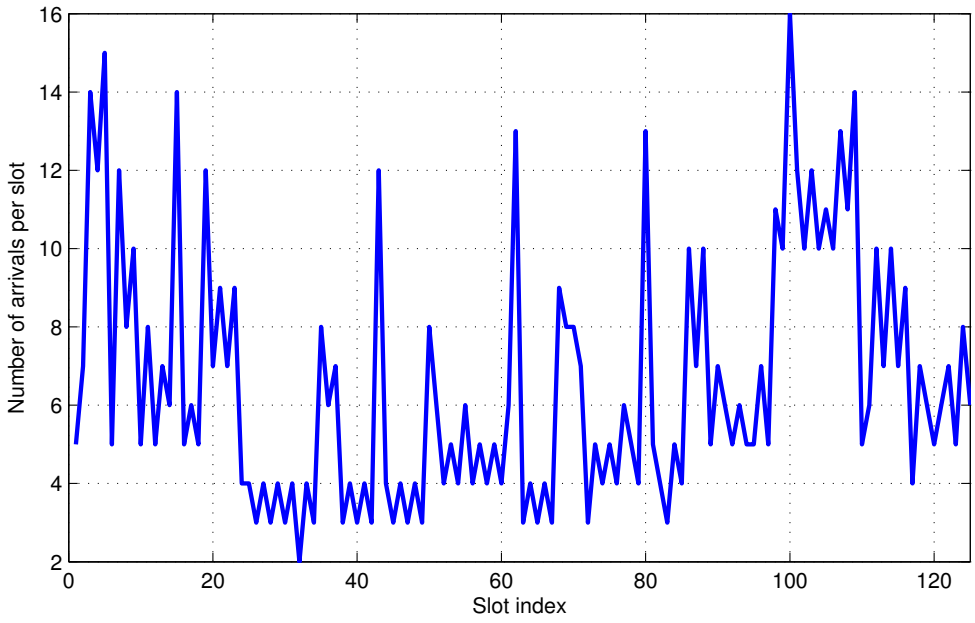
Table 3.6 Default values for the third test scenario.

Parameter	**Parameter**
$k_e = 0.005\ (Joule/(MHz)^2)$	$Q = 4$
F=F1$[MHz]$	$\overline{L}_{tot} = 70(Mbit)$
M=$\{20, 30, 40\}$	$\Omega_i = 0.5\ (mW)$

3.2.4.1.2 Test Workload In order to account for the effects of the reconfiguration costs and the time-fluctuations of the offered workload on the energy performance of the tested schedulers, as in [91], we model the offered workload as an independent identically distributed (i.i.d.) random sequence $\{L_{tot}(m), m = 0, 1, \ldots\}$, (where m is the index of input workload), whose samples are uniformly distributed over the interval $[\overline{L}_{tot} - a, \overline{L}_{tot} + a]$, with $\overline{L}_{tot} \equiv 8\ (Mbit)$. By setting the spread parameter a to 0 $(Mbit)$, 2 $(Mbit)$, 4 $(Mbit)$, 6 $(Mbit)$ and 8 $(Mbit)$, we obtain Peak-to-Mean Ratios (PMRs) of 1 (i.e., the offered workload is of constant size), 1.25, 1.5, 1.75 and 2.0, respectively. Each tested point has been evaluated



(a) Synthetic workload.



(b) An enterprise cluster in Microsoft [106].

Fig. 3.11 Sample traces of I/O workload.

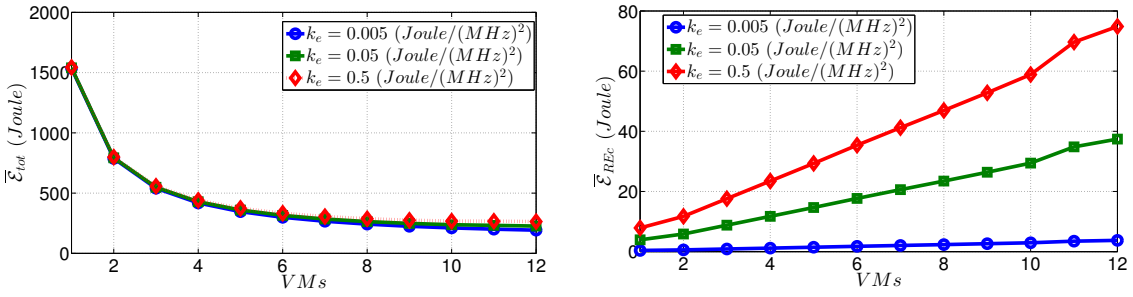
by averaging over 1000 independent runs. Moreover, in order to test the capability of the proposed scheduler when the arrival process exhibits time-correlation, we also considered the aforementioned synthetic traces of workload and the real-world arrival trace of Figs. 3.11a

and 3.11b, respectively. Fig.3.11a reports the synthetic workload of 100 slot-periods for the corresponding PMR and the $\overline{L}_{tot}$ are 2 and 8 $(Mbit)$, respectively. Fig.3.11b reports the real-world trace of I/O workload from an enterprise cluster in Microsoft [106] and refers to the I/O workload taken from four RAID volumes of an enterprise storage cluster in Microsoft (see Section IV.A of [106]). In Fig.3.11b, the corresponding PMR and time-correlation coefficient are 2.49 and 0.85, respectively. The measured arrival rate is in multiple of the slot period and the reported trace covers more than 120 slot-periods. The numerical tests carried out in this paper refer to the communication-plus-computing infrastructure of Table 3.4.

3.2.4.1.3 Setting of benchmark schedulers Testing the sensitivity of the performance of the proposed scheduler to the PMR of the offered workload is the goal of the numerical results of these simulations. Specifically, they aim at unveiling the impact of the PMR on the average energy consumption of the proposed scheduler and comparing it against the state-of-the-art scheduler, namely, the *STAS* [9]. Intuitively, we expect that the energy savings attained by dynamic scheduler increase when multi-rate reconfigurable VMs are used, especially at large PMR values. However, we also expect that non negligible reconfiguration costs may reduce the attained energy savings and that the experienced reductions tend to increase for large PMRs.

About the simulated *STAS*, we note that current virtualized data centers usually rely on static resource provisioning, where, by design, a *fixed* number of VMs constantly run at the maximum processing rate f_i^{max} [6]. The goal is to constantly provide the exact computing capacity needed for satisfying the peak workload $L_{tot}^{max} \triangleq (\overline{L}_{tot} + a)$ $(Mbit)$.

In order to put under the right perspective the following performance comparisons, two main explicative remarks are in order. First, although the *STAS* does not experience reconfiguration costs, it induces resource overbooking. Hence, the resulting per-job average communication-plus-computing energy consumption gives a benchmark for numerically evaluating the energy efficiency (i.e., the percent energy gaps) of dynamic schedulers. Second,



(a) Per-VM $\overline{\mathscr{E}}_{tot}$ (b) Per-VM $\overline{\mathscr{E}}_{Rec}$

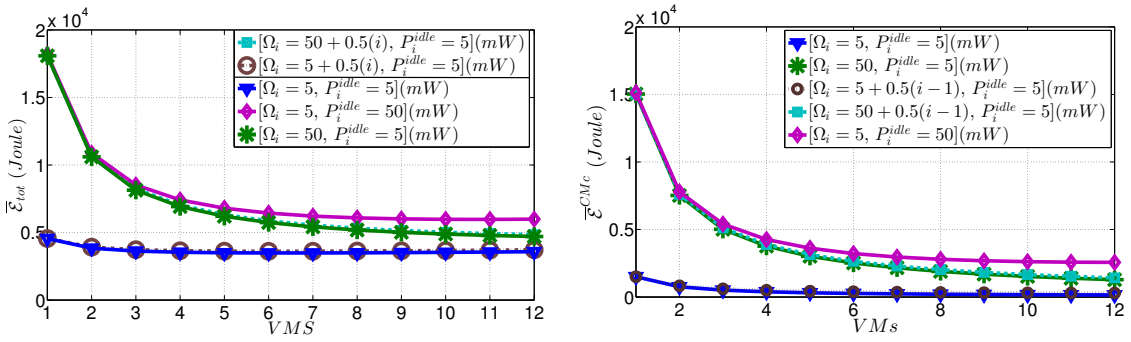
Fig. 3.12 3.12a: $\overline{\mathscr{E}}_{tot}$ and 3.12b:$\overline{\mathscr{E}}_{REc}$ for the application simulation Table 3.4 at $\Omega_i = 5(mW)$, $i = 1,\ldots,M$, $T = 0.1\ (s)$ and $F = F1$ in *GreenNetDC* and the I/O workload in [106] is considered.

the capacity planning studies in [6] refer to static schedulers and this provides an additional motivation for considering the energy performance of the *STAS* as a benchmark. Explanation of the modified *STAS* respected to the *GreenNetDC* model which is applied as a benchmark is presented in Appendix D.

3.2.4.2 Experimental Results

In order to measure the performance of *GreenNetDC*, we evaluated *GreenNetDC* under operating scenarios detailed in the following subsections.

3.2.4.2.1 Performance effects of VMs hibernation and dynamic reconfiguration The first experiment focuses on the real discrete value for the DVFS-based CPU which are taken from Table 3.4. Fig.3.12 presents the average total energy consumption of the *GreenNetDC* at various switching costs k_e for I/O workload of Fig.3.11b. Fig.3.12a shows the effects of the reconfiguration cost on the average total energy for various values of k_e. This plot indicates that the average total energy increases by an increases in k_e value, while it strongly decreases by increasing the number of VMs. Also, the response time for each workload is about 5-10 ms. Fig.3.12b shows that the average energy of switching (i.e., $\overline{\mathscr{E}}_{REc}$) smoothly increases for increasing the number of VMs, especially for higher k_e's. In the second scenario which uses Table 3.5 parameters, we evaluated the average (per-job) energy consumed by the system to



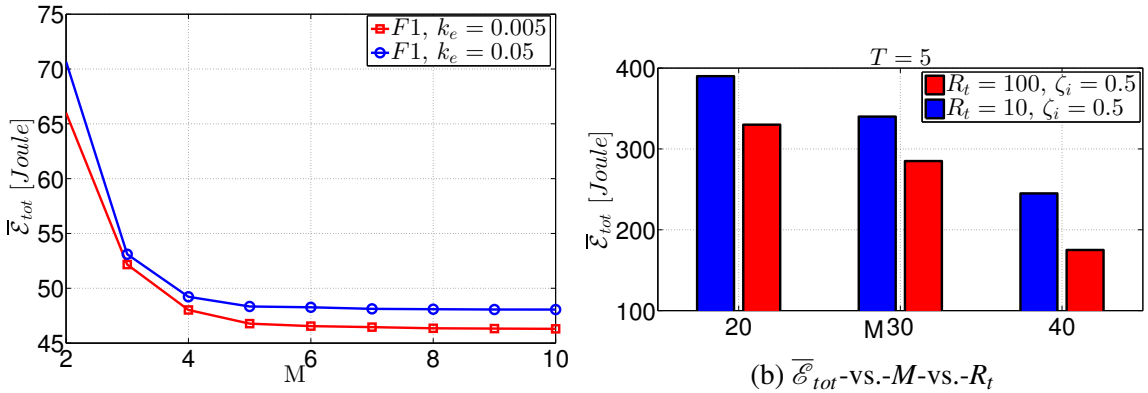
(a) Per-VM $\overline{\mathscr{E}}_{tot}$. (b) Per-VM $\overline{\mathscr{E}}^{CMc}$.

Fig. 3.13 Effects of the link quality on the energy consumptions of the *GreenNetDC* in the homogeneous cases (continue plots) and the heterogeneous cases (dashed plots); The application scenario of Table 3.4 at $k_e = 0.005\,(Joule/(Mbit/s)^2)$, and $F = F1$ with the workload in Fig.3.11a is considered.

vary the number of available VMs, and the variation of k_e with the frequency speeds $F1$ which is shown in Fig.3.14a. Based on the synthetic traces of workload in Fig.3.14a, comparisons of VMs with the CPU range $F2$ confirms that by increasing the VMs the energy reduces which ranges from 80% (case of $k_e = 0.005$ with the lower plot) to 85% (case of $k_e = 0.05$ with the upper plot). These results proceed the expectations [6] that noticeable energy savings may be attained by jointly changing the available computing-plus-communication resources.

In the third scenario, we evaluate the energy consumption for the huge amount of workloads and VMs. Figs. 3.14 presents the total average consumed energy among aforementioned techniques for 20, 30, and 40 VMs and high incoming workload. In Fig.3.14b, we use the third scenario with various SLA parameters T, R_t and the communication coefficient ζ in order to energy reduction rate of the proposed method while facing various SLA ranges. Fig.3.14b shows that, by fixing the T and ζ, while we smooth the R_t data center communication boundary 10 times higher, the proposed scheduler saves more energy approximately 15% in high VMs. It confirms that the scheduler could save energy depends on the various boundary assigned to it.

3.2.4.2.2 Performance effects of the communication costs The second experiment tests *GreenNetDC* under various Ω_i and evaluates $\overline{\mathscr{E}}^{*}_{tot}$ under the following settings: *i)* $\Omega_i = 5$



(a) Impact of $\overline{\mathscr{E}}_{tot}$ by reconfiguration parameter

(b) $\overline{\mathscr{E}}_{tot}$-vs.-$M$-vs.-$R_t$

Fig. 3.14 $\overline{\mathscr{E}}_{tot}$ for the second/third test scenarios for the proposed method in 3.14a for various k_e (second scenario) and in 3.14b for various R_t in fixed T and ζ_i (third scenario).

(mW); *ii)* $\Omega_i = 50\ (mW)$; *iii)* $\Omega_i = [5+0.5(i-1)]\ (mW)$; and, *iv)* $\Omega_i = [50+0.5(i-1)]$ (mW), $i = 1,\ldots,M$ with $P_i^{idle} = 5\ (mW)$ and; *v)* homogeneous $\Omega_i = 5\ (mW)$ with high idle end-to-end link power $P_i^{idle} = 50\ (mW)$. As expected, larger Ω_i's penalize the overall energy performance of the *GreenNetDC*. Interestingly, since $\overline{\mathscr{E}}_{tot}$ is the *minimum* energy when up to M VMs may be turned ON, at fixed P_i^{idle}'s, $\overline{\mathscr{E}}^*_{tot}$ decreases for increasing M and, then it approaches a minimum value that does not vary when M is further increased (see the flat segments of the first uppermost plots of Fig.3.13a and Fig.3.13b).

3.2.4.2.3 Performance effects of dynamic computation costs In Fig.3.15, we processed *GreenNetDC* with different C_{eff} as a input coefficient for each computing part and evaluated the total energy and its corresponding energy terms (the first term in (3.41.1)). In detail, while the capacitance load (C_{eff}) increases 10 times, the average total energy or $\overline{\mathscr{E}}_{tot}$ which is depicted in Fig.3.15 (the upper lined plot) decreases with increasing M, while the average computing energy ($\overline{\mathscr{E}}_{CPc}$) increases for the capacitance load 10 and 100 but the ratio is so low (see the dashed lines of Fig.3.15).

3.2.4.2.4 Performance effects of discrete computation rates In this subsection, we investigated *GreenNetDC*' t_{ij} with different T and the results are shown in Fig.3.16. The

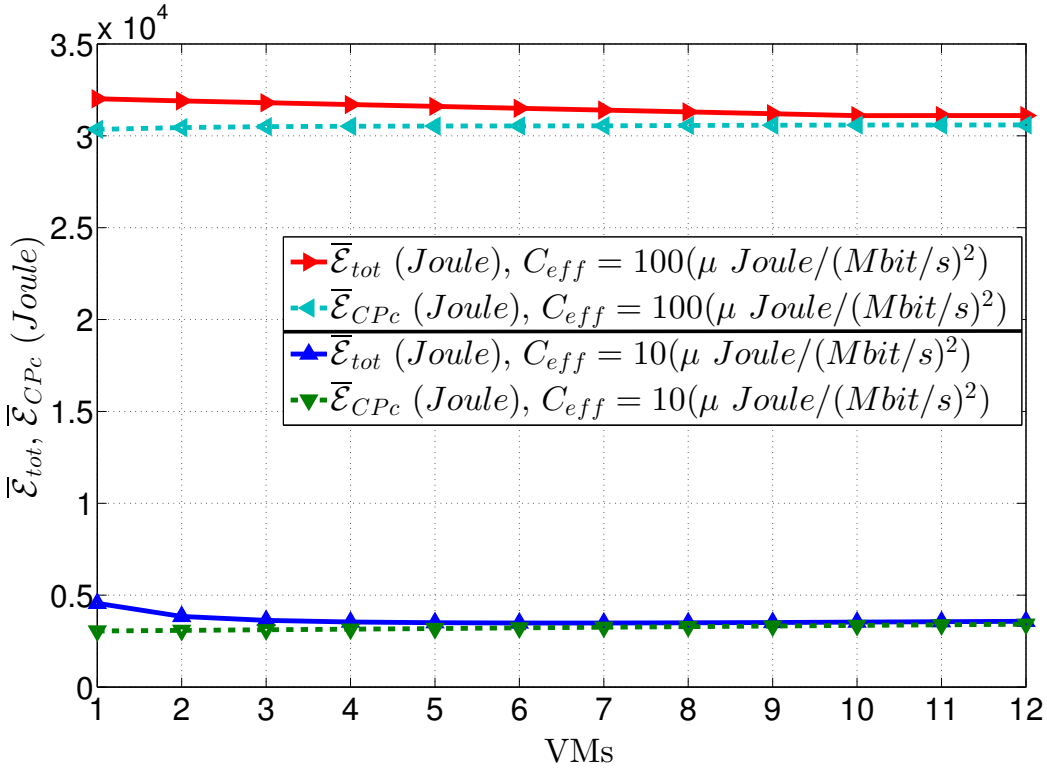
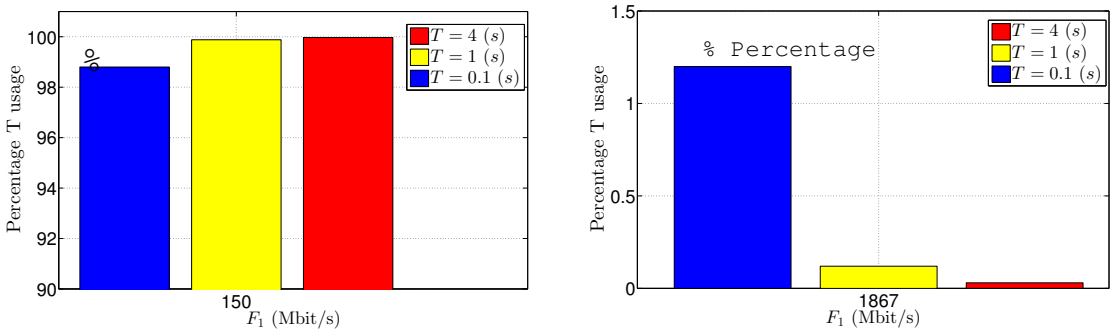


Fig. 3.15 $\overline{\mathscr{E}}_{tot}$ and $\overline{\mathscr{E}}_{CPc}$ for the application simulation Table 3.4 with $\Omega_i = 5(mW)$, $P_i^{idle} = 50$ (mW), $i = 1,\ldots,M$, $k_e = 0.005\ (Joule/(Mbit/s)^2)$, and $F = F1$ in *GreenNetDC*.

results are the normalized percentages of the optimum time assigned to available frequencies. Interestingly, Fig.3.16 shows that the mast part of the testes durations is assigned to the idle mode and less time is consumed for VM processing. Moreover, the summation of the durations that each VM can work with *three*-higher discrete ranges of the frequency (i.e., $t_{ij} \mid i = 1,\ldots,M,\ j = 3,4,5$) is approximately zero. It means, each VM is able to respond to its workload fraction within its first two ranges of active discrete frequencies (i.e., $t_{ij} \mid i = 1,\ldots,M,\ j = 1,2$) and no need to work with its high discrete ranges of frequencies for processing. In detail, Figs. 3.16a and 3.16b indicate that by increasing the processing time (i.e., less computing hard-deadline considered), VMs manage their processing time precisely and can be in idle mode for longer durations and are able to handle the requests even with the second active discrete frequency (e.g., $t_{i1} \mid i = 1,\ldots,M$) about 6 times lower than the first active discrete frequency (e.g., $t_{i0} \mid i = 1,\ldots,M$)(see the two lower bars of Fig.3.16b).

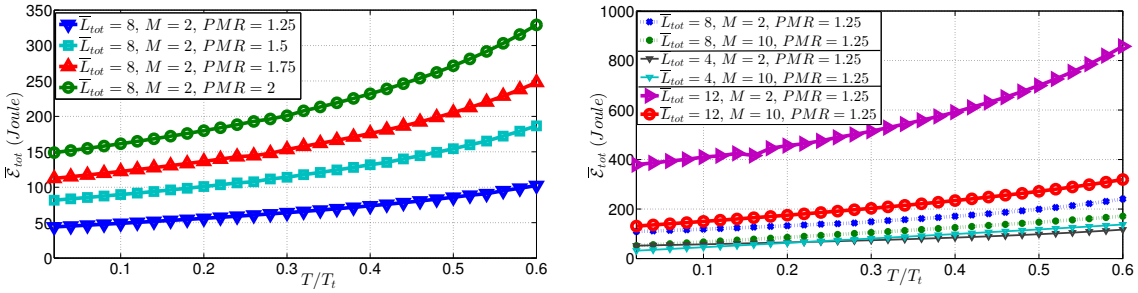


(a) Percentage of T usage in idle mode for each VM.

(b) Percentage of T usage in 2nd discrete range of frequency.

Fig. 3.16 3.16a: T's Percentage for VMs' idle mode frequency ($F_{i1} = 150$ $(Mbit/s)$, $i = 1,\ldots,M$); and, 3.16b: T's Percentage for VMs' 2nd discrete ranges for frequency ($F_{i2} = 1867$ $(Mbit/s)$, $i = 1,\ldots,M$) for the application simulation Table 3.4 with $\Omega_i = 5$ (mW), $k_e = 0.005(Joule/(Mbit/s)^2)$, $F = F1$ and various $T = \{0.1, 1, 4\}$ in *GreenNetDC*, (i.e., we omit 2 remaining largest ranges, because, there is no time assigned for these ranges in 1000 incoming workloads).

3.2.4.2.5 Computing-vs.-communication energy tradeoff We expect that small T's values give rise to a higher per-VM computing frequencies, while extremely large T values induce high end-to-end communication rates (see eq. (3.41.5)). However, we also expect that due to the adaptive power-rate control provided by the optimal scheduler, there exists a broad range of T values that attain an optimized tradeoff. Fig.3.17 confirms the aforementioned expectations. Specifically, the plots of Fig.3.17a confirm that by increasing the PMR, the energy is affected with higher slop (see the plots in Fig.3.17a). The plots of Fig.3.17b refer to the application simulation in Table 3.4 at $M = 2, 10$, $\overline{L}_{tot} = \{4, 8, 12\}$ $(Mbit)$, $C_{eff} = 1$, $P_i^{idle} = 50$, $\Omega_i = 5$, $i = 1,\ldots,M$, $T_t = 5$ and $F = F1$. An examination of these plots supports three main conclusions. First, the proposed scheduler attains the minimum energy consumption for values of the ratio (T/T_t) falling into the (quite broad) interval $[0.02, 0.6]$, especially for $M = 10$. Second, the effects of the ratio (T/T_t) on the energy performance of the scheduler are negligible when the *GreenNetDC* operates far from the boundary of the feasibility region dictated by (3.54.1)-(3.54.2) (see the two lowermost curves of Fig.3.17b with $\overline{L}_{tot} = 4$ $Mbit$). Interestingly, the increasing behavior of the curves of Fig.3.17b gives



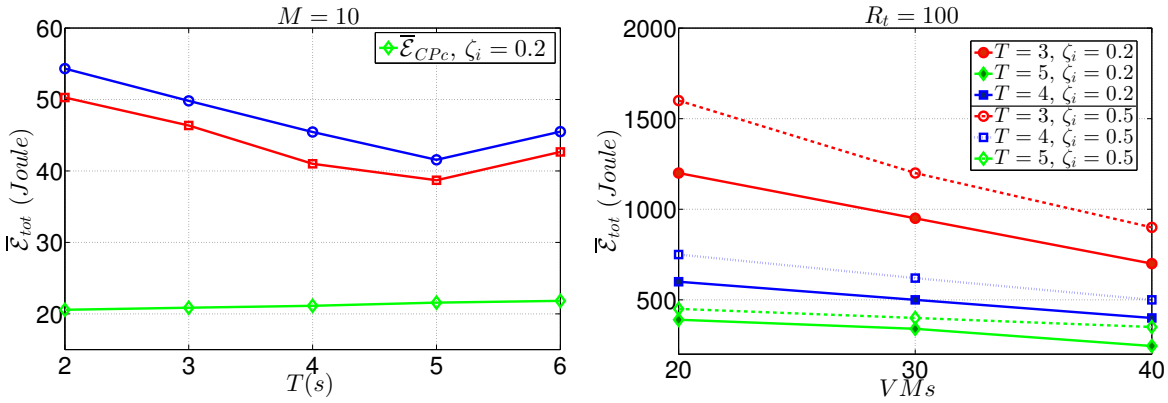
(a) $\overline{\mathscr{E}}_{tot}\ (Joule)$ for various PMR-vs.-T/T_t at *fixed* $\overline{L}_{tot}$.

(b) $\overline{\mathscr{E}}_{tot}\ (Joule)$ at fixed $PMR = 1.25$ vs. T/T_t at *various* $\overline{L}_{tot}$.

Fig. 3.17 3.17a: $\overline{\mathscr{E}}_{tot}\ (Joule)$ for various $PMR = \{1.25, 1.5, 1.75, 2\}$ vs. T/T_t with fixed $\overline{L}_{tot} = 8\ (Mbit)$ and 2 VMs; and, 3.17b: $\overline{\mathscr{E}}_{tot}\ (Joule)$ for fixed $PMR = 1.25$ vs. T/T_t with various $\overline{L}_{tot} = \{4, 8, 12\}\ (Mbit)$ and various VMs $M = \{2, 10\}$, for the application scenarios of Table 3.4 at $\Omega_i = 5(mW), k_e = 0.005(Joule/(Mbit/s)^2), F = F1$ and various T in *GreenNetDC*.

practical evidence that the computation energy dominates the overall energy consumption at decreasing (T/T_t), while the network energy becomes *substantial* for $(T/T_t) \to 1$, implying that the range of (T/T_t) is less than 1 (see eq. (3.41.5)). Third, if we keep $M = 10$ fixed and increase the $\overline{L}_{tot}$, we will understand the $\overline{\mathscr{E}}_{tot}$ increasing ratio is not dominant compared with the same workload increasing for low VMs. It means, while workload increases, CPUs should work with maximum speed to serve workload and this leads to increasing the energy consumption of the data center. Fig.3.18a tests the computing-vs-communication energy trade-off for the second scenario of Table 3.5 for different ranges of T. It confirms that small T's values give rise to a higher per-VM computing frequencies which leads to increasing the $\mathscr{E}_{tot}$. While T increases, the proposed scheduler uses processing time in order to decrease $\mathscr{E}_{tot}$ about 20% (see the two upper plots of Fig.3.18a). While extremely large T values induce high end-to-end communication rates which it leads to increasing the $\mathscr{E}_{tot}$ (see eq. (3.41.5)). Fig.3.18b shows that, by fixing the R_t and grouping the ζ, while we increase the T (computing time for the processing of each server) average total energy per-VM saves 200 to 500 *Joule* even with huge amount of T and ζ_i.

3.2.4.2.6 Performance comparisons under synthetic workload traces In this subsection, we compare the performance of *GreenNetDC* with the *NetDC* [23], *Lyapunov* method

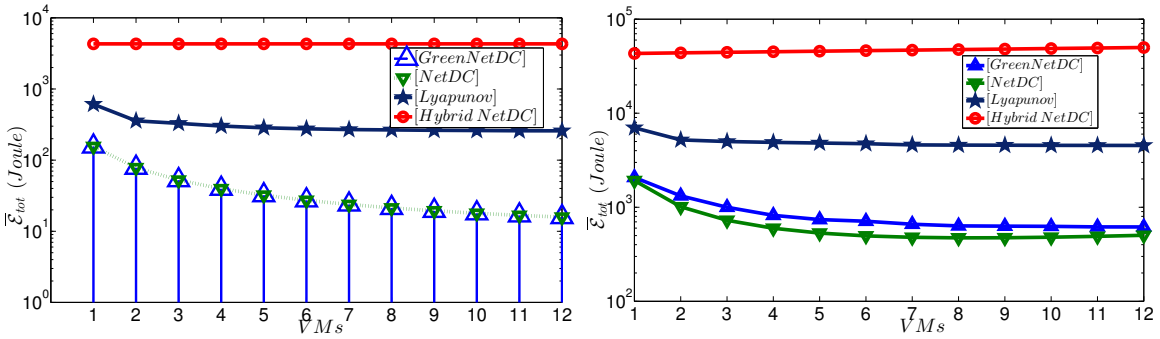


(a) Computing-vs.-Communication tradeoff. (b) $\overline{\mathscr{E}}_{tot}$-vs.-$M$-vs.-$T$-vs.-$\zeta$ (the 3rd scenario).

Fig. 3.18 $\overline{\mathscr{E}}_{tot}$ for the second/third test scenario for the proposed method in 3.18a and 3.18b for various T and ζ, respectively.

in [91] and the *HybridNetDC* [17] ones in terms of the synthetic workload traces, energy-savings, and SLA attainments.

3.2.4.2.6.1 Synthetic workload traces comparisons: From a formal point of view, *HybridNetDC* implements the solution of *OP*, e.g., the constrained minimization problem in (3.41.1)-(3.41.5) over the variable $\{t_{ij}\}$ at $R_i = R_t$, $i = 1,\ldots,M$. The plots of Fig.3.19a refer to the application scenario of Table 3.4 at $C_{eff} = 10$, $P_i^{idle} = 50, \Omega_i = 5$, $i = 1,\ldots,M$, $T_t = 5$, $T = 1$ (s), and $F = F3$. We observe that at low processing frequencies, our approach is approximately 1%-2% worse than *NetDC*, about 50% better than *Lyapunov* and approximately 90% better than *HybridNetDC* approach. In the low frequency range, the only difference among *GreenNetDC* and *NetDC* is the computing and reconfiguration parts in (3.41.1): *GreenNetDC*($\overline{\mathscr{E}}_{CPc}$)$\simeq$ *NetDC*($\overline{\mathscr{E}}_{CPc}$) but *GreenNetDC*($\overline{\mathscr{E}}_{REc}$)$>$ *NetDC*($\overline{\mathscr{E}}_{REc}$), and the communication energy cost for both of them are the same. Fig.3.19b presents the $\overline{\mathscr{E}}_{tot}$ $(Joule)$ for aforementioned approaches at high frequencies and fixed $Q = 4$. Our approach is still better than *Lyapunov*[91], *HybridNetDC*[17] and the difference between *NetDC*[23] and *GreenNetDC* decreases as M increases and/or increasing Ω's (i.e., heterogeneous Ω). The difference is approximately 6-8%, because, *NetDC*[23] exploits continuous range of frequencies that allows higher degree of freedom. As a result, the gap between the proposed



(a) $\overline{\mathscr{E}}_{tot}$ for $F = F3$ and $\Omega_i = 5\,(mW)$. (b) $\overline{\mathscr{E}}_{tot}$ for $F = F2$, $\Omega_i = 50 + 0.5(i-1)\,(mW)$.

Fig. 3.19 $\overline{\mathscr{E}}_{tot}\,(Joule)$ for *GreenNetDC*, *NetDC*[23], *Lyapunov*[91] and *HybridNetDC*[17] for the application scenario of Table 3.4 at $k_e = 0.005(Joule/(Mbit/s)^2)$, $P_i^{idle} = 50\,(mW)$.

scheduler and *NetDC* increases with an increase in frequency and end-to-end link coefficient. Therefore, *GreenNetDC* is better performing under homogeneous end-to-end link and at high processing frequencies. We should emphasize that *NetDC* is presented in order to work with the continues ranges of frequencies which is not feasible, unrealistic and impossible in reality while *GreenNetDC* could be one of the best practical solutions in networked data centers. Also, in the second simulation comparisons, in order to evaluate the energy reduction due to scaling up/down of the computing, reconfiguration and communication rates by increasing the VMs (i.e., we process the results for the one time implementation over 10 VMs). In detail, Fig.3.20 presents the average of 1000 offered workloads (i.e., $m = 1000$) for the total energy $\mathscr{E}_{tot}$, Computation energy $\mathscr{E}_{CPc}$, reconfiguration energy $\mathscr{E}_{REc}$, and communication energy $\mathscr{E}^{CMc}$ for our approach, IDEAL, Standard, Lyapunov-based method in [91] and recent work done in this area in [23], in sub-figures 3.20a, 3.20b, 3.20c, 3.20d, respectively. Specifically, Fig.3.20a points out that, by increasing the VM number, the average total cost for all approaches decreases because while the number of VMs is increased, the abilities to respond to the offered workloads will be higher, so, less quota of L_{tot} will be assigned to each VM (i.e., $F_{ij}t_{ij}$), and the needed frequency and time for computation decreases enormously and also their correlated cost declines. In Fig.3.20a, the average energy-saving of the proposed method is approximately 50%, 60% compared to Lyapunov-based and Standard schedulers, respectively. Furthermore, in Fig.3.20b, while

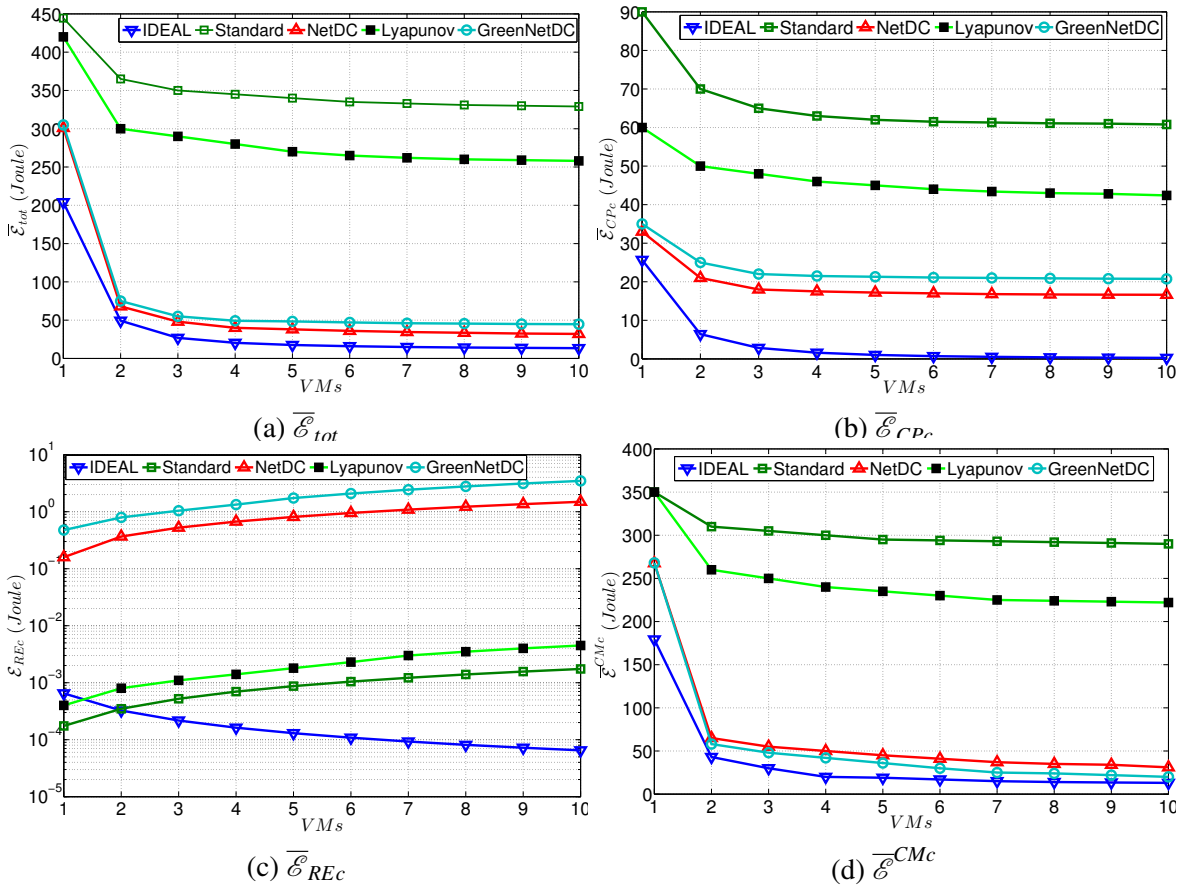


Fig. 3.20 Average energy terms of eq. (3.41.1) with PMR=1.25 for the second scenario for the proposed method-vs.- IDEAL in [68]-vs.- Standard [54]-vs.- NetDC in [23]-vs.- Lyapunov in [91].

$M > 2$ increases, it does effect the cost of the computing too-much, because the system is able to manage the running time for each active discrete frequency even while M is low ($M < 4$) or (high, $M > 20$), and most of the time system for each offered workload goes to the Idle mode or F_0 and most times are in that state and less time will be assigned for the Q of remains discrete time, system can easily response its quota $F_{ij}t_{ij}$ with less effects of M. In Fig.3.20c, it is obvious that our approach considers two costs (internal-switching and external-switching) for each $VM(i)$ in each incoming workload. So it results in increased reconfiguration cost compared to *NetDC* [23] and Lyapunov-based scheduler in [91], which consider just probabilities of previous and next active discrete frequencies for each $VM(i)$ (i.e., external-cost of our approach). Lastly, Fig.3.20d points out that the communication cost of the proposed techniques is less than others and near to IDEAL, because according to

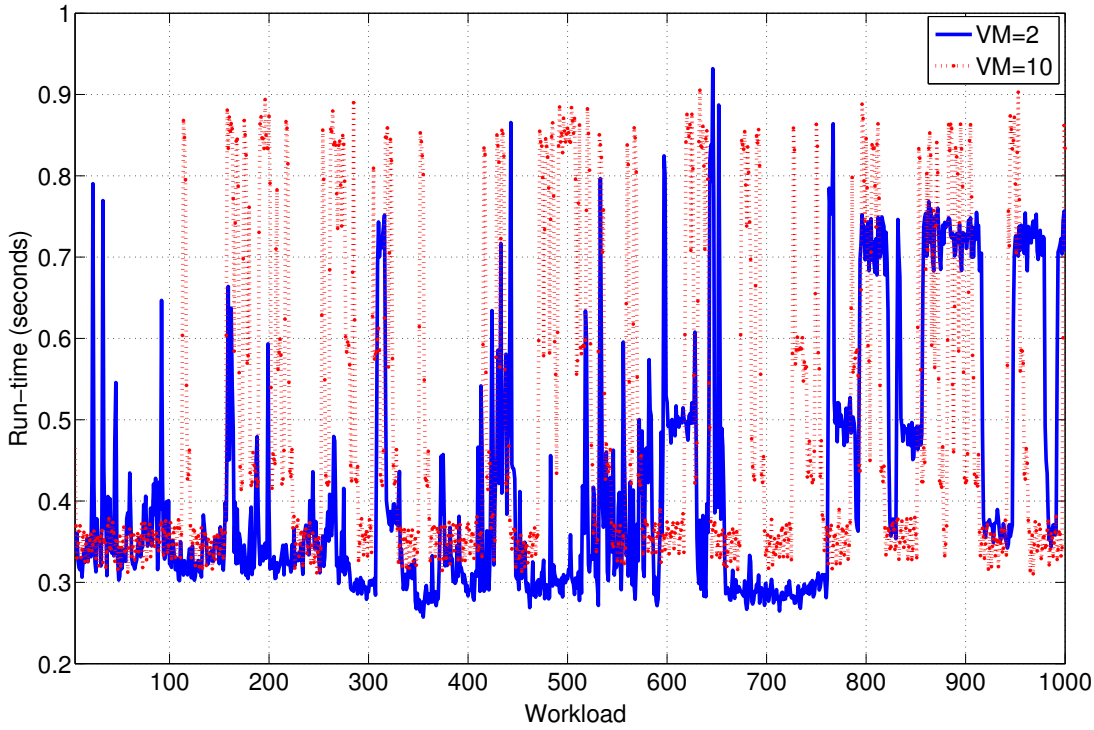


Fig. 3.21 System time for 1000 workload in the second scenario

the third term of the (3.41.1), the optimization problem tries to find the optimum objective variables while more resources are available. Fig.3.20d shows that the proposed scheduler is about 10%, 50%, 65% better than NetDC[23], Lyapunov[91], and Standard [54] schedulers, respectively. Indeed, the proposed scheduler is able to find proper times of the active discrete frequencies for each offered workload (it means that, the summation $\sum F_{ij}t_{ij}$ are the same for all approaches and equal to L_{tot}). Figure 3.21 reports the execution-time for each of 1000 offered workload (i.e., L_{tot}) for $M = 2$ and $M = 10$ in the second scenario. Specifically, while M increases the matrices capacities in programming increased and the time will be risen. And, in Fig.3.22, it is obvious that our approach considers two costs (internal-switching and external-switching) for each $VM(i)$ in each incoming workload. So it results in increased reconfiguration cost compared to *NetDC* [23] and Lyapunov-based scheduler in [91], which consider just probabilities of previous and next active discrete frequencies for each $VM(i)$ (i.e., external-cost of our approach).

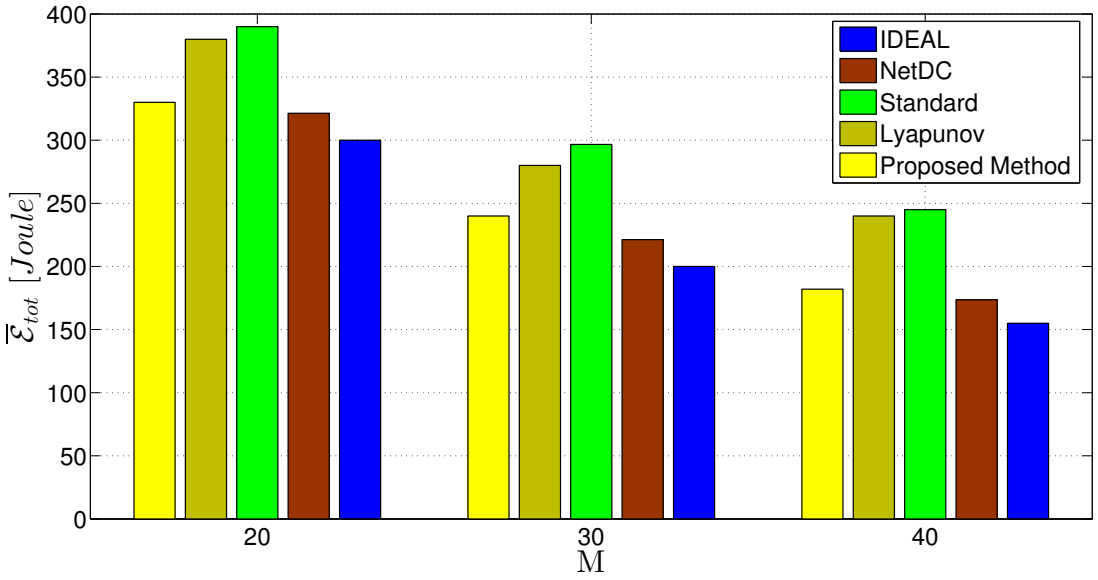


Fig. 3.22 Average energy terms of eq. (3.41.1) with PMR=1.25 for the third scenario for the proposed method-vs.- IDEAL in [68]-vs.- Standard [54]-vs.- *NetDC* in [23]-vs.- Lyapunov in [91].

3.2.4.2.6.2 Energy-saving at various PMRs: We tested our proposed method with a related state-of-the-art scheduler: *STAS* [9]. This is motivated by the fact that current data centers usually rely on *static* resource provisioning, where, by design, a *fixed* number of VMs constantly run at the *maximum* processing rate f_i^{max}, in order to constantly provide the computing capacity needed to satisfy the peak input workload. Table 3.7 reports the average energy savings (in percent) provided by the the *GreenNetDC*, *Lyapunov*[91]and *HybridNetDC*[17] schedulers over the static one (*STAS*)[9]. In *STAS*, it is enough to consider f_i^{max} instead of each discrete frequency F_{ij}, T instead of t_{ij} for i-th VM and switch cost is calculated just for each incoming workload. So, just changing from the idle mode to the f_i^{max} for each VM at first stage is considered. These results refer to $\Omega_i = 0.5$, $k_e = 0.05$, $T = 1$, $T_t = 5$, $f_i^{max} = 933$ $(Mbit/s)$, $\overline{L}_{tot} = 8$, $M = 12$, $Q = 4$, and $F = F2$. In order to guarantee that the *STAS* [9], *HybridNetDC*[17], *Lyapunov*[91] and *GreenNetDC* schedulers retain the *same* energy performance under *constant* offered 1000 incoming workloads, the numerical results of Table 3.7 have been evaluated by forcing the aforementioned schedulers to utilize the *same* number of VMs which are activated by the *STAS*. An examination of

Table 3.7 Average energy reductions attained by *GreenNetDC*, *Lyapunov* and *HybridNetDC* schedulers over the *STAS* [9].

PMR	**GreenNetDC**	**Lyapunov**[91]	**HybridNetDC**[17]
1.25	73%	62%	54%
1.5	72%	58%	52%
1.75	70%	56%	49%
2	66%	53%	45%

the numerical results reported in Table 3.7 leads to three conclusions. First, the average energy saving of the proposed scheduler over the *STAS* approaches 73%, even when the VMs are equipped with a limited number $Q = 4$ of discrete processing frequencies and the reconfiguration energy overhead is accounted for. This result confirms that *GreenNetDC* produces an effective means for leveraging the sudden time-variations exhibited by the workload. Second, the performance loss suffered by the *GreenNetDC* scheduler with respect to the *Lyapunov*, and *HybridNetDC* tends to increase about 6% for growing PMRs due to rising workload fluctuations and this increment is even lower than other schedulers. Third, the average energy reductions attained by *GreenNetDC* for various PMRs compared to *Lyapunov* and *HybridNetDC* are limited up to $11\% - 14\%$ and $19\% - 21\%$, respectively.

3.2.4.2.6.3 Keep SLA comparisons: Lastly, we consider our approach in the case where the SLA is defined as a fraction of difference between the requested frequencies for all VMs $f_i^{max}(m)$ and the actually alloted frequency $f_i(m)$ for each VMs relative to the total requested frequency over the life-time of the VMs (e.g., see [9, 74])

$$SLA(m) = \frac{\sum_{i=1}^{M} f_i^{max}(m) - f_i(m)}{\sum_{i=1}^{M} f_i^{max}(m)}, \tag{3.57}$$

where m is the slot index. Table 3.8 shows the SLA violation rates for the *GreenNetDC*, *NetDC*, and *Hybrid NetDC* under the workload of Fig.3.12b. *GreenNetDC* is able to decrease the percentage of SLA violation rate substantially than other techniques. The

Table 3.8 Average SLA violation percentage in the *GreenNetDC*, *NetDC*, and *Hybrid NetDC*.

GreenNetDC	**NetDC**[23]	**Lyapunov**[91]	**HybridNetDC**[17]
15%	20%	25%	28%

GreenNetDC minimizes the amount of SLA violations by using discrete ranges of VM frequencies and it is able to adapt by tracking the previous optimum frequency achieved in each VM. In a nutshell, a VM learns to decide when it becomes overloaded according to the dynamic workload.

Chapter 4

TCP/IP-based QoS-aware Energy-efficient Scheduler to Support Vehicular Cloud Services

In this chapter, we propose and test an efficient dynamic resource provisioning scheduler which applied in Networked Data Centers (NetDCs) which are connected to (possibly, mobile) clients through TCP/IP-based vehicular backbones. The goal is to maximize the energy-efficiency, while meeting hard QoS requirements on the delivered transmission rate and processing delay. The resulting optimal cross-layer resource scheduler is adaptive, and jointly performs: *i)* admission control of the offered input traffic; *ii)* balanced control and dispatching of the admitted workload; *iii)* dynamic reconfiguration and consolidation of the Dynamic Voltage and Frequency Scaling (DVFS)-enabled Virtual Machines (VMs) instantiated onto the parallel computing platform; and, *iv)* rate control of the traffic injected into the vehicular backbone. Necessary and sufficient conditions for the feasibility and optimality of the proposed scheduler are also provided in closed-form. The salient features of the proposed scheduler are that: *i)* it is adaptive and admits distributed scalable implementation; *ii)* it is capable to provide hard QoS guarantees, in terms of minimum/maximum *instantaneous* rate of the traffic delivered to the client, *instantaneous* rate-jitter and total processing delay; and, *iii)* it explicitly accounts for the dynamic interaction between computing and networking resources, in order to maximize the resulting energy efficiency and the TCP/IP mobile

connection throughput. Actual performance of the proposed scheduler in the presence of :*i)* client mobility; *ii)* wireless fading; *iii)*reconfiguration and consolidation costs of the underlying networked computing platform; and, *iv)*abrupt changes of the transport quality of the available TCP/IP mobile connection, are numerically tested and compared against the corresponding ones of some state-of-the-art schedulers, under both synthetically generated and measured real-world workload traces.

Motivated by these considerations, in this section, we develop and test a new scheduler for minimizing the energy consumption induced by computing, communication and reconfiguration costs in Internet-based virtualized DCs which utilize end-to-end TCP/IP mobile energy-constrained connections under hard limits on the per-job total processing time. Our scheduler performs dynamic load balancing and uses online job decomposition for adaptive resource management. It leads to the optimum processing speeds and bandwidth rates on a per-VM basis, as well as the proper workload quota for each VM on a per-job basis. Furthermore, it also performs admission control and adaptive management of the transmission rate of the Cloud-to-Vehicular TCP/IP mobile connections of Fig.4.1, in order to meet QoS constraints at the minimum energy wasting. Consider that our energy model is non-convex, hence, we develop a mathematical approach to turn nonconvexity into convexity. A remarkable feature of the resulting scheduler is its adaptive nature and scalability. The model in 4.1 operates at the Middleware layer and Software as a service (SaaS) is the provided service mode. The data center includes M servers. Each physical server is equipped with a Virtual Machine Manager (i.e., a hypervisor) and hosts multiple VMs. Input/output limited-capacity buffers control the load dispatcher before/after Cloud processing.

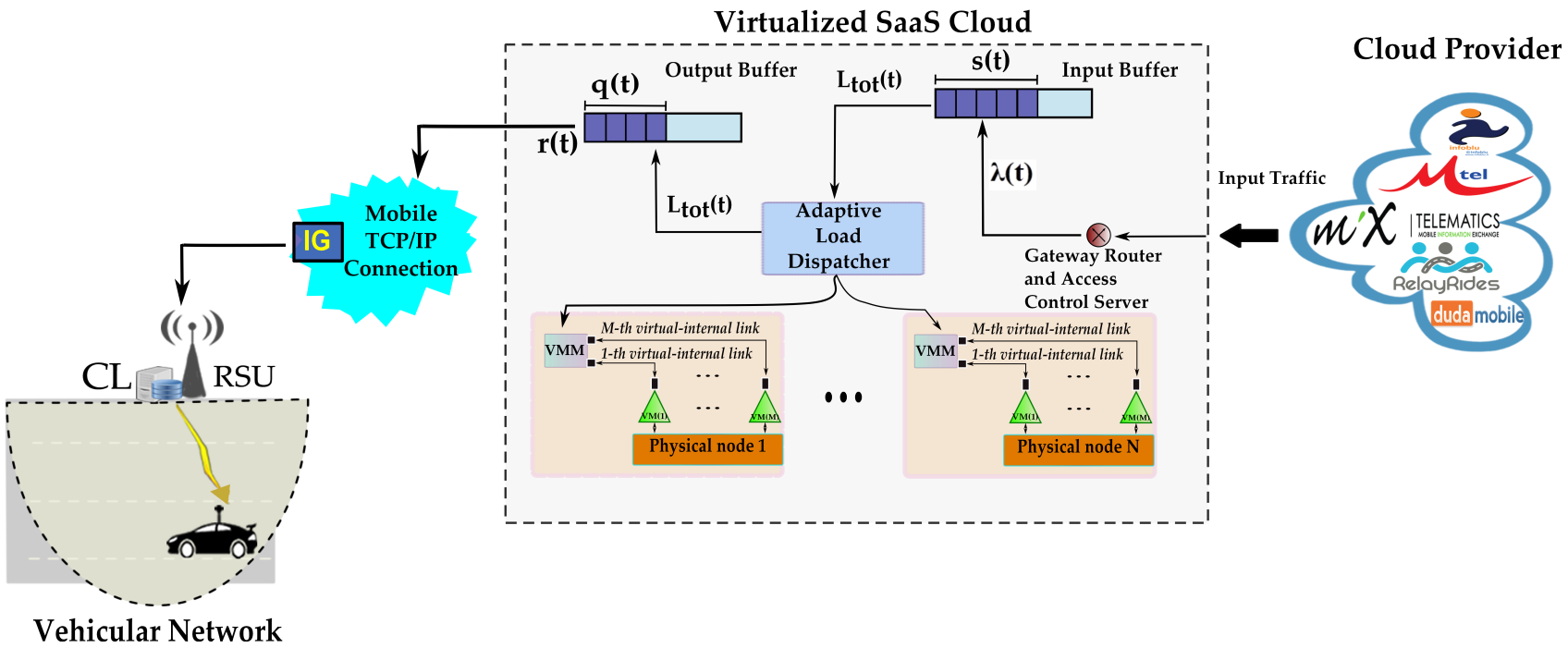


Fig. 4.1 The considered TCP/IP connection vehicular Cloud architecture. RSU:=Road Side Unit; CL:=CloudLet; IG:=Internet Gateway; SaaS:=Software as a Service.

4.1 The Considered VCC Infrastructure

Several recent research efforts target V2I networked infrastructures. Among these, the *FleetNet* and *Network on Whells (NoW)* European projects investigated the overall communication architecture for the integration of Cloud infrastructures and Internet-assisted vehicular networks [31]. Specifically, about the main building blocks of this architecture, a number of static RSUs are deployed along the road and are evenly spaced apart of D (m). Each RSU serves a spatial cluster of radius R (m), with $D \gtreqless 2R$, and *i)* acts as AP for the VCs currently traveling on the served cluster; *ii)* relay node of the wireless backbone to the Internet and the remote Cloud; and *iii)* it may be equipped with computing capability, so as to (occasionally) play the role of a CloudLet [99]. As pointed out in [31], this basic architecture is capable to support the Internet connection of resource-limited vehicular devices to the remote cloud. The *i-th* RSU (i.e., $RSU(i)$), is equipped with two bidirectional communication ports (namely, port-1 and port-2) and each port operates in a half-duplex way. Moreover, depending on the deployed networked infrastructure, each port may be equipped with a single or even multiple Network Interface Cards (NICs), which work over a same frequency band.

According to the emerging *Communication Access for Land Mobiles (CALM)* standard, the set of the available carrier frequencies falls into the 5.9 GHz band. Moreover, the overall current allotted spectrum is of 25 (MHz) and it is forecast that, in the next years, will be limited up to 75 (MHz) [49]. Hence, vehicular networking will continue to represent, indeed, a spectrally crowded scenario for the support of infotainment applications [99].

In the vehicular framework of Fig.4.1, time is slotted, T_s (s) is the slot duration, t is the discrete-time slot index and the *t-th* slot spans the semi-open time interval $[tT_s,(t+1)T_s), t \geq 0$. According to this assumption, the VCs of Fig.4.1 may change their spatial positions and the network nodes (that is, RSUs and VCs) may start to transmit *only* at the beginning of each slot t. In order to cope with the (aforementioned) bandwidth limitation, the communication traffic over the vehicular backbone of Fig.4.1 is Time Division Duplexed (TDD) and it is

organized into super-frames of duration T_{SF} (s). Each super-frame comprises an uplink phase and a downlink phase, whose (possibly, unequal) relative time-durations are tuned at the network setup.

At this regard, we note that, in order to avoid (or, at least) mitigate inter-flow co-channel interference, during the uplink phase of each super-frame, $RSU(i)$ may perform adaptive spatial transmit beam-forming or use multiple highly directive transmit antennas for implementing suitable antenna steering (see [2] and references therein for some recent results on this topic). This is no longer feasible during the downlink phase, in which $RSU(i)$ plays a "passive" role and limits up to detect the already occurred packets collisions. Hence, since the broadcast bandwidth from the serving RSU to the served VCs is typically larger than the access bandwidth from the served VCs to the serving RSU [67],

In the following, we focus on the Cloud-to-Vehicular (e.g., downlink phase), most deeply involved into the integrated computing-plus-communication problem, in order to *jointly* perform on a per-slot basis: *i)* the admission control of the input (e.g., exogenous) workload at the Cloud input; *ii)* the balanced dispatching of the admitted workload over the available Cloud computing nodes; *iii)* the consolidation and resource configuration of the VMs hosted by the Cloud; *iv)* the control of the transmission rates of the LAN hosted by the Cloud; and, finally, *v)* the control of the rate of the traffic injected into the TCP/IP mobile connection of Fig.4.1.

4.1.1 Input traffic and queue model in virtualized Clouds

In Internet virtualized DCs, each processing unit executes the currently assigned task by self-managing own local virtualized storage/computing resources. When a request for a new job is submitted from the Cloud providers to the virtualized cloud which contains virtualized networked data center (VNetDC), the resource controller dynamically performs both admission control and allocation of the available virtual resources [66]. Hence, according to the typical architecture recently presented in [83], Fig.4.1 reports the main blocks which

compose the Middle-ware layer of the considered VNetDC. Roughly speaking, it is composed by: *i)* an Admission Control Server (ACS or Adaptive load dispatcher); *ii)* an input buffer of size N_I; *iii)* a reconfigurable computing Cloud managed by the Virtual Machine Manager (VMM) and the related switched Virtual LAN; *iv)* an output queue of size N_O; *v)* the mobile end-to-end TCP/IP connection; *vi)* an adaptive controller that dynamically manages all the available computing-communication resources and also performs the admission control of the input/output traffic flows.

Specifically, at the end of slot t, new input requests arrive at the input of the ACS of Fig.4.1. This happens according to a random real-valued arrival process $\{Job(t) \in R_0^+, t \geq 0\}$, that is limited up to $Job_{max} \in R_0^+$ Information Units (IUs) per slot (e.g., $Job(t) \leq Job_{max}$, $t \geq 0$)[1]. The arrival process is assumed to be independent from the current backlogs of the input/output queues of Fig.4.1. However, we *do not* assume any a priori knowledge about the statistics of arrival request $\{Job(t) \in R_o^+, t \geq 0\}$. For example, $\{Job(t)\}$ could be a Constant Bit Rate (CBR) input traffic, or it could be a Markov-modulated process with time-varying instantaneous rate. This models a general scenario with unpredictable and (possibly) time-varying input workloads. Let $\lambda(t) \in R_o^+(IU/slot)$ be the number of IUs out of $Job(t)$ that are admitted into the input queue of Fig.4.1 at the end of slot t. We assume that any new request that is not admitted by the ACS of Fig.4.1 is declined. Thus, we have: $0 \leq \lambda(t) \leq Job(t)$, so that: $1 - (\lambda(t)/Job(t))$ is the fraction of the input workload that is rejected at slot t. Furthermore, we consider a (time-slotted) $G/G/1/N_I$ fluid system and a $G/G/1/N_O$ fluid one for modeling the input and output queues of Fig.4.1, respectively. Due to the admission control, both queues are loss-free and they implement the FIFO service discipline. Let $s(t) \in R_0^+$ and $q(t) \in R_0^+$ be the numbers of workloads stored by the input and output queues of Fig.4.1 *at the beginning* of slot t. Furthermore, let L_{tot} $(bit/slot)$ be the size of the incoming job that: *i*) is drained from the input queue *at the beginning* of slot t; and, *ii*)

[1]The meaning of an IU is application dependent. It may represent a bit, byte, segment or even an overall large-size application task (for example, a large image). We anticipate that, in the carried out tests of Section 4.5, IUs are understood as Mbytes.

is injected into the output queue at *the end* of slot t. Finally, let $r(t)(bit/slot)$ be the workload that is drained from the output queue and transmitted over the mobile TCP/IP connection of Fig.4.1 *during* slot t. Hence, the time evolutions of the backlogs $\{s(t) \in (R)_0^+, t \geq 0\}$, $\{q(t) \in (R)_0^+, t \geq 0\}$ of the input and output queues are dictated by the following Lindley's equations [70]:

$$s(t+1) = [s(t) - L_{tot}(t)]^+ + \lambda(t), t \geq 0, \tag{4.1}$$

$$q(t+1) = [q(t) - r(t)]^+ + L_{tot}(t), t \geq 0. \tag{4.2}$$

Since data arrive at the input and output queues at the end of slot t, we must have: $L_{tot}(t) \leq s(t)$, and: $r(t) \leq q(t)$. Goal of the output queue is to effectively cope with the congestion and mobility-induced fluctuations of the bandwidth offered by the mobile TCP/IP connection (e.g., end-to-end vehicular backbone of Fig.4.1), in order to save transmit energy.

Due to the real-time nature of the considered application scenario, full processing of the input file must be carried out within an assigned deterministic working time T_t (s) (s means second). Hence, in our framework, a real-time job is characterized by: *i)* the size L_{tot} of the file to be processed; *ii)* the maximum tolerated processing delay T_t; and, *iii)* the job granularity, that is, the (integer-valued) maximum number $M_T \geq 1$ of independent parallel tasks embedded into the submitted job.

4.1.2 The TCP/IP vehicular cloud architecture

Let $M \geq 1$ be the maximum number of VMs that are available at the Middleware layer of Fig.4.1. In principle, each VM may be modeled as a virtual server, that is capable to process f_i bits per second (i is the VM identifier and its maximum value is M). Depending on the size $L(i)$ (bit) of the task to be currently processed by $VM(i)$, the corresponding processing rate f_i may be adaptively scaled at run-time through DVFS. It may assume values over the interval $[0, f_i^{\max}]$, where $f_i^{\max}$ (bit/s) is the maximum allowed processing rate of $VM(i)$.

The TCP-based architecture partitions the overall scheduling problem into two smaller sub-problems. These sub-problems are addressed by two controllers, e.g., the *global controller* and the *local controller*. A local controller manages each PM or physical server, in order to optimize its working state by observing its current utilization. Moreover, the architecture has a global controller (e.g., a supervisor), in order to optimize the VM placement sub-problem. The local controller resides on a PM. It monitors the CPU and VM utilization and classifies the PM into one of the $Q+1$ sets of discrete frequency ranges: $\{F_0,\ldots,F_Q = f^{\max}\}$ which is elicited from the DVFS technology. Q is the number of allowed processing frequencies between the minimum and maximum for each VM. The global controller collects the states of the PMs from the local controllers and builds a global best-plan by using the TCP-based scheduling algorithm, which is described in the next section. The global controller sends commands to the VMMs for the optimization of the VM placement. The commands fix which VMs on a source PM should be activated and send back the processed job to the global controller (e.g., the load balancer). The VMM performs actual dispatching of VMs tasks, on the basis of the commands from the local controllers. Due to the real-time nature of the considered application scenario, the time allowed each VM to fully process each submitted task is fixed in advance at $T\ (s)$, *regardless* of the actual size $L(i)$ of the task currently assigned to the $VM(i)$.

4.1.2.1 Offered workload and VMM

Being L_{tot} the overall size of the current input job, let $L(i) \geq 0$, $i = 1,\ldots,M$, be the size of the task that Load Dispatcher of Fig.4.1 assigns to the $VM(i)$. Hence, the following constraint: $\sum_{i=1}^{M} L(i) = L_{tot}(t)$, guarantees that the overall job is *partitioned* into (at the most) M parallel tasks at time slot t. The set of the attributes which characterize each VM is:

$$\{\Phi(\eta_i), \Delta, E_c^{max}(i), E_c^{idle}(i), f_i^{max}\},\ i = 1,2,\ldots,M, \tag{4.3}$$

where Δ is the maximum allowed execution time (in seconds); $E_c^{idle}(i)$ is the static energy consumed by $VM(i)$ in the idle state ($Joule$); $E_c^{max}(i)$ is the maximum energy consumed by $VM(i)$; f_i^{max} is the maximum processing speed of $VM(i)$ (bit/s). $\Phi(\eta_i)$ is the utilization function with utilization factor η_i of $VM(i)$, which is typically defined as in [5]: $\Phi(\eta_i) \triangleq (f_i/f_i^{max})^2$.

4.1.3 Computing energy and reconfiguration cost in virtualized Clouds

In the following sub-section, we detail the adopted energy model, which accounts for the computational and reconfiguration costs.

4.1.3.1 Computational cost

The adopted computing energy model is based on the VMs' states. In [77], it is shown that there is a linear relationship between the *CPU utilization* of $VM(i)$ and the corresponding computing energy consumption, so that we can write:

$$E_c(i,t) = E_c^{idle}(i) + (f_i/f_i^{max})^2 (E_c^{max}(i) - E_c^{idle}(i)). \tag{4.4}$$

Interestingly, when $f_i = 0$, $VM(i)$ wastes E_c^{idle} energy.

4.1.3.2 Reconfiguration cost

We note that switching from the processing frequency $f_i(t-1)$ (the working processing speed of $VM(i)$ at $(t-1)$-th slot) to $f_i(t)$ (the working processing speed of $VM(i)$ in t-th slot) entails an energy overhead of $E_{dyn}(i,t)$ [5]. Although, the actual behavior of the function $E_{dyn}(i,t)$ depends on the adopted DVFS technique [91, 83], a quite common practical model is the following one [91, 83]:

$$E_{dyn}(i,t) = k_e \left(f_i(t) - f_i(t-1)\right)^2, \ (Joule) \tag{4.5}$$

where k_e $(Joule/(Hz)^2)$ denote the reconfiguration cost induced by an unit-size frequency switching, which is typically limited up to few hundreds of $\mu J's$ per $[MHz]^2$ [83].

4.1.4 Intra-data center communication

We assume that the i-th virtual end-to-end connection (e.g., the i-th virtual link) in Fig.4.1 is bidirectional, symmetric and operates in a *half-duplex* way [6]. A joint analysis of the computing-plus-networking energy consumption in *delay-tolerant* data centers is performed in [77], where the effects of inter and intra-cloud networking infrastructures are evaluated. Two main conclusions arise from [77]. First, the energy consumption due to data transport may represent a large part of the total energy consumption, especially at medium/high usage rates. Second, the energy consumption of cloud systems needs to be analyzed by *simultaneously* accounting for data computing and data transport. Motivated by these considerations, we consider the TCP New Reno protocol [88] to model the managed end-to-end intra-DC transport connections. Under the Congestion Avoidance state, the power drained by each connection may be evaluated as in [21]:

$$P_i^{net}(t) = \Omega_i \left(\overline{RTT_i}\, R_i(t)\right)^2, \qquad i = 1,\ldots,M, \ (Watt), \tag{4.6}$$

where $\overline{RTT_i}$ is the average round-trip-time of the i-th intra-data center end-to-end connection, $R_i(t)$ is the communication rate of the i-th virtual link at the t-th slot and Ω_i is the i-th virtual link power coefficient. The latter is dictated by the maximum segment size of the transmitted packets and the number of per-ACK acknowledged segments [65, 83]. Hence, the corresponding one-way transmission delay is: $D_i(t) = L_i(t)/R_i(t)$, where $L_i(t)$ is the workload assigned to $VM(i)$ at the t-th slot. The overall two-way communication-plus-computing delay induced by the i-th end-to-end connection of Fig.4.1 equates: $2D(i)$, so that the hard constraint on the overall per-job execution time reads as in: $\max_{1\leq i\leq M}\{2D(i)\} \leq T_t$. Thus, the corresponding one-way communication energy $E_{LAN}(i,t)$ wasted by the i-th virtual

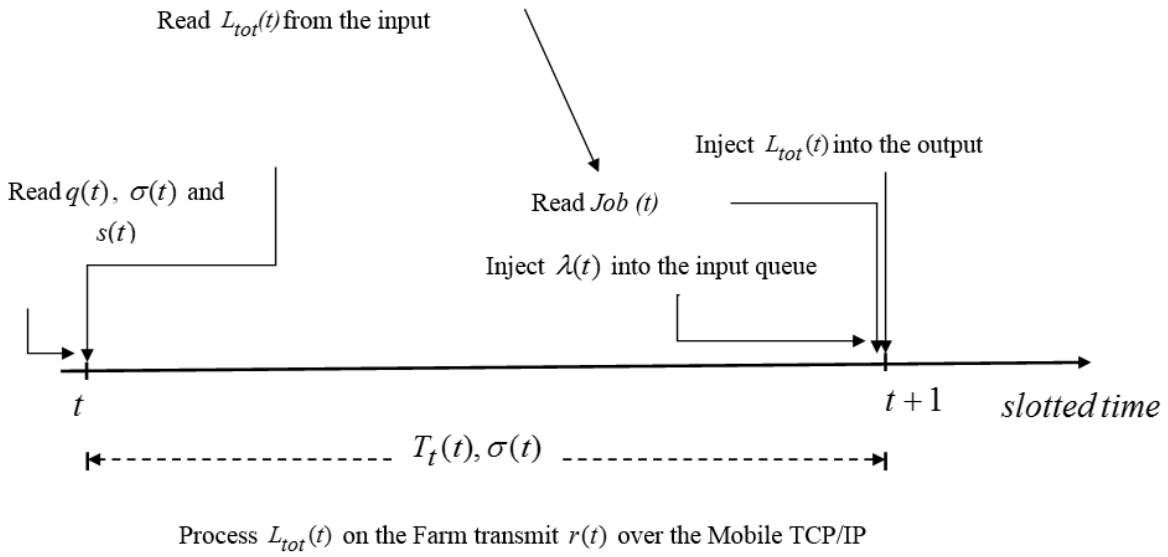


Fig. 4.2 Time chart

link at slot t is:

$$E_{LAN}(i,t) = P_i^{net}(t)(L_i(t)/R_i(t)) \ (Joule). \tag{4.7}$$

4.1.5 Goodput-vs.-energy in TCP/IP mobile connections

In the following, we explore the TCP/IP mobile connection. The TCP/IP mobile connection of Fig.4.1 models the combined effects of the *overall* stack of protocols implemented at the Transport, Network, MAC and PHY layers. The underlying (possibly, multi-hop) physical wireless channel may be affected by multiple access interference, noise and fading phenomena that are assumed *constant* over each slot (i.e. we assume a *block-faded* physical channel that operates in the steady-state [34]). The resulting state $\sigma(t) \in \mathscr{R}_0^+$ at the t-th slot for the overall end-to-end connection of Fig.4.1 is modeled as a real-valued nonnegative r.v., with a *priory unknown* steady-state pdf $p_\sigma(\sigma)$ [2]. The state value $\sigma(t)$ is assumed to be known at

[2] An example of $\sigma(t)$ that models the steady-state behavior of the TCP/IP wireless connection working in the Congestion-Avoidance state is reported in section 4.1. We stress that, in this framework, the r.v. $\sigma(t)$ reflects the fading and congestion-induced fluctuations suffered by the goodput (i.e., the *net* throughput) of the overall end-to-end mobile TCP/IP connection of Fig.4.1.

the controller of Fig.4.1 at the *beginning* of slot t (see Fig.4.2), so that, the connection state information is available at the controller for performing resource allocation on a *per-slot* basis. In agreement with the behavior of legacy TCP/IP connection working in the Congestion Avoidance state [59], we also assume that, at each slot t, $\sigma(t)$ is strictly positive, e.g.,

$$\sigma(t) \geq \sigma_{min} \geq 0, \forall t \geq 0, \tag{4.8}$$

where $\sigma_{min} > 0$ reflects the minimum non-vanishing bandwidth guaranteed by the mobile TCP/IP connection of Fig.4.1 when it operates in the steady-state[3]. Finally, the (possibly, *time-correlated*) random sequence $\{\sigma(t) \in [\sigma_{min}, \infty), t \geq 0\}$ and, then, the overall queuing system of Fig.4.1 are considered to operate under stationary, and argotic conditions (i.e. they work in the *steady-state*)[4]. According to the seminal results of Wolf and Loyes [59], under the aforementioned steady-state operating conditions, there exists a *unique* stationary solution to the recursions in (4.1),(4.2), as detailed by the following Loyes' Lemma [59]:

Lemma 1 *: For any given service policies $L_T(t)$ and $r(t)$, connection state statistics, admission control policy $\lambda(t)$, and exogenous arrival rate $Job(t)$, under queue stability conditions, there exist* unique *r.v.s $\boldsymbol{s}(\infty)$ and $\boldsymbol{q}(\infty)$ such that, for* any *choice of the initial conditions $\boldsymbol{s}(0)$, $\boldsymbol{q}(0)$, the backlog sequences $\{s(t), t \geq 0\}$ and $\{q(t), t \geq 0\}$ in* (4.1)*,* (4.2) *converge in* finite time *to the joint probability distribution of the r.v.'s $\boldsymbol{s}(\infty)$ and $\boldsymbol{q}(\infty)$.* □

Regarding the goodput offered by the vehicular connection of Fig. 4.1, we observe that the transmission rate $r(t)$ over the TCP/IP mobile connection of Fig.4.1 depends on both of the transmit energy $E_W(t)$ and the state $\sigma(t)$ of the connection as in [83]:

$$r(t) = \sigma(t)(E_W(t))^{1/2}, (byte/slot), \tag{4.9}$$

[3]The explicit expression assumed by σ_{min} for a TCP/IP wireless connection working in the congestion-avoidance state is reported in section 4.5.

[4]We anticipate that this assumption will be relaxed in section 4.1, where we deal with the on-the-fly adaptive implementation of the optimal controller of Fig.4.1.

where the state of the connection at slot t is defined as [83]:

$$\sigma(t) \triangleq (K_0(z(t))^{1/2})/CRTT(t), t \geq 1, \tag{4.10}$$

so that:

$$E_W(t) \equiv E_W\Big(r(t), \sigma(t)\Big) = (r(t)/\sigma(t))^2. \tag{4.11}$$

In eq. (4.9), $z(t)$ is the mobility function of the vehicular client linked to the mobile TCP/IP connection of Fig.4.1. It may be modeled as a time-correlated log-distributed sequence [37]: $z(t) \triangleq a_0 10^{0.1x(t)}, \forall t \geq 1$, where $a_0 \approx 0.9738$ and $\{x(t), t \geq 1\}$ is a time-correlated, stationary, zero-mean, unit-variance Markov random sequence, whose probability density function is evenly distributed over the interval $[-\sqrt{3}, \sqrt{3}]$ [37]. The positive constant K_0 in (4.9) captures the performance of the FEC-based error-recovery system implemented at the Physical layer of Fig. 4.1 [83], and $CRTT(t)$ is the corresponding overall round-trip-time in the mobile TCP/IP connection of Fig.4.1. It is calculated iteratively by using the Jacobson's formula as in [83]:

$$CRTT(t) = (1 - j_P)CRTT(t-1) + (j_P)\Delta_{IP}(t), t \geq 1, CRTT(0) = 0, \tag{4.12}$$

where $j_p = 0.25$[83] and $\Delta_{IP}(t)$ is the measured instantaneous packet-delays (in multiple of slot period).

Furthermore, in micro-cellular land-mobile applications, the corresponding correlation coefficient $h \triangleq E\{\mathbf{z}(t)\mathbf{z}(t-1)\}$ that describes the time correlation of the fluctuations of the vehicular connection of Fig.4.1 that can be evaluated as in [37]:

$$h = (0.82)^{vT_s/100}, \tag{4.13}$$

Table 4.1 Main taxonomy of the chapter.

Symbol	**Meaning/Role**
$MSS\ (bit)$	Maximum size for each TCP segment
$N_I(N_O)\ (IU)$	Size of the input (output) buffer
$r_{min}(R_{max})\ (IU/slot)$	Minimum (maximum) instantaneous value for the TCP/IP mobile connection state of Fig.4.1
$Job_{max}\ (IU)$	Maximum instantaneous value of the incoming traffic to the data center
$s(t)(q(t))\ (IU)$	Length of the input (output) buffer or queue at the beginning of slot t
$\Delta_{IP}(t)\ (s)$	Instantaneous packet delay of the TCP/IP mobile connection of Fig.4.1 in the t-th slot
v	Speed of the vehicular client of Fig.4.1
$CRTT(t)\ (s)$	Overall round-trip-time of the i-th TCP/IP mobile connection of Fig.4.1 in the t-th slot
$\overline{RTT_i}\ (s)$	Average round-trip-time of the i-th end-to-end connection
$f_i^{\max}(bit/s)$	Maximum allowed computing frequency for the i-th CPU
$f_i(bit/s)$	Computing frequency for the i-th CPU
$L_{tot}(t)\ (bit)$	Job size (workload size) in the t-th slot
$R_i(bit/s)$	Communication rate of the i-th end-to-end connection
$R_t(bit/s)$	Aggregate communication rate of the Virtual LAN of Fig.4.1
$\Delta(s)$	Per-job maximum allowed computing time
$T_t(s)$	Per-job maximum allowed computing-plus-communication time
$T_s(s)$	Maximum allowed transmit time throw TCP/IP mobile connection
$P_{Idle}(Watt)$	Idle power
$P_i^{net}(Watt)$	Power consumed by the i-th end-to-end connection
$E_c^{idle}(i)(Watt)$	Energy consumed by the i-th CPU in the idle mode
$E_{tot}(t)(Joule)$	Total consumed energy in t-th slot
$E_W(t)(Joule)$	Total transmit energy in TCP/IP mobile connection for the t-th slot
$E_C(i,t)(Joule)$	Total computing energy in t-th slot for i-th VM
$E_{dyn}(i,t)(Joule)$	Total reconfiguration energy in t-th slot for i-th VM
$E_{LAN}(i,t)(Joule)$	Total network energy in t-th slot for i-th VM
M	Maximum number of available VMs

where $T_s(s)$ is the slot duration and $v\ (m/s)$ is the speed of the vehicular client of Fig.4.1.

Overall, the goal we go to pursue is the minimization (on a per-slot basis) of the resulting total communication-plus-computing energy, formally defined as in:

$$E_{tot}(t) \triangleq \left(\sum_{i=1}^{M} E_c(i,t) + E_{dyn}(i,t) + E_{LAN}(i,t) \right) + E_W(t). \tag{4.14}$$

Table 4.1 summarizes the main notations used in this chapter.

4.2 The Afforded Resource Management Optimization Problem

The proposed scheduler aims at minimizing the per-job total energy consumption in (4.14) by selecting the best resource allocation based on the current state $\sigma(t)$ of the TCP/IP vehicular connection. In detail, we build a solution which is capable to obtain the optimum values for the processing speeds of the VMs, the transmission rates of the intra-DC of Fig.4.1 and the transmission rate of the mobile TCP/IP connection of Fig.4.1.

Thus, the overall considered optimization problem assumes the following form:

$$\min_{\mathscr{A}} \quad \theta E_{tot}(t) - (1-\theta) r(t), \tag{4.15.1}$$

subjected to:

$$\sum_{i=1}^{M} L_i(t) = L_{tot}(t), \tag{4.15.2}$$

$$E\Big\{E_W(\sigma, r)\Big\} \leq E_{ave}, \tag{4.15.3}$$

$$0 \leq L_i(t) \leq \Delta f_i^{max}, \qquad i = 1, \ldots, M, \tag{4.15.4}$$

$$D_i(t) \leq \frac{(T_t - \Delta)}{2}, \qquad i = 1, \ldots, M, \tag{4.15.5}$$

$$0 \leq f_i(t) \leq f_i^{max}, \qquad i = 1, \ldots, M, \tag{4.15.6}$$

$$0 \leq R_i(t) \leq R_i^{max}, \qquad i = 1, \ldots, M, \tag{4.15.7}$$

$$r_{min} \leq L_{tot}(t) \leq s(t), \tag{4.15.8}$$

$$s(t+1) \leq N_I, \tag{4.15.9}$$

$$0 \leq \lambda(t) \leq \lambda_{max}, \tag{4.15.10}$$

$$r_{min} \leq q(t+1) \leq N_O, \tag{4.15.11}$$

$$L_{tot}(t) \leq (T_s - \Delta)(R_i^{max}/2). \tag{4.15.12}$$

where $\mathscr{A} \triangleq \{L_i(t), f_i(t), i = 1,\ldots,M, r(t), \lambda(t)\}$ is the set of the (2M+2) variables to be optimized and $\theta \in [0,1]$ acts as normalization weight.

About the reported problem, some explicative remarks may be of interest. The objective function in problem (4.15.1) in general weighted linear combination of the per-slot total energy consumption and the goodput of the TCP/IP vehicular connection of Fig.4.1. Setting of θ allows us to assign relative priorities between goodput and energy consumption.

The constraints in (4.15.2) and (4.15.4) guarantee that the overall input workload $L_{tot}(t)$ offered to the Cloud at slot t is partitioned over the available VMs in a feasible manner. The constraint in (4.15.3) limits up to E_{ave} $(Joule)$ the per-slot overall energy $E_W(.;.)$ available for transmitting over the mobile TCP/IP connection of Fig.4.1. The set of time constraints in (4.15.5) enforces the virtualized Cloud of Fig.4.1 to fully process the offered workload $L_{tot}(t)$ within the *hard* deadline of T_t (s). Eqs. (4.15.6) and (4.15.7) limit the processing rate of $VM(i)$ and the communication rate of the i-th link of the intra-DC of Fig.4.1, respectively. Eqs. (4.15.8), (4.15.10) and (4.15.9) account for the current and next backlog $s(t)$ and $s(t+1)$ of the input queue (see Fig.4.1), the maximum allowed input rate $\lambda_{max}(byte/slot)$ and the size of the input buffer N_I, respectively. Likewise, eq. (4.15.11) accounts for the finite size N_O of the output buffer of Fig.4.1. Equation (4.15.12), limits the processing rate of the $VM(i)$ and account for the aggregate transmission rate of the DC's LAN of Fig.4.1. Finally, in the reported formulations, the *instantaneous* goodput $r(t)$ conveyed by the mobile connection of Fig.4.1 must fall into the range $[r_{min}, r_{max}]$ where r_{max} is the maximum instantaneous value allowed for goodput. Formally speaking, the (nonnegative) threshold r_{min} $(byte/slot)$ fixes (in a *hard* way) the *minimum* goodput to be guaranteed by the mobile pipe of Fig.4.1, while the (positive) threshold r_{max} $(byte/slot)$ dictates (in a *hard* way) the maximum allowed instantaneous goodput.

Remark 2 - Generalization of the optimization problem formulation.

Depending on the actually considered TCP/IP mobile cloud platform and sustained Software-as-a-services, some generalizations of the reported problem optimization formulation are possible.

First, several reports point out that the energy consumption of other non-IT equipments (e.g., cooling equipments) is roughly proportional to that in (4.14) through a constant factor power usage effectiveness (PUE), which represents the (measured) ratio of the total energy wasted by the data center to the energy consumed by the corresponding computing-plus-networking equipments [3].

Second, the size $\tilde{L_i}^{(0)}(t)$ of the workload output by $VM(i)$ at the end of the computing phase may be different from the corresponding one $L_i(t)$ received in input at the beginning of the computing phase in the t-th slot. Therefore, after introducing the i-th inflating/deflating constant coefficient $\vartheta_i \triangleq \tilde{L}_i^{(0)}(t)/L_i(t) \lesseqgtr 1$, the basic problem optimization formulation may be generalized by simply replacing the term: $2L_i(t)$ in (4.15.1), (4.15.5) by the following one: $(1+\vartheta_i)L_i(t), i = 1,\ldots,M$.

Third, the summation in (4.15.1) of the computing and reconfiguration energies may be replaced by *any* two energy functions: $H(f_1,\ldots,f_M)$ and $V(f_1,\ldots,f_M)$ which are jointly convex in $\{f_i, i = 1,\ldots,M\}$. Just as an application example, as in [105], let us assume that all the VMs are instantiated onto the same physical server and, due to imperfect isolation, they directly compete for acquiring CPU cycles. In this (limit) case, the sum-form in (4.14) for the computing energy falls short, and the total energy $E_c(i,t)$ wasted by the physical host server may be modeled as in [105]: $E_c(i,t) = E_c^{\max}\left\{(1-b)\left[\sum_{i=1}^{M}\left(\frac{f_i}{f_i^{max}}\right)\right]^C + b\right\}$, where $E_c^{\max}$ is the maximum energy consumed by the physical server, $E_c^{idle} \triangleq b\,E_c^{\max}$ is the corresponding energy consumed by the physical server in the idle state, while the inner summation is the *aggregate utilization* of the server by all hosted VMs. Since the above expression of $E_c(.)$ is jointly convex in $\{f_i, i = 1,\ldots,M\}$ for $C \geq 1$, the solving approach of the next section still applies verbatim. □

4.3 The Resulting Adaptive Resource Scheduler

We start presenting some structural properties of the afforded problem, described in Lemma 2, and its feasibility conditions in Proposition 8, respectively.

Lemma 2 *Structural properties of the afforded problem Let the aforementioned assumptions on the system's model be fulfilled. Thus,*

i) in the steady-state, the following chain of inequalities holds:

$$Job_{max} \geq E\{Job(t)\} \geq^{(a)} E\{\lambda(t)\} =^{(b)} E\{L_{tot}(t)\} =^{(c)} E\{r(t)\} \geq^{d} r_{min}, \quad (4.16)$$

ii) the energy function $E_W^*(\sigma, r)$ in (4.15.3) *is strictly increasing for $r \geq 0$, strictly decreasing for $\sigma \geq 0$ and strictly convex in $r \geq 0$;*

iii) the instantaneous energy $E_W(t) \equiv E_W(r(t), \sigma(t))$ in (4.15.3) *wasted for transmitting over the mobile TCP/IP connection of Fig.4.1 is upper bounded in a* hard *way as in*

$$E_W(\sigma_{min}, r_{max}) \geq E_W(t), \quad \forall t \geq 1, \quad (4.17)$$

where σ_{min} and r_{max} are the minimum connection state of the TCP/IP end-to-end backbone and the maximum desired goodput for the rate of the TCP/IP vehicular connection, respectively.

Proof:

The reported proof exploits arguments based on the considered TCP/IP vehicular cloud architecture as is shown in Fig.4.1.

i) Since the input and output queue of Fig.4.1 are in tandem, loss-free and operate in the steady-state, the equalities in (b), (c) of (4.16) must hold. Afterwards, (d) follows from the lower bound in (4.15.8), while (a) stems from the performed admission control on the arrival input traffic.

ii) Since $r(t)$ in (4.11) is strictly increasing in $\sigma \geq 0$ and $E_W \geq 0$ and strictly concave in E_W, its inverse function r^{-1} with respect to E_W is strictly increasing and strictly convex in $r \geq 0$, and it is strictly decreasing in $\sigma \geq 0$.

iii) From (4.11) it follows that the energy function $E_W(\cdot)$ attains its maximum at $\sigma = \sigma_{min}, r = r_{max}$. This proves the validity of (4.17).

This completes the proof of Lemma 2. □

About σ_{min}, it reflects the minimum non-vanishing bandwidth guaranteed by the TCP/IP vehicular connection of Fig.4.1 when it operates in the steady-state. Interestingly, we can calculate it in closed form as:

$$\sigma(t) \geq \sigma_{min} \triangleq K_0 \left((a_0 10^{-0.1(3)1/2})^{1/2} / \Delta_{IP}^{max} \right), t \geq 1, \tag{4.18}$$

where K_0 accounts for the performance of the FEC-based error recovery scheme implemented by the PHY layer of the connection of Fig. 4.1 [65], and Δ_{IP}^{max} is the maximum delay actually experienced by the connection at the slot t. Likewise, according to the eqs. (4.18) (4.8) and (4.9): $r_{min} \triangleq \sigma_{min}\sqrt{E_{ave}}$.

Since inactive VMs may be turned ON at the consolidation instants in response to workload increments, it suffices to afford the feasibility issues of the resource allocation problem in (4.15) under the assumption that all the available VMs are turned ON at $t = 0$. So doing, we obtain the feasibility conditions for the afforded problem.

Proposition 8 *Feasibility conditions*

Under the reported assumptions, the following four inequalities:

$$E_{\sigma}\{E_W(\sigma, r_{min})\} \leq E_{ave}, \tag{4.19.1}$$

$$\sum_{i=1}^{M} f_i^{max}\Delta \geq r_{min}, \tag{4.19.2}$$

$$(R_i^{max}/2)(T_s - \Delta) \geq N_I, \tag{4.19.3}$$

$$Job(t) \geq r_{min}, \forall t \geq 0, \tag{4.19.4}$$

provide a set of sufficient and necessary conditions for the feasibility of the constrained optimization problem in (4.15).

Proof: Since the optimization problem The reported proof highlight the optimization problem (4.15.1) feasibility conditions.

a) Since Lemma 1 proves that $E_W(\sigma, r)$ is strictly increasing for $r \geq 0$, the condition in (4.19.1) is both necessary and sufficient for guaranteeing that the constraint in (4.15.1) is always met.

b) Since the l.h.s. of (4.19.2) represents the maximum workload that the Cloud of Fig.4.1 may process during a slot period (see (4.15.4)), we have:

$$L_{tot}(t) \leq \sum_{i=1}^{M} f_i^{max}\Delta, \qquad \forall t \geq 0. \tag{4.20}$$

Hence, since (see eq. (4.2)): $q(t+1) \geq L_{tot}(t)$, after posing: $L_{tot}(t) \equiv \sum_{i=1}^{M} f_i^{max}\Delta$, eq. (4.19.2) suffices to guarantee that: $q(t+1) \geq r_{min}$, so that the constraint in (4.15.11) is met. About the necessary part of the condition in (4.19.2), let us consider an instant $\underline{t}$ such that: $r(\underline{t}) = q(\underline{t})$. Hence, since (see eq. (4.2)): $q(\underline{t}+1) = L_{tot}(\underline{t})$, and $L_{tot}(\underline{t})$ is upper bounded as in (4.20), it follows that (4.19.2) is necessary for having: $q(\underline{t}+1) \geq r_{min}$.

c) Since the size of the input queue is fixed at N_I, we have that: $L_{tot}(t) = \sum_i^M L_i(t) \leq N_I, t \geq 0$, and this inequality proves that the condition in (4.19.3) suffices for guaran-

teeing the feasibility of the constraint in (4.15.12). About the necessary part, let us consider an instant $\underline{t}$ such that (see eqs. (4.1),(4.15.8)): $s(\underline{t}) = L_{tot}(\underline{t}) = N_I$. Hence, in order to guarantee that the aggregate capacity R_i^{max} of the intra-Cloud LAN of Fig.4.1 is capable to transport N_I IUs within $(T_s - \Delta)$ seconds, the condition in (4.19.3) must necessarily hold.

d) Since the input and output queues of Fig.4.1 work in a pipelined fashion, the condition in (4.19.4) suffices for having: $q(t+1) \geq r_{min}, \forall t \geq 0$, that, in turn, is sufficient for guaranteeing that: $r(t) \geq r_{min}, \forall t \geq 0$. About the necessary part of the condition in (4.19.4), let us consider an instant $\underline{t}$ such that (see eqs. (4.1),(4.2)): $s(\underline{t}) = L_{tot}(\underline{t})$, and: $q(\underline{t}+1) = r(\underline{t}+1)$. Hence, since the upper bounds in (4.15.8) and (4.15.9) lead to the following two chains of inequalities: $Job(\underline{t}) \geq \lambda(\underline{t}) = s(\underline{t}+1) \geq L_{tot}(\underline{t}+1)$, and: $q(\underline{t}+2) = L_{tot}(\underline{t}+2) \geq r(\underline{t}+2)$, it directly follows that the condition: $Job(\underline{t}) \geq r_{min}$ is necessary for having: $r(\underline{t}+2) \geq r_{min}$. □

Proposition 9 *(Preserve task granulity)*

Let be $r_{max} > r_{min}$ and let r_{max} (IU) and r_{min}(IU) comprise integer numbers of tasks. Let the feasibility conditions of Proposition 8 be met, and let $r_L(t)$, $r_H(t)$ the transmit policies so defined:

$$r_L(t) \triangleq \chi round(r^*(t)/\chi); r_H(t) \triangleq \chi \min\Big\{ceil(r^*(t)/\chi); round(q^*(t)/\chi)\Big\}; \tag{4.21}$$

$$\overline{E}_L(t) \triangleq E_\sigma\Big\{E_W(\sigma, r_L(t))\Big\}, \quad and \quad \overline{E}_H(t) \triangleq E_\sigma\Big\{E_W(\sigma, r_H(t))\Big\}, \tag{4.22}$$

be the conditional average transmit energies required by the policies $r_L(\cdot)$ and $r_H(\cdot)$, respectively. Finally, let $P_0(t)$ the following probability:

$$P_0(t) \triangleq \frac{\overline{E}_H(t) - E_{ave}}{\overline{E}_H(t) - \overline{E}_L(t)} \tag{4.23}$$

Thus, the randomized transmit policy so defined:

$$\tilde{r}(t) \triangleq \begin{cases} r_L(t), & P_0(t), \quad t \geq 1 \\ r_H(t), & 1 - P_0(t), \quad t \geq 1 \end{cases} \tag{4.24.1}$$

retains the following structural properties:

i) $\tilde{r}(t)$ *conveys an integer number of tasks for any* t*;*

ii) $r_{min} \leq \tilde{r}(t) \leq r_{max}, \quad t \geq 1$*;*

iii) $\tilde{r}(t) \leq q^*(t), \quad t \geq 1$*;*

iv) $E_\sigma\{E_W(\sigma, \tilde{r}(t))\Big\} = E_{ave}$*;*

v) $\tilde{r}(\cdot)$ *is the (possibly, not unique) policy with the largest conditional average value* $E_\sigma\{\tilde{r}(t)\}$ *over the set of integer-valued policies that meet the constraints in* (4.15.1)-(4.15.12).

Proof:

i) Since both $r_L(\cdot)$ *and* $r_H(\cdot)$ *in eq.* (4.21) *convey integer numbers of tasks by design, the same property is also inherited by* $\tilde{r}(\cdot)$ *in eq.* (4.24).

ii) Since both r_{min} *and* r_{max} *embrace integer numbers of tasks and* $r^*(t) \in [r_{min}, r_{max}]$*, from eq.* (4.21) *it follows that* $r_L(t)$ *and* $r_H(t)$ *fall into the interval* $[r_{min}, r_{max}]$ *for any* t*. Hence, due to eq.* (4.24)*, the same property is inherited by* $\tilde{r}(\cdot)$.

iii) Due to (4.2)*,* $r^*(t)$ *and, then,* $r_L(t)$ *and* $r_H(t)$ *are limited up to* $q^*(t)$*, so that the same property is inherited by* $\tilde{r}(\cdot)$ *in eq.* (4.24).

iv) Equation (4.23) *may be rewritten as in:* $P_0(t)\overline{E}_L(t) + (1 - P_0(t))\overline{E}_H(t) = E_{ave}$*, that, in turn, directly proves the statement.*

v) Since $r^*(t)$ *is not constrained to be an integer multiple of* χ*, as in [72, section II], the conditional average transmit rate of any integer-valued feasible transmit policy is*

upper bounded by: $E_\sigma\{r^*(t)\}$*, so that* $E_\sigma\{\tilde{r}(t)\} \leq E_\sigma\{r^*(t)\}$*. Furthermore, as in [72, section II],* $r_L(t)$ *and* $r_H(t)$ *in eq.* (4.21) *are the nearest integer-valued neighboring rates that lower and upper limit* $r^*(t)$*, while simultaneously meeting the (required) point-wise constraints:* $r_{min} \leq r_L$*,* $r_H \leq r_{max}$ *and* $r_L \leq q^*$*,* $r_H \leq q^*$*. Finally, as in [72, section II], the time-sharing factor* $P_0(t)$ *in eq.* (4.23) *is the only one that allows to meet with the equality the constraint on the (conditional) average transmit energy (see* (4.23)*), and, then, it concurs to guarantee the feasibility of the policy* $\tilde{r}(\cdot)$*. Overall, as in [72, section II],these three properties suffice to prove the claim.*

Although optimal, actual implementation of the policy eq. (4.24) may be hampered by the fact that the closed-form evaluation of the conditional expectations in eq. (4.22) requires the a priori knowledge of the (generally unknown) pdf $p_\infty(\sigma)$. As detailed in the sequel, $P_0(t)$ may be iteratively computed and updated by resorting again to the gradient-based algorithm, but this somewhat increases the implementation complexity of the overall controller. Specifically, let $N_H(t-1)(N_L(t-1))$ be the integer-valued number of times such that the policy $r_H(\cdot)(r_L(\cdot))$ is actually applied over the closed time interval $[0,t-1]$. Furthermore, let $1_{[t;H]}$ be the binary variable that takes the unit value *iff* the policy $r_H(\cdot)$ is actually applied over the t-th slot. Hence, the Newton-type t-th iterate for updating $P_0(t)$ is carried out at the beginning of the t-th slot. It reads as

$$P_0(t) = P_0(t-1) - \beta_0(t)\left[P_0(t-1) - \left(\left(\overline{E}_H(t-1) - E_{ave}\right)/\left(\overline{E}_H(t-1) - \overline{E}_L(t-1)\right)\right)\right], \quad t \geq 1, P_0(0) = 1, \tag{4.25}$$

where,

$$N_H(t-1) = N_H(t-2) + 1_{[t-1;H]}, t \geq 2, N_H(0) = 0, \tag{4.26}$$

$$\overline{E}_H(t-1) \triangleq \Big(1/N_H(t-1)\Big) \sum_{i=1}^{(t-1)} E_W\Big(r_H(i),\sigma(i)\Big) \equiv$$
$$\Big(1/N_H(t-1)\Big)\Big[N_H(t-2)\overline{E}_H(t-2) + 1_{[t-1;H]} E_W\Big(r_H(t-1),\sigma(t-1)\Big)\Big], \tag{4.27}$$
$$t \geq 2, \overline{E}_H(0) \equiv E_W\Big(r_{max},\sigma_{min}\Big),$$

together with

$$\overline{E}_L(t-1) = \Big(1/N_L(t-1)\Big)\Big[N_L(t-2)\overline{E}_L(t-2) + \Big(1 - 1_{[t-1;H]}\Big) E_W\Big(r_L(t-1),\sigma(t-1)\Big)\Big],$$
$$t \geq 2, \overline{E}_L(0) = 0, \tag{4.28}$$

Finally, analogously to 5 line #15, the (possibly, time-varying) step-size $\beta_0(t) \in \mathscr{R}_0^+$ in (4.25) may be still updated according to [60, equation (2.4)], in order to cope with the (possible) occurrence of abrupt changes of the statistics of the r.v. $\sigma(\cdot)$. Before proceeding, two remarks are in order. First, under the feasibility condition in (4.19.3), the constraint in (4.15.12) is automatically met. Second, since the steady-state pdf $p_\infty(\sigma)$ of the wireless TCP/IP connection of Fig.4.1 may be a priori unknown, the condition in (4.19.1) could be hard to test. However, since $E_W(\sigma,r)$ is not increasing in σ, the following relationship: $E_W(\sigma_{min},r_{min}) \leq E_{ave}$, provides a simpler to test (only) sufficient condition for the feasibility of the average energy constraint in (4.15.1).

To solve the optimization problem in (4.15), we must find the optimum $L_{tot}^*(t)$ allotted by the admission control module of Fig.4.1. It can be formally proved that $L_{tot}^*(t)$ may be calculated according the following equation (4.29)

$$L_{tot}^*(t) \triangleq \begin{cases} 0 & \text{for } q(t) > N_O + r_{min} - r_{max}, \\ \mathscr{W}\mathscr{L}(t) & \text{for } q(t) \leq N_O + r_{min} - r_{max}, \end{cases} \tag{4.29}$$

with $\mathscr{WL}(t) \triangleq \min\left\{s(t); \max\{r_{min}; \overline{r}(t-1)\}\right\}$ where, according to the Jacobson's formula, we have that:

$$\overline{r}(t) \triangleq (1-J_P)\overline{r}(t-1) + J_P\, r(t), t > 1, \tag{4.30}$$

is the average measured rate of the underlying mobile TCP/IP connection. Moreover, the optimal admitted workload sent over the Cloud (i.e., the input buffer of Fig.4.1) during slot t can be calculated according to the following formula:

$$\lambda^*(t) = \min\{Job(t), N_I - s(t) + L^*_{tot}(t)\}, \tag{4.31}$$

which is comprises by J_P threshold of the effect of the instantaneous rate of the previous slot and $(1-J_P)$ rate of average of $(t-1)$ slots in the mobile TCP/IP connection in the Fig.4.1. where $\overline{r}(t)$ is the average goodput rate from the first slot until slot t for the output queue which is called *throughput* or *goodput* of the whole system. For example, if we are at time-slot $t = 10$ we need to sum all the r values for all the time-slots from $\tau = 0$ to $\tau = 10$ which is calculated in previous slots. Also, when we go to the next slot ($t = 11$) we do not need to calculate all the old goodput values for the previous slots just we can use the following formula instead of the formula (4.30)

$$\overline{r}(t) \triangleq \frac{\overline{r}(t-1)(t-1) + r(t)}{t} \tag{4.32}$$

Therefore, $r(t)$ is correlated to the previous $r(t-1)$ and $\sigma(t)$ is related to channel connection state at slot t. The $L^*_{tot}(t)$ is depicted in figure 4.3: After calculating $L^*_{tot}(t)$, we need as in (4.14) to calculate the general optimal transmit rate $r^*(t)$ over the TCP/IP mobile connection, which, in turn, equates:

$$r^*(t) = \left[\dot{E}_W^{-1}\left(\sigma, \frac{(1-\theta)}{MU(t)}\right)\right]_{r_{min}}^{\min\left\{r_{max}; q\right\}}. \tag{4.33}$$

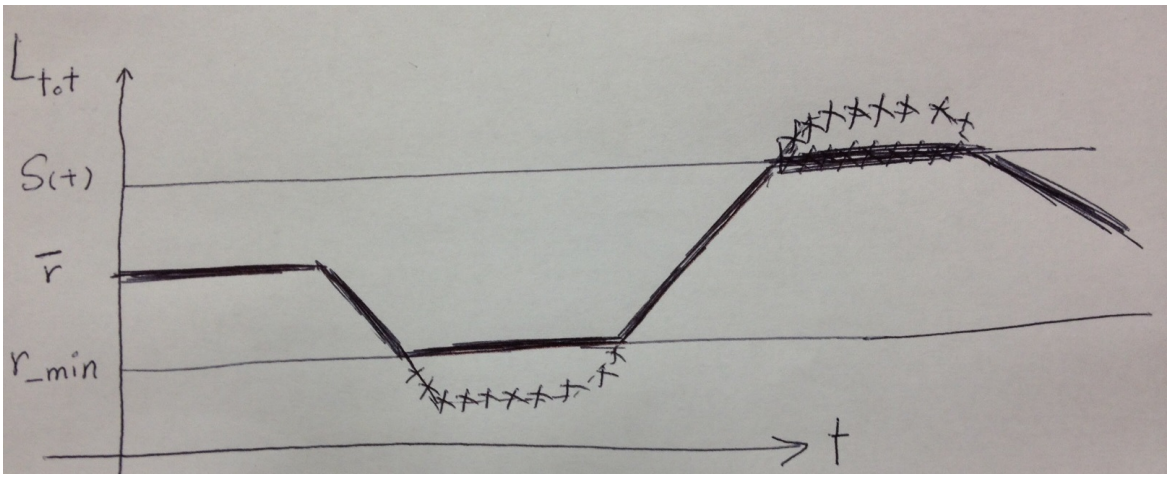


Fig. 4.3 $L_{tot}(t)$ is calculated by time-slot, the x-axis is time-slot and y-axis is the $L_{tot}(t)$ which is based on two hard-limits (r_{min} and $s(t)$) : the dotted lines $(-)$ is not exist and just the dark-lines is the result for the input/output workload of output /input queue.

where $\dot{E}_W^{-1}$ is the inverse of the derivative of $E_W(t)$ in eq. (4.11) and the projection operator $[x]_a^b \triangleq \min\{b;\max\{a;x\}\}$ in (4.16), so that the resulting $r^*(t)$ in (4.16) reads as in:

$$r^*(t) \equiv \left[\frac{(1-\theta)\sigma(t)^2}{2MU(t)}\right]_{r_{min}}^{\min\left\{r_{max},\, q(t)\right\}}. \tag{4.34}$$

where $MU(t)$ is a suitable positive step-size sequence adaptively updated slot-by-slot to meet the constraint (4.15.3) (through the standard gradient based updates:

$$MU(t+1) = [MU(t) + (k(t)/t)(E_W^*(t) - E_{ave})]. \tag{4.35}$$

In principle, the actual choice of $\{MU(t)\}$ impacts on the rate of convergence and tracking capability of the time slots to attain (4.15.3), and the step-size parameter $k(t)$ can be adaptively optimized slot-by-slot to improve the speed of convergence (see [16]).

About the QoS issues, we point out that, regardless of the type of service offered by the available wireless TCP/IP connection of Fig.4.1, the ultimate goal of the DC of Fig.4.1 is to guarantee a minimum level of QoS (e.g., service level agreement (SLA) [24]) to the vehicular clients. A combined exploitation if the bounds N_I, N_O on the buffer sizes and limits (4.15.8),

(4.15.11) on the minimum and maximum rates delivered by the output queue of Fig.4.1 leads to the following hard QoS guarantees.

Proposition 10 *Hard QOS guarantees*

Let the feasibility conditions of Proposition 8 be met. Let T_{QI}, T_{SI}, T_{QO}, and T_{SO} be the r.v.s that measure the random queue delay of the input queue, the service time of the input queue, the queue delay of the output queue and the service time of the output queue (in multiple of the slot time), respectively, thus,

i) the random total delay $T_{TOT} \triangleq T_{QI} + T_{SI} + T_{QO} + T_{SO}$ (in multiple of the slot period) is limited in a hard *way as*

$$T_{TOT} \leq ((N_I + N_O)/r_{min}) + 2; \tag{4.36.1}$$

ii) The instantaneous jitter affecting the transmit rate is upper bounded as in:

$$|r(t_1) - r(t_2)| \leq (r_{max} - r_{min}), \text{ for any } t_1, t_2 \tag{4.36.2}$$

iii) Furthermore, the corresponding unconditional average *jitter: $\sigma_r \triangleq \sqrt{E\{r(t) - \bar{r}\}^2}$ $(IU/slot)$ is limited as*

$$\sigma_r \leq \sqrt{(r_{max})^2 - (r_{min})^2}. \tag{4.36.3}$$

We developed an iterative method, which calculates the optimum L_i and f_i for each $VM(i)$ in each t. After some iterations, we will reach the optimal solution of the considered problem, which consists of a set of optimal parameters: $\{L_i^*, f_i^*, i = 1, \ldots, M\}$ that are expressed in function *AdLoadDispatch*. Indeed, we iterate our method n-times (i.e., n is the loop counter for searching proper optimum workload and frequency for each VM). The pseudocode for this proposed solving method is presented in **Algorithm 2**. In detail, for the consolidation state we introduce a function *ConsolDispatch* which is described in the next section.

The **Algorithm 3** details the applied admission control. In detail, at each time slot, we need to perform lines 3 to 9, in order to compute the optimum allowed workload which should

Algorithm 2 Proposed Algorithm

Input Jobs
Output $\mathscr{A}^*$
for $t \geq 1$ **do**
    if $\sim$*Feasibility* in Proposition 8 **then**
        error('Program is not feasible')
    else
        ComCopResAlloc(t)
        Calculate $W(t)$ using (4.37)
        if $Lth \leq W(t) \leq Hth$ **then**
            AdLoadDispatch(t)
        end if
        if *not* $(Lth \leq W(t) \leq Hth)$ **then**
            ConsolDispatch(t)
        end if
    end if
    Calculate DVFS in Remark 1, (4.15.1)
end for
return E^*_{tot}, $\mathscr{A}^*$

Algorithm 3 ComCopResAlloc()

Input t-th Job
Output $L^*_{tot}(t)$, Admission Control parameters
Compute $\bar{r}(t)$ by using (4.30) as Jacobson formula
Compute $L^*_{tot}(t)$ by using (4.29)
Compute $\lambda(t)$ using (4.31)
Compute $r^*(t)$ using modified (4.34)
Compute $E^*_W(t)$ using second part (4.11)
Compute $MU(t+1)$ using (4.35)
Compute $s(t+1)$ and $q(t+1)$ using (4.1) and (4.2), respectively
return $L^*_{tot}(t)$, Admission Control parameters

be dispatched to active VMs and, then, injected into the mobile TCP/IP connection. Here, $Job(t)$ denotes the input traffic before the admission control (i.e., at the input of the input buffer) and MU is the step-size parameter adopted to control the attainment of the equality constraint in (4.15.3). $MU(t)$ is updated on the basis of its current value (it was calculated based on the previous round/iteration) and the variation of MU can be managed with the coefficient k/t. It means that, when the time slot increases, the step-size parameter turn to be near to its previous values like [76].

Algorithm 4 AdLoadDispatch()

Input $L^*_{tot}(t)$, f_i^{max}, $iter^{max}$, Δ
Output $\mathscr{A}^*$, and optimized instantiates energy of section 4.1
$G(L_i) \triangleq \sum_{i=1}^{M} L_i(t) - L^*_{tot}(t)$
$N \leftarrow 1$, $a \leftarrow 0$
$itr = [\log(f_i^{max} - a)/\min(P_{LAN})]$
while $N \leq itr$ **do** for each $i \in 1, \ldots, M$ in parallel
    $c \leftarrow (a + f_i^{max})/2$
    Calculate $G(c\Delta)$, $G(a\Delta)$
    if $G(c\Delta) = 0$ or $f_i^{max} - a < 0$ **then**
        return $c, G(c\Delta)$
        Stop
    end if
    if $G(a\Delta)G(c\Delta) < 0$ **then**
        $b = c$, $G(b\Delta) = G(c\Delta)$
    else
        $a = c$, $G(a\Delta) = G(c\Delta)$
    end if
    $N++$
end while
Return $c, G(c\Delta)$

Algorithm 4 is an instance of the bisection method to solve (4.15.2) by using higher and lower range for each VM: f_i/L_i and obtain a rough approximation to a solution which is $\mathscr{A}^*$.

4.4 Dynamic Turning ON/OFF VMs

Data center consolidation is a popular strategy to further reduce energy consumption by turning OFF the under utilized VMs. The effectiveness of data center consolidation in

driving costs out of IT is shown in the popularity of this strategy. The IT organizations apply consolidation to reduce IT assets by using more efficient technologies. Recently, some of the consolidation technologies used in data centers comprise server virtualization, storage virtualization, cloud computing, and using tools for process automation. In this paper, we use server virtualization as a dynamic control to improve energy efficiency in cloud data center. In this paper, we use VM consolidation as a dynamic control tool to improve energy efficiency in virtualized cloud data center. We focus on consolidation policies that work at multiple time scales and allow consolidation only at instants falling into a predefined index set: $\hat{T}_{CN} \triangleq \{\hat{t}_0, \hat{t}_1 \ldots\}$ of consolidation instants. Roughly speaking, this means that the set $\mathscr{S}$ of active VMs is chosen at $\hat{t}_i \in \hat{T}_{CN}$, and, then, it is held fixed up to the instant: $t = (\hat{t}_{i+1} - 1)$. The set of active VMs may potentially change at $\hat{t}_{i+1}$. Hence, in our framework, while consolidation decisions are taken at *slower* time-scale, all the other resource allocation decisions (e.g., admission control, workload dispatching, queue updating, and allocation of the computing-plus-communication resources) are performed on a *per-slot* basis in order to evaluate online the actual utilization of the corresponding active VMs, we introduce W as a *VM consolidation factor* to monitor each slot and find the consolidation slots. W is defined as in

$$W(t) \triangleq \frac{1}{|\mathscr{S}(t)|} \sum_{i \in \mathscr{S}(t)} \frac{L_i(t)}{L_i^{max}}, \tag{4.37}$$

where $L_i^{max} \triangleq \Delta f_i^{max}$ is the maximum processed workload of $VM(i)$ and $\mathscr{S}(t)$ is the set of turn ON VMs at the t-th slot. The W factor falls in to the interval zero and 1. In detail, for this method, two thresholds are set ($\mathscr{I} \triangleq (Lth, Hth)$), a lower and an upper thresholds, respectively). If the consolidation factor W drops below the lower threshold (Lth), then some VMs according to the consolidate multi-gradient iterative method and all the VMs residing on that servers should be removed (i.e., workloads also removed) so that the host can be switched off in order to save energy. If W is higher than the upper threshold, then some new servers should be on and some of the VMs residing on the current servers should be removed

and migrate to the new servers in order to avoid SLA violations. Without loss of generality, we assume that the $VM(i)$ is turned ON at slot t when the corresponding processing speed $f_i \geq f_i^{ON}$, where f_i^{ON} is an assigned positive wake up frequency. Hence, at each consolidation instant $\hat{t} \in \hat{T}_{CN}, \hat{t} \geq 1$, we may model the overall energy consumption of the $VM(i)$ as in

$$E_c(i,t) = E_c^{idle}(i)u_{-1}(f_i(\hat{t})) + (f_i/f_i^{max})^2(E_c^{max}(i) - E_c^{idle}(i)), \quad i \in \{1,\ldots,M\}, \hat{t} \in \hat{T}_{CN}, \tag{4.38}$$

where $u_{-1}(x)$ is the unit-size Heaviside' function (e.g., $u_{-1}(x)$ is unit for $x > 0$ and it vanishes for $x \leq 0$). The first term in (4.38) accounts for the saving (consumption) of the static computing energy that arises from turning OFF (ON) the $VM(i)$, $i = 1,\ldots,M$. In order to model the corresponding time-overhead induced by the turning ON/OFF operations, let $T_{ON}(s)$ and $T_{OFF}(s)$ be the times needed to turn ON/OFF a VM, respectively. Hence, under the assumptions: $T_{ON} \leq \Delta$ and $T_{OFF} \leq T_t$, the following set of constraints must hold at each consolidation instant $\hat{t} \in \hat{T}_{CN}$, $\hat{t} \geq 1$:

$$T_{ON}\, u_{-1}\Big(f_i(\hat{t})\Big) + \Big(\frac{L_i(\hat{t})}{f_i(\hat{t})}\Big) \leq \Delta, \quad i \in \overline{S}(\hat{t}-1), \tag{4.39.1}$$

$$\Big(\frac{L_i(\hat{t})}{f_i(\hat{t})}\Big) \leq \Delta, \quad i \in S(\hat{t}-1), \tag{4.39.2}$$

$$\sum_{i=1}^{M} L_i(\hat{t}) = L_{tot}(\hat{t}), \quad i \in \overline{S}(\hat{t}-1). \tag{4.39.3}$$

where $\overline{S}(\hat{t}-1)$ is the set of the VMs that are turned OFF at slot $(\hat{t}-1)$ while $S(\hat{t}-1)$ is the set of the VMs that are turned ON at slot $(\hat{t}-1)$. Specifically, the constraint in (4.39.1) applies only to the set of VMs that are inactive at $(\hat{t}-1)$ (e.g., to the VMs falling into the set $\overline{S}(\hat{t}-1)$). This constraint explicitly accounts for the time-overhead requested for turning ON the inactive $VM(i)$. The constraint in (4.39.2) applies only to the VMs that are already active at $(\hat{t}-1)$. Since it forces vanishing workload $L_i(\hat{t})$ at $f_i(\hat{t}) = 0$, (4.39.2) guarantees that no workload is carried out by the $VM(i)$ during each slot t at which it is turning OFF. This is also

the reason why T_{OFF} is not explicitly present in (4.39.2). Finally, (4.39.3) still guarantees workload conservation at each consolidation instant. Before processing some explicative remarks about the assumed cloud architecture and its venality limits may be of interest. In time-varying environments characterized by (possibly, abrupt and random) time-fluctuations of the offered workload L^*_{tot} (see **Algorithm 3**), the per-job evaluation and online tracking of the consolidation state in the following may be performed by resorting to a gradient-based updating in [83]. The details is simplified in the following **Algorithm 5**. In this algorithm, $l \geq 1$ is an integer value iteration index, $\hat{t}$ is the t-th time slot which is in consolidation state, $\gamma^{(l-1)}(t)$, is a (suitable) l-variant nonnegative step-size sequence, and the dummy iterates in the **Algorithm 5** hold (i.e., lines 9-16). In this Algorithm, $L_i^{(l)}(\hat{t})$, $\tilde{L}_i^{(l)}(\hat{t})$, and $\overline{L}_i^{(l)}(\hat{t})$ are the instantaneous, estimated, and average/expectation workload in each VM in the $\hat{t}$-slot.

where two functions *ConsolCpuDyn* and *ConsolLAN* describe the average consumed energy for the servers and intra-DC links in the DC in the consolidation state of the Virtualized Cloud in Fig.4.1, respectively. For more information on the aforementioned functions follow the Appendix F. The final point is that the complexity of the proposed joint scheduler is linear in the number M of the utilized VMs and is lower than the schedulers in [21, 91].

4.5 Test Results and Performance Comparisons

This section presents the tested energy performance of the proposed scheduler for a set of synthetic and real-world input traffic traces and, then, compare it with the corresponding ones of the recent DVFS-based techniques in [83], the Lyapunov-based scheduler in [91], the static and hybrid schedulers in [69, 6], and the NetDC scheduler in [23].

4.5.1 Simulated Cloud setup

The simulations have been carried out by exploiting the numerical software of the MATLAB platform. They emulate 10 quad-core Dell PowerEdge servers, equipped with 3.06 GHz Intel

Algorithm 5 ConsolDispatch()

Input $L^*_{tot}(t)$, f_i^{max}, $iter^{max}$, f_i^{ON}, T_{ON}, Δ, k, α_{max}, $\mathscr{I}$, c, $\mathscr{S}(t)$, $Consolthr$
Output $\mathscr{A}^*$ for t-th slot
$l \leftarrow 1$, $\tilde{L}_i^{(0)}(\hat{t}) \leftarrow L_i^*(\hat{t}-1)$, $\mu^{(0)}(\hat{t}) \leftarrow \mu^*(\hat{t}-1)$
$\hat{t} \in \mathscr{S}(t)$, $\tilde{f}_i^{(0)}(\hat{t}) \leftarrow f_i^*(\hat{t}-1)$, $r^{(0)}(\hat{t}) \leftarrow r^*(\hat{t}-1)$
$\gamma^{(0)}(\hat{t}) \leftarrow \alpha_{max}$, $Ceff \leftarrow 0$
while $l \leq itr^{max}$ & $W(t) \notin [L_{th}, H_{th}]$ **do**
    $Ceff \leftarrow (k/l)$ //An coefficient to manage updating γ and f_i
    $Y^{(l-1)}(\hat{t}) \leftarrow \sum_{i=1}^{M} L_i^{(l-1)}(\hat{t}) - L^*_{tot}(\hat{t})$
    $\mu^{(l)}(\hat{t}) \leftarrow \left[\mu^{(l-1)}(\hat{t}) - \gamma^{(l-1)}(\hat{t}) Y^{(l-1)}(\hat{t})\right]_+$
    $f_i^{(l)}(\hat{t}) \leftarrow [f_i^{(l-1)}(\hat{t}) - Ceff \times ConsolCpuDyn^{(l-1)}(f_i^{(l-1)}, r^{(l-1)})]_0^{f_i^{max}}$
    $\tilde{L}_i^{(l)}(\hat{t}) \leftarrow \left[\tilde{L}_i^{(l-1)}(\hat{t})\text{-}ConsolLAN^{(l-1)}\left(\tilde{L}_i^{(l-1)}, \mu^{(l)}\right)\right]_0^{L_i^{max}}$
    $\bar{L}_i^{(l)}(\hat{t}) \leftarrow \left[\Delta f_i^{(l)}(\hat{t}) - \{T_{ON} f_i^{(l)}(\hat{t}) u_{-1}(f_i^{(l)}(\hat{t}))\}\right]_0^{L_i^{max}}$
    $L_i^{(l)}(\hat{t}) \leftarrow \left[L_i^{(l)}(\hat{t})\right]_{\tilde{L}_i^{(l)}(\hat{t})}^{\bar{L}_i^{(l)}(\hat{t})}$
    $r^{(l)}(\hat{t}) \leftarrow \left[\mu^{(l)}(\hat{t}) - ConsolLAN^{(l)}\left(\tilde{L}_i^{(l)}, \mu^{(l)}\right)\right]_+$
    $\gamma^{(l)}(\hat{t}) \leftarrow \left[\gamma^{(l-1)}(\hat{t}) - c \times Ceff \times Y^{(l-1)}(\hat{t})\right]_0^{\alpha_{max}}$
    $V^{(l)}(\hat{t}) \leftarrow \left(1 - \gamma^{(l-1)}(\hat{t})\right) V^{(l-1)}(\hat{t}) - Y^{(l-1)}(\hat{t})$
    Calculate $W(t)$ using (4.23)
    if $Y^{(l)}(\hat{t}) < Consolthr$ & $W(t) \in [L_{th}, H_{th}]$ **then**
        return $\mathscr{A}(t)$
        break
    end if
end while
return $\mathscr{A}(t)$

Xeon CPU and 4GB of RAM. All the emulated servers are connected through commodity Fast Ethernet NICs. In all carried out tests, we configure the VMs with 512MB of RAM and emulate the TCP New Reno protocol for implementing the needed VM-to-VM transport connections. The simulated parameters are listed in Table 4.2. In order to measure the actual performance of the proposed joint scheduler, we introduce the per-IU average total delay $\overline{T}^*_{tot}$ performance index (measured in multiple of the slot time), formally defined as in:

$$\overline{T}^*_{tot} \triangleq \frac{\lim\limits_{t\to\infty} \frac{1}{t}\left(\sum_{\tau=1}^{t} s^*(\tau)\right)}{\lim\limits_{t\to\infty} \frac{1}{t}\left(\sum_{\tau=1}^{t} \lambda^*(\tau)\right)} + \frac{\lim\limits_{t\to\infty} \frac{1}{t}\left(\sum_{\tau=1}^{t} q^*(\tau)\right)}{\lim\limits_{t\to\infty} \frac{1}{t}\left(\sum_{\tau=1}^{t} L^*_{tot}(\tau)\right)} + 2.(slot) \tag{4.40}$$

Table 4.2 Simulation setup.

Parameters		
$\Delta = 1\ (s)$	$f_i = \{0,5,50,70,90,f_i^{max}\}\ (Mbit/s)$	
$J_P = \{0.25,0.75\}$	$\lambda_{max} = 10^3$	$\alpha = 10^{-2}$
$N_I = N_O = 240\ (Mbyte)$	$I = 1, N_{to} = 100$	$T_t = 5\ (s)$
$k_e = \{0.05,0.005\}\ (J/(MHz)^2)$	$R_i^{max} = 1000\ (Mbit/s)$	$MSS = 120$
$f_i^{max} = \{10,105\}\ (Mbit/s)$	$E_c^{max} = \{40,60\}\ (Joule)$	$E_c^{idle} = 5\ (Joule)$
$r_{max} = 16\ (Mbyte/slot)$	$E_{ave} = \{0.05,0.1,0.15,0.2,0.3,0.4,0.5,0.6,0.75,1\}(Joule)$	
$r_{min} = \{0.01,3.43\}\ (Mbyte/slot)$	$T_s = 2\ (s)$	$h = 0.95$
$\theta = \{0.01,0.1,0.25,0.5,0.75,0.9,0.99\}$	$\Omega_i = 0.5\ (Watt)$	$T_{ON} = 10^{-3}(s)$
$f_i^{ON} = 10^{-6}\ (Mbit/s)$	$\overline{Job}$=8 $(Mbyte)$	$k = 1000$

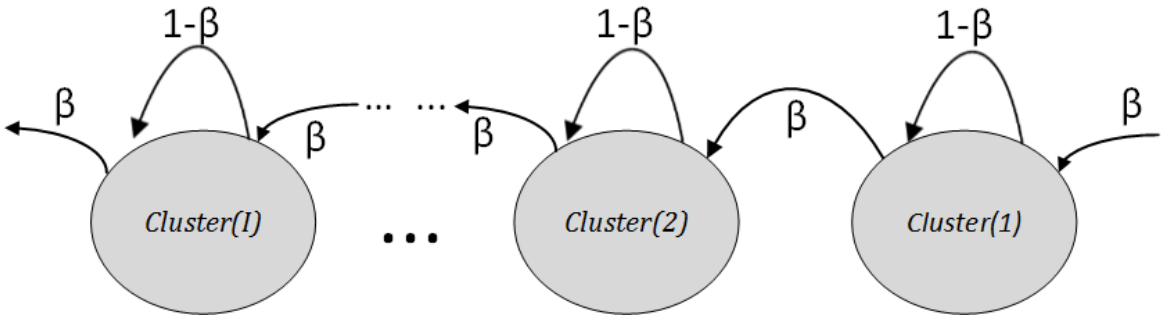


Fig. 4.4 The adopted Markovian random walk for the simulated client mobility.

In eq. (4.40), the first ratio is the average delay: $\overline{T}_{QI}^* = \overline{s}^*/\overline{\lambda}^*$(slot) induced by the input buffer of Fig.4.1, the second ratio is the measured average delay: $\overline{T}_{QO}^* = \overline{q}^*/\overline{L}_{tot}^*$(slot) of the output buffer, and: $\bar{T}_{SI} + \bar{T}_{SO} = 2$ are the overall corresponding average service times.

4.5.2 Simulated vehicular setup

Each vehicular client (VC) of Fig.4.1 may be served only by the RSU of the cluster over which is currently traveling (see Fig.4.1). The location of j-th vehicular client $VC(j)$ during slot t is described by the10following set of binary variables (see Fig.4.5):

$$m_{ji}(t) \triangleq \begin{cases} 1, & \text{if } VC(j) \text{ is in } cluster(i) \text{ during slot } t, \\ 0, & \text{otherwise,} \end{cases} \tag{4.41}$$

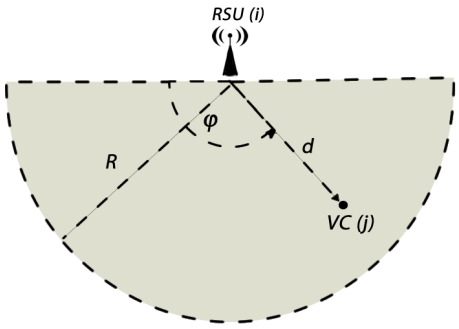


Fig. 4.5 Position of $VC(j)$ over Cluster i.

with $j = 1,\ldots,N_{to}, \quad i = 1,\ldots,I$. In eq. (4.41), N_{to} and I are the total number of VCs and the RSUs in the simulated vehicular scenario, so that the following constraint:

$$\sum_{i=1}^{I} m_{ji}(t) \leq 1, \qquad j = 1,\ldots,N_{to}, \tag{4.42}$$

accounts for the fact that, during each slot time t, $VC(j)$ may travel, at most, over a single cluster. In particular, the summation at the left-hand-side (l.h.s.) of (4.42) vanishes when the inter-RSU distance D is larger than $2R$ and, at slot t, $VC(j)$ is out of the RSU coverage (see Fig.4.1). The topology of the simulated vehicular backbone is of linear-type and the vehicular traffic flows parallel to the backbone path in an one-way direction. As in Section 3.2 of [67], we resort to the so called *Markovian random walk* with *random positioning* for simulating the VC mobility. According to this model, the inter-cluster mobility of each vehicular client is described by the transition among adjacent spatial clusters. It is modelled by the spatially homogeneous Markov chain of Fig.4.4, in which each state corresponds to a spatial cluster of radius R (m) centered at the serving RSU (see Fig.4.5). At the beginning of slot t, each VC autonomously either moves to the next cluster with spatial transition probability β, or stays in the current cluster with probability $(1-\beta)$. Afterwards, in order to compute the current spatial position over the selected cluster, at first, each VC randomly picks up its distance $d \in [0,R]$ from the serving RSU. Then, if a cluster transition is not occurred, the VC randomly selects its angular position φ over the (restricted) interval $(0,\varphi']$, where φ' is

its angular position in the previous slot time. Otherwise, if a cluster transition is happened, the VC randomly choices φ over the full interval $[0,\pi]$ (see Fig.4.5). So doing, we have that, in the steady-state, each cluster sustains the *same* average number $\bar{N}$ $(client/cluster)$ of vehicular clients. Furthermore, according to eq. (6) of [67], the aforementioned parameters β and $\bar{N}$ may be numerically evaluated as in (see Figs.4.4, 4.5):

$$\beta = \bar{v}/v_{max}, \quad \text{and,} \quad \bar{N} = A_{jam}\,(1-\beta)\,(R^2\pi/2), \tag{4.43}$$

where $\bar{v}$ (respectively, v_{max}) is the average (respectively, maximum) client speed, while A_{jam} $(vehicle/m^2)$ is the maximum spatial density of vehicles when the vehicular traffic of Fig.4.1 begins to halt (that is, at vanishing β). The reported numerical results are averaged over all the simulated active Cloud-to-RSU-to-VC connections.

4.5.3 Performance results

In the first test scenario, we run the proposed scheduler and evaluate $\overline{E}_{tot}$ under a synthetic workload with $\overline{Job} = 8$ $(Mbyte)$ for 2000 slots. Fig.4.6 reports the average total consumed energy for various values of M. Specifically, Fig.4.6 points out that: *i)* $\overline{E}_{tot}$ decreases for increasing M; *ii)* $\overline{E}_W$ and $\overline{E}_{tot}$ increase for increasing $\bar{r}$; and *iii)* $\bar{r}$, $\overline{E}_W$, and $\overline{E}_{tot}$ decrease for increasing θ. The second set of simulations in Fig.4.7 presents the total average consumed energies $\overline{E}_{tot}$ at $M = 2$, 5, 10, 15, 20 for various values of $\overline{E}_W$. Fig.4.7 points out that, by increasing the number of VMs, $\overline{E}_{tot}$ decreases. The interesting point is that, even at very low E_{ave}, the admission control attempts to decrease $\overline{E}_{tot}$. Fig.4.8 shows that, in the steady state, $\overline{L}_{tot}$ is the same as $\bar{r}$, $\overline{\lambda}$. $\bar{s}$ decreases. Furthermore, Fig.4.8 points out that changing J_P has no effect on $\bar{r}$ and $\overline{L}_{tot}$, so that, the system is capable to work, even under huge input workload.

4.5.3.1 Fraction of declined requests versus experienced delay

The plots of Figs.4.9 and 4.10 report the (numerically evaluated) average fraction: $\overline{F}^*_D = \lim_{t\to\infty}(1/t)(\sum_{\tau=1}^{t}(\lambda^*(\tau))/(Job(\tau)))$ of the offered input traffic that is served by the proposed

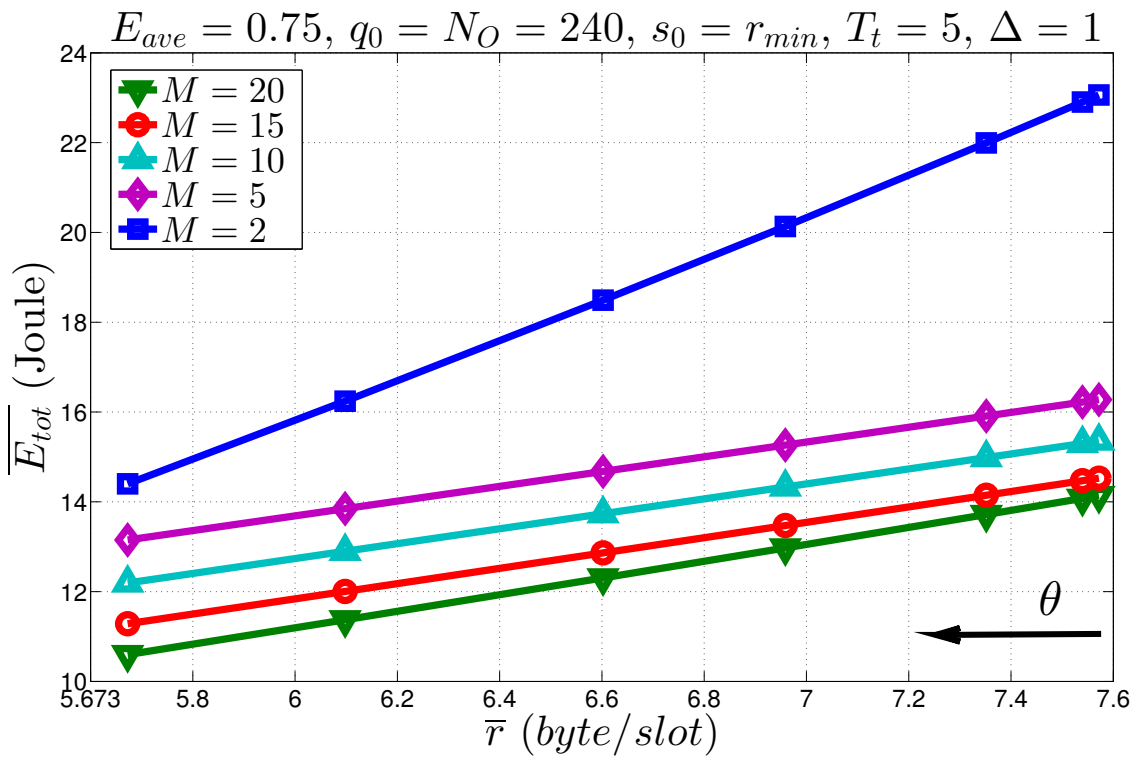


Fig. 4.6 $\overline{E}_{tot}$-vs.-$\bar{r}$ under synthetic input traffic.

scheduler and the corresponding average total delay: $\overline{T}^*_{tot}(slot)$ suffered by the admitted workload. These plots refer to the case of $k_e = 5(mJ/(MHz)^2)$, $\theta = 0.5$, $M = 10$, and $h = 0.95$. These plots allow us to gain insight about the tradeoff between the admission capability and the induced delay of the proposed scheduler. Specifically, three main conclusions may be drawn from the plots of Figs.4.9, 4.10. First, the average fraction of the rejected traffic decreases (i.e., served traffic increases) for increasing values of the storage capacity $N_I = N_O$ of the input/output buffers of Fig.4.1 (see Fig.4.9). Second, the fraction $\overline{F}^*_D$ quickly increases/decreases for increasing values of E_{ave} and/or σ_{min} (see Fig.4.10). Third, according to the Little' law, the corresponding average total delay suffered by the admitted traffic increases for increasing values of $N_I = N_O$ (see Fig.4.9). However, the increasing rate (quickly) decreases for increasing values of E_{ave} and/or σ_{min} (see Fig.4.10), especially for values of the buffers' capacity limited up to 50 $(Mbyte)$.

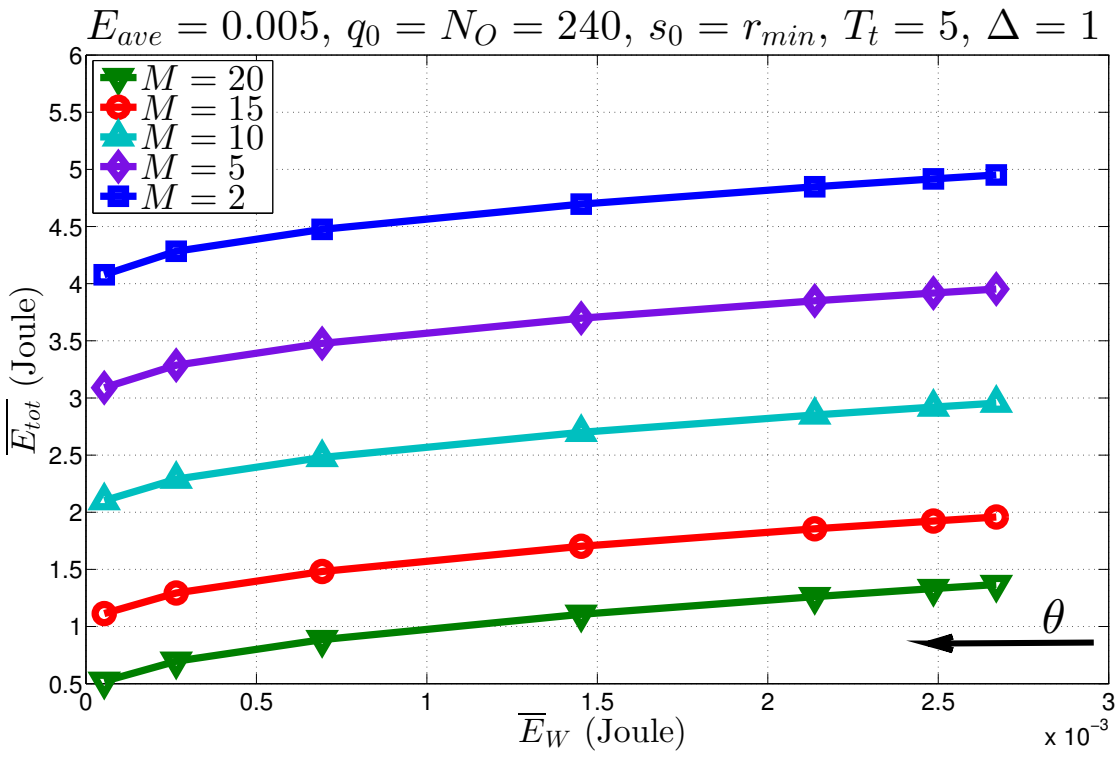
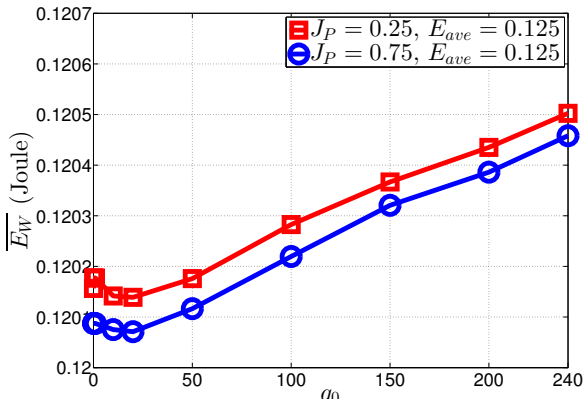


Fig. 4.7 $\overline{E}_{tot}$-vs.-$\overline{E}_W$ for the synthetic input traffic and various M at $E_{ave} = \{0.0625, 0.005\}$ $(Joule)$ with various θ.

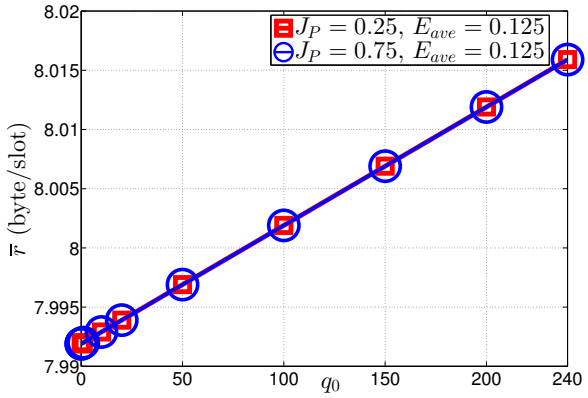
4.5.4 Mobility effects

Goal of this set of numerical tests is to acquire insight about the effects induced by the client mobility on the resulting average total delay $\overline{T}^*_{tot}$, when the target performance is assigned in terms of average goodput $\overline{r}^*$ and average total consumed energy $\overline{E}_{tot}$. For this purpose, the application scenario of Table 4.2 has been still considered at $M = 10$, $r_{min} = 3.43$, $k_e = 0.05$ $(J/(MHz)^2)$ and $\theta = 0.5$. The corresponding numerically evaluated performance is reported in Table 4.3. An examination of these results leads to three main conclusions.

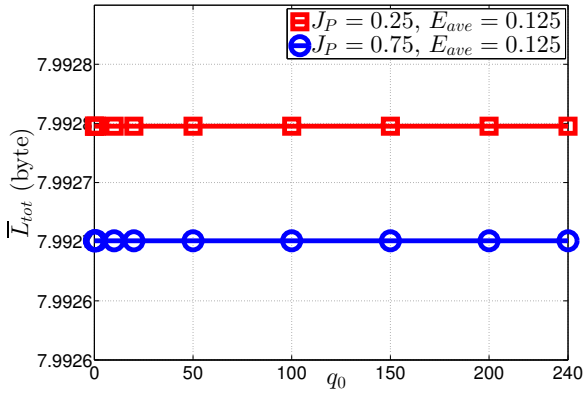
First, the minimum sizes $N_I = N_O$ of the input/output buffers required to meet the target $\overline{r}^*$=8.037 and $\overline{E}_{tot}$= 36.09 (quickly) increases for increasing values of h (e.g., for decreasing values of the client speed v). This is due to the fact that larger h's increase the coherence time of the sequence $\{\sigma(t)\}$ of the connection state (see eqs. (4.9), (4.13)), so that $\{\sigma(t)\}$ tends to be "less ergodic". Thus, in order to offset this effect without penalizing energy



(a) $[\overline{E}_W$



(b) $[\overline{r}$



(c) $[\overline{L}_{tot}$

Fig. 4.8 Admission control: q_0 denotes output buffer length in initial slot, with r_{min}=3.4323, $max(CRTT)$= 8.448 and Δ_{IP}^{max}= 10.

performance, the scheduler requires more buffering capacity, that, in turn, tends to *penalize* the resulting delay performance (see the last column of the first/second part in Table 4.3).

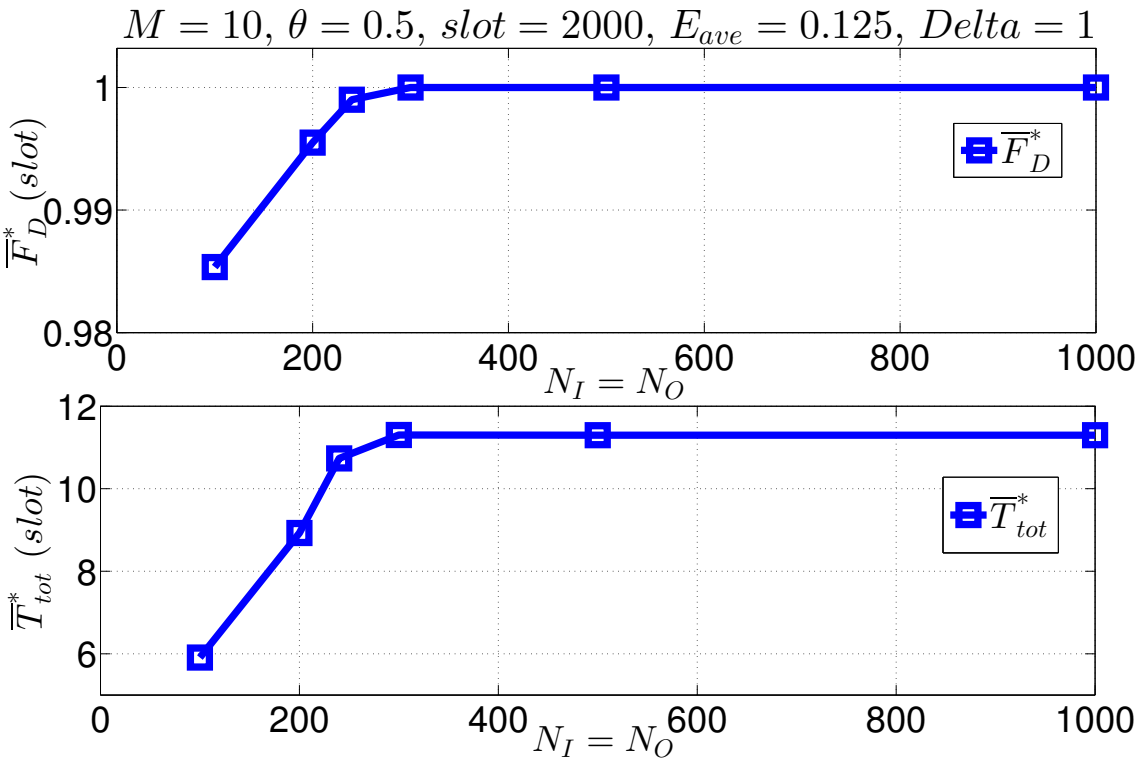


Fig. 4.9 $\overline{F}^*_D$ and $\overline{T}^*_{tot}(slot)$ versus the capacity $N_I = N_O$ $(Mbyte)$ of the input/output queues of Fig.4.1 for the application scenario of Section 4.5.3.1.

Second, at least in the simulated scenario, the requested buffers' size $N_I = N_O$ tends to scale as: $O(-1/\log(h))$ for $h \geq 0.87$. These considerations point out that, in practical, a right tradeoff between goodput and delay performance should depend on the value of h.

4.5.5 Performance tests and comparisons under real-world time-correlated input traffic

In order to test the consolidation capability of the proposed scheduler when the arrival process exhibits time-correlation, we have considered the real-world arrival trace of Fig.4.11.

Fig.4.11 reports the real-world trace of Fig.2 of [106] and refers to the I/O workload taken from four RAID volumes of an enterprise storage cluster in Microsoft (see Section IV.A of [106]). The numerical tests carried out in this sub-section refer to the communication-

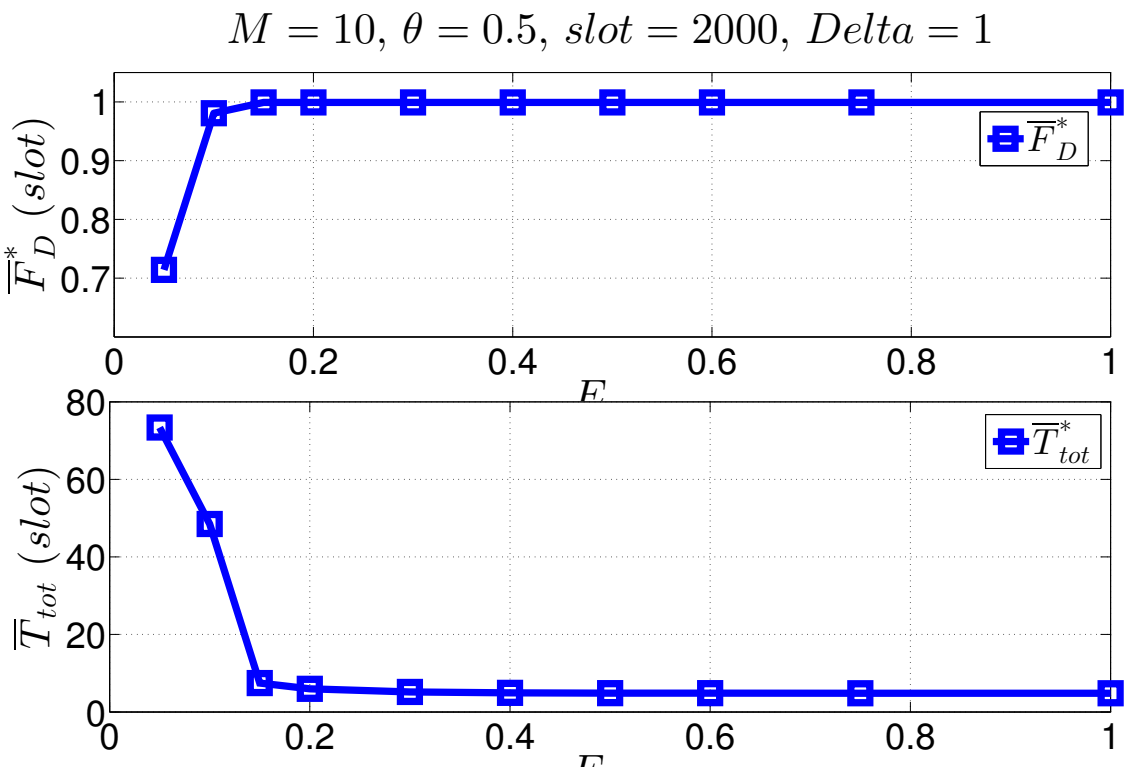


Fig. 4.10 $\overline{F}_D^*$ and $\overline{T}_{tot}^*(slot)$ (in multiple of the slot period) versus E_{ave} $(Joule)$ for the application scenario of Section 4.5.3.1.

computing infrastructure of Table 4.2 at $k_e = 0.5$ $(Joule/(MHz)^2)$ and $\Delta = 1.2$ (s). Furthermore, in order to maintain the peak workload fixed at 16 $(Mbit/slot)$, we assume that each arrival of Fig.4.11 carries out a traffic of 0.533 $(Mbit)$. Unless otherwise stated, the numerical results presented in the following subsections refer to the setting of Table 4.2 at $M = 10$, $h = 0.95$, $r_{min} = 0.01$, $r_{max} = 1600$ $(byte/slot)$ and $\theta = 0.5$.

The workload trace of Fig.4.11 is *smoother* than those previously considered in as synthesis workload in simulation results (i.e., synthesis workload) in the first scenario. Hence, we expect that the corresponding performance gaps of the proposed and sequential schedulers [69, 6] over the aforementioned tested related works one are somewhat less than those reported in the second set of simulations for the case of i.i.d. workload. However, we have tested that, even under the (strongly) real-world correlated workload trace of HTTP

Table 4.3 Buffers' size v.s.-delay tradeoff at target $\overline{r}^*$ and $\overline{E}_{tot}$ for various values of the mobility-depending correlation coefficient h in (4.13) for the synthetic and real-world input traffic traces [7]. The application scenario of Table 4.2 is considered. The reported values of h corresponded to client speeds v of 126, 30, and 2.8 (km/h), respectively.

Workloads	**h**	$\overline{r}^*(byte/slot)$	$\overline{E}_{tot}(Joule)$	(q_{min})	$\overline{\overline{T}}^*_{TOT}(slot)$
$\overline{Job}=8$, PMR=1.25	0.870	8.03	36.20	4	$\frac{10.20}{7.99}+\frac{18.3}{8.01}+2=4.28$
	0.967	8.037	36.16	70	$\frac{12.06}{7.99}+\frac{21.67}{8.01}+2=6.21$
	0.997	8.037	36.09	240	$\frac{10.13}{7.99}+\frac{32.37}{8.01}+2=7.30$
Real-workload [7]	0.870	8.9	42.36	100	$\frac{240}{8.9}+\frac{100}{8.9}+2=40.2$
	0.967	8.9	41.8	110	$\frac{240}{8.9}+\frac{115}{8.9}+2=41.9$
	0.997	8.9	40.2	150	$\frac{240}{8.8}+\frac{161}{8.8}+2=47.6$

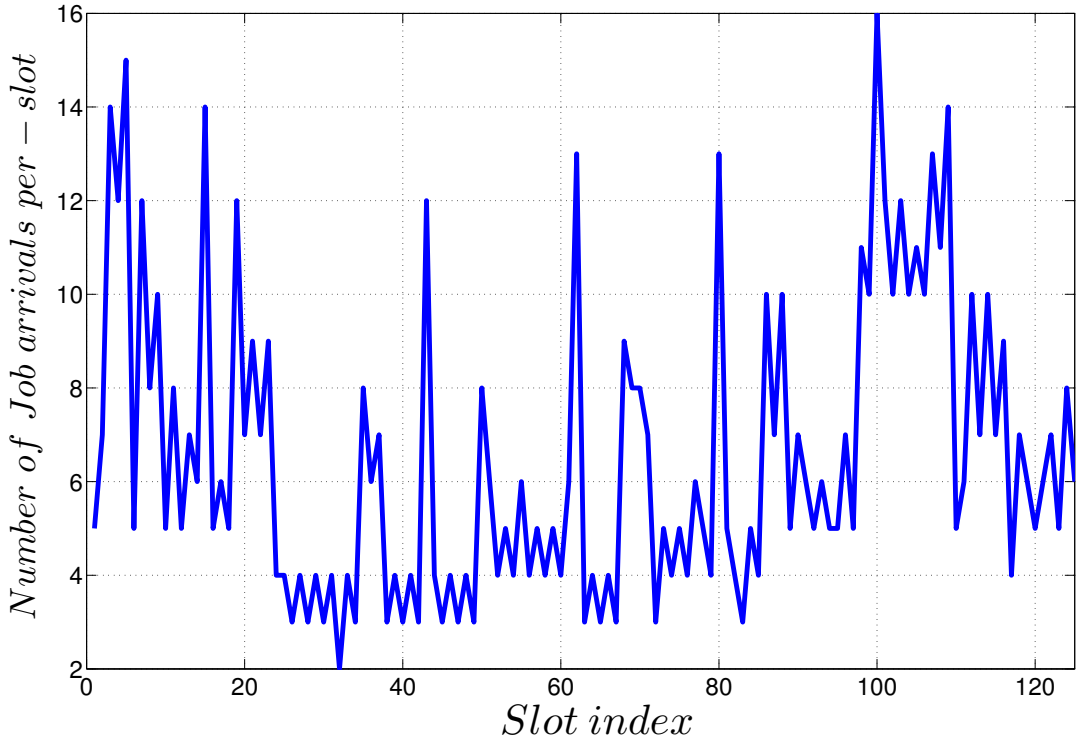


Fig. 4.11 Sampled trace of an I/O workload from an enterprise cluster in Microsoft [106]. The measured arrival rate is in multiple of the slot period and the reported trace covers more than 120 slot-periods.

sessions at Web servers of the 1998 Soccer world Cup [61], the average energy reductions of the proposed scheduler over the static and hybrid ones [69, 6] still approaches 40% and

19%, respectively. The corresponding energy saving of the sequential scheduler [6] over the proposed one remains limited up to $5\% - 5.5\%$.

Quite different performance gaps are expected under the workload trace of Fig.4.11. In fact, in this case, the (numerically evaluated) PMR is larger than those of the i.i.d. synthetic traces and equates $PMR = 2.49$, while the corresponding time-covariance coefficient is low. In agreement with the quasi-random behavior of the trace of Fig.4.11, we have tested that the corresponding energy reduction of the proposed scheduler over the static and hybrid ones [69, 6] approach 68% and 28% respectively.

4.5.5.1 Adaptive consolidation and convergence to the optimum

The numerical trial of this subsection aim at testing both the actual convergence to the optimum and the corresponding convergence speed of the proposed iterative consolidation algorithm. For this purpose, we have evaluated and compared the performance of the proposed iterative consolidation algorithm of Section 4.5 against the theoretically optimal one. By design, at each consolidation instant $\hat{t} \in \mathscr{S}(t)$, the latter computes the optimal set of the consolidated VMs by performing an exhaustive research over the set O of all VMs' configurations. The performance comparisons have been carried out by implementing three different event-driven policies, where consolidation at slot $\hat{t}$ is triggered when the average consolidation factor $W(\hat{t}-1)$ of the VMs that are active at $(\hat{t}-1)$ falls out of the intervals: *i)* $\mathscr{I}^{(1)} = [0.01, 0.99]$ (Case 1); *ii)* $\mathscr{I}^{(2)} = [0.2, 0.8]$ (Case 2); and, *iii)* $\mathscr{I}^{(3)} = [0.3, 0.7]$ (Case 3). The corresponding measured performance loss (in percent) suffered by the proposed iterative consolidation algorithm of Section 4.5 against the optimal one is reported in Table 4.4 for the cases of $k_e = 5(mJ/(MHz)^2)$ (e.g., case of low frequency-scaling cost) and $k_e = 500(mJ/(MHz)^2)$ (e.g., case of high frequency-scaling costs). An examination of the results of Table 4.4 leads to three main conclusions. First, the actual occurrence rate of the consolidation events increases by passing from Case 1 to Case 3, and this induces a reduction

Table 4.4 Average performance loss (in percent) of the proposed consolidation algorithm against the exhaustive optimal one. The application scenario of Table 4.2 is considered.

	Case 1	Case 2	Case 3
$k_e = 500\,(mJ/(MHz)^2)$	1.2%	0.8%	0.5%
$k_e = 5\,(mJ/(MHz)^2)$	0.87%	0.70%	0.3%

in the performance penalty suffered by the proposed consolidation algorithm, whose chances of convergence to the optimal consolidated configurations increase, indeed, for increasing rate of the occurrence of the consolidation events. Second, the performance penalty suffered by the iterative consolidation algorithm is nearly negligible and, at least in the carried out tests, it remains, indeed, limited up to 1%, even at high k_e's. Third, since the rate of the occurrence of VM underutilization increases for growing k_e's and too frequent occurrence of VM underutilization phenomena tends to increase the resource reconfiguration actions to be carried out at the consolidation instants, the performance penalty suffered by the iterative consolidation algorithm of Section 4.5 tends to somewhat increase for growing k_e's. At this regard, we have numerically experienced that, at least in the carried out tests, the iterative consolidation algorithm of Section 4.5 converges to the optimum at each consolidation instant $\hat{t}$. The plot of Fig.4.12 reports the corresponding (numerically evaluated) time-behavior of the number $W(t)$ of the active VMs for the Case 2 of Table 4.4 at $k_e = 5\ (mJ/(MHz)^2)$ and $M = 10$. A comparison of the plots of Fig.4.12 with the arrival trace of Fig.4.11 confirms that, after a delay of about $7-8$ slot periods, the iterative consolidation algorithm of section 4.5 is capable to track the abrupt changes exhibited by the exogenous workload by scaling up/down the number of active VMs.

4.5.6 Performance Comparison

This subsection compares the energy performance of the proposed adaptive scheduler against the corresponding ones of some state-of-the-art schedulers e.g., the DVFS-based scheduler of [83], the Lyapunov-based scheduler of [91], the static and hybrid schedulers of [69,

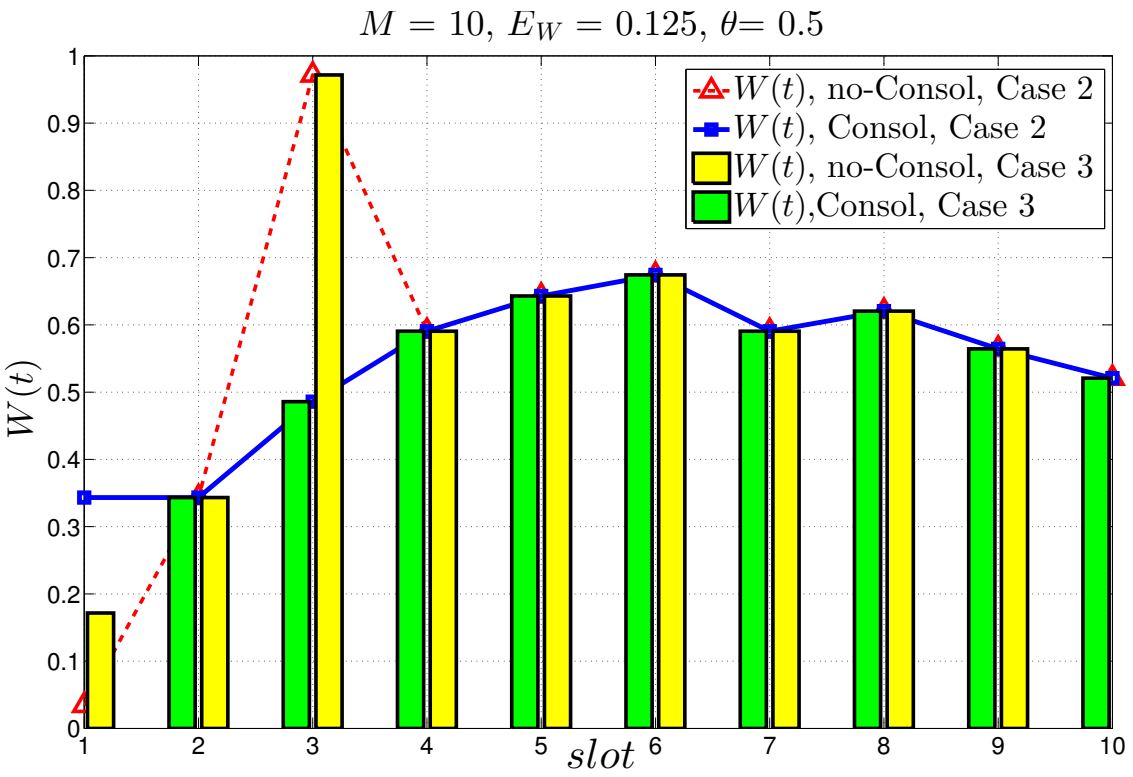


Fig. 4.12 Time-behavior of the numerical tested number $W(t)$ of the active VMs under the application scenario of Table 4.2 at $M = 10$ for the Cases 2 and 3 of Table 4.3. By design, all the available VMs are turned ON at $t = 0$.

6], and the NetDC scheduler of [23]. The goal of this last set of numerical tests is to (numerically) evaluate and compare the reductions in the average energy consumption: $\overline{E}_c(i,t) + \overline{E}_{LAN}(i,t)$ of the computing plus intra-cloud communication energy induced by the dynamic frequency scaling and VMs' consolidation operations performed by the proposed adaptive scheduler. This is motivated by the fact that current data centers usually rely on *static* resource provisioning, where, by design, a *fixed* number of VMs constantly run at the *maximum* processing rate f_i^{max}, in order to constantly provide the computing capacity needed to satisfy the peak input workload Job_{max}. Although the resulting static scheduler does not suffer by the reconfiguration costs arising from dynamic frequency scaling and consolidation, it induces overbooking of the computing resources [6, 69]. Hence, the average intra-cloud communication-plus-computing energy consumption: $\overline{E}^{(STS)} \triangleq \overline{E}^{(STS)}_{CPU} + \overline{E}^{(STS)}_{LAN}$ of the *STatic*

Scheduler (STS) provides a benchmark for numerically evaluating the actual energy savings that the proposed scheduler of Sections 4.5 and 4.1 attains by performing dynamic frequency scaling and VMs' consolidation.

In the application framework which considered in Figs.4.11 and 4.12, the energy: $E^{(STS)}(t) \triangleq E^{(STS)}_{CPU}(t) + E^{(STS)}_{LAN}(t)(Joule)$ consumed by the STC for processing the current input workload $Job(t)$ (in $(bit/slot)$) is computable in closed-form. Specifically, after resorting to the channel formula for calculating $E_{LAN}(i,t)(Joule)$, the overall $E^{(STS)}(t)$ equates

$$E^{(STS)}(t) = M_{STS}\left(E^{max}_{c} + \frac{M_p}{(M_{STS})^2}\right) + \left(Job^3_{max} - Job^3_{min}\right) + E_{ave}, (Joule) \tag{4.44}$$

where: $M_{STS} \triangleq round(Job_{max}/\Delta f^{max}_i)$ is the number of constantly running VMs that the STS must utilize for satisfying the peak/minimum input workload Job_{max}/Job_{min} and $M_p \triangleq (2\Omega_i \overline{RTT}_i/(T_t - \Delta))$. Hence, the corresponding average energy $E^{(STS)}$ consumed by the STS may be directly evaluated by performing the sample-average of (4.44) over the exogenous workload conveyed by the arrival trace of Fig.4.11. Afterwards, In order to carry out fair performance comparisons, in the performed tests of this subsection, we have implemented the proposed scheduler by directly setting: $Job(t) = \lambda(t) = r(t), t \geq 0$, together with $\theta = 1$ and $\sigma(t) = \infty, t \geq 0$. In so doing, the objective function in (4.15.1) to be minimized by the proposed scheduler of sections 4.5 and 4.1 reduces to the summation: $E_c(i,t) + E_{LAN}(i,t)$ of the corresponding computing-plus-communication energy consumed by the cloud infrastructure Fig.4.1. Hence, on the basis of the carried out numerical tests, we have experienced that: *i)* the average energy reduction of the proposed scheduler over the STS is of about 47% at $k_e = 500(mJ/(MHz)^2)$, while it increases to about 56% at $k_e = 5(mJ/(MHz)^2)$ (e.g., when the energy overhead induced by the dynamic frequency-scaling is low); and, *ii)* at least in the carried numerical tests, a fraction of about $25\% - 30\%$ of these energy savings is provided by VMs' consolidation operations. These results confirm that the proposed iterative

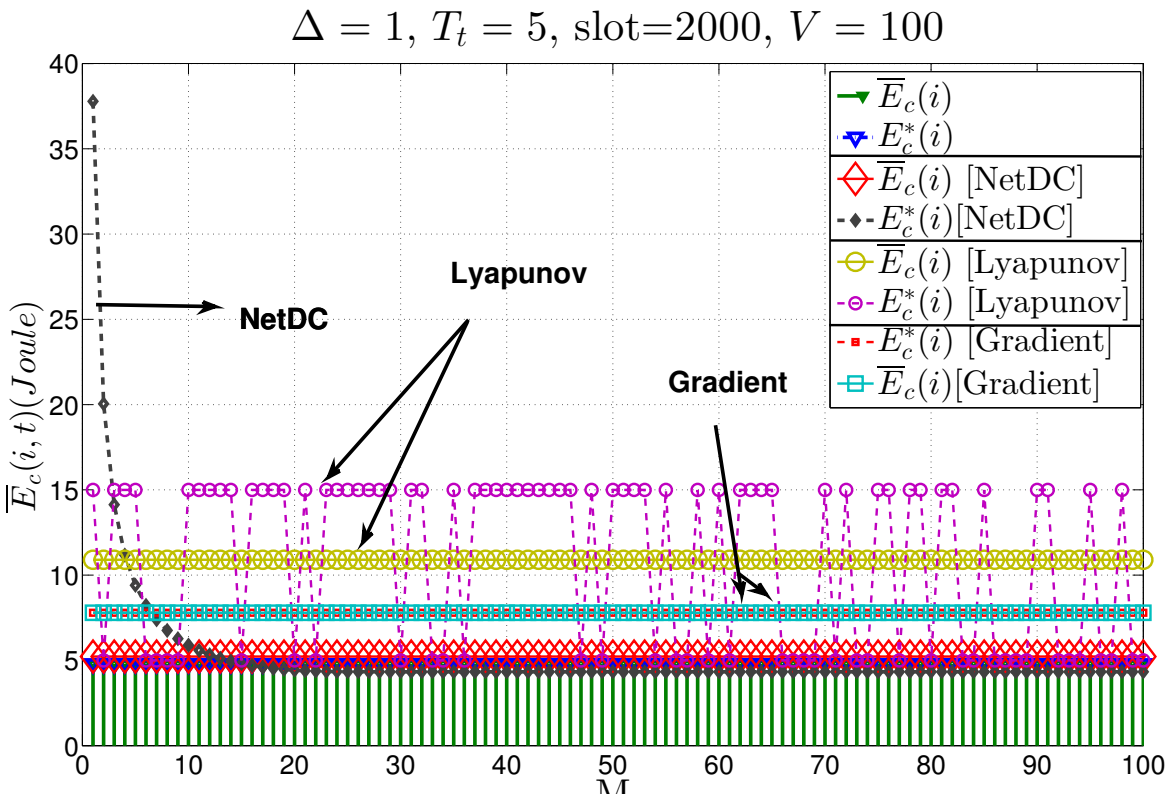


Fig. 4.13 $\overline{E}_c(i)$ $(Joule)$ for the proposed DVFS-equipped method-vs.-NetDC method in [23]-vs.- Lyapunov method in [91] and vs.- gradient-based iterative method in [83].

consolidation algorithm provides an effective means for leveraging the sudden time-variations exhibited by the input traffic of Fig.4.1.

In the last simulations, in order to evaluate the energy reduction due to scaling up/down of the computing and reconfiguration rates, we have also implemented three schedulers recently proposed in [23], [91], [83]. The resulting comparisons are presented in Figs.4.13 and 4.14. According to Fig.4.13, the average energy savings of the proposed scheduler (i.e., the green colored continue plot marked by $-\nabla-$) over the Lyapunov-based one in [91] (i.e., the two upper most plots marked by $-\circ-$ and labeled "Lyapunov"), the NetDC scheduler [23] (i.e., black dashed plots marked by $-\Diamond-$ and labeled by "NetDC") and the gradient-based iterative scheduler [83] (i.e., dashed red and cyan continue plot marked by $-\Box-$ and labeled by "Gradient") are about 60%, 10%, and 33%, respectively. This confirms that the proposed method is capable to adapt to the time-varying behaviors of the input traffic.

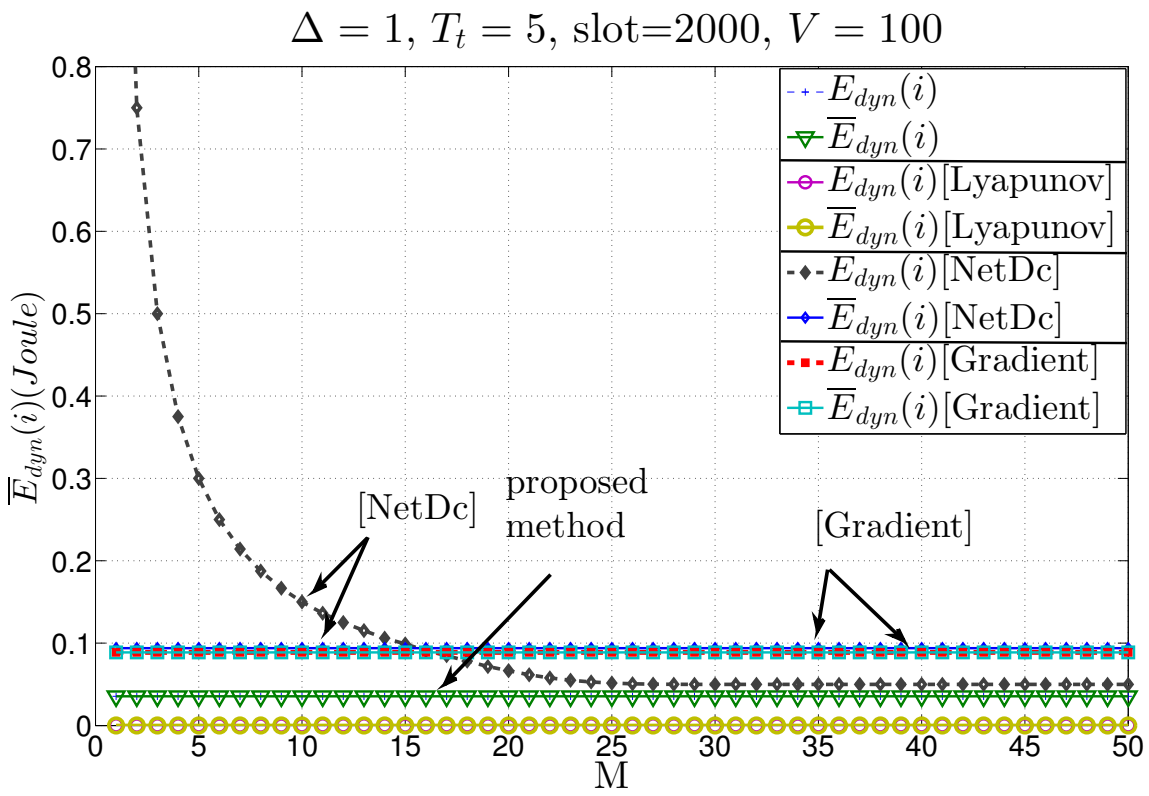


Fig. 4.14 $\overline{E}_{dyn}(i)$ $(Joule)$ for the proposed method (i.e., using DVFS)-vs.- NetDC method in [23]-vs.- Lyapunov method in [91] and vs.- gradient-based iterative scheduler [83].

Furthermore, Fig.4.14 shows that the gaps among the average reconfiguration costs of the proposed scheduler, the NetDC one in [23] and the gradient-based scheduler in [83] are negligible (i.e., less than 0.05). Moreover, by looking Figs.4.13 and 4.14, we can conclude that this gap is unable to fill the corresponding gap of the computing part. As a conclusion, [91] has much higher computing cost compared to the proposed scheduler, even under lower reconfiguration costs. Lastly, consider that [23, 91] are able to control the processed speed by using DVFS but does not optimize the energy wasted by the TCP/IP vehicular connection. Furthermore, [91] manages time in a average manner and it is unable to manage the online/instantaneous traffic fluctuations, which is handled by our scheduler.

Chapter 5

Conclusion and Hint for the Future Research

5.1 Conclusions

The rapid growth in demand for computational power driven by modern service applications combined with the shift to the Cloud computing model have led to the establishment of large-scale virtualized data centers. Such data centers consume enormous amounts of electrical energy resulting in high operating costs and carbon dioxide emissions. Dynamic consolidation of virtual machines (VMs) using online job scheduling, intelligence resource provisioning and switching idle nodes to the sleep mode allow Cloud providers to optimize resource usage and reduce energy consumption. This Dissertation is focused on introducing some state-of-the art methods to minimize the communication-plus-computing energy which is wasted by processing streams of Big Data under hard real-time constraints on the per-job computing-plus-communication delays. In this thesis, we studied and implemented two kinds of scheduling techniques for the virtual machines (VMs) management to preserve energy provisioning over networked data centers (NetDCs). The proposed schedulers operate at the Middleware layer of the underlying protocol stack and performs the adaptive joint allocation of the task sizes, computing rates, communication rates and powers of the underlying Virtualized Networked data centers. The carried out performance comparisons and sensitivity

tests highlight that the average energy savings provided by our implemented schedulers over the state-of-the-art static ones. Actual performance of the proposed controllers have been numerically tested under both synthetic and measured traces of the exogenous workload, by also considering various mobility conditions and settings of the networked computing farms.

The first contribution of this thesis is the design and implementation of the approach which is called *VNetDC* that is explored in the third chapter. We developed the optimal minimum-energy scheduler for the joint adaptive load balancing and provisioning of the computing-plus-communication resources. *VNetDC* platforms have been considered which operate under hard real-time constraints. The carried out numerical performance tests highlight that the average energy savings of the proposed scheduler over the static and hybrid ones may be larger than 60% and 25% respectively, even when the PMR of the offered workload is less than two. Interestingly, the corresponding average energy loss with respect to the corresponding sequential scheduler equipped with perfect knowledge of the future workload is typically limited up to $4\% - 6\%$ especially when the offered workload exhibits not negligible time-correlation. A main strength of the proposed scheduler (*VNetDC*) is its capability to adapt to the time-varying statistical features of the offered workload *without* requiring any a priori assumption and/or knowledge about the statistics of the processed data. A possible shortcoming is that the proposed scheduler does not perform forecast of the future workload, in order to better copy with the reconfiguration costs. Hence, improving the energy performance of the proposed scheduler by also performing reliable workload forecast is a first topic for future research. At this regard, we also observe that, when the focus shifts to non real-time Internet-assisted mobile applications, the current work may be extended along three main directions. First, being the *VNetDC* considered here constrained to work in real-time, it subsumes that a single job per slot is processed, in order to avoid *random* queue delays. However, under soft (e.g., average) latency constraints, the energy consumption of the *VNetDC* could be, in principle, reduced by allowing multiple jobs to be queued. Hence,

guaranteeing an optimized average delay-vs.-energy consumption tradeoff under soft latency constraints is, indeed, a first research topic. Second, due to the hard real-time constraints, in our framework, the size L_{tot} of the input job is measured at the beginning of the corresponding slot and, then, it remains constant over the overall slot duration. However, under soft latency constraints, intra-slot job arrivals may take place and, then, live migrations of VMs could provide an additional means for reducing the energy consumptions. The development of adaptive mechanisms for planning at run-time the minimum-energy live migrations of VMs is a second research topic of potential interest. Finally, emerging *mobile* applications require that jobs processed by data centers are timely delivered to the mobile clients through TCP/IP mobile connections. In this environment, the energy-efficient adaptive management of the delay-vs.-throughput tradeoff of the TCP/IP mobile connections becomes an additional topic for further research which is explored in the next chapter.

The second contribution of this thesis which is shown in the third chapter is the design and implementation of the new DVFS-based scheduler named *GreenNetDC*, for the joint adaptive load balancing and provisioning of the computing rates, communication rates and communication powers in energy-efficient networked data centers working under discrete DVFS. Although the resulting optimization problem is inherently nonconvex, we unveiled and exploited its loosely coupled structure for attaining the analytical characterization of the optimal solution. The carried out performance comparisons highlight that the average energy savings provided by our proposed scheduler over the state-of-the-art STAS may be larger than 66%, even when the PMR of the offered workload is limited to 2 and the number Q of different processing rates equipping each VM is limited to 4. Interestingly, the average energy saving of our scheduler with respect to *HybridNetDC* and *Lyapunov* is 90% and 50%, respectively. *GreenNetDC* is able to meet the hard-deadline constraints by having each server spend much more time in idle mode and thus, saving more energy. Finally,

GreenNetDC is able to meet more than 85% SLAs of the incoming workload, which is better than *NetDC*, *Lyapunov* and *Hybrid NetDC*.

The third and the final contribution of this thesis which is shown in the forth chapter is the design and implementation of a adaptive TCP/IP-based scheduler in the Networked Data centers. As it has been already shown in the literature survey in the second chapter that the VM consolidation problem is strictly NP-hard and it is important to propose an integrated and adaptive scheduler which balanced usage of join computing-communication resources inside the networked data centers. Decide fast and effective consolidation techniques are crucial for the cloud data centers, therefore, this chapter focuses on the milt-gradient based stochastic approximation method which is applied in the proposed scheduler for the joint adaptive tuning of the: *i)* admitted traffic; *ii)* delivered throughput; and, *iii)* resource reconfiguration and consolidation, of virtualized Cloud platforms linked to vehicular TCP/IP connections. The overall goal is the energy-saving support of QoS demanding computing-intensive delay-sensitive services, that utilize the vehicular TCP/IP connections for delivering remotely processed workload to vehicular clients. Remarkable features of the developed joint scheduler are that: *i)* its implementation is distributed and adaptive and the resulting complexity fully scales with the number of the available VMs; *ii)* it minimizes the energy consumed by the overall platform for computing, intra-Cloud communication and transmission over the vehicular TCP/IP connection; and, *iii)* despite the unpredictable time-varying nature of the underlying TCP/IP vehicular pipe, it is capable to provide hard QoS guarantees, in terms of minimum delivered instantaneous throughput, *maximum* instantaneous rate-jitter and *maximum* queuing-plus-computing delay. Actual performance of the proposed scheduler has been numerically tested under both synthetic and real-world traces for the input traffic under various mobility conditions and settings of the networked Cloud. The carried out tests highlight that the average computing savings provided by the proposed scheduler over

the *NetDC*, Lyapunov-based, and gradient-based ones are of the order of 60%, 10%, 33%, respectively.

5.2 Future Directions of the Research

This work can be extended in some directions of potential interest. First of all, closed networked multi-tier computing infrastructures may be considered for the support of delay-tolerant session-based services. Since in this application scenario, intra-slot traffic arrivals could be allowed, live migration of VMs could be also forecast for attaining additional energy savings. We would like to evaluate all the migration techniques on VMs with huge working set size in order to rise the effectiveness of the existing schedulers and come up with another effective migration technique to mitigate huge downtime as well as application performance degradation.

Enhancing Flexibility of the TCP/IP-based adaptive scheduler by enabling priority queues: modern applications could have complex data delivery requirement. Supporting priority queues could help these applications with their complex requirements. We are planning to provide support for built-in priority queues in TCP/IP-based adaptive scheduler. The upgaraded version of this scheduler will be able to support traffics with different priorities. That means the user can make sure that at each time on each server, the workload with the higher priority will be delivered to the VC earlier. Priority queue support opens up new usage directions for proposed adaptive scheduler. Many applications can benefit from this feature.

Apply New AI-based Prediction Techniques: Scientist are doing several effort in order to establish ongoing applications of inspired techniques for development of novel data center architecture. Support vector machines (SVMs) are one of the most popular algorithms in machine learning. SVMs often provide significantly better classification performance than other machine learning algorithms on reasonably sized data sets. Currently SVM is popular among the researchers in electricity distribution systems, and they have applications such

as power-quality classification, power transformer fault diagnosis, etc. However, not much work have been conducted on using SVMs for building power models. Besides, some new state-of-the-art AI techniques such as Deep learning [44] and NSGA-II [73] prove to produce exceptional results in classification tasks which indicates its promise in creating future high resolution energy models for data centers[15].

Distributed Monitoring the VNetDCs: Monitoring has proven to be essential for distributed systems. It is very important to understand how the resources of a distributed system are utilized by applications. Monitoring resources of a large scale distributed system is not trivial with a centralized traditional monitoring solution. Distributed buffers could play an essential part to provide a distributed solution for large scale system monitoring [101]. One of the future directions of this work is to design and implement a distributed monitoring system using *GreenNetDC*/VNetDC. Providing an efficient system, will be able to serve a distributed monitoring solution well.

Chapter 6

Accomplishments

The papers that have been published are an important metric to measure the progress and to highlight the accomplishments achieved so far. We then draw the conclusions achieved so far. The papers and documents that have been published based on our working progress are listed as follows:

Conference Papers:

(1) **Mohammad Shojafar**, Nicola Cordeschi, Danilo Amendola, Enzo Baccarelli, "Energy-saving adaptive computing and traffic engineering for real-time-service data centers", *Communication Workshop (ICCW), 2015 IEEE International Conference on*, London, UK, Page: 1800-1806, 2015.

(2) **Mohammad Shojafar**, Nicola Cordeschi, Jemal H. Abawajy, Enzo Baccarelli, "Adaptive Energy-Efficient QoS-Aware Scheduling Algorithm for TCP/IP Mobile Cloud", *Global Communication Workshop (GLOBECOM), 2015 IEEE International Conference on*, San Diego, USA, Pages: 1-6, 2015.

(3) **Mohammad Shojafar**, Aliasghar Rahmani Hosseinabadi, M. Kardgar, Shahab Shamshirband, "TETS: A Genetic-based Scheduler in Cloud Computing to Decrease Energy and Makespan", *Hybrid Intelligent Systems (HIS 2015), 15th International Conference on*, Seoul, South Korea, Pages:1-11, 2015.

(4) Saeed Javanmardi, **Mohammad Shojafar**, Danilo Amendola, Nicola Cordeschi, Hue Liu, Ajith Abraham, "Hybrid Job scheduling Algorithm for Cloud computing Environment", Volume 303, Pages: 43-52, Springer, *IBICA*, 2014.

(5) Nicola Cordeschi, Danilo Amendola, **Mohammad Shojafar**, Enzo Baccarelli, "Performance evaluation of primary-secondary reliable resource-management in vehicular networks", *The 25th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications(IEEE PIMRC 2014)*, Page: 959-964, 2014.

Journal Papers:

(6) **Mohammad Shojafar**, Nicola Cordeschi, Enzo Baccarelli,"Energy-efficient Adaptive Resource Management for Real-time Vehicular Cloud Services", *IEEE Transactions on Cloud Computing (TCC)*, 2016.

(7) **Mohammad Shojafar**, Saeed Javanmardi, Saeid Abolfazli, Nicola Cordeschi, "FUGE: A Joint Meta-heuristic Approach To Cloud Job Scheduling Algorithm Using Fuzzy Theory And A Genetic Method", Volume 18, Issue 2, Pages: 829-844, Springer, *Cluster Computing (CLUS)*, June 2015.

(8) Nicola Cordeschi,**Mohammad Shojafar**, Enzo Baccarelli, "Energy-saving self-configuring networked data centers", *Computer Networks*, ISSN: 1389-1286, Volume 57, Issue 17, Pages: 3479–3491, *Elsevier Science*, The Netherlands, 2013.

(9) Nicola Cordeschi, **Mohammad Shojafar**, Danilo Amendola, Enzo Baccarelli, "Energy-efficient adaptive networked datacenters for the QoS support of real-time applications", ISSN: 0920-8542, Volume 71, Issue 2, Pages: 448-478, Springer, *The Journal of Supercomputing (SUPE)*, 2014.

(10) Nicola Cordeschi, Danilo Amendola, **Mohammad Shojafar**, Enzo Baccarelli, "Distributed and Adaptive Resource Management in Cloud-assisted Cognitive Radio Vehicular Networks with Hard Reliability Guarantees", *Vehicular Communications*, Volume 2, Issue 1, Page: 1–12, *Elsevier Science*, The Netherlands, 2015.

Book Chapters:

(11) **Mohammad Shojafar**, Nicola Cordeschi, Enzo Baccarelli, ”Resource Scheduling for Saving Energy in Reconfigurable Internet Data Centers”, *Handbook of Research on Next-Generation High Performance Computing*, IGI Global, to be appear 2016.

(12) Nicola Cordeschi, **Mohammad Shojafar**, Danilo Amendola, Enzo Baccarelli, ”Energy-Saving QoS Resource Management of Virtualized Networked Data Centers for Big Data Stream Computing”, Handbook in *Emerging Research in Cloud Distributed Computing Systems*, IGI Global, Pages: 122-155, 2015.

Magazine:

(13) Enzo Baccarelli, Nicola Cordeschi, Alessandro Mei, Massimo Panella, **Mohammad Shojafar**, Julinda Stefa, ”Energy-efficient Dynamic Traffic Offloading and Reconfiguration of Networked Datacenters for Big Data Stream Mobile Computing: Review, Challenges, and a Case Study”, *IEEE Network Magazine*, Vol. 30, Iss. 2, pp. 54-61, March-April 2016.

References

[1] Al-Fares, M., Loukissas, A., and Vahdat, A. (2008). A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review*, 38(4):63–74.

[2] Alcaraz, J., Vales-Alonso, J., and Garcia-Haro, J. (2009). Control-based scheduling with QoS support for vehicle to infrastructure communications. *Wireless Communications, IEEE*, 16(6):32–39.

[3] Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and Sridharan, M. (2011). Data center tcp (dctcp). *ACM SIGCOMM computer communication review*, 41(4):63–74.

[4] Andrae, A. and Corcoran, P. M. (2013). Emerging trends in electricity consumption for consumer ict.

[5] Azodolmolky, S., Wieder, P., and Yahyapour, R. (2013). Cloud computing networking: challenges and opportunities for innovations. *Communications Magazine, IEEE*, 51(7):54–62.

[6] Baliga, J., Ayre, R. W., Hinton, K., and Tucker, R. S. (2011). Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE*, 99(1):149–167.

[7] Ballani, H., Costa, P., Karagiannis, T., and Rowstron, A. (2011). Towards predictable datacenter networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 242–253. ACM.

[8] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. (2013). *Nonlinear programming: theory and algorithms*. John Wiley & Sons.

[9] Beloglazov, A., Buyya, R., Lee, Y. C., Zomaya, A., et al. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in computers*, 82(2):47–111.

[10] Bilal, K., Malik, S. U. R., Khan, S. U., and Zomaya, A. Y. (2014). Trends and challenges in cloud datacenters. *IEEE Cloud Computing*, (1):10–20.

[11] Brown, D. J. and Reams, C. (2010). Toward energy-efficient computing. *Communications of the ACM*, 53(3):50–58.

[12] Buyya, R., Beloglazov, A., and Abawajy, J. (2010). Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*.

[13] Buyya, R., Vecchiola, C., and Selvi, S. T. (2013). *Mastering cloud computing: foundations and applications programming*. Newnes.

[14] Chiang, M., Low, S. H., Doyle, J. C., et al. (2007). Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312.

[15] Chilimbi, T., Suzue, Y., Apacible, J., and Kalyanaraman, K. (2014). Project adam: Building an efficient and scalable deep learning training system. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 571–582.

[16] Cordeschi, N., Amendola, D., and Baccarelli, E. (2014a). Reliable adaptive resource management for cognitive cloud vehicular networks. *Vehicular Technology, IEEE Transactions on*, 64(6):2528 –2537.

[17] Cordeschi, N., Amendola, D., De Rango, F., and Baccarelli, E. (2014b). Networking-computing resource allocation for hard real-time green cloud applications. In *WD, 2014 IFIP*, pages 1–4. IEEE.

[18] Cordeschi, N., Amendola, D., Shojafar, M., and Baccarelli, E. (2014c). Performance evaluation of primary-secondary reliable resource-management in vehicular networks. In *Personal, Indoor, and Mobile Radio Communication (PIMRC), 2014 IEEE 25th Annual International Symposium on*, pages 959–964. IEEE.

[19] Cordeschi, N., Amendola, D., Shojafar, M., and Baccarelli, E. (2015a). Distributed and adaptive resource management in cloud-assisted cognitive radio vehicular networks with hard reliability guarantees. *Vehicular Communications*, 2(1):1–12.

[20] Cordeschi, N., Patriarca, T., and Baccarelli, E. (2012). Stochastic traffic engineering for real-time applications over wireless networks. *Journal of Network and Computer Applications*, 35(2):681–694.

[21] Cordeschi, N., Shojafar, M., Amendola, D., and Baccarelli, E. (2014d). Energy-efficient adaptive networked datacenters for the qos support of real-time applications. *The Journal of Supercomputing*, pages 1–31.

[22] Cordeschi, N., Shojafar, M., Amendola, D., and Baccarelli, E. (2015b). Energy-saving qos resource management of virtualized networked data centers for big data stream computing. *Emerging Research in Cloud Distributed Computing Systems*, page 122.

[23] Cordeschi, N., Shojafar, M., and Baccarelli, E. (2013). Energy-saving self-configuring networked data centers. *Computer Networks*, 57(17):3479–3491.

[24] Cugola, G. and Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3):15.

[25] Dabbagh, M., Hamdaoui, B., Guizani, M., and Rayes, A. (2014). Energy-efficient cloud resource management. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 386–391. IEEE.

[26] Dabbagh, M., Hamdaoui, B., Guizani, M., and Rayes, A. (2015). Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment. *Network, IEEE*, 29(2):56–61.

[27] Das, T. and Sivalingam, K. M. (2013). Tcp improvements for data center networks. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pages 1–10. IEEE.

[28] Enokido, T. and Takizawa, M. (2012). An extended power consumption model for distributed applications. In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 912–919. IEEE.

[29] Eriksson, J., Balakrishnan, H., and Madden, S. (2008). Cabernet: vehicular content delivery using wifi. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 199–210. ACM.

[30] Faruque, S. (2008). Traffic engineering for multi rate wireless data. In *Electro/Information Technology, 2008. EIT 2008. IEEE International Conference on*, pages 280–283. IEEE.

[31] Festag, A., Noecker, G., Strassberger, M., Lübke, A., Bochow, B., Torrent-Moreno, M., Schnaufer, S., Eigner, R., Catrinescu, C., and Kunisch, J. (2008). NoW–Network on Wheels: Project objectives, technology and achievements. *Proceedings of 5rd International Workshop on Intelligent Transportation (WIT*, pages 211–216.

[32] Gao, Y., Guan, H., Qi, Z., Wang, B., and Liu, L. (2013). Quality of service aware power management for virtualized data centers. *Journal of Systems Architecture*, 59(4):245–259.

[33] Ge, R., Feng, X., and Cameron, K. W. (2005). Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 34. IEEE Computer Society.

[34] Glisic, S. and Lorenzo, B. (2009). *Advanced wireless networks: cognitive, cooperative & opportunistic 4G technology*. John Wiley & Sons.

[35] Grant, M., Boyd, S., and Ye, Y. (2008). Cvx: Matlab software for disciplined convex programming.

[36] Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. (2009). Vl2: a scalable and flexible data center network. In *ACM SIGCOMM computer communication review*, volume 39, pages 51–62. ACM.

[37] Gudmundson, M. (1991). Correlation model for shadow fading in mobile radio systems. *Electronics letters*, 27(23):2145–2146.

[38] Guérout, T., Monteil, T., Da Costa, G., Calheiros, R. N., Buyya, R., and Alexandru, M. (2013). Energy-aware simulation with dvfs. *Simulation Modelling Practice and Theory*, 39:76–91.

[39] Gulati, A., Merchant, A., and Varman, P. J. (2010). mclock: handling throughput variability for hypervisor io scheduling. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, pages 1–7. USENIX Association.

[40] Gunaratne, C., Christensen, K., Nordman, B., and Suen, S. (2008). Reducing the energy consumption of ethernet with adaptive link rate (alr). *Computers, IEEE Transactions on*, 57(4):448–461.

[41] Guo, C., Lu, G., Wang, H. J., Yang, S., Kong, C., Sun, P., Wu, W., and Zhang, Y. (2010). Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International COnference*, page 15. ACM.

[42] Gupta, V., Nathuji, R., and Schwan, K. (2011). An analysis of power reduction in datacenters using heterogeneous chip multiprocessors. *ACM SIGMETRICS Performance Evaluation Review*, 39(3):87–91.

[43] Hansmann, U. (2003). *Pervasive computing: The mobile world*. Springer Science & Business Media.

[44] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[45] Hirzel, M., Soulé, R., Schneider, S., Gedik, B., and Grimm, R. (2014). A catalog of stream processing optimizations. *ACM Computing Surveys (CSUR)*, 46(4):46.

[46] Hotta, Y., Sato, M., Kimura, H., Matsuoka, S., Boku, T., and Takahashi, D. (2006). Profile-based optimization of power performance by using dynamic voltage scaling on a pc cluster. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8–pp. IEEE.

[47] Javanmardi, S., Shojafar, M., Shariatmadari, S., and Ahrabi, S. S. (2014). Fr trust: a fuzzy reputation–based model for trust management in semantic p2p grids. *International Journal of Grid and Utility Computing*, 6(1):57–66.

[48] Jing, S.-Y., Ali, S., She, K., and Zhong, Y. (2013). State-of-the-art research study for green cloud computing. *The Journal of Supercomputing*, 65(1):445–468.

[49] Karagiannis, G., Altintas, O., Ekici, E., Heijenk, G., Jarupan, B., Lin, K., and Weil, T. (2011). Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *Communications Surveys & Tutorials, IEEE*, 13(4):584–616.

[50] Keiser, G. (1989). *Local area networks*. McGraw-Hill New York.

[51] Kenway, S., Priestley, A., Cook, S., Seo, S., Inman, M., Gregory, A., and Hall, M. (2008). Energy use in the provision and consumption of urban water in australia and new zealand. *Water Services Association of Australia (WSAA): Sydney, Australia*.

[52] Kilper, D. C., Atkinson, G., Korotky, S. K., Goyal, S., Vetter, P., Suvakovic, D., and Blume, O. (2011). Power trends in communication networks. *IEEE Journal of Selected Topics in Quantum Electronics*, 2(17):275–284.

[53] Kim, K. H., Beloglazov, A., and Buyya, R. (2009). Power-aware provisioning of cloud resources for real-time services. In *Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*, page 1. ACM.

[54] Kimura, H., Sato, M., Hotta, Y., Boku, T., and Takahashi, D. (2006). Emprical study on reducing energy of parallel programs using slack reclamation by dvfs in a power-scalable high performance cluster. In *CLUSTER'06*, pages 1–10. IEEE.

[55] Koller, R., Verma, A., and Neogi, A. (2010). Wattapp: an application aware power meter for shared data centers. In *Proceedings of the 7th international conference on Autonomic computing*, pages 31–40. ACM.

[56] Koomey, J. (2011). Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*, page 9.

[57] Krug, L., Shackleton, M., and Saffre, F. (2014). Understanding the environmental costs of fixed line networking. In *Proceedings of the 5th international conference on Future energy systems*, pages 87–95. ACM.

[58] Kumbhare, A. G., Simmhan, Y., and Prasanna, V. K. (2014). Plasticc: Predictive look-ahead scheduling for continuous dataflows on clouds. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, pages 344–353. IEEE.

[59] Kurose, J. F. (2005). *Computer Networking: A Top-Down Approach Featuring the Internet, 3/E*. Pearson Education India.

[60] Kushner, H. J. and Yang, J. (1994). Analysis of adaptive step size sa algorithms for parameter tracking. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 1, pages 730–737. IEEE.

[61] Li, K. (2008). Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed. *Parallel and Distributed Systems, IEEE Transactions on*, 19(11):1484–1497.

[62] Lin, M., Wierman, A., Andrew, L. L., and Thereska, E. (2013a). Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking (TON)*, 21(5):1378–1391.

[63] Lin, M., Wierman, A., Andrew, L. L., and Thereska, E. (2013b). Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking (TON)*, 21(5):1378–1391.

[64] Liu, H., Jin, H., Xu, C.-Z., and Liao, X. (2013). Performance and energy modeling for live migration of virtual machines. *Cluster computing*, 16(2):249–264.

[65] Liu, Q., Zhou, S., and Giannakis, G. B. (2004). Cross-layer combining of adaptive modulation and coding with truncated arq over wireless links. *Wireless Communications, IEEE Transactions on*, 3(5):1746–1755.

[66] Lu, T., Chen, M., and Andrew, L. L. (2013). Simple and effective dynamic provisioning for power-proportional data centers. *Parallel and Distributed Systems, IEEE Transactions on*, 24(6):1161–1171.

[67] Luan, T. H., Ling, X., and Shen, X. S. (2012). Provisioning QoS controlled media access in vehicular to infrastructure communications. *Ad Hoc Networks*, 10(2):231–242.

[68] Mathew, V., Sitaraman, R. K., and Shenoy, P. (2012). Energy-aware load balancing in content delivery networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 954–962. IEEE.

[69] Mishra, A., Jain, R., and Durresi, A. (2012). Cloud computing: networking and communication challenges. *Communications Magazine, IEEE*, 50(9):24–25.

[70] Mitra, D. and Wang, Q. (2005). Stochastic traffic engineering for demand uncertainty and risk-aware network revenue management. *IEEE/ACM Transactions on Networking (TON)*, 13(2):221–233.

[71] Nathuji, R. and Schwan, K. (2007). Virtualpower: coordinated power management in virtualized enterprise systems. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 265–278. ACM.

[72] Neely, M. J., Modiano, E., and Rohrs, C. E. (2003). Power allocation and routing in multibeam satellites with time-varying channels. *IEEE/ACM Transactions on Networking (TON)*, 11(1):138–152.

[73] Nesmachnow, S., Perfumo, C., and GoirI, I. (2015). Holistic multiobjective planning of datacenters powered by renewable energy. *Cluster Computing*, pages 1–19.

[74] Papagianni, C., Leivadeas, A., Papavassiliou, S., Maglaris, V., Cervello-Pastor, C., and Monje, A. (2013). On the optimal allocation of virtual resources in cloud computing networks. *Computers, IEEE Transactions on*, 62(6):1060–1071.

[75] Pooranian, Z., Shojafar, M., Abawajy, J. H., and Abraham, A. (2015). An efficient meta-heuristic algorithm for grid computing. *Journal of Combinatorial Optimization*, 30(3):413–434.

[76] Pooranian, Z., Shojafar, M., Abawajy, J. H., and Singhal, M. (2013). Gloa: a new job scheduling algorithm for grid computing. *IJIMAI*, 2(1):59–64.

[77] Portnoy, M. (2012). *Virtualization essentials*, volume 19. John Wiley & Sons.

[78] Qian, Z., He, Y., Su, C., Wu, Z., Zhu, H., Zhang, T., Zhou, L., Yu, Y., and Zhang, Z. (2013). Timestream: Reliable stream computation in the cloud. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 1–14. ACM.

[79] Rajaraman, A., Ullman, J. D., Ullman, J. D., and Ullman, J. D. (2012). *Mining of massive datasets*, volume 77. Cambridge University Press Cambridge.

[80] Rizvandi, N. B., Taheri, J., Zomaya, A. Y., and Lee, Y. C. (2010). Linear combinations of dvfs-enabled processor frequencies to modify the energy-aware scheduling algorithms. In *CCGRID*, pages 388–397. IEEE.

[81] Shojafar, M., Canali, C., Lancellotti, R., and Abolfazli, S. (March 2016a). An energy-aware scheduling algorithm in dvfs-enabled networked data centers. In *The 6th International Conference on Cloud Computing and Services Science (CLOSER 2016)*, volume 2, pages 387–397.

[82] Shojafar, M., Cordeschi, N., Abawajy, J. H., and Baccarelli, E. (2015a). Adaptive energy-efficient qos-aware scheduling algorithm for tcp/ip mobile cloud. In *Global Communications (GLOBECOM), 2015 IEEE International Conference on*, pages 1–6. IEEE.

[83] Shojafar, M., Cordeschi, N., Amendola, D., and Baccarelli, E. (2015b). Energy-saving adaptive computing and traffic engineering for real-time-service data centers. In *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pages 1800–1806. IEEE.

[84] Shojafar, M., Cordeschi, N., and Baccarelli, E. (2016b). Energy-efficient adaptive resource management for real-time vehicular cloud services. *IEEE Transactions on Cloud Computing (TCC)*, PP(99):1–14.

[85] Shojafar, M., Cordeschi, N., and Baccarelli, E. (2016c). Resource scheduling for saving energy in reconfigurable internet data centers. *Research on Next-Generation High Performance Computing*, page pp.

[86] Shojafar, M., Cordeschi, N., Singhal, M., and Baccarelli, E. (2016d). Adaptive network-aware minimum-energyscheduling for time-constrained multimedia processing in cloud systems. *IEEE Transactions on Cloud Computing (TCC)*, PP(99):1–14.

[87] Sinnen, O. (2007). *Task scheduling for parallel systems*, volume 60. John Wiley & Sons.

[88] Tamm, O., Hermsmeyer, C., and Rush, A. M. (2010). Eco-sustainable system and network architectures for future transport networks. *Bell Labs Technical Journal*, 14(4):311–327.

[89] Tsitsiklis, J. N., Bertsekas, D. P., Athans, M., et al. (1986). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812.

[90] Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., and Tantawi, A. (2007). Analytic modeling of multitier internet applications. *ACM Transactions on the Web (TWEB)*, 1(1):2.

[91] Urgaonkar, R., Kozat, U. C., Igarashi, K., and Neely, M. J. (2010). Dynamic resource allocation and power management in virtualized data centers. In *NOMS'10*, pages 479–486. IEEE.

[92] Vasudevan, V., Phanishayee, A., Shah, H., Krevat, E., Andersen, D. G., Ganger, G. R., Gibson, G. A., and Mueller, B. (2009). Safe and effective fine-grained tcp retransmissions for datacenter communication. In *ACM SIGCOMM computer communication review*, volume 39, pages 303–314. ACM.

[93] Von Laszewski, G., Wang, L., Younge, A. J., and He, X. (2009). Power-aware scheduling of virtual machines in dvfs-enabled clusters. In *CLUSTER'09*, pages 1–10. IEEE.

[94] Wang, D., Ren, C., Govindan, S., Sivasubramaniam, A., Urgaonkar, B., Kansal, A., and Vaid, K. (2013a). Ace: Abstracting, characterizing and exploiting datacenter power demands. In *Workload Characterization (IISWC), 2013 IEEE International Symposium on*, pages 44–55. IEEE.

[95] Wang, L., Zhang, F., Arjona Aroca, J., Vasilakos, A. V., Zheng, K., Hou, C., Li, D., and Liu, Z. (2014). Greendcn: A general framework for achieving energy efficiency in data center networks. *Selected Areas in Communications, IEEE Journal on*, 32(1):4–15.

[96] Wang, L., Zhang, F., Hou, C., Arjona Aroca, J., and Liu, Z. (2013b). Incorporating rate adaptation into green networking for future data centers. In *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*, pages 106–109. IEEE.

[97] Warneke, D. and Kao, O. (2011). Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. *Parallel and Distributed Systems, IEEE Transactions on*, 22(6):985–997.

[98] Weiser, M., Welch, B., Demers, A., and Shenker, S. (1996). Scheduling for reduced cpu energy. In *Mobile Computing*, pages 449–471. Springer.

[99] Whaiduzzaman, M., Sookhak, M., Gani, A., and Buyya, R. (2014). A survey on vehicular cloud computing. *Journal of Network and Computer Applications*, 40:325–344.

[100] Whitehead, B., Andrews, D., Shah, A., and Maidment, G. (2014). Assessing the environmental impact of data centres part 1: background, energy use and metrics. *Building and Environment*, 82:151–159.

[101] Wu, C. and Buyya, R. (2015). *Cloud Data Centers and Cost Modeling: A Complete Guide To Planning, Designing and Building a Cloud Data Center*. Morgan Kaufmann.

[102] Xia, L., Cui, Z., Lange, J. R., Tang, Y., Dinda, P. A., and Bridges, P. G. (2012). Vnet/p: Bridging the cloud and high performance computing through fast overlay networking. In *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*, pages 259–270. ACM.

[103] Xu, H. and Li, B. (2014). Reducing electricity demand charge for data centers with partial execution. In *Proceedings of the 5th international conference on Future energy systems*, pages 51–61. ACM.

[104] Zaharia, M., Das, T., Li, H., Shenker, S., and Stoica, I. (2012). Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. In *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing*, pages 10–10. USENIX Association.

[105] Zhou, Z., Liu, F., Jin, H., Li, B., Li, B., and Jiang, H. (2013a). On arbitrating the power-performance tradeoff in saas clouds. In *INFOCOM, 2013 Proceedings IEEE*, pages 872–880. IEEE.

[106] Zhou, Z., Liu, F., Xu, Y., Zou, R., Xu, H., Lui, J., and Jin, H. (2013b). Carbon-aware load balancing for geo-distributed cloud services. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*, pages 232–241. IEEE.

[107] Zhu, D., Melhem, R., and Childers, B. R. (2003). Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. *Parallel and Distributed Systems, IEEE Transactions on*, 14(7):686–700.

[108] Zhuo, J. and Chakrabarti, C. (2008). Energy-efficient dynamic task scheduling algorithms for dvs systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(2):17.

Appendix A

Derivations of Equations (3.23.1)-(3.25)

Since the constraint in (3.12.7) is already accounted for by the feasibility condition (3.20.2), without loss of optimality, we may directly focus on the resolution of optimization problem in (3.18) under the constraints in (3.12.2)-(3.12.5). Since this problem is strictly convex and all its constraints are linear, the Slater's qualification conditions hold [8, chap.5], so that the Karush-Khun-Tucker (KKT) conditions [8, chap.4] are both necessary and sufficient for analytically characterizing the corresponding unique optimal global solution. Before applying these conditions, we observe that each power-rate function in (3.19) is increasing for $L_i \geq 0$, so that, without loss of optimality, we may replace the equality constraint in (3.12.3) by the following equivalent one: $\sum_{i=1}^{M} L_i \geq L_{tot}$. In doing so, the Lagrangian function of the afforded problem reads as in

$$\mathscr{L}\left(\{L_i, f_i, \nu_i, \mu\}\right) \equiv \mathscr{Z}\left(\{L_i, f_i\}\right) \sum_{i=1}^{M} \nu_i \left(L_i - f_i \Delta + L_b(i)\right) + \mu \left(L_t - \sum_{i=1}^{M} L_i\right), \qquad \text{(A.1)}$$

where $\mathscr{Z}\left(\{L_i, f_i\}\right)$ indicates the objective function in (3.18), ν_i's and μ are nonnegative Lagrange multipliers and the box constraints in (3.12.4),(3.12.5) are managed as implicit

ones. The partial derivatives of $\mathscr{L}(.;.)$ with respect to f_i, L_i are given by

$$\frac{\partial \mathscr{L}(.)}{\partial f_i} = \frac{\mathscr{E}_i^{max}}{f_i^{max}} \frac{\partial \Phi_i(f_i/f_i^{max})}{\partial \eta_i} + 2k_e\left(f_i - f_i^0\right) - \nu_i \Delta, \tag{A.2}$$

$$\frac{\partial \mathscr{L}(.)}{\partial L_i} = 2\frac{\partial}{\partial R_i} P_i^{net}\left(\frac{2L_i}{T_t - \Delta}\right) + \nu_i - \mu, \tag{A.3}$$

$i = 1, \ldots, M$, while the complementary conditions [8, chap.4] associated to the constraints present in (A.1) read as in

$$\nu_i \left[L_i - f_i \Delta + L_b(i)\right] = 0,\ i = 1, \ldots, M;\ \ \mu\left(L_{tot} - \sum_{i=1}^{M} L_i\right) = 0. \tag{A.4}$$

Hence, by equating (A.2) to zero we directly arrive at (3.23.1), that also accounts for the box constraint: $f_i^{\min} \leq f_i \leq f_i^{\max}$ through the corresponding projector operator. Moreover, a direct exploitation of the last complementary condition in (A.4) allows us to compute the optimal μ^* by solving the algebraic equation in (3.25). In order to obtain the analytical expressions for L_i^* and ν_i^*, we proceed to consider the two cases of $\nu_i^* > 0$ and $\nu_i^* = 0$. Specifically, when ν_i^* is positive, the i-th constraint in (3.12.2) is bound (see (A.4)), so that we have

$$L_i^* = \Delta f_i^* - L_b(i), \quad \text{at } \nu_i^* > 0. \tag{A.5}$$

Hence, after equating (A.3) to zero, we obtain the following expression for the corresponding optimal ν_i^*:

$$\nu_i^* = \mu^* - 2\left[\frac{\partial P_i^{net}}{\partial R_i}\left(\frac{2L_i^*}{T_t - \Delta}\right)\right], \text{ at } \nu_i^* > 0. \tag{A.6}$$

Since L_i^* must fall into the closed interval $[0, \Delta f_i^* - L_b(i)]$ for feasible *CPOPs* (see (3.12.2),(3.12.5)), at $\nu_i^* = 0$, we must have: $L_i^* = 0$ or $0 < L_i^* < \Delta f_i^* - L_b(i)$. Specifically, we observe that, by definition, vanishing L_i^* is optimal when $[\partial \mathscr{L}/\partial L_i]_{L_i=0} \geq 0$. Therefore, by imposing that the derivative in (A.3) is nonnegative at $L_i^* = \nu_i^* = 0$, we obtain the following condition for the

resulting optimal μ^*:

$$\mu^* \leq 2\left[\partial P_i^{net}(R_i)/\partial R_i\right]_{R_i=0} \triangleq TH(i), \quad \text{at } v_i^* = L_i^* = 0. \tag{A.7}$$

Passing to consider the case of $v_i^* = 0$ and $L_i^* \in]0, \Delta f_i^* - L_b(i)[$, we observe that the corresponding KKT condition is *unique*, it is necessary and sufficient for the optimality and requires that (A.3) vanishes [8, chap.4]. Hence, the application of this condition leads to the following expression for the optimal L_i^* (see (A.3)):

$$L_i^* = \frac{(T_t - \Delta)}{2}\left(\partial P_i^{net}(R_i)/\partial R_i\right)^{-1}(\mu^*/2), \text{ at } v_i^* = 0 \text{ and } 0 < L_i^* < (\Delta f_i^* - L_b(i)). \tag{A.8}$$

Equation (A.8) vanishes at $\mu^* = TH(i)$ (see (A.7)) and this proves that the function: $L_i^*(\mu^*)$ vanishes at $\mu^* = TH(i)$. Therefore, since (A.7) already assures that vanishing L_i is optimal at $v_i^* = 0$ and $\mu^* \leq TH(i)$, we conclude that the expression in (A.8) for the optimal L_i^* must hold when $v_i^* = 0$ and $\mu^* \geq TH(i)$. This structural property of the optimal scheduler allows us to merge (A.7),(A.8) into the following equivalent expression:

$$L_i^* = \frac{(T_t - \Delta)}{2}\left[\left(\partial P_i^{net}(R_i)/\partial R_i\right)^{-1}(\mu^*/2)\right]_+, \text{ for } v_i^* = 0, \tag{A.9}$$

so that Equation (3.23.2) directly arises from (A.5),(A.9). Finally, after observing that v_i^* cannot be negative by definition, from (A.6) we obtain (3.24). This completes the proof of *Proposition 4*.

Appendix B

Proof of Proposition 5

The reported proof exploits arguments based on the Lyapunov Theory which are similar, indeed, to those already used, for example, in [8, sections 3.4, 8.2], [104, Appendix II]. Specifically, after noting that the feasibility and strict convexity of the *CPOP* in (3.18) guarantees the existence and uniqueness of the equilibrium point of the iterates in (3.27) and (3.28), we note that *Proposition 4* assures that, for any assigned $\mu^{(n)}$ and $\left\{L_i^{(n-1)}\right\}$, Equations (3.28.1)-(3.28.3) give the corresponding optimal values of the primal and dual variables $\left\{f_i^{(n)}, L_i^{(n)}, \nu_i^{(n)}\right\}$. Hence, it suffices to prove the global asymptotic convergence of the iteration in (3.27). To this end, after posing

$$U^{(n-1)}\left(\left\{L_i^{(n-1)}\right\}\right) \equiv U^{(n-1)} \triangleq \left[\sum_{i=1}^{M} L_i^{(n-1)} - L_{tot}\right]^2, \tag{B.1}$$

we observe that $U^{(n-1)} > 0$ for $\left\{L_i^{(n-1)}\right\} \neq \{L_i^*\}$ and $U^{(n-1)} = 0$ at the optimum, i.e., for $\left\{L_i^{(n-1)}\right\} = \{L_i^*\}$[1]. Hence, since $U^{(n-1)}(.)$ in (B.1) is also radially unbounded (that is, $U^{(n-1)}(.) \to \infty$ as $||\sum_{i=1}^{M} L_i^{(n-1)} - L_{tot}|| \to \infty$), we conclude that (B.1) is an admissible Lyapunov's function for the iterations in (3.27). Hence, after posing $U^{(n)}\left(\left\{L_i^{(n)}\right\}\right) =$

[1] *Proposition 4* proves that, for any assigned $\mu^{(n)}$, the relationship in (3.28.3) gives the corresponding optimal $L_i^{(n)}$, $i = 1, \ldots, M$. This implies, in turn, that $U^{(n-1)}(.)$ in (B.1) vanishes if and *only* if the global optimum is attained, that is, at $L_i^{(n-1)} = L_i^*$, for any $i = 1, \ldots, M$.

$U^{(n)} \triangleq \left[\sum_{i=1}^{M} L_i^{(n)} - L_{tot}\right]^2$, according to the Lyapunov's Theorem [8, section 3.10], we must prove that the following (sufficient) condition for the asymptotic global stability of (3.27) is met:

$$U^{(n)} < U^{(n-1)}, \text{ for } n \to \infty. \tag{B.2}$$

To this end, after assuming $U^{(n-1)} > 0$, let us consider, at first, the case of

$$\left(\sum_{i=1}^{M} L_i^{(n-1)} - L_{tot}\right) > 0. \tag{B.3}$$

Hence, since $\alpha^{(n-1)}$ is positive, we have (see (3.27)): $\mu^{(n)} < \mu^{(n-1)}$, that, in turn, leads to (see (3.28.3)): $L_i^{(n)} < L_i^{(n-1)}$, for any $i = 1,\ldots,M^2$. Therefore, in order to prove (B.2), it suffices to prove that the following inequality holds for large n:

$$\left(\sum_{i=1}^{M} L_i^{(n)} - L_{tot}\right) \geq 0. \tag{B.4}$$

To this end, we observe that: *i)* $\{\alpha(n-1)\}$ in (3.27) vanishes for $n \to \infty$; and, *ii)* $L_i^{(n)}$ is limited up to $\Delta f_i^{\max}$, for any $i = 1,\ldots,M$ (see the constraints in (3.12.2), (3.12.4)). As a consequence, the difference: $\left(\mu^{(n)} - \mu^{(n-1)}\right)$ may be done vanishing as $n \to \infty$. Hence, after noting that the functions in (3.27), (3.28) are continue by assumption, a direct application of the Sign Permanence Theorem guarantees that (B.4) holds when the difference in (B.3) is positive.

By duality, it is direct to prove that (B.2) is also met when the difference in (B.3) is negative. This completes the proof of *Proposition 5*.

[2] About this point, the formal assumption of section 3.2.3 guarantees that: *i)* $\pi_i^{-1}(.)$ in (3.28.2) is strictly increasing in $\nu_i^{(n)}$ over the feasible set $\left[f_i^{\min}, f_i^{\max}\right]$; *ii)* $\left(\partial P_i^{net}(R_i)/\partial R_i\right)^{-1}(.)$ in (3.28.3) is strictly increasing in $\mu^{(n)}$; and, *iii)* $\nu_i^{(n)}$ in (3.28.1) is strictly increasing in $\mu^{(n)}$ for $\nu_i^{(n)} > 0$. Hence, the condition: $\mu^{(n)} < \mu^{(n-1)}$ guarantees that: $L_i^{(n)} < L_i^{(n-1)}$, for any $i = 1,\ldots,M$.

Appendix C

Derivations of the *GreenNetDC* OP solution

Since the constraint in (3.41.3) is already accounted for by the feasibility condition (3.52), without loss of optimality, we may directly focus on the solution of the optimization problem in (3.41.1) under the constraints in (3.41.2), (3.41.4), (3.41.5). Since this problem is strictly convex and all its constraints are linear, the Slater's qualification conditions hold [8, chap.5]. We observe that each power-rate function in the third term of (3.41.1) is nondecreasing for: $F_{ij}t_{ij} \geq 0$, so that, without loss of optimality, we may replace the equality constraint in (3.41.2) by the following equivalent one: $\sum_{i=1}^{M}\sum_{j=1}^{Q} F_{ij}t_{ij} \geq L_{tot}$. Moreover, the frequencies for each VM is known so it plays the role of a coefficient. Therefore, the *OP* may be simplified as in followings:

$$\min_{t_{ij}} \sum_{i=1}^{M}\sum_{j=0}^{Q}\left[AC_{eff}F_{ij}^{3}t_{ij}\right] + \sum_{i=1}^{M}\mathscr{E}_{REc}(i) + \sum_{i=1}^{M}\sum_{j=1}^{Q}(T_t - T)P_i^{CMc}\left(\frac{2F_{ij}t_{ij}}{T_t - T}\right), \tag{C.1.1}$$

$$\text{s.t.: } \sum_{j=0}^{Q} t_{ij} - T = 0, \quad i = 1,\ldots,M, \tag{C.1.2}$$

$$L_{tot} - \sum_{i=1}^{M}\sum_{j=1}^{Q} F_{ij}t_{ij} = 0, \tag{C.1.3}$$

$$t_{ij} - T \leq 0, \;\; i = 1, \ldots, M, \;\; j = 0, \ldots, Q. \tag{C.1.4}$$

After denoting the objective function in (C.1.1) by $\mathscr{Z}(\{t_{ij}\})$, we have: $\mathscr{Z}(\{t_{ij}\}) \triangleq$ (C.1.1).

$$\mathscr{Z}(\{t_{ij}\}) \triangleq \sum_{i=1}^{M} \sum_{j=0}^{Q} \left[AC_{eff} F_{ij}^{3} t_{ij} \right] + \sum_{i=1}^{M} \mathscr{E}_{REc}(i) + \sum_{i=1}^{M} \sum_{j=1}^{Q} (T_t - T) P_i^{CMc} \left(\frac{2F_{ij} t_{ij}}{T_t - T} \right), \tag{C.2}$$

hence, the Lagrangian function of the problem (C.1) is constructed as in

$$\begin{aligned} \mathscr{L}\left(\{t_{ij}, \nu_{ij}, \mu_i\}\right) \equiv \mathscr{Z}(\{t_{ij}\}) + \sum_{i=1}^{M} \mu_i \left(\sum_{j=0}^{Q} t_{ij} - T \right) + \\ \mu_{M+1} \left(L_{tot} - \sum_{i=1}^{M} \sum_{j=1}^{Q} F_{ij} t_{ij} \right) + \sum_{i=1}^{M} \sum_{j=0}^{Q} \nu_{ij} \left(t_{ij} - T \right) \end{aligned} \tag{C.3}$$

, ν_i's and μ are nonnegative Lagrange multipliers and the box constraints in (3.41.4),(3.41.5) are managed as implicit ones. The partial derivative of $\mathscr{L}(.)$ with respect to t_{ij} is given by the partial derivative of $\mathscr{Z}(.)$ with respect to t_{ij} is given by

$$\begin{aligned} \frac{\partial \mathscr{Z}(.)}{\partial t_{ij}} = AC_{eff} F_{ij}^{3} + (T_t - T) \frac{\partial P_i^{CMc}(.)}{\partial t_{ij}}, \\ i = 1, \ldots, M, \; j = 0, \ldots, Q. \end{aligned} \tag{C.4}$$

Hence, the $\mathscr{Z}(.)$ is linear and by equating the partial derivatives of (C.3) to zero, we can solve the resulting algebraic equation with respect to t_{ij}. So doing, we calculate the $M(Q+1)$ variables by solving the aforementioned linear problem, which is the same as the Gauss-Jordan system which is produced by the M equations in (C.1.2) and the equations in (C.1.3) and (C.3).

Appendix D

Derivations of STAS solution over *GreenNetDC*

In the application architecture considered in Fig. 3.10, the energy consumption for the static scheduler (STAS) (i.e., it is enough to consider f_i^{max} instead of each discrete frequency F_{ij} and T instead of t_{ij} for i-th VM, simultaneously; switch cost is calculated just for each incoming workload, also, VMs are working with their maximum frequencies, so just changing from the idle mode to the f_i^{max} for each VM at first stage is considered.) over *SOP* optimization problem in (3.41.1) named $\overline{\mathscr{E}}_{tot}^{(STAS)}$ is computable in closed-form and equates

$$\overline{\mathscr{E}}_{tot}^{(STAS)} \triangleq M_s \left[AC_{eff} f_i^{max} T + k_e (f_i^{max})^2 + \mathscr{X}(F_{ij}, t_{ij}) \right] \tag{D.1}$$

where $\mathscr{X}(f_i^{max}, T)$ is the average static value for the aforementioned arguments which borrows Proposition 6 and (3.41) and calculates as

$$\mathscr{X}(F_{ij}, t_{ij}) \triangleq E\left\{(T_t - T)P_i^{CMc}\left(\frac{2F_{ij}t_{ij}}{T_t - T}\right)\right\} = E\left\{(T_t - T)\Omega_i\left(\overline{RTT}_i\frac{2F_{ij}t_{ij}}{T_t - T}\right)^2\right\} =$$
$$M_p E\left\{\left(F_{ij}t_{ij}\right)^2\right\} = M_p E\left\{\left(\frac{\overline{L}_{tot}}{M_s}\right)^2\right\} = M_p \int_{\overline{L}_{tot}-a}^{\overline{L}_{tot}+a}\left(\frac{\overline{L}_{tot}}{M_s}\right)^2 p(\overline{L}_{tot})\partial(\overline{L}_{tot}) =$$
$$M_p \int_{\overline{L}_{tot}-a}^{\overline{L}_{tot}+a}\left(\frac{\overline{L}_{tot}}{M_s}\right)^2 \frac{1}{2e}\partial(\overline{L}_{tot})] = \frac{1}{6e}M_p M_s^{-2}\left[\left(\overline{L}_{tot}+a\right)^3 - \left(\overline{L}_{tot}-a\right)^3\right] \tag{D.2}$$

where $M_s \triangleq \left[(\overline{L}_{tot}+a)/(T f_i^{max})\right]$ is the number of constantly running machines that *STAS* uses for satisfying the peak load : $T f_i^{max}$. Also, $M_p \triangleq \left[\Omega_i (T_t - T)^{-1}\left(2\overline{RTT}_i\right)^2\right]$, and $F_{ij}t_{ij} \equiv \overline{L_{tot}}/M_s$. The second equivalent describes that expected value of fraction workload for each VM in the system can be equal to the average fraction of incoming workload of the system. We should highlight that, average total incoming workload to the system is uniformly distributed (i.i.d) so the integral can be simplified as $1/2e$.

Appendix E

Derivations of modified *NetDC* and *HybridNetDC* solutions

NetDC approach considers the workload fractions for each VM in each physical server. Besides, it pays close attention to the frequencies fluctuations for each incoming workload to the system. Also, this approach along *GreenNetDC* are able to works for various kinds of incoming workload with the aforementioned PMR, and are able to adapt the system for the online incoming workload respecting the QoSs and SLA constraints. Because of several similarities, we were interested in comparing *GreenNetDC* with *NetDC* [23]. To make the comparison meaningful, we modeled the *NetDC* respecting to the presented architecture in Fig.3.10 and made a minor change to the *NetDC*. We modified CPU and end-to-end link parts (i.e., the optimization problem is similar *OP* in (3.41)). Specifically, we add idle mode frequency and correlated-idle-power (i.e., P^{idle}) to each CPU, add idle mode power for the end-to-end links and use end-to-end link model equation (3.38). Moreover, *HybridNetDC* which is elicited from *NetDC* approach is similar to the *NetDC* except the communication part ($R_i = R_t \; i = 1, \ldots, M$). Specifically, *Hybrid NetDC* [17] considers fixed value communication part for each parallel end-to-end link. From a formal point of view, *HybridNetDC* implements

the solution of the constrained minimization problem in (3.41.1)-(3.41.5) over the variables $\{f_i, t_i\}$ at $R_i = R_t$.

Appendix F

Derivations of the Applied Functions in Algorithm 5

To calculate the consolidation state, we need to introduce some functions which are crucial for finding solution $\mathscr{A}$ of optimization problem 4.13.

i) $u_{-1}(x,a)$ and $H_{-1}(x,a)$ is the unit-size diversion of unit-size Heaviside' functions which are defined as:

$$\begin{aligned} u_{-1}(x,a) &\triangleq \frac{1}{2}\left[1+\tanh\left(\frac{x-a}{\delta}\right)\right], \delta>0, \\ H_{-1}(x,a) &\triangleq \frac{1}{\delta(\cosh(\frac{x-a}{\delta}))^2}, \\ \delta &\cong 0.5\times 10^{-6}, \quad x\geq 0 \end{aligned} \tag{F.1}$$

It is enough to use $u_{-1}(f_i, f_i^{ON})$ which express the coefficient of $E_c^{idle}(i)$ energy usage for the consolidation for each VM. f_i^{ON} is the $VM(i)$'s processing speed that works in consolidations state. Besides, we can use $H_{-1}(f_i, f_i^{ON})$ to demonstrate the portion of the CPU energy uses in the consolidation state of the system.

ii) we define $\Gamma \triangleq \theta E_{tot}(t) - (1-\theta)r(t)$ as a utilization of the scheduler.

iii) we define ρ function and add it with the Γ and make the equivalent optimization problem with the same solution for the consolidation slots:

$$\begin{aligned}\rho(\{L_i, f_i, i=1,\ldots,M\}) \triangleq \sum_{i=1}^{M} \Big[\max\{0;-L_i\}\Big]^2 + \\ \Big[\max\{0;-f_i\}\Big]^2 + \Big[\max\{0;(L_i - L_i^{max})\}\Big]^2 + \\ \Big[\max\{0;(f_i - f_i^{max})\}\Big]^2.\end{aligned} \tag{F.2}$$

Therefore, we have

$$\min_{\mathscr{A}} \quad (\Gamma + \rho) \tag{F.3.1}$$

subjected to:

$$\text{eqs. (4.15.2), (4.15.3) (4.15.8), (4.19), Lemmas 1, 2} \tag{F.3.2}$$

which is resolve using multi-gradient based stochastic approximation method (details are explained in the Algorithms 4 and 5). In the following, we define some functions in order to apply them in the **Algorithm 5**.

As is shown in eq. (F.4), the first function called *ConsolCpuDyn* which is updated iteratively, expresses the appropriate frequency for root of the gradient-based movement along the function of optimized E_{tot} for each time-slot.

$$ConsolCpuDyn^{(l)} \triangleq \hat{\pi}_i\left(f_i^{(l)}(\hat{t}); r^{(l)}(\hat{t}), \Delta\right) - \left(\theta 2k_e f_i^{(0)}(\hat{t}) + \Delta r^{(l)}(\hat{t})\right) \tag{F.4}$$

where

$$\begin{aligned}\hat{\pi}_i(.) \triangleq & \theta\left[\frac{1}{2}H_{-1}(f_i, f_i^{ON}) + \left(\frac{E_c^{max}(i) - E_c^{idle}(i)}{f_i^{max}}\right)\left(\frac{f_i}{f_i^{max}}\right)^2 + 2k_e f_i\right] \\ & + 2g\left[\max\left(0; f_i - f_i^{max}\right) - \max(0; -f_i)\right] + \\ & \mathbf{1}_{[i\in\overline{S}(\hat{t}-1)]}\left[r_i T_{ON} u_{-1}(f_i, f_i^{ON})\right].\end{aligned} \tag{F.5}$$

Moreover, the second function named *ConsolLAN* explains the portion of energy consumption for the consolidation state at l-th iteration respect to the other multi-gradient step-size parameters (note that the slots which system goes to the consolidation state is belong to the $\overline{S}$ set and t is called $\hat{t}$). Therefore we have

$$\begin{aligned} ConsolLAN(l) \triangleq \theta \dot{E}_{LAN}(\tilde{L}_i^{(l)}(\hat{t}))+ \\ \Big[\max(0;(\tilde{L}_i^{(l)}(\hat{t}) - L_i^{max})) - \max(0;-\tilde{L_i}^{(l)}(\hat{t}))\Big] - \mu^{(l)}(\hat{t}), \end{aligned} \tag{F.6}$$

where $\dot{E}_{LAN}$ is the derivation of the communication function of (4.7) as

$$\dot{E}_{LAN}(.) \triangleq \frac{\partial E_{LAN}}{\partial L_i}. \tag{F.7}$$