Energy-saving QoS resource management of virtualized networked data centers for Big Data Stream Computing

Nicola Cordeschi, Mohammad Shojafar, Danilo Amendola and Enzo Baccarelli

Dept. of Information, Electrical and Telecommunication (DIET) engineering, ``Sapienza'' University of Rome, Via Eudossiana 18, 00184 Rome, Italy. Phone +39 06 44585366 FAX no. +39 06 4873300. Emails: {nicola.cordeschi, shojafar, amendola, enzo.baccarelli}@diet.uniroma1.it

ABSTRACT

In this research contribution, we develop the scheduler which optimizes the energy-vs.-performance trade-off in Software-as-a-Service (SaaS) Virtualized Networked Data Centers (VNetDCs) that support real-time Big Data Stream Computing (BDSC) services. Our objective is to minimize the communication-plus-computing energy which is wasted by processing streams of Big Data under hard real-time constrains on the per-job computing-plus-communication delays. For this purpose, the proposed scheduler operates 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 available VNetDC. In order to deal with the inherently nonconvex nature of the resulting resource management optimization problem, we develop a solving approach that leads to the lossless decomposition of the afforded problem into the cascade of two simpler sub-problems. The resulting optimal scheduler is amenable of scalable and distributed adaptive implementation. The performance of a Xen-based prototype of the scheduler is tested under several Big Data workload traces and compared with the corresponding ones of some state-of-the-art static and sequential schedulers.

Key words: Virtualized networked data centers, Energy-saving, Adaptive resource management, Real-time Big Data Stream Computing (BDSC), Virtual Machine (VM).

INTRODUCTION

Energy-saving computing through Virtualized Networked Data Centers (VNetDCs) is an emerging paradigm that aims at performing the adaptive energy management of virtualized Software-as-a-Service (SaaS) computing platforms. The goal is to provide QoS Internet services to large populations of clients, while minimizing the overall computing-plus-networking energy consumption (Cugola & Margara, 2012; Baliga, Ayre, Hinton, & Tucker, 2011; Mishra, Jain, & Durresi, 2012). As recently pointed out in (Mishra et al. 2012; Azodomolky, Wieder, & Yahyapour, 2013; Wang et al. 2014), the energy cost of communication gear for current data centers may represent a large fraction of the overall system cost and it is induced primarily by switches, LAN infrastructures, routers and load balancers.

However, actual virtualized data centers subsume the (usual) Map/Reduce-like batch processing paradigm and they are not designed for supporting networking-computing intensive real-time services, such as, for example, emerging Big Data Stream Computing (BDSC) services (Cugola et al. 2012). In fact, BDSC services retain the following (somewhat novel and unique) characteristics (Cugola et al. 2012; Scheneider, Hirzel, & Gedik, 2013; Qian, He, Su, Wu, Zhu, & Zhang, 2013; Kumbhare, 2014); (i) the incoming data (i.e., the offered workload) arrive continuously at volumes that far exceed the storage capabilities of individual computing machines. Furthermore, all data must be timely proceed but, typically, a few of data require to be stored. This means that the (usual) storing-then-computing batch paradigm is no longer feasible; (ii) since BDSC services acquire data from massive collections of distributed clients in a stream form, the size of each job is typically unpredictable and also its statistics may be quickly time-varying; and, (iii) the offered workload is a real-time data stream, which needs real-time computing with latencies *firmly* limited up to a few of seconds (Qian et al. 2013; Kumbhare, 2014). Imposing hard limits on the overall per-job delay requires, in turn, that the overall VNetDC is capable to quickly adapt its resource allocation to the current (a priori unpredictable) size of the incoming big data.
In order to attain energy saving in such kind harsh computing scenario, the joint balanced provisioning and adaptive scaling of the networking-plus-computing resources is demanded. This is the focus of this work, 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 formulas 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 workload offered by the supported Big Data Stream application and converges to the optimal resource allocation without the need to be restarted.

Related Work

Updated surveys of the current technologies and open communication challenges about energy-efficient data centers have been recently presented in (Mishra et al. 2012; Balter, 2013). Specifically, power management schemes that exploit Dynamic Voltage and Frequency Scaling (DVFS) techniques for performing resource provisioning are the focus of (Chen & Kuo, 2005; Kim, Buyya, & Kim, 2007; Li, 2008). Although these contributions consider hard deadline constraints, they do not consider, indeed, the performance penalty and the energy-vs.-delay tradeoff stemming from the finite capacity of the utilized network infrastructures and do not deal with the issues arising from perfect/imperfect Virtual Machines (VMs) isolation in VNetDCs.
Energy-saving dynamic provisioning of the computing resources in virtualized green data centers is the topic of (Liu, Zhao, Liu, & He, 2009; Mathew, Sitaraman, & Rowstrom, 2012; Padala, You, Shin, Zhu, Uysal, Wang, Singhal, & Merchant, 2009; Kusic & Kandasamy, 2009; Govindan, Choi, Urgaonkar, Sasubramanian, & Baldini, 2009; Zhou, Liu, Jin, Li, Li, & Jiang, 2013; Lin, Wierman, Andrew, & Thereska, 2011; Laszewski, Wang, Young, & He, 2009). Specifically, (Padala et al. 2009) formulates 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 problems, where the target values are a priori unknown. Roughly speaking, the common approach pursued by (Mathew et al. 2012; Kusic et al. 2009; Govindan et al. 2009) and (Lin et al. 2011) is to formulate the afforded minimum-cost problems as sequential optimization problems and, then, solve them by using limited look-ahead control. Hence, the effectiveness of this approach relies on the ability to accurately predict the future workload and degrades when the workload exhibits almost

unpredictable time fluctuations. In order to avoid the prediction of future workload, (Zhou et al. 2013) resorts to a Lyapunov-based techniques, 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.

A suitable exploitation of some peculiar features of the network topology of current NetDCs is at the basis of the capacity planning approach recently proposed in (Wang et al. 2014). For this purpose, a novel traffic engineering-based approach is developed, which aims at reducing the number of active switches, while simultaneously balancing the resulting communication flows. Although the attained reduction of the energy consumed by the networking infrastructures is, indeed, noticeable, the capacity planning approach in (Wang et al. 2014) 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 which perform static resource allocation is, indeed, the focus of (Baliga et al. 2011; Tamm, Hersmeyer, & Rush, 2010), 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 static schedulers, but do not attempt to optimize it through the dynamic joint scaling of the available communication-plus-computing resources. Providing computing support to the emerging BDSC services is the target of some quite recent virtualized management frameworks such as, S4 (Neumeyer, Robbins, & Kesari, 2010), D-Streams (Zaharia, Das, Li, Shenker, & Stoica, 2012) and Storm (Loesing, Hentschel, & Kraska, 2012). While these management systems are specifically designed for the distributed runtime support of large scale BDSC applications, they do not still provide automation and dynamic adaptation to the time-fluctuations of Big Data streams. Dynamic adaptation of the available resources to the time-varying rates of the Big Data streams is, indeed, provided by the (more recent) Time Stream (Qian et al. 2013) and PLAStiCC (Kumbhare, 2014) management frameworks. However, these frameworks manage only the computing resources and do not consider the simultaneous management of the networking resources.
The rest of this chapter is organized as follows. After modeling in Section 2 the considered virtualized NetDC platform, in Section 3 we formally state the afforded CCOP and, then, we solve it and provide the analytical conditions for its feasibility. In Section 4, we present the main structural properties of the resulting optimal scheduler and analytically characterize the (possible) occurrence of hibernation phenomena of the instantiated VMs. Furthermore, we also develop an adaptive implementation of the optimal scheduler and prove its convergence to the optimum. In Section 5, we test the sensitivity of the average performance of the proposed scheduler on the Peak-to-Mean Ratio (PMR) and time-correlation of the (randomly time-variant) offered Big Data workload and, then, we compare the obtained performance against the corresponding ones of some state-of-the-art static and sequential schedulers. After addressing some possible future developments and hints for incoming research in Section 6, the conclusive Section 7 recaps the main results.
About the adopted notation, $[x]_a^b$ indicates $\min\{\max\{x; a\}; b\}$, $[x]_+$ indicates $\max\{x;0\}$, $\triangleq$ means equality by definition, while $\mathbf{1}_{[A]}$ is the (binary-valued) indicator function of the A event (that is, $\mathbf{1}_{[A]}$ is unit when the A event happens, while it vanishes when the A event does not occur).

THE CONSIDERED VNETDC PLATFORM

A networked virtualized platform for real-time parallel computing is composed by multiple clustered virtualized processing units interconnected by a single-hop virtual network and managed by a central controller (see, for example, Fig.1 of (Azodomolky et al. 2013)). 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 to the VNetDC, the central resource controller dynamically performs both admission control and allocation of the available virtual resources (Almeida, Almeida, Ardagna, Cunha, & Francalanci, 2010). Hence, according to the emerging communication-computing system architectures for the support of real-time BDSC services (see, for example, Figs. 1 of (Azodomolky et al. 2013) and (Ge, Feng, Feng, & Cameron, 2007)), a VNetDC is composed by multiple reconfigurable VMs, that operate at the Middleware layer of the underlying protocol stack and are interconnected by a throughput-limited switched Virtual Local Area Network (VLAN). The topology of the VLAN is of star-type, and, in order to guarantee inter-VM communication, the Virtual Switch of Fig.1 acts as a gather/scatter central node. The operations of both VMs and VLAN are jointly managed by a Virtual Machine Manager (VMM), which performs task scheduling by dynamically allocating the available virtual computing-plus-communication resources to the VMs and Virtual Links (VLs) of Fig.1. A new job is initiated by the arrival of a data of size L_{tot} *(bit)*. Due to the (an aforementioned) hard real-time nature of the supported BDSC services, full processing of each input job must be completed within *assigned* and *deterministic* processing time which spans T_t seconds.

Hence, in our framework, a real-time job is characterized by: *i)* the size L_{tot} of the data segment 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 which are available at the Middleware layer of Fig.1. In principle, each VM may be modeled as a virtual server that is capable to process f_c bits per second (Portnoy, 2012). 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 runtime, and it may assume values over the interval $[0, f_c^{\max}]$, where $f_c^{\max}$ *(bit/s)* is the per-VM maximum allowed processing rate[1]. 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 $\Delta(s)$, *regardless* from 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 (Portnoy, 2012).

Hence, by definition, the *utilization factor* η of the VM equates (Portnoy, 2012): $\eta \triangleq f_c / f_c^{\max} \in [0,1]$. Then, as in (Kim et al. 2009; Almeida et al. 2010; Koller, Verma, & Neogi, 2010), let $E_c = E_c(f_c)$ *(Joule)* be the overall energy consumed by the VM to process a single task of duration Δ at the processing rate f_c, and let $E_c^{\max} = E_c(f_c^{\max})$ *(Joule)* be the corresponding maximum energy when the VM operates at the maximum processing rate $f_c^{\max}$. Hence, by definition, the (dimensionless) ratio is the so-called Normalized Energy Consumption of the considered VM (Warneke & Kao, 2011). 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.1 (Kim, Beloglazov, & Buyya, 2009; Zhu, Melhem, & Childers, 2003).

$$\Phi(\eta) \triangleq \frac{E_c(f_c)}{E_c^{\max}} = \Phi(\frac{f_c}{f_c^{\max}}), \tag{1}$$

Specifically, the following three main conclusions are widely agreed by several recent studies (Kansal, Zhao, Liu, Kothari, & Bhattacharya, 2010; Ge et al. 2007; Cordeschi, Shojafar, & Baccarelli, 2013; Stoess, Lang, & Bellosa, 2007), (Laszewski et al. 2009). First, under the assumption of isolation of the

[1] Since L_{tot} is expressed in *(bit)*, we express $f_c^{\max}$ *(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.

VMs, the energy consumed by the physical servers is the summation of the energies wasted by the hosted VMs. Second, in practice, the per-VM normalized energy consumption $\Phi(\eta)$ follows the *c*-powered behavior:

$$\Phi(\eta) = (1-b)\eta^c + b, \qquad 0 \le \eta \le 1. \tag{2}$$

In (2), the exponent parameter $c \ge 1$ is application-depending and it is empirically evaluated at runtime (e.g., *c=1* in (Kansal et al. 2010) and *c=2* in (Lin et al. 2011)), while $b \in [0,1)$ is the fraction of the per-VM maximum energy consumption $E_c^{\max}$ which is wasted by the VM in the idle state, that is, $E_c^{idle} = b / E_c^{\max}$. Third, from an application point of view, the aforementioned per-VM energy parameters $E_c^{\max}$, E_c^{idle} and *c* may be estimated at runtime. In particular, the *Joulemeter* system described in section 5 of (Kansal et al. 2010) is a practical tool that estimates at runtime the per-VM energy parameters by performing suitable measurements atop the VMM of Fig.1. Finally, before proceeding, we anticipate that the validity of the analytical developments of Sections 3 and 4 is not limited up to the energy model in (2) and it extends, indeed, to the more general case in which $\Phi(\eta)$ is an increasing and convex function of η.

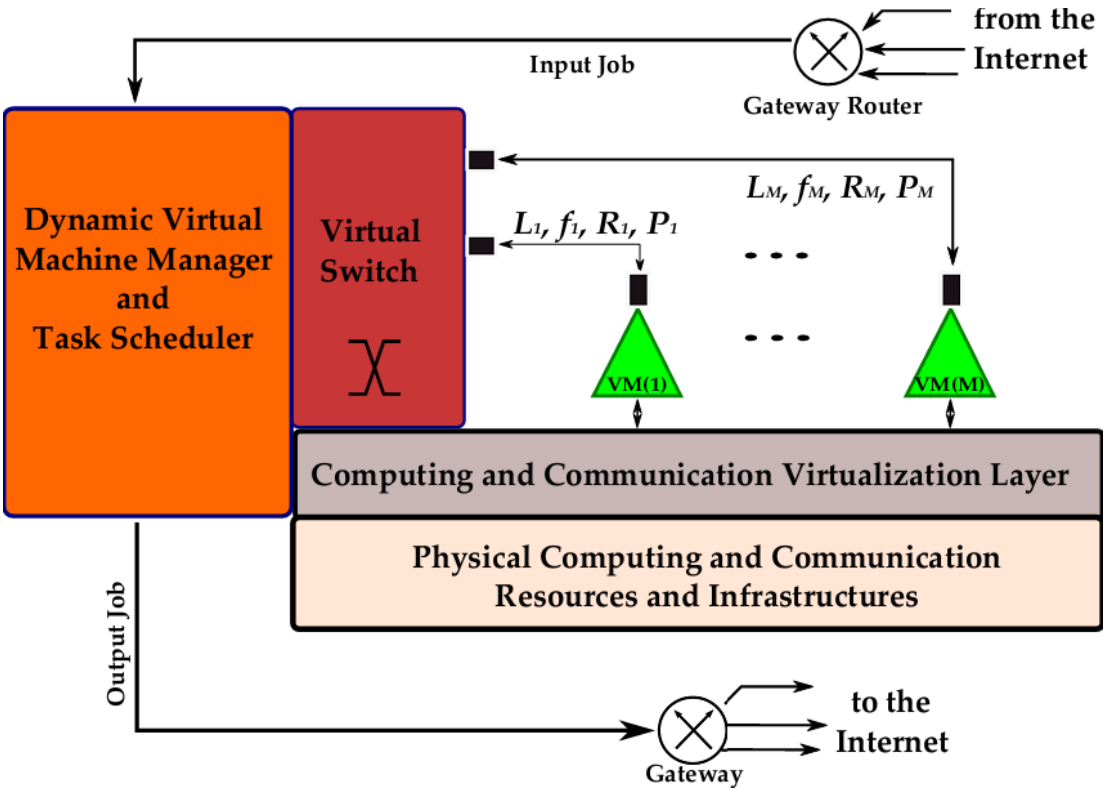

Fig.1: A typical VNetDC architecture for the support of real-time BDCSC services. The Virtual LAN (VLAN) is atop the Virtualization Layer. Black boxes indicate Virtual Network Interface Cards (VNICs) ending point-to-point TCP-based connections. Physical Ethernet adapters connect the VLAN to the underlying physical network (Azodomolky et al. 2013). The reported architecture is instantiated at the Middleware layer.

Big Data streaming workload and power-limited virtualized communication infrastructures)

Let $M= min\ \{M_V, M_T\}$ 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.1 assigns to the *i*-th VM, e.g., *VM(i)*. Hence, the following constraint: $\sum_{i=1}^{M} L_i = L_{tot}$ guarantees that the overall job L_{tot} is *partitioned* into (at the most) *M* parallel tasks. In order to hold at the minimum the transmission delays from (to) the Scheduler to (from) the connected VMs of Fig.1, as in (Baliga et al. 2011), (Azodomolky et al. 2013), we assume that *VM(i)* communicates to the Scheduler through a dedicated (i.e., contention-free) reliable virtual link, that operates at the transmission rate of R_i *(bit/s)*, *i=1,…, M* and it is equipped with suitable Virtual Network Interface Cards (VNICs) (see Fig.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 $P_i^{net}(i)$ consumed by the transmit VNIC and Switch and the corresponding power P_i^{net} wasted by the receive VNIC (see Fig.1).

About the actual value of P_i^{net}, we noted 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 suites (mainly, the TCPNewReno one) for attaining end-to-end (typically, multi-hop) reliable communication (Mishra et al. 2012). At this regard, we note that the data center-oriented versions of the legacy TCPNewReno suite proposed in (Vasudevan, Phanishayee, & Shah, 2009; Alizadeh, Greenberg, Maltz, & Padhye, 2010; Das & Sivalingam, 2013) 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 TCPNewReno protocol. This means, in turn, that the average throughput R_i *(bit/s)* of the *i*-th virtual link of Fig.1 (i.e., the *i*-th end-to-end transport connection) equates (Kurose & Ross, 2013, section 3.7; (Jin, Guo, Matta, & Bestavros, 2003)

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

As it is well-known, *MSS (bit)* in (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 (e.g., $\overline{RTT_i}$ less than *1 (ms)* in typical data centers (Vasudevan et al. 2009)), and $\overline{P}_i^{Loss}$ is the average segment loss probability experienced by the *i*-th connection section 3.7 in (Kurose et al. 2013). At this regard, several studies point out that $\overline{P}_i^{Loss}$ scales down for increasing $\overline{P}_i^{net}$ as in (Liu, Zhou, &Giannakis, 2004); Cordeschi et al 2012a; Baccarelli, Cordeschi, & Patriarca, 2012b; Baccarelli & Biagi, 2003; Baccarelli, Biagi, Pelizzoni, &Cordeschi, 2007)

$$\overline{P}_i^{Loss} = (g_i P_i^{net})^{-d}, \qquad i=1,\ldots,M. \tag{4}$$

where $g_i (W^{-1})$ is the coding gain-to-receive noise power ratio of the *i*-th end-to-end connection, while the positive exponent *d* measures the diversity gain provided by the frequency-time interleavers implemented at the Physical layer. Explicit closed-form expressions for g_i and *d* are reported, for example, in (Liu et al. 2004; Cordeschi, Patriarca, & Baccarelli, 2012a) and (Baccarelli et al. 2012b) for various operating settings. Hence, after introducing (4) into (3), we obtain

$$P_i^{net} = \Omega_i (\overline{RTT_i} R_i)^{\alpha}, \qquad i = 1,\ldots,M. \tag{5}$$

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

Before proceeding, we point out that the α-powered (convex) formula in (5) featuring the power-vs.-throughput relationship of the *i*-th virtual link of Fig.1 holds *regardless* from the actual (possibly, multi-hop) topology of the adopted physical network (e.g., Fat-tree, BCube, DCell (Wang et al. 2014)). Formally speaking, the validity of (5) relies on the (minimal) assumption that TCP-based transport connections working in the Congestion Avoidance state are used for implementing the virtual links of Fig.1.

Reconfiguration cost in VNetDCs

Under the per-job delay constraints imposed by Big Data stream services (Zhou et al. 2013), the VMM of Fig.1 must carry out two main operations at runtime, namely, Virtual Machine management and load balancing. Specifically, goal of the Virtual Machine management is to adaptively control the Virtualization Layer of Fig.1. In particular, the set of the (aforementioned) VM's attributes:

$$\{\Delta, f_c^{\max}(i), \Phi_i(\eta_i), E_c^{\max}(i), L_b(i), i = 1,\ldots,M\}, \tag{6}$$

are dictated by the Virtualization Layer and, then, they are passed to the VMM of Fig.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 (Warneke et al. 2011).

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)* (Laszewski et al. 2009; Kim et al. 2007). Although the actual behavior of the function $\varepsilon(f_1; f_2)$ may depend on the adopted VMM and the underlying DVFS technique and physical CPUs (Portnoy, 2012), any practical $\varepsilon(f_1; f_2)$ function typically retains the following general properties (Laszewski et al. 2009; Kim et al. 2007): *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:

$$\varepsilon(f_1; f_2) = k_e (f_1 - f_2)^2 \qquad (Joule) \tag{7}$$

where k_e *(Joule/(Hz)* ^{2}*)* dictates the resulting per-VM reconfiguration cost measured at the Middleware layer (see Fig.1). For sake of concreteness, in the analytical developments of the following section 3, we directly subsume the quadratic model in (6). The generalization to the case of $\varepsilon(.;.)$ functions that meet the aforementioned (more general) analytical properties is, indeed, direct.

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

Due to the hard delay-sensitive feature of Big Data stream services, the Virtualization Layer of Fig.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 (e.g., 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.1 with a per-VM queue system that implements the (recently proposed) *mClock* scheduling discipline section 3 in (Gulati, Merchant, & Varman, 2010). Interestingly enough, Table 1 of (Gulati et al. 2010) 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 DVSF-enabled physical CPUs (see the *Algorithm 1* of (Gulati et al. 2010) for a code of the *mClock* scheduler).

About the TCP-based networking virtualization, several (quite recent) contributions (Ballami, Costa, Karagiannis, & Rowstron, 2011; Greenberg et al. 2011; Guo et al. 2010; Xia, Cui, & Lange, 2012) point out that the most appealing property of 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 (Azodomolky et al. 2013)). The Middleware layer architecture of Fig.1 of the considered virtualized NetDC is aligned, indeed, with this requirement and, then, it is *general enough* to allow the implementation of *agile* data centers.

Specifically, according to (Azodomolky et al. 2013), the VNetDC of Fig.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* (Ballami et al. 2011, Fig.5) provides a contention-free switched LAN abstraction atop tree-shaped physical network topologies, while *VL2* (Greenberg et al. 2011) works atop physical Clos' networks. Furthermore, *SecondNet*e (Guo et al. 2010, section 3) and *VNET/P* (Xia et al. 2012, section 4.2) provide bandwidth-guaranteed virtualized Ethernet-type contention-free LAN environments atop *any* TCP-based end-to-end connection. For this purpose, *SeconNet* implements Port-Switching based Source Routing (PSSR) (Guo et al. 2010, section 4), while *VNET/P* relies on suitable Layer2 Tunneling Protocols (L2TPs) (Xia et al. 2012, section 4.3).

An updated survey and comparison of emerging contention-free virtual networking technologies is provided by Table 2 of (Azodomolky et al. 2013). Before proceeding, we anticipate that the solving approach developed in section 3.1 still holds verbatim when the summation: $\sum_{i=1}^{M} E_i^{net}(i)$ in (7) is replaced by a single energy function $\chi(.)$ which is jointly convex and not decreasing in the variables $\{L_i\}$(see (15.1)). 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 (e.g., contentions among competing TCP flows) (Portnoy, 2012).

Remark 1 - Discrete DVFS

Actual VMs are instantiated atop physical CPUs which offer, indeed, only a *finite* set: $\mathrm{A} \triangleq \left\{\hat{f}^{(0)} \triangleq 0, \hat{f}^{(1)}, \ldots, \hat{f}^{(Q-1)} \triangleq f_c^{\max}\right\}$of Q discrete computing rates. Hence, in order to deal with both continuous and discrete reconfigurable physical computing infrastructures without} introducing loss of optimality, we borrow the approach formerly developed, for example, in (Neely, Modiano, & Rohs, 2003; Li, 2008). Specifically, after indicating by $\mathrm{B} \triangleq \left\{\hat{\eta}^{(0)} \triangleq 0, \hat{\eta}^{(1)}, \ldots, \hat{\eta}^{(Q-1)} \triangleq 1\right\}$ the discrete values of η which correspond to the frequency set A, we build up a per-VM *Virtual Energy Consumption* curve $\hat{\Phi}(\eta)$ by resorting to a piecewise linear interpolation of the allowed Q operating points: $\left\{\left(\hat{\eta}^{(j)}, \Phi(\hat{\eta}^{(j)})\right), j = 0, \ldots, (Q-1)\right\}$. 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 (Neely et al. 2003). 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 (Neely et al. 2003; Li, 2008), any point $\{(\hat{\eta}^*, \Phi(\eta^*))\}$, with $\hat{\eta}^{(j)} < \eta^* < \hat{\eta}^{(j+1)}$, on the virtual curve may be actually attained by time-averaging over Δ secs (i.e., on a *per-job* basis) the corresponding surrounding vertex points: $\{(\hat{\eta}^{(j)}, \Phi(\hat{\eta}^{(j)}))\}$ and $\{(\hat{\eta}^{(j+1)}, \Phi(\hat{\eta}^{(j+1)}))\}$. Due to the piecewise linear behavior of the virtual curve, as in (Neely et al. 2003; Li, 2008), it is *guaranteed* that the average energy cost of the discrete DVFS system equates that of the corresponding virtual one over *each* time interval of duration Δ (e.g., on a per-job basis).

□

OPTIMAL ALLOCATION OF THE VIRTUAL RESOURCES

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

$$E_{tot} \triangleq \sum_{i=1}^{M} E_c(i) + \sum_{i=1}^{M} E^{net}(i) \qquad (Joule), \tag{8}$$

under the (aforementioned) *hard* constraint T_t on the allowed per-job execution time. This last depends, in turn, on the (one-way) delays *{D(i), i=1,..., M}* introduced by the VLAN and the allowed per-task processing time Δ. Specifically, since the *M* virtual connections of Fig.1 are typically activated by the virtual switch of Fig.1 (e.g., the load dispatcher) in a parallel fashion (Scheneider et al. 2013), the overall two-way communication-plus-computing delay induced by the *i*-th connection of Fig.1 equates: *2D(i)*+ Δ, so that the hard constraint on the overall per-job execution time reads as in:

$$\max_{1 \le i \le M} \{2D(i)\} + \Delta \le T_t. \tag{9}$$

Thus, the overall CCOP assumes the following form:

$$\min_{\{R_i, f_i, L_i\}} \sum_{i=1}^{M} \left\{ \Phi_i\left(\frac{f_i}{f_i^{\max}}\right) E_i^{\max} + k_e (f_i - f_i^{\,0})^2 + 2P_i^{net}(R_i)\left(\frac{L_i}{R_i}\right) \right\}, \tag{10.1}$$

s.t:

$$(L_i + L_b(i)) \le f_i \Delta, \qquad i = 1,...,M, \tag{10.2}$$

$$\sum_{i=1}^{M} L_i = L_{tot},, \tag{10.3}$$

$$0 \le f_i \le f_i^{\max}, \qquad i = 1,...,M, \tag{10.4}$$

$$L_i \ge,0, \qquad i = 1,...,M, \tag{10.5}$$

$$\frac{2L_i}{R_i} + \Delta \le T_t, \qquad i = 1,...,M, \tag{10.6}$$

$$\sum_{i=1}^{M} R_i \le R_t, \tag{10.7}$$

$$R_i \ge,0, \qquad i = 1,...,M, \tag{10.8}$$

About the stated problem, the first two terms in the summation in (10.1) account for the computing-plus-reconfiguration energy $E_c(i)$ consumed by the *VM(i)*, while the third term in (10.1) is the communication energy $E^{net}(i)$ 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 (10.1) 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, $k_e(f_i - f_i^0)^2$ in (10.1) accounts for the resulting reconfiguration cost. The constraint in (10.2) guarantees that *VM(i)* executes the assigned task within Δ secs, while the (global) constraint in (10.3) assures that the overall job is partitioned into *M* parallel tasks. According to (9), the set of constraints in (10.6) forces the VNetDC of Fig.1 to process the overall job within the assigned hard deadline T_t. Finally, the global constraint in (10.7) limits up to R_t *(bit/s)* the aggregate transmission rate which may be sustained by the underlying VLAN of Fig.1, so that R_t is directly dictated by the actually considered VLAN standard (Azodomolky et al. 2013; Scheneider et al. 2013). Table 1 summarizes the main taxonomy used in this paper.

Table 1: Main taxonomy of the paper.

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
Δ (s)	Per-job maximum allowed computing time
T_t (s)	Per-job maximum allowed computing-plus-communication time
$E_i^{\max}$ $(Joule)$	Per-job maximum energy consumed by *VM(i)*
$f_i^{\max}$ (bit/s)	Maximum computing rate of *VM(i)*
$E_c(i)(Joule)$	Per-job maximum energy consumed by *VM(i)*
$E^{net}(i)(Joule)$	Per-job communication energy consumed by the *i*-th virtual link
$E_{tot}(Joule)$	Per-job total consumed energy

Remark 2 - 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 (Kim et al. 2009; Almeida et al. 2010; Balter, 2013, Chap. 27; Koller et al. 2010; Gandhi, Balter, &Adam, 2010; Loesing et al. 2012) . 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 (Balter, 2013, Chap. 27; (Loesing et al. 2012).
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 (Balter, 2013, Chap. 27; Loesing et al. 2012) point out that turning the idle servers OFF *increases*, indeed, the resulting average energy consumptions when the corresponding setup time *I* is larger than $2T_t$, and this conclusion holds also when the long-run (i.e., not instantaneous) average utilization factor of the servers is low and of the order of about 0.3 (see Table 27.1 of (Balter, 2013)). Second, in order to not induce 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 (10.6)). 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 (Zaharia et al. 2012)), the results of the aforementioned performance analysis induce to refrain to turn the idle servers OFF, at least when the setup times 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 (Kim et al. 2009; Almeida et al. 2010, section 3.1; Koller et al. 2010) unveil that, under real-time constraints, there is still large room for attaining energy savings by *adaptively* setting the set of the utilization factors in (2) of the available VMs, so as to properly track the instantaneous size L_{tot} of the offered workload. This is, indeed, the energy-management approach we pursue in the following sections, where we focus on the (still quasi unexplored) topic of the energy-saving adaptive configuration of the VNetDC of Fig.1.

□

Remark 3 - Generalization of the *CCOP*'s formulation.

Depending on the actually considered VNetDC platform and sustained BDSC services, some generalizations of the reported CCOP's 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 (8) through a constant factor 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 (Greenberg, Hamilton, & Maltz, 2009).

Second, the size $\tilde{L}_i^{(0)}(bit)$ of the workload output by *VM(i)* at the end of the computing phase may be different from the corresponding one L_i received in input at the beginning of the computing phase. Hence, after introducing the i-th inflating/deflating constant coefficient $\theta_i \triangleq \tilde{L}_i^{(0)} / L_i \gtrless 1$, the basic CCOP's formulation may be generalized by simply replacing the term: $2Li$ in (10.1), (10.6) by the following one: $(1+\theta_i)L_i, i=1,\ldots,M$.

Third, the summation in (10.1) 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 (Zhou et al. 2013), 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 (8) for the computing energy falls short, and the total energy E_{CPU} wasted by the physical host server may be modeled as in (Zhou et al. 2013):

$$\mathrm{E}_{\mathrm{CPU}} = \mathrm{E}_{\mathrm{CPU}}^{\max}\left\{(1-b)\left[\sum_{i=1}^{M}\left(\frac{f_i}{f_i^{\max}}\right)\right]^{c} + b\right\},$$ where $\mathrm{E}_{\mathrm{CPU}}^{\max}$ is the maximum energy consumed by the physical server, $\mathrm{E}_{\mathrm{CPU}}^{idle} \triangleq b\mathrm{E}_{\mathrm{CPU}}^{\max}$ 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 $\mathrm{E}_{\mathrm{CPU}}$ is jointly convex in $\{f_i,\ i=1,\ldots,M\}$ for $c \geq 1$, the solving approach of the next section 3.1 still applies verbatim.

□

Solving approach and optimal provisioning of the virtual resources

The CCOP in (10) 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 (10.1) 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} that *does not* depend on R_i. Therefore, neither guaranteed-convergence iterative algorithms nor closed-form expressions are, to date, available to compute the optimal solution $\left\{\hat{L}_i, \hat{R}_i, \hat{f}_i, i=1,\ldots,M\right\}$ of the CCOP.

However, we observe that the computing energy in (10.1) depends on the variables $\{f_i\}$, while the network energy depends on the variables $\{R_i\}$. Furthermore, since the variables $\{L_i\}$ are *simultaneously* present in the sets of constraints in (10.2) and (10.6), they affect *both* the computing and network energy consumptions. Formally speaking, this implies, that, for any assigned set of values of $\{L_i\}$, the minimization of the computing and network energies reduces to two *uncoupled* optimization sub-problems over the variables $\{f_i\}$ and $\{R_i\}$, respectively. These sub-problems are *loosely coupled* (in the sense of (Chiang, Low, Calderbank, & Doyle, 2007)), and (10.2), (10.6) are the coupling constraints.
Therefore, in order to compute the optimal solution of the CCOP, in the following, we formally develop a solving approach that is based on the lossless decomposition of the CCOP into the (aforementioned) CMOP and CPOP.
Formally speaking, for any assigned nonnegative vector $\vec{L}$ of the task's sizes, the CMOP is a (generally nonconvex) optimization problem in the communication rate variables $\{R_i,\ i=1,...,M\}$, so formulated:

$$\min_{\{R_i\}} \sum_{i=1}^{M} \frac{2P_i^{net}(R_i)}{R_i} L_i, \tag{11.1}$$

s.t:

CCOP's constraints in (10-6)-(10-8) (11.2)

Let $\{R_i^*(\vec{L}), i=1,...,M\}$ be the optimal solution of the CMOP in (11), and let

$$S \triangleq \left\{ \vec{L} \in (\Re_0^+)^M : \left(\frac{L_i}{R_i^*(\vec{L})} \right) \le \frac{(T_t - \Delta)}{2}, i=1,...,M; \sum_{i=1}^{M} R_i^*(\vec{L}) \le R_t \right\}, \tag{12}$$

be the region of the nonnegative M-dimensional Euclidean space which is constituted by all $\vec{L}$'s vectors that meet the constraints in (10.6)-(10.8). Thus, after introducing the dummy function: $\chi(L_1,...,L_M) \triangleq \sum_{i=1}^{M} \frac{2P_i^{net}(R_i^*)}{R_i^*} L_i$, the CPOP is formally stated as in:

$$\min_{\{R_i, f_i, L_i\}} \sum_{i=1}^{M} \left\{ \Phi_i \left(\frac{f_i}{f_i^{\max}} \right) E_i^{\max} + k_e (f_i - f_i^0)^2 \right\} + \chi(\vec{L}), \tag{13.1}$$

s.t:

CCOP's constraints in (10-2)-(10-5) and $\vec{L} \in S$ (13.2)

Let $\{L_i^*, R_i^*, f_i^*, i=1,...,M\}$ be 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 (10.1)-(10.8) admits the *same* feasibility region and the *same* solution of the cascade of the CMOP and CPOP, that is, $\left\{ \hat{L}_i, \hat{R}_i, \hat{f}_i, i=1,...,M \right\} = \left\{ L_i^*, R_i^*, f_i^*, i=1,...,M \right\}$.

Proof:

By definition, the region S in (12) fully accounts for the set of the CCOP constraints in (10.6)-(10.8), so that the constraint $\vec{L}$ is equivalent to the set of constraints in (10.6)-(10.8). Since these constraints are also accounted for by the CMOP, this implies that S fully characterizes the coupling induced by the

variables $\{L_i, i=1,\ldots,M\}$ between the two sets of constraints in (10.2)-(10.5) and (10.6)-(10.8). Therefore, the claim of *Proposition 1* directly arises by noting that, by definition, $\chi(L_1,\ldots,L_M)$ is the result of the constrained optimization of the objective function in (10.1) over the variables $\{R_i, i=1,\ldots,M\}$, and $\chi(L_1,\ldots,L_M)$ is part of the objective function of the resulting CPOP in (13.1).

■

About the feasibility of the CMOP, the following result holds (see (Cordeschi, Amendola, Shojafar, & Baccarelli, 2014) for the proof).

Proposition 2

Let each function $P_i^{net}(R_i)/R_i$ in (11.1) be continuous and not decreasing for $R_i \geq 0$. Hence, for any assigned vector $\vec{L}$, the following two properties hold:

i. The CMOP in (11.1) is feasible *if and only if* the vector $\vec{L}$ meets the following condition:

$$\sum_{i=1}^{M} L_i \leq \left(R_t (T_t - \Delta)\right)/2; \tag{14}$$

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

$$R_i^*(\vec{L}) = R_i^*(L_i) = \left(2L_i (T_t - \Delta)\right), i = 1,\ldots,M. \tag{15}$$

Being the condition in (14) necessary and sufficient for the feasibility of the CMOP, it fully characterizes the feasible region *S*in (12). This last property allows us to recast the CPOP in the following equivalent form:

$$R_i^*(\vec{L}) = R_i^*(L_i) = \left(2L_i (T_t - \Delta)\right), i = 1,\ldots,M. \tag{16.1}$$

s.t.:

CCOP's constraints in (10-2)-(10-5) and (14) (16.2)

where (see (15)):

$$\chi(L_1,\ldots,L_M) = (T_t - \Delta)\sum_{i=1}^{M} P_i^{net}\left(\frac{2L_i}{T_t - \Delta}\right). \tag{17}$$

Since the constraint in (14) involves only the offered workload, it may be managed as a feasibility condition and this observation leads to the following formal feasibility result (proved in the final *Appendix A* of (Cordeschi et al. 2013)).

Proposition 3

The CCOP in (10) is feasible if and only if the CPOP in (16) is feasible. Furthermore, the following set of *(M+1)* conditions:

$$L_b(i) \leq \Delta f_i^{\max}, i = 1,\ldots,M, \tag{18.1}$$

$$L_{tot} \leq \min\left\{\sum_{i=1}^{M}\left(f_i^{\max}\Delta - L_b(i)\right), \frac{R_t}{2}(T_t - \Delta)\right\}, \tag{18.2}$$

is necessary and sufficient for the feasibility of the CPOP.

About the solution of the CPOP, after denoting by $\pi_i(.)$ the following dummy function:

$$\pi_i(f_i) \triangleq \left(\frac{E_i^{\max}}{f_i^{\max}}\right)\frac{d\Phi_i}{d\eta}\left(\frac{f_i}{f_i^{\max}}\right) + 2k_e f_i, i = 1,\ldots,M, \tag{19}$$

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

$$TH(i) \triangleq 2\left(\partial P_i^{net}(R_i)/\partial R_i\right)\Big|_{R_i=0}, i = 1,\ldots,M. \tag{20}$$

Hence, after indicating by $\left(\partial P_i^{net}(R_i)/\partial R_i\right)^{-1}(y)$ the final 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 (18) be met. Let the $\chi(.)$ function in (17) be strictly jointly convex and let each function: $P_i^{net}(R_i)/R_i, i = 1,\ldots,M,$ be continuous and increasing for $R_i \geq 0$. Hence, the global optimal solution of the CPOP is unique, and it is analytically computable as in:

$$f_i^* = \left[\pi_i^{-1}(2k_e f_i^0 + \nu_i^*\Delta)\right]_{f_i^{\min}}^{f_i^{\max}}, \tag{21.1}$$

$$L_i^* = \mathbf{1}_{[\nu_i^*>0]}\left((f_i^*\Delta - L_b(i))\right) + \mathbf{1}_{[\nu_i^*=0]}\left[\frac{(T_t-\Delta)}{2}\left(\frac{\partial P_i^{net}(R_i)}{\partial R_i}\right)^{-1}\left(\frac{\mu^*}{2}\right)\right]_+. \tag{21.2}$$

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

$$\nu_i^* \triangleq \left[\mu^* - \frac{2\partial P_i^{net}}{\partial R_i}\left(R_i = \frac{2L_i^*}{(T_t-\Delta)}\right)\right]_+. \tag{22}$$

Finally, $\mu^* \in \Re_0^+$ in (21.2) is the unique nonnegative root of the following algebraic equation:

$$\sum_{i=1}^{M} L_i^*(\mu) = L_{tot}, \tag{23}$$

where $L_i^*(.)$ is given by the r.h.s. of (21.2}, with μ^* replaced by the dummy variable μ.

ADAPTIVE IMPLEMENTATION OF THE OPTIMAL SCHEDULER

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

Hibernation effects

Formally speaking, the *i*-th VM is hibernated when $L_i^* = 0$ (i.e., no exogenous workload is assigned to *VM(i)*) and the corresponding processing rate f_i^* is *strictly larger* than the minimum one: $f_i^{\min} \triangleq L_b(i)/\Delta$ requested for processing the background workload $L_b(i)$ (see (10.2)). 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 (21.1) when the following hibernation condition is met:

$$\mu^{*} \le TH(i). \tag{24}$$

Adaptive implementation of the optimal scheduler and convergence properties

From an application point of view, remarkable features of the optimal scheduler of *Proposition 4* are that: *i)* it leads to *distributed* and *parallel* computation (with respect to the *i*-index) of the *3M* variables $\{ f_i^{*}, L_i^{*}, R_i^{*}, i=1,..., M\}$; and, *ii)* its implementation complexity is *fully* independent from the (possibly, very large) size L_{tot} of the offered workload.

Moreover, in Big Data Streaming environments characterized by (possibly, abrupt and unpredictable) time-fluctuations of the offered workload L_{tot} (see section 5), the per-job evaluation and adaptive tracking of the Lagrange multiplier in (23) may be performed by resorting to suitable primal-dual iterates. Specifically, due to the (strict) convexity of the CPOP, Theorem 6.2.6 of (23) guarantees that the Karush-Khun-Tucker (KKT) point of CPOP is unique and coincides with the saddle point of the corresponding Lagrangian function in (A.1) of *Appendix B* (Bazaraa, Sherali, & Shetty, 2006, pp.272-273). Furthermore, the saddle point is the equilibrium point of the equations' system which is obtained by equating to zero the gradients of the Lagrangian function in (A.1) performed with respect to the primal and dual variables (Bazaraa et al. 2006, Th.6.2.6).

Hence, as in (Srikant, 2004, sections 3.4, 3.8; Zaharia et al. 2012), we apply the primal-dual iterative algorithm for computing the saddle-point of the Lagrangian function in (A.1)[2]. Since the gradient of the Lagrangian function in (A.1) with respect the dual variable μ is: $\left(L_{tot} - \sum_{i=1}^{M} L_i \right)$ and the closed-form KKT relationships in (21.1), (21.2), (22) hold, the primal-dual algorithm reduces to the following quasi-Newton iterates:

$$\mu^{(n)} = \left[\mu^{(n-1)} - \alpha^{(n-1)} \left(\sum_{i=1}^{M} L_i^{(n-1)} - L_{tot} \right) \right]_{+}, \tag{25}$$

with $\mu^{(0)} \ge 0, L^{(0)} \ge 0$. In (25), $n \ge 1$ 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 (21.1),(21.2) and (22)):

$$\nu_i^{(n)} \equiv \left[\mu^{(n)} - \frac{2\partial P_i^{net}}{\partial R_i} \left(R_i = \frac{2L_i^{(n-1}}{(T_t - \Delta)} \right) \right]_{+}, \tag{26.1}$$

$$f_i^{(n)} = \left[\pi_i^{-1} (2k_e f_i^{0} + \nu_i^{(n)} \Delta) \right]_{f_i^{\min}}^{f_i^{\max}}, \tag{26.2}$$

$$L_i^{(n)} = \mathbf{1}_{[\nu_i^{(n)} > 0]} \left((f_i^{(n)} \Delta - L_b(i)) \right) + \mathbf{1}_{[\nu_i^{(n)} = 0]} \left[\frac{(T_t - \Delta)}{2} \left(\frac{\partial P_i^{net}(R_i)}{\partial R_i} \right)^{-1} \left(\frac{\mu^{(n)}}{2} \right) \right]_{+}. \tag{26.3}$$

Regarding the asymptotic global convergence to the optimum of the primal-dual iterates in (25), (26), the following formal result holds (see the final *Appendix B* for the proof).

Proposition 5

[2] 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 *simultaneously* and moving towards the saddle-point [23, pp.407-408]

Let the feasibility condition in (18) be met and let $\{\alpha^{(n-1)}\}$ in (25) be positive and vanishing for $n \to \infty$, i.e., $\lim_{n\to\infty} \alpha^{(n-1)} = 0^+$. Hence, the primal-dual iterates in (25), (26) converge to the global optimum for $n \to \infty$, regardless of the starting point $\mu^{(0)} \geq 0, L^{(0)} \geq 0$.

Formally speaking, *Proposition 5* points out that the adaptive version in (25), (26) of the proposed scheduler attains the global convergence to the solving point in *Proposition 4* of the considered optimization problem. The numerical plots of section 5.2 confirm the actual global convergence of the iterates in (25), (26). In principle, the actual choice of $\{\alpha^{(n-1)}\}$ in (25) also impact on the rate of convergence and the tracking capability of the iterates. At this regard, we note that an effective choice for coping with the unpredictable time-variations of the workload offered by Big Data streaming applications is provided by the gradient-descendant algorithm in (Kushner et al. 1995) for the adaptive updating of the step-size in (25). In our framework, this updating reads as in (Kushner et al. 1995, Equation (2.4)):

$$\alpha^{(n)} = \max\left\{0; \min\left\{\beta; \alpha^{(n-1)} - \gamma V^{(n-1)}\left(\sum_{i=1}^{M} L_i^{(n-1)} - L_{tot}\right)\right\}\right\}, \tag{27}$$

where β and γ are positive constants, while $V^{(n-1)}$ is updated as in (Kushner et al. 1995, Equation (2.5)):

$$V^{(n)} = \left(1-\alpha^{(n-1)}\right)V^{(n-1)} - \left(\sum_{i=1}^{M} L_i^{(n-1)} - L_{tot}\right), \tag{28}$$

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 (25), (26) 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 (25), (26) are performed by starting from the optimal resource allocation computed at the previous slot. Then, after attaining the convergence, the iterates stop and the corresponding resource reconfiguration takes place. Hence, since a *single* reconfiguration action is performed during each slot-time T_t, the resulting reconfiguration cost of the adaptive implementation of the scheduler *does not* depend, indeed, on the convergence times and/or trajectories of the iterates (25), (26) (see Fig.3).

PROTOTYPE AND PERFORMANCE EVALUATION

In order to evaluate the per-job average communication-plus-computing energy $\overline{E}_{tot}^{*}$ consumed by the proposed scheduler, we have implemented a prototype of the adaptive scheduler of section 4.2, with paravirtualized Xen 3.3 as VMM and Linux 2.6.18 as guest OS kernel (see Fig.1). The adaptive scheduler has been implemented at the driver domain (i.e., *Dom0*) of the legacy Xen 3.3. Interestingly enough, 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 TCPReno congestion control suite and Xen's I/O buffer management.

The implemented experimental setup comprises four Dell PowerEdge servers, 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 TCPNewReno suite for implementing the needed VM-to-VM transport connections.

Test Workload Patterns

In order to stress the effects of the reconfiguration costs and time-fluctuations of the Big Data workload, as in (Zhou et al. 2013), we begin to model the workload size as an independent identically distributed (i.i.d.) random sequence $\{\overline{L}_{tot}(mT_t), m0,1,\ldots,\}$, whose samples are T_t-spaced apart r.v.'s evenly distributed

over the interval $[\bar{L}_{tot} - a, \bar{L}_{tot} + a]$, with $\bar{L}_{tot}$ = 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* (e.g., the offered workload is of constant size), *1.2*, *1.5*, *1.75* and *2.0*, respectively. About the dynamic settings of $\{f_i^0\}$ in (10.1), at the first round of each batch of the carried out tests, 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 test results properly *account* for the reconfiguration cost in (7). Furthermore, all the reported test results have been obtained by implementing the adaptive version in (25)-(28) of the optimal scheduler, with the duration of each *n*-indexed iterate set to $(T_t/30)$ secs. Finally, the performed tests subsume the power-rate function in (5) at $\alpha = 1.2$, together with the computing energy function in (2) with *c=2.0* and *b=0.5*.

Impact of the VLAN setup and tracking capability of the scheduler

Goal of a first set of tests is to evaluate the effects on the per-job average energy $\bar{E}_{tot}^*$ consumed by the optimal scheduler induced by 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)*, $E_i^{\max} = 60$ *(Joule)*, $\Delta = 0.1$ *(s)*, $RTT_i = 700$ (μs) and L_b*(i)=0*. Afterwards, since the quality of the virtual links (e.g., end-to-end TCP connections) of Fig.1 is captured by the corresponding coefficients Ω_i (see (5)), we have evaluated the average consumed energy $\bar{E}_{tot}^*$ under the following settings: *i)* $\Omega_i = 0.2$ *(W)*; *ii)* $\Omega_i = 0.5$ *(W)*; *iii)* $\Omega_i = [0.5 + 0.25(i-1)]$ *(W)*; and, *iv)* $\Omega_i = [0.5 + 0.5(i-1)]$ *(W)*, *i=1,…,M*. The obtained numerical plots are drawn in Fig.2. As it could be expected, larger Ω_i's penalized the overall energy performance of the emulated VNetDC. Interestingly, since $\bar{E}_{tot}^*$ is, by definition, the *minimum* energy when up to *M* VMs may be turned on, at fixed P_i^{net}'s, $\bar{E}_{tot}^*$ 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 two uppermost plots of Fig.2).

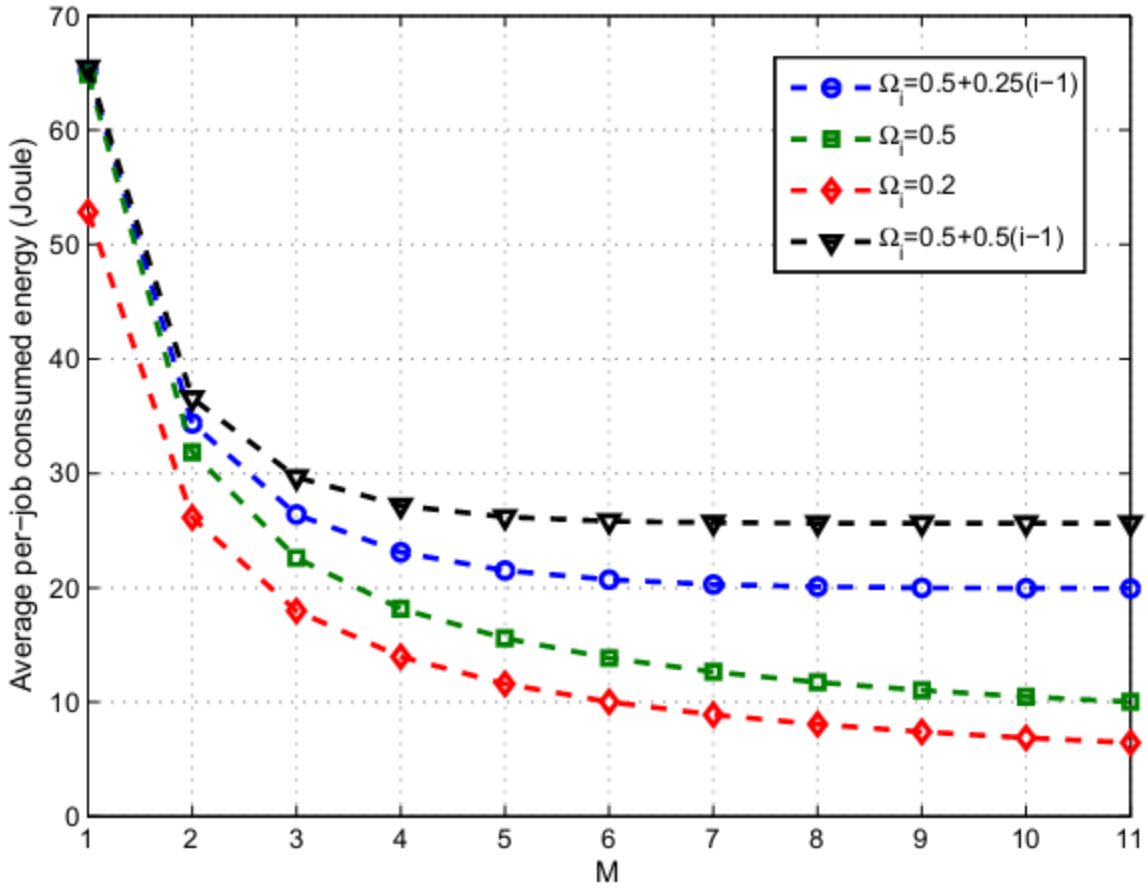

Fig.2: Effects of the link quality on $\overline{E}^{*}_{tot}$ for the application scenario of section 5.2 at *PMR= 1.25.*

Finally, in order to appreciate the sensitivity to the parameters β, γ of the adaptive version in (27), (28) of the implemented scheduler, Fig.3 reports the measured time-behavior of $\mu^{(n)}$ in (25) when the workload offered by the supported Big Data stream application 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 tested at *M= 10* and $\Omega_i = 0.5$ *(W)*. The solid piece-wise linear plot of Fig.3 marks the steady-state optimal values of μ. These optimal values have been obtained by *directly* solving the equation (23) through offline centralized numerical methods. Overall, an examination of the plots of Fig.3 supports two main conclusions. First, the implemented adaptive version of the scheduler quickly reacts to abrupt time-variations of the offered 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 we report the time-evolutions of $\mu^{(n)}$ at $\beta = 0.004, 0.008, 0.01, 0.04$ and $\gamma = 0.4$.

As already noted in (Kushner et al. 1995), also in our framework, the *sensitivity* of the adaptive version of the optimal scheduler on β, γ is *negligible*, at least for values of β, γ in (27) ranging over the intervals $[10^{-3}, 10^{-1}]$ and *[0.1, 0.6]*, respectively.

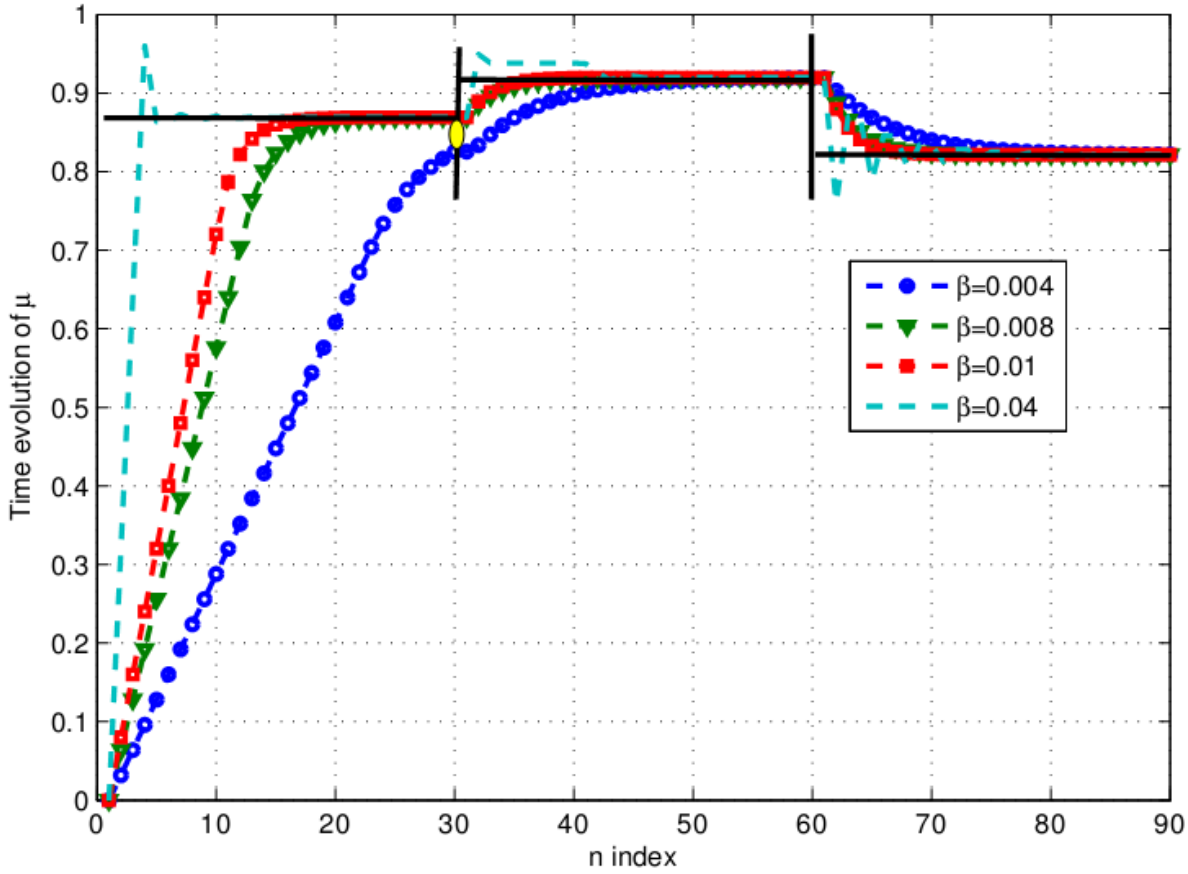

Fig.3: Time evolution (in the *n*-index) of $\mu^{(n)}$ in (25) for the application scenario of section 5.2.

Computing-vs.-communication tradeoff

Intuitively, we expect that small Δ's values give rise to high per-VM computing frequencies (see (10.2)), while too large Δ's induce high end-to-end communication rates (see (10.6)). However, we also expect that, due to the adaptive power-rate control provided by the implemented scheduler (see (21.2), (22)), there exists a broad range of Δ's values that attains an optimized tradeoff. The plots of Fig.4 confirm, indeed, these expectations. They refer to the application scenario of section 5.2 at *M=2, 10*, $\bar{L}_{tot} = 4{,}12$ *(Mbit), a=1,3 (Mbit)* and $\Omega_i = [0.5 + 0.25(i-1)]$ *i=1,…, M (W)*. An examination of these plots supports two main conclusions. First, the energy consumption of the 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 by (18.1)-(18.2) (see the two lowermost curves of Fig.4). Interestingly enough, the increasing behavior of the curves of Fig.4 gives practical evidence that the computing energy dominates the overall energy consumption at vanishing Δ / T_t (see (10.2)), while the network energy becomes substantial at $(\Delta / T_t) \to 1^-$ (see (10.6)).

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-frequency techniques; and, *ii)* the energy penalty induced by the frequency-switching over a finite number Q of allowed per-VM processing rates (see (7)). For this

purpose, the same operating scenario of the above section 5.2 has been considered at $k_e = 5\times10^{-4}$ *(Joule/(MHz)^2)* and $\Omega_i = [0.5 + 0.25(i-1)]$ *(W) i=1,…,M*. The energy curves obtained at: *i)* $Q = +\infty$ (i.e., continuous computing rates); *ii) Q=6* (i.e., discrete computing rates with six allowed computing frequencies 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.5. Interestingly, we have ascertained that the not monotonic behavior of the uppermost curve of Fig.5 is the result of two effects that are dominating at *Q=2*. First, at *Q=2*, each active VM is forced to operate at $f_i^{\max}$, so that the increment in the computing energy induced by the activation of an additional VM scales up as $E_i^{\max}$. Second, at *Q=2*, the energy overhead in (8) 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.5 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.5 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.5 is limited up to *15%*.

Performance comparisons under synthetic time-uncorrelated workload traces

Testing the sensitivity of the performance of the implemented scheduler on the PMR of the offered workload is the goal of the tests of this sub-section. Specifically, they aim at unveiling the impact of the PMR of the offered workload 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 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 implemented the communication-computing platform of section 5.2 at $\Omega_i = 0.9$ *(W)*, 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 (Baliga et al. 2011; Tamm et al. 2010), where, by design, a *fixed* number M_s of VMs constantly runs at the maximum processing rate $f^{\max}$. The goal is to constantly provide the exact computing capacity needed for satisfying the peak workload $L_{tot}^{\max} \triangleq \bar{L}_{tot} + a$ *(Mb)*. Although the resulting static scheduler *does not* experience reconfiguration costs, it induces overbooking of the computing resources. Hence, the per-job average communication-plus-computing energy consumption $\bar{E}_{tot}^{(STAS)}$ *(Joule)* of the STAS gives a benchmark for numerically evaluating the energy savings actually attained by dynamic schedulers.

About the simulated SES, it exploits (by design) *perfect* future workload information over a time-window of size $I \geq 2$ (measured in multiple of the slot period T_t), in order to perform *off-line* resource provisioning at the minimum reconfiguration cost. Formally speaking, the SES implements the solution of the following sequential minimization problem:

$$\min_{\{R_i(m),\, f_i(m),\, L_i(m)\}} \sum_{m=1}^{I} \sum_{i=1}^{M} \left\{ \Phi_i \left(\frac{f_i(m)}{f_i^{\max}} \right) E_i^{\max} + k_e \left(f_i(m) - f_i(m-1) \right)^2 + 2P_i^{net}(R_i(m)) \left(\frac{L_i(m)}{R_i(m)} \right) \right\} \tag{29}$$

under the constraints in (10.2)-(10.8). We have evaluated the solution of the above sequential problem by resorting to numerical computing tools.

Table 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 Δ *=0.05 (s), 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 2 have

been evaluated by forcing the sequential and the proposed schedulers to utilize the *same* number of VMs 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 2 leads to four main conclusions.

First, the average energy savings 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.4. 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% (Cordeschi et al. 2014).

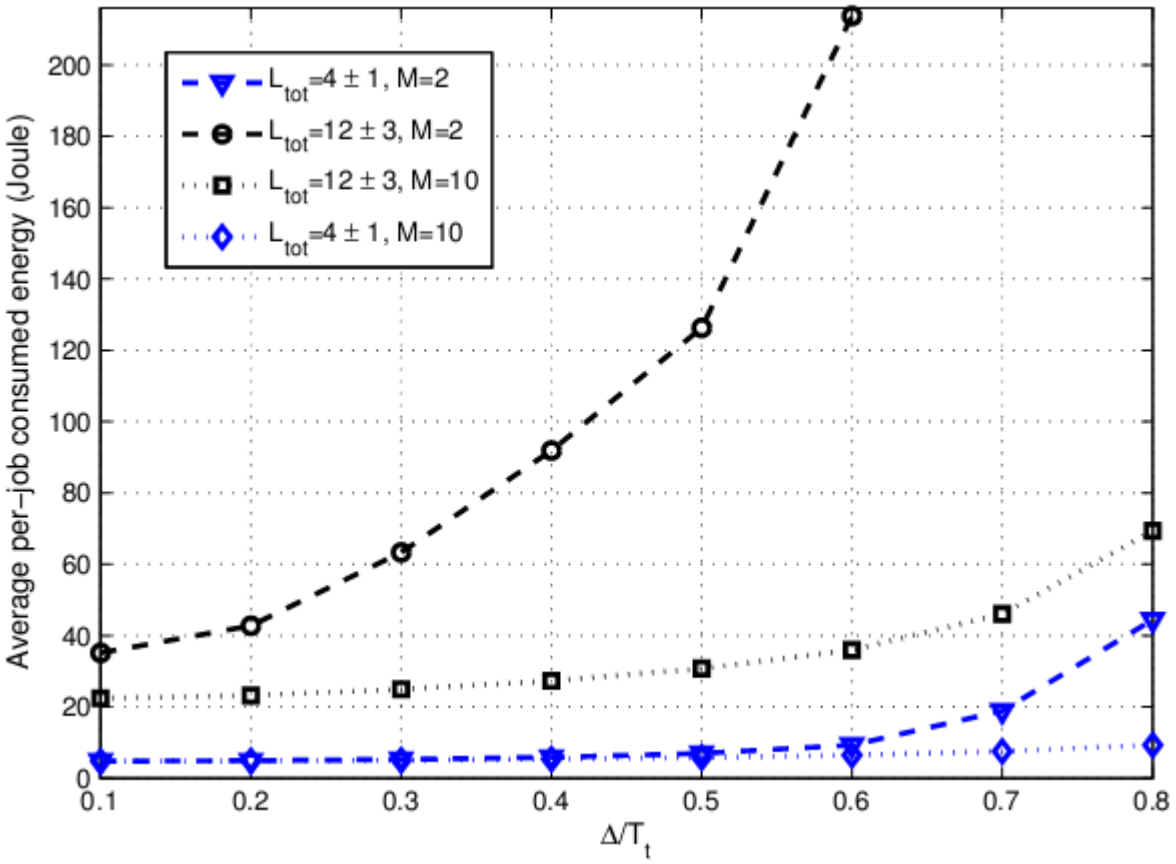

Fig.4: Impact on $\overline{E}_{tot}^{*}$ of the computing-vs.-communication delay tradeoff.

Table 2: Average energy reductions attained by the proposed and the sequential schedulers one over the static one at k_e=0.005 *(Joule/(MHz)^2)*, and $f_i^{\max} = 80$ *(Mbit/s)*.

PMR	**Proposed scheduler at** $\Delta = 0.05(s)$	**SES at** $\Delta = 0.05(s)$	**Proposed scheduler at** $\Delta = 0.1(s)$	**SES at** $\Delta = 0.1(s)$
1	0%	0%	0%	0%
1.25	51%	54%	65%	68%
15	45%	48%	62%	65%
1.75	63%	67%	57%	62%
2	57%	62%	50%	56%

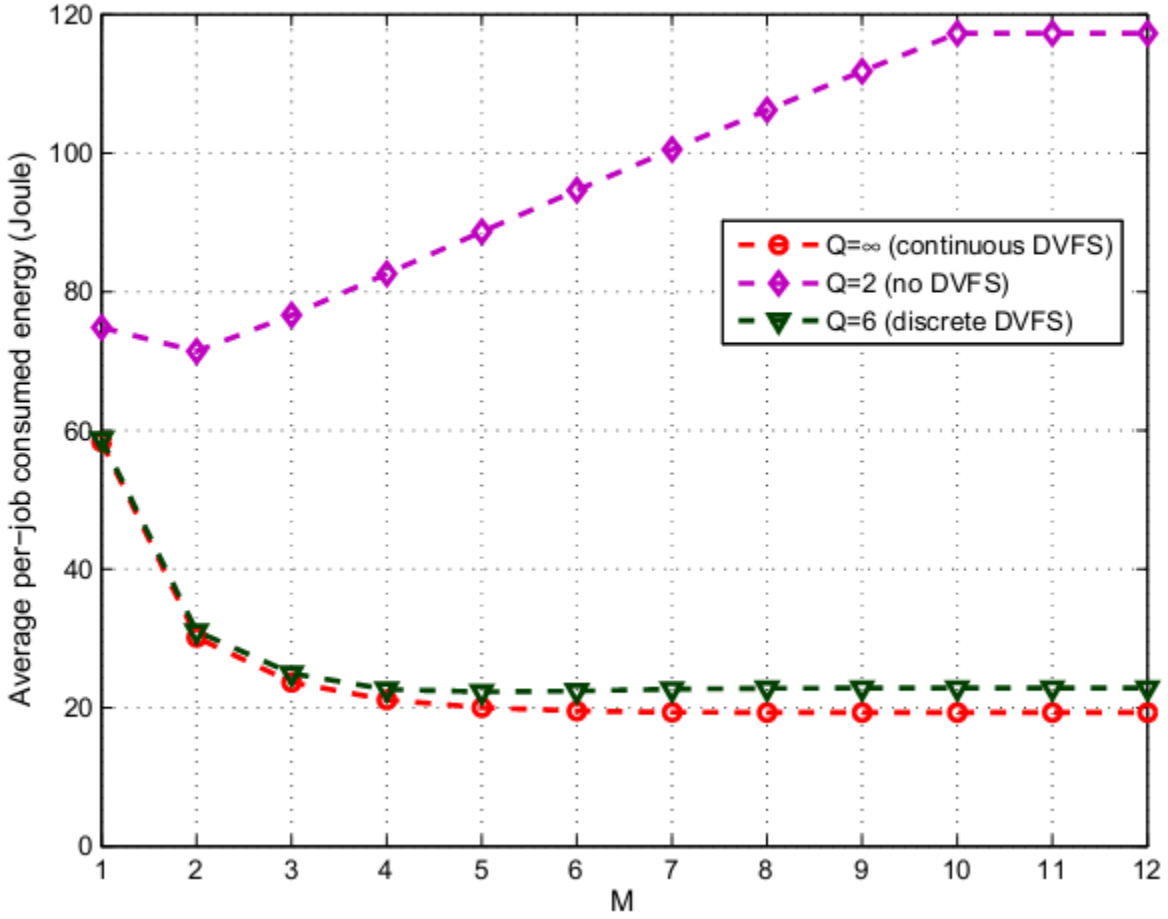

Fig.5: Effects of continuous/discrete computing rates on the energy performance of the platform of Fig. 1. The frequency-switching energy penalty in (7) is explicitly accounted for.

Performance comparisons under time-correlated real-world workload traces

These conclusions are confirmed by the numerical results of this subsection, that refer to the *real-world* (e.g., not synthetic) workload trace of Fig.6. This is the same real-world workload trace considered in Fig.14.a of (Urgaonkar, Pacifici, Shenoy, Spreitzer, & Tantawi, 2007). The numerical tests of this subsection refer to the communication-computing infrastructure of Section 5.5 at k_e*=0.5 (Joule/ (MHz)* ^{2}*)* and Δ *=1.2 (s)*. Furthermore, in order to maintain the peak workload still fixed at *16 (Mbit/slot)*, we assume that each arrival of Fig.6 carries out a workload of *0.533(Mbit)*.

Since the (numerically evaluated) PMR of the workload trace of Fig.6 is *limited* up to *1.526*, and the corresponding time-covariance coefficient ρ is *large* and approaches *0.966*, the workload trace of Fig.6 is *smoother* (e.g., it exhibits *less* time-variations) than those previously considered in Section 5.5. 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 5.5 for the case of time-uncorrelated workload.

However, we have tested that, even under the (strongly) time-correlated workload trace of Fig.6, the average energy reduction of the proposed scheduler over the static one still approach 40%, while the corresponding average energy saving of the sequential scheduler over the proposed one remains limited up to 5.5%-5%.

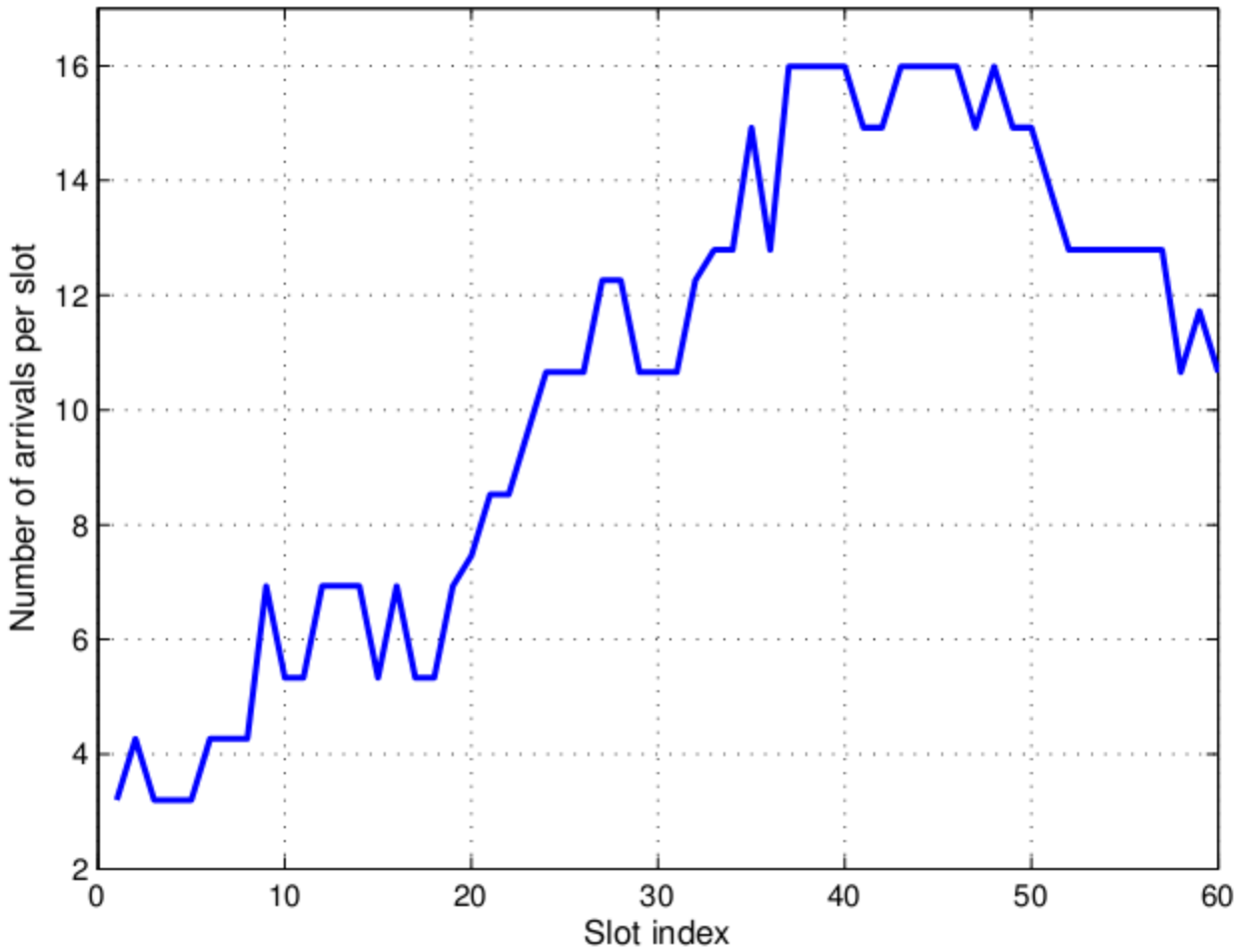

Fig.6: Measured workload trace (Urgaonkar et al. 2007, Fig.14.a). The corresponding PMR and covariance coefficient ρ equate 1.526 and 0.966, respectively.

ONGOING DEVELOPMENTS AND HINTS FOR FUTURE RESEARCH

The focus of this contribution is on the adaptive minimum-energy reconfiguration of data center resources under hard latency constraints. However, when the focus shifts to delay-tolerant Internet-assisted mobile applications (e.g., mobile Big Data applications), the current work may be extended along three main directions of potential interest.

First, being the VNetDC considered here tailored on hard real-time applications, it subsumes that a single job accedes the VNetDC during each slot, so to avoid at all random queue effects. However, under soft latency constraints, the energy efficiency of the VNetDC could be, in principle, improved by allowing multiple jobs to be temporarily queued at the Middleware layer. Hence, guaranteeing optimized energy-vs.-queue delay trade-offs under soft latency constraints are, indeed, a first research topic.

Second, due to the considered hard real-time constrains, in our framework, the size L_{tot} of the incoming job is measured at the beginning of the corresponding slot and, then, it remains constant over the slot duration T_t. As a consequence, the scheduling policy considered here is of clairvoyant-type, and this implies, in turn, that migrations of VMs are not to be considered. However, under soft delay constraints, intra-slot job arrivals may take place. Hence, the optimal resource provisioning policy could be no longer of clairvoyant-type, so that live migrations and VMs replacement could become effective means to further reduce the energy costs. The development of adaptive mechanisms for planning at runtime minimum-energy live migrations of VMs is a second research topic of potential interest.

Third, emerging mobile BDSC applications require that the information processed by data centers is timely delivered to the requiring clients through TCP/IP mobile connections (Cugola et al. 2012). In this application scenario, the energy-efficient adaptive management of the delay-vs.-throughput trade-off of TCP/IP mobile connections becomes an additional topic for further research.
Lastly, the final practical goal of these research lines is to implement in SW a revised VMM kernel, that: *(i)* operates at Dom0 of the legacy Xen 3.3 suite; and, *(ii)* is capable to jointly manage the computing/communication resources hosted onto the data center, together with the data center-to-mobile client TCP/IP connections.

CONCLUSIONS

In this contribution, we developed the optimal scheduler for the joint adaptive load balancing and provisioning of the computing rates, communication rates and communication powers in energy-efficient virtualized NetDCs which support real-time Big Data streaming services. Although the resulting optimization problem is inherently nonconvex, we unveil and exploit its loosely coupled structure for attaining the analytical characterization of the optimal solution. The carried out performance comparisons and sensitivity tests highlight that the average energy savings provided by our implemented scheduler over the state-of-the-art static one may be larger than 60%, even when the PMR of the offered workload is limited up to 2 and the number Q of different processing rates equipping each VM is limited up to 4-5. Interestingly, the corresponding average energy loss of our scheduler with respect to the corresponding sequential one is limited up to 4%-6%, especially when the offered workload exhibits not negligible time-correlation.

APPENDIX A: DERIVATIONS OF EQUATIONS (21.1)-(23)

Being the constraint in (10.7) already accounted for by the feasibility condition in (18.2), without loss of optimality, we may directly focus on the resolution of optimization problem in (16) under the constraints in (10.2)-(10.5). Since this problem is strictly convex and all its constraints are linear, the Slater's qualification conditions hold (Bazaraa et al. 2006, Chap.5, 23), so that the KKT conditions (Bazaraa et al. 2006, Chap.5) 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 (17) is increasing for $L_i \geq 0$, so that, without loss of optimality, we may replace the equality constraint in (10.3) 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

$$\ell(\{L_i,\, f_i, \nu_i, \mu\}) = Z(\{L_i,\, f_i\}) \sum_{i=1}^{M} \nu_i \left(L_i - f_i \Delta + L_b(i)\right) + \mu \left(L_t - \sum_{i=1}^{M} L_i \right), \tag{A.1}$$

where $Z(L_i,\, f_i)$ indicates the objective function in (16.1), ν_i's and μ are nonnegative Lagrange multipliers and the box constraints in (10.4), (10.5) are managed as implicit ones. The partial derivatives of $\ell(.;.)$ with respect to f_i, L_i are given by

$$\frac{\partial \ell(.)}{\partial f_i} = \frac{E_i^{\max}}{f_i^{\max}} \frac{\partial \Phi_i(f_i / f_i^{\max})}{\partial \eta_i} + 2k_e (f_i - f_i^{0}) - \nu_i \Delta, \tag{A.2}$$

$$\frac{\partial \ell(.)}{\partial L_i} = 2 \frac{\partial P_i^{net}}{\partial R_i} \left(\frac{2L_i}{T_t - \Delta} \right) + \nu_i - \mu, i = 1, \ldots, M, \tag{A.3}$$

while the complementary conditions (Bazaraa et al. 2006, Chap.4) associated to the constraints present in (A.1) read as in

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

Hence, by equating (A.2) to zero, we directly arrive at (21.1), that also account for the box constraint: $f_i^{\min} \le f_i \le f_i^{\max}$ 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 (23). 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 (10.2) is bound (see (A.4)), so that we have:

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

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

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

Since L_i^* must fall into the closed interval $[0, \Delta f_i^* - L_b(i)]$ for feasible CPOPs (see (10.2), (10.5)), 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 $\left[\partial \ell / \partial L_i\right]_{L_i=0} \ge 0$. 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_i^*:

$$\mu_i^* \le 2\left[\partial P_i^{net}(R_i)/\partial R_i\right]_{R_i=0} \equiv TH(i), \quad at \quad \nu_i^* = L_i^* = 0, i = 1,\ldots,M, \tag{A.7}$$

Passing to consider the case of $\nu_i^* = 0$ 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 (Bazaraa et al. 2006, Chap.4). Hence, the application of this condition leads to the following expression for the optimal L_i^* (see (A.3)):

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

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

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

so that equation (21.2) directly arises from (A.5), (A.9). Finally, after observing that ν_i^* cannot be negative by definition, from (A.6) we obtain (22), where the projector operator accounts for the nonnegative value of ν_i^*. 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 sections 3.4, 8.2 in (Srikant, 2004). Specifically, after noting that the feasibility and strict convexity of the CPOP in (16) guarantees the existence and uniqueness of the equilibrium point of the iterates in (25) and (26), we note that *Proposition 4* assures that, for any assigned $\mu^{(n)}$ and $\left\{L_i^{(n-1)}\right\}$, equations (26.1)-(26.3) give the corresponding (unique) optimal values of the primal and dual variables $\left\{f_i^{(n)}, \nu_i^{(n)}, L_i^{(n)}\right\}$. Hence, it suffices to prove the global asymptotic convergence of the iteration in (25).

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 $\left\{L_i^{(n-1)}\right\} = \{L_i^*\}$ [3]. Hence, since $U^{(n-1)}(.)$ in (B.1) is also radially unbounded (that is, $U^{(n-1)}(.) \to \infty$ as $|| \sum_{i=1}^{M} L_i^{(n-1)} - L_{tot} || \mapsto \infty$), we conclude that (B.1) is an admissible Lyapunov's function for the iterate in (25). Hence, after posing $U^{(n)}\left(\left\{L_i^{(n)}\right\}\right) = U^{(n)} \triangleq \left[\sum_{i=1}^{M} L_i^{(n)} - L_{tot}\right]^2$, according to the Lyapunov's Theorem (Srikant, 2004, section 3.10), we must prove that the following (sufficient) condition for the asymptotic global stability of (25) is met:

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

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 (25)): $\mu^{(n)} < \mu^{(n-1)}$, that, in turn, leads to (see (26.3)): $L_i^{(n)} < L_i^{(n-1)}$, for any *i=1,…,M*[4]. Therefore, in order to prove (B.2), it suffices to prove that the following inequality holds for large *n*:

$$\left(\sum_{i=1}^{M} L_i^{(n)} - L_{tot}\right) \geq 0. \tag{B.4}$$

---

[3] *Proposition 4* proves that, for any assigned $\mu^{(n)}$, the relationship in (26.3) 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,…,M*.

[4] About this point, the formal assumption of section 2 guarantees that: *i)* $\pi^{-1}(.)$ in (26.2) is continuous and 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 (26.3) is continuous and strictly increasing in $\mu^{(n)}$; and, *iii)* $\nu_i^{(n)}$ in (26.1) is continuous for $\mu^{(n)} \geq 0$ and 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,…,M*.

To this end, we observe that: *i)* $\{\alpha(n-1)\}$ in (25) vanishes (by assumption) for $n \to \infty$; and, *ii)* $L_i^{(n)}$ is limited up to $\Delta f_i^{\max}$, for any *i=1,…, M* (see the constraints in (10.2), (10.4)).

As a consequence, the difference: $(\mu^{(n)} - \mu^{(n-1)})$ may be done vanishing (i.e., as small as desired) as $n \to \infty$. Hence, after noting that the functions in (25), (26) are continuous by assumption (see the footnote 4), 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*. From an application point of view, we have numerically ascertained that, in our setting, the step-size sequence in (27) leads to the global convergence in the steady-state and allows fast tracking in the transient-state.

REFERENCES

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

Baliga, J., Ayre, R.W.A., Hinton, K., &Tucker, R.S. (2011). Green Cloud Computing: Balancing Energy in Processing, Storage and Transport. *Proc. of IEEE, 99(1),* 149-167.

Mishra, A., Jain, R., & Durresi, A. (2012). Cloud Computing: Networking and Communication challenges. *IEEE Comm. Magazine*, 24-25.

Azodomolky, S., Wieder, P., & Yahyapour, R. (2013). Cloud computing networking: Challanges and Opportunities for Innovations. *IEEE Comm. Magazine*, 54-62.

Warneke, D., & Kao, O. (2011). Exploiting Dynamic Resource Allocation for efficient parallel data processing in the Cloud. *IEEE Tr. on Paral. and Distr. Systems, 22(6),* 985-997.

Tamm, O., Hersmeyer, C., Rush, A.M. (2010). Eco-sustainable system and network architectures for future transport networks. *Bell Labs.Techn.J., 14*, 311-327.

Liu, J. Zhao, F. Liu, X. & He, W. (2009). Challenges towards elastic power management in internet data centers. *Proc. on IEEE Conf. Distr. Comput. Syst. Workshops*, Los Alamitos (USA), 65-72: IEEE.

Kim, K.H., Beloglazov, A., & Buyya, R. (2009) Power-aware provisioning of Cloud resources for real-time services. *Proc.of ACM MGC'09*: ACM.

Scheneider, S., Hirzel, M., & Gedik, B. (2013). Tutorial: stream processing optimizations. *ACM DEBS*, 249-258.

Lu, T., Chen, M., & Andrew, L.L.H. (2012). Simple and effective dynamic provisioning for power-proportional data centers. *IEEE Transactions on Parallel and Distributed Systems, 24(6),* 1161-1171.

Mathew, V., Sitaraman, R., & Rowstrom, A. (2012). Energy-aware load balancing in content delivery networks. *INFOCOM, 954-962:* IEEE.

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

Kushner, H.J. & Yang, J. (1995). Analysis of adaptive step-size SA algorithms for parameter tracking. *IEEE Transactions on Automatic Control, 40(8),* 1403-1410.

Kurose, J.F., & Ross, K.W. (2013) *Computer Networking - A top-down approach featuring the Internet*, 6th Ed.: Addison Wesley.

Almeida, J., Almeida, V., Ardagna, D., Cunha, I., & Francalanci, C. (2010) Joint admission control and resource allocation in virtualized servers. *Jour. Paral. Distr. Computing*, 70, 344-362.

Balter, M.H. (2013). Performance modeling and design of computer systems: Cambridge Press.

Padala, P., You, K.Y., Shin, K.G., Zhu, X., Uysal, M., Wang, Z.… Merchant, M. (2009). Automatic control of multiple virtualized resources. *Proc.of EuroSys.*

Kusic, D., & Kandasamy, N. (2009). Power and performance management of virtualized computing environments via look-ahead control. *Proc. of ICAC*.

Govindan, S., Choi, J., Urgaonkar, B., Sasubramanian, A., & Baldini, A., (2009). Statistical profiling-based techniques for effective power provisioning in data centers. *Proc of EuroSys*.

Zhou, Z., Liu, F., Jin, H., Li, B., & Jiang, H. (2013). On arbitrating the power-performance tradeoff in SaaS clouds. *Proceedings of IEEE INFOCOM*, 872-880: IEEE.

Lin, M., Wierman, A., Andrew, L., & Thereska, E. (2011) Dynamic right-sizing for power-proportional data centers. *INFOCOM*:IEEE.

Neely, M.J., Modiano, E., & Rohs, C.E. (2003). Power Allocation and Routing in Multi Beam Satellites with Time-varying channels. *IEEE/ACM Tr. on Networking, 19(1),* 138-152.

Bazaraa, M.S., Sherali, H.D., Shetty, C.M. (2006). *Nonlinear Programming*, 3rd Ed.: Wiley.

Koller, R., Verma, A., Neogi, A. (2010) WattApp: an application aware power meter for shared data centers. *ICAC'10:* IEEE.

Portnoy, M. (2012). *Virtualization Essentials*: Wiley.

Chiang, M., Low, S.H., Calderbank, A.R., & Doyle, J.C. (2007). Layering as optimization decomposition: a mathematical theory of network architectures. *Proc. of IEEE, 95(1),* 255-312.

Chen, J.J., & Kuo, T.W. (2005). Multiprocessor Energy-efficient Scheduling for real-time tasks with different power characteristics. *ICCP'05*, 13-20: IEEE.

Laszewski, G., Wang, L., Young, A.J., He, X., (2009). Power-aware scheduling of Virtual Machines in DVFS-enabled Clusters. *Proceeding of CLUSTER'09*: IEEE.

Neumeyer, L., Robbins, B., & Kesari, A. (2010). S4: Distributed stream computing platform", In Intl. Workshop on Knowledge Discovery Using Cloud and Distributed Computing Platforms. *ICDMW '10,* 170-177: IEEE.

Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., & Tantawi, A. (2007). Analytic modeling of multitier Internet applications. *ACM Tr. on the Web*, 1(1).

Kim, K.H., Buyya, R., & Kim, J. (2007). Power aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled clusters. *IEEE International Symposium of CCGRID*, 541-548: IEEE.

Li, K. (2008). Performance analysis of Power-aware Task scheduling Algorithms on Multiprocessor Computers with Dynamic Voltage and Speed. *IEEE Tr. On Par. Distr. Systems, 19(11),* 1484-1497.

Srikant, R. (2004). *The Mathematics of Internet Congestion Control*, Birkhauser.

Zaharia, M., Das, T., Li, H., Shenker, S., & Stoica, I. (2012). Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. *Hotcloud*.

Gulati, A., Merchant, A., & Varman, P.J. (2010). mClock: Handling throughput variability for Hypervisor IO scheduling. *OSDI'10*.

Ballami, H., Costa, P., Karagiannis, T., & Rowstron, A. (2011). Towards predicable datacenter networks. *SIGCOMM '11*.

Greenberg, A. et al. (2011). VL2: a scalable and flexible data center Network. *Comm. Of the ACM, 54(3),* 95-104.

Guo, C. et al. (2010). SecondNet: A Data Center Network virtualization architecture with bandwidth guarantees. *ACM CoNEXT:* ACM.

Xia, L., Cui, Z., Lange, J. (2012). VNET/P: bridging the Cloud and High Performance Computing through fast Overaly networking. *HPDC:* IEEE.

Cordeschi, N., Amendola, D., Shojafar, M., & Baccarelli, E. (2014) Performance evaluation of primary-secondary reliable resource-management in vehicular networks. *IEEE PIMRC 2014*, Washington, DC, USA, Accepted.

Wang, L., Zhang, F., Aroca, J.A., Vasilakos, A.V., Zheng, K. , Hou, C., …, Liu, Z. (2014). Green DCN: A general framework for achieving energy efficiency in Data Denter Networks. *IEEE JSAC, 32(1).*

Gandhi, A., Balter, M.H, & Adam, I. (2010). Server farms with setup costs. *Perf. Evaluation .11(67),* 1123-1138.

Loesing, S., Hentschel, M., & Kraska, T. (2012). Storm: an elastic and highly available streaming service in the cloud. *EDBT-ICDT '12*, 55-60: IEEE.

Qian, Z., He, Y., Su, C., Wu, Z., Zhu, H., Zhang, T. (2013). TimeStream: reliable stream computation in the cloud. *in EuroSys,* 1-14.

Kumbhare, A. (2014). PLAstiCC: predictive look-ahead scheduling for continuous data flaws on clouds. *CCGRID:* IEEE.

Kansal, A., Zhao, F., Liu, J., Kothari, N., Bhattacharya, A. (2010). Virtual Machine Power Metering and Provisioning. *SoCC '10,* 39-50: IEEE.

Ge, R., Feng, X., Feng, W., Cameron, K.W. (2007). CPU MISER: A Performance-Directed, Run-Time System for Power-Aware Clusters. *IEEE ICPP07,* 1-8: IEEE.

Vasudevan, V., Phanishayee, A., & Shah, H. (2009). Safe and effective fine-grained TCP stream retransmissions for datacenter communication. *ACM SIGCOMM*, 303-314: ACM.

Alizadeh, M., Greenberg, A., Maltz, D.A., & Padhye, J. (2010). Data center TCP (DCTCP). *ACM SIGCOMM:* ACM.

Das, T., & Sivalingam, K.M. (2013). TCP improvements for data center networks", *Communication Systems and Networks (COMSNETS)*, 1-10.

Jin, S., Guo, L., Matta, I., & Bestavros, A. (2003). A spectrum of TCP-friendly window-based congestion control algorithms. *IEEE/ACM Transactions on Networking (TON), 11(3),* 341-355.

Liu, Q., Zhou, S., Giannakis, G.B. (2004). Cross-Layer combining of adaptive modulation and coding with truncated ARQ over wireless links. *IEEE Transactions on Wireless Communications, 3(5),* 1746-1755.

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

Baccarelli, E., Cordeschi, N., & Patriarca, T. (2012). QoS Stochastic Traffic Engineering for the wireless support of real-time streaming applications. *Computer Networks, 56(1),* 287-302.

Cordeschi, N., Shojafar, M., & Baccarelli, E. (2013). Energy-saving self-configuring networked data centers, *Computer Networks, 57(17),* 3479-3491.

Baccarelli, E., & Biagi, M. (2003). Optimized Power Allocation and Signal Shaping for Interference-Limited Multi-antenna 'Ad Hoc' Networks. *Springer, Personal Wireless Communications*, 138-152.

Baccarelli, E., Biagi, M., Pelizzoni, C., & Cordeschi, N. (2007). Optimized power allocation for multiantenna systems impaired by multiple access interference and imperfect channel estimation. *IEEE Transactions on Vehicular Technology, 56(5),* 3089-3105.

Stoess, J., Lang, C., & Bellosa, F. (2007). Energy Management for Hypervisor-Based Virtual Machines. *USENIX Annual Technical*, 1-14.

Greenberg, A., Hamilton, J., & Maltz, D.A. (2009). The cost of a cloud: research problems in data center networks. *ACM SIGCOMM, 39(1),* 68-73.

KEY TERMS AND DEFINITIONS

Network virtualization: Network virtualization is categorized as either external virtualization, combining many networks or parts of networks into a virtual unit, or internal virtualization, providing network-like functionality to software containers on a single network server.

Stream Computing: A high-performance computer system that analyzes multiple data streams from many sources live. The word stream in stream computing is used to mean pulling in streams of data; processing the data and streaming it back out as a single flow.

Data center: A data center is a facility used to house computer systems and associated components, such as telecommunications and storage systems.

Virtual Machine Manager (VMM): Virtual Machine Manager allows users to create, edit, start and stop VMs, view and control of each VM's console and see performance and utilization statistics for each VM.

Dynamic Voltage and Frequency Scaling (DVFS): 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.

Load Balancing: algorithms seek to distribute workloads across a number of servers in such a manner that the average time taken to complete execution of those workloads is minimized.

Cloud brokers: They accept risk associated with reserving dynamically priced resources in return for charging higher but stable prices.

Reconfiguration cost: reconfiguration cost measured at the Middleware layer of VNetDc and refer to the cost of changing the frequency of each VM while it loaded for new workload compare to its frequency for the previous workload.