Energy-efficient Adaptive Resource Management for Real-time Vehicular Cloud Services

Mohammad Shojafar, Member, IEEE, Nicola Cordeschi and Enzo Baccarelli

Abstract—Providing real-time cloud services to Vehicular Clients (VCs) must cope with delay and delay-jitter issues. Fog computing is an emerging paradigm that aims at distributing small-size self-powered data centers (e.g., Fog nodes) between remote Clouds and VCs, in order to deliver data-dissemination real-time services to the connected VCs. Motivated by these considerations, in this paper, we propose and test an energy-efficient adaptive resource scheduler for Networked Fog Centers (NetFCs). They operate at the edge of the vehicular network and are connected to the served VCs through Infrastructure-to-Vehicular (I2V) TCP/IP-based single-hop mobile links. The goal is to exploit the locally measured states of the TCP/IP connections, in order to maximize the overall communication-plus-computing energy efficiency, while meeting the application-induced hard QoS requirements on the minimum transmission rates, maximum delays and delay-jitters. The resulting energy-efficient scheduler jointly performs: (i) admission control of the input traffic to be processed by the NetFCs; (ii) minimum-energy dispatching of the admitted traffic; (iii) adaptive reconfiguration and consolidation of the Virtual Machines (VMs) hosted by the NetFCs; and, (iv) adaptive control of the traffic injected into the TCP/IP mobile connections. The salient features of the proposed scheduler are that: (i) it is adaptive and admits distributed and scalable implementation; and, (ii) it is capable to provide hard QoS guarantees, in terms of minimum/maximum instantaneous rates of the traffic delivered to the vehicular clients, instantaneous rate-jitters and total processing delays. Actual performance of the proposed scheduler in the presence of: (i) client mobility; (ii) wireless fading; and, (iii) reconfiguration and consolidation costs of the underlying NetFCs, is numerically tested and compared against the corresponding ones of some state-of-the-art schedulers, under both synthetically generated and measured real-world workload traces.


1 INTRODUCTION AND REFERENCE VEHICULAR SCENARIO

Vehicular Cloud Computing (VCC) is a promising paradigm that aims at merging Mobile Cloud Computing (MCC) and Vehicular Networking (VN), in order to give rise to integrated communication-computing platforms [1]. While safety applications have been the prime motivation behind VN, the final target of VCC is to shrink the spectrum of the supported services, in order to include the emerging Future Internet Applications [2]. These are infotainment applications (such as, for example, Netflix and VTube), that exploit Infrastructure-to-Vehicle (I2V) links for providing data dissemination and content delivery services to connected Vehicular Clients (VCs) [3]. Being of streaming type, these applications are delay and delay-jitter sensitive [4], so that the large delay and delay-jitter typically induced by Wide Area Networks (WANs) preclude the direct utilization of remote clouds as servers (see, for example, Chapter 8 of [5]).

Hence, a new paradigm is emerging to meet the QoS requirements of vehicular Future Internet Applications, generally referred to as Fog Computing (shortly, Fog) [6], [7]. In vehicular scenarios, Fog nodes are virtualized networked data centers (e.g., NetFCs), which are hosted by Road Side Units (RSUs) at the edge of the vehicular network, so to give rise to a three-tier Cloud-Fog-VC hierarchical architecture (see Fig.1a). Similar to Cloud centers, Fog nodes provide bandwidth, compute support, storage and application services to the connected VCs on a per-VC basis. However, the distinguishing features of the Fog paradigm are the following four ones [6], [7]: (i) Edge location and location awareness: being deployed in proximity of the VCs, Fog nodes may efficiently exploit the awareness of the connection states for the support of delay and delay-jitter sensitive applications, such as VTube-like interactive applications [3]; (ii) Geographical deployment: Fog nodes support services which demand for widely distributed geographical deployments, as, for example, vehicular video streaming services; (iii) Predominance of wireless access and support for the mobility: Fog nodes may exploit Infrastructure-to-Vehicular (I2V) single-hop IEEE 802.11-like wireless links for data dissemination; and, (iv) Low energy consumption: since, in vehicular applications, Fog nodes are hosted by RSUs, they are equipped with capacity-limited batteries, that should be (hopefully) re-charged through solar cells. Hence, attaining energy efficiency is a central target in the Fog paradigm [7].

1.1 Technical contribution overview and main idea of the paper

Motivated by the aforementioned considerations, we focus on the vehicular scenario of Fig.1a. In this scenario, Fog-equipped RSUs broadcast locally processed data to smartphone-equipped VCs through point-to-point TCP/IP-based I2V connections, which rely, in turn, on one-hop IEEE802.11-type wireless links [8]. Regarding the technical contribution of the work and the novelty of the pursued approach, we point out that the resource management prob-
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2551747, IEEE Transactions on Cloud Computing

lemed afforded in this paper jointly embraces (see Fig.1b): (i) the adaptive control of the input and output traffic flows by coping with the random (possibly, unpredictable) fluctuations of the input traffic to be processed and the states of the utilized TCP/IP connections; (ii) the adaptive reconfiguration of the per-VM task sizes and processing rates; (iii) the adaptive reconfiguration of the intra-Fog per-link communication rates; and, (iv) the adaptive consolidation of the virtualized physical servers hosted by the Fog nodes. The pursued objective is the minimization of the average communication-plus-computing energy wasted by each TCP/IP-mobile connection and its minimum transmission rate; and, (ii) the per-connection maximum computing-plus-communication delay and jitter induced by the NetFC platform of Fig.1b. Due to the possible high number of involved VCs, RSUs and hosted VMs, we also require that the resulting adaptive scheduler allows distributed and scalable implementation. Overall, due to its self-sensing and self-configuring capabilities and distributed nature, we anticipate that the proposed resource scheduler is an instance of the emerging paradigm of the (networked) Cognitive Computing [3].

1.2 Related work and outline of the paper

An examination of the (recently published) related work supports the conclusion that the adaptive optimization of the aforementioned scheduling actions has been till now pursued by following three main parallel (e.g., disjoint) research directions, so that the proposed cognitive computing-inspired joint approach seems to be somehow new.

Specifically, a first research direction focuses on task-scheduling algorithms for the minimization of the energy in reconfigurable data centers that serve static clients [9], [10], [11]. For this purpose, in [9], an energy-efficient greedy-type scheduling algorithm is presented. It maps jobs to VMs and, then, maps VMs to Dynamic Voltage Frequency Scaling (DVFS)-enabled servers under upper bounds on the allowed per-job execution time. Interestingly, the resulting scheduler relies on a suitable version of the First-Fit-Decreasing (FFD) heuristic, in order to perform the most energy efficient VM-to-server mapping and assignment of the processing rates to the servers. Its energy performance is, indeed, noticeable. However, we note that: (i) the application scenario of [9] does not consider, by design, mobile clients; and, (ii) the admission control of the input workload is not performed and the bandwidths of the intra-datacenter links are assumed fixed. Energy-efficient joint reconfiguration of the overall communication-plus-computing resources in virtualized data centers is the topic of [10], where an approach based on the stochastic nonlinear optimization is pursued. Although the resulting scheduler is adaptive and guarantees bounded per-job execution times, we point out that: (i) the application scenario of [10] does not consider mobile clients; and, (ii) resource consolidation and admission control of the input traffic are not performed. Energy-saving dynamic provisioning of the computing resources in virtualized green data centers is the topic of [11]. Specifically, in order to avoid the prediction of future input traffic, the authors of [11] resort to a Lyapunov-based technique that dynamically scales up/down the sizes of the VM’s tasks and the corresponding processing rates by exploiting the available queue information. Although the pursued approach does not suffer from prediction errors, it relies on an inherent execution delay-vs.-utility tradeoff, that does not allow us to account for hard deadline on the execution times.

A second research direction deals with the energy-efficient traffic offloading from VCs to serving RSUs by exploiting the underlying Vehicular-to-Infrastructure (V2I) wireless connections [12], [13]. In this framework, the focus of [12] is on the optimized task and processing rate mapping of an assigned Task Interaction Graph (TIG) to a computing graph composed by networked reconfigurable computing nodes. As in our approach, hard constraints on the overall execution times are considered by [12]. However, unlike our approach, we point out that: (i) the focus of [12] is on the traffic offloading from mobile devices to remote clouds, so that the resulting task scheduler does not support, by design, data dissemination and content delivery services; (ii) the joint task and computing rate mapping afforded in [12] is, by design, static; and, (iii) the scheduler in [12] does not perform real-time reconfiguration and/or consolidation of the computing resources hosted by the serving cloud. The authors of [13] develop a distributed scheduler for performing mobile-to-access point traffic offloading. The scheduler enforces cooperation among the mobile devices, in order to minimize the average energy consumption of the mobile devices under per-device hard upper bounds on the volumes of the traffic to be offloaded to the remote cloud. For this purpose, in [13], it is assumed that mobile devices can execute tasks locally, offload tasks to other cooperative mobile devices or to the remote cloud, in accordance with the decision taken by the proposed scheduler. Interestingly, it exploits a Lyapunov-based optimization approach, in order to attain a good tradeoff between the average energy wasted by the mobile devices for local/remote traffic offloading and the volume of the Internet traffic generated towards the remote cloud. Thus, similarly to our work, the energy-vs.-traffic tradeoff is the focus of [13]. However, we point out that: (i) the work in [13] does not consider a mobile scenario; (ii) the scheduler in [13] does not enforce per-client QoS guarantees on the maximum execution-plus-communication delay and/or the minimum processing rate; and, (iii) the energy wasted by the remote cloud for processing the offloaded traffic is not included in the objective function of [13].

A third research direction focuses on the energy consumption of hand-held wireless devices when traffic offloading over 3G/4G/WiFi connections is performed [14], [15], [16]. Specifically, [14] focuses on the CPU energy consumption of wireless devices by profiling the volume of computation that can be performed under a given energy budget. Interestingly, this paper supports the conclusion that DVFS does affect the computing energy efficiency of wireless devices, but not radically. Interestingly, it is showed in [14] that the number of performed CPU cycles mainly depends on the sizes of the tasks and the burst factors of the task streams to be processed. According to this observation, a main result of [14] is the characterization of the relationship between task sizes, average number of
needed CPU cycles and burst factors of the processed task streams. Similar results are presented in [15] and [16], where analytical models for the average energy consumptions of 3G/WiFi and 4G connections are presented and tested through field trials. Overall, the contributions in [14], [15] and [16] do not afford, by design, resource management and/or scheduling optimization problems and the carried out energy analysis does not consider the effect of QoS constraints on the allowed computing-plus-communication delay.

The remainder of this paper is organized as follows. After presenting the architecture of the considered TCP/IP-based vehicular Fog platform in Section 2, the proposed joint scheduler is developed in Sections 3 and 4. Implementation aspects and implementation complexity of the proposed scheduler are the topics of Section 5. Numerical results attained through performance tests are presented in Section 6, while some conclusions are drawn in Section 7. Due to space limitations, detailed proofs of the most analytical results are not reported here. The interested reader may refer to [17].

Regarding the main adopted notation, \( E\{\cdot\} \) indicates expectation, \( \triangleq \) means “equal by definition”, \( [x]_a \) and \( [x]_+ \) indicate: \( \min\{b; \max\{a; x\}\} \), and: \( \max\{0; x\} \), respectively. Finally, \( E\{\varphi(\sigma, s, q)\} \) denotes the expectation of the multi-argument function \( \varphi(\sigma, s, q) \) carried out over the probability density function (pdf) of the random variable (r.v.) \( \sigma \).

2 The considered VCC infrastructure

Several recent research efforts target I2V networked infrastructures for the support of data dissemination and content delivery services [18]. The main building blocks of these architectures are a number of static RSUs, which are deployed along the road and are spaced apart by \( Di \) (m). Each RSU serves a spatial cluster of radius \( Ra \) (m). It acts as a serving point for the VCs currently traveling on the served cluster and it is equipped with computing capability, in order to play the role of a Fog node [18]. The RSUs may be also inter-connected by a wireless backbone (see the horizontal orange rays in Fig.1a), in order to allow inter-Fog data exchange when the served VCs move over adjacent spatial clusters [18], [19]. According to the Dedicated Short-Range Communication (DSRC) standard for the Intelligent Transportation System (ITS) [2], an RSU provides (possibly, multiple) I2V point-to-point connections to the served VCs.

At this regard, we shortly note that the current radio band of vehicular networks is located at 5.9 GHz and partitioned into seven channels. One control channel is designed to establish I2V transport connections, while the remaining service channels are employed to download data through point-to-point single-hop IEEE 802.11p radio links (see the RSU-to-vehicle green ray at the bottom of Fig.1a) [8]. For this purpose, the transmission time over each service channel is partitioned into slots, \( T_S \) (s) is the slot duration, \( t \) is the discrete-time slot index, and the \( t \)-th slot spans the semi-open time interval \([tT_S, (t+1)T_S]\), \( t \geq 0 \).

2.1 Input traffic and queue model in virtualized Fog nodes

In a virtualized Fog node, each VM processes the currently assigned task by self-managing own local virtualized storage/computing resources. When a request for a new job is submitted to the Fog node, the corresponding resource scheduler adaptively performs both admission control and allocation of the available virtual resources. Hence, according to the typical architecture presented, for example, in [7], Fig.1b reports the main blocks which operate at the Middleware layer of the considered Fog node. Roughly speaking, a Fog node is composed by: (i) an Admission Control Router (ACR); (ii) an input buffer of size \( N_I \); (iii) a reconfigurable computing platform and the related switched Virtual LAN (VLAN); (iv) an output queue of size \( N_O \); and, (v) an adaptive scheduler, that reconfigures and consolidates the available computing-communication resources and also performs the control of the input/output traffic flows.

Specifically, at the end of slot \( t \), input requests of processing new jobs arrive at the input of the ACR of Fig.1b. This happens according to a random real-valued arrival process \( \{a(t) \in \mathbb{R}^+, t \geq 0\} \), that is limited up to \( a_{max} \in \mathbb{R}^+ \) Information Units (IUs) per slot\(^1\). The arrival process \( \{a(t), t \geq 0\} \) is assumed to be independent from the current backlogs of the input/output queues of Fig.1b. Furthermore, since, in our framework, it may be the aggregation of multiple (possibly, heterogeneous) traffic flows generated by the remote cloud, nearby Fog nodes and VCs [19], we do not introduce any a priori assumption about the statistical behavior of \( \{a(t)\} \). According to this position, let \( \lambda(t) \in \mathbb{R}^+ \) (IU/slot) be the number of IUs out of \( a(t) \) that are admitted into the input queue of Fig.1b at the end of slot \( t \), respectively. We assume that any new arrival that is not admitted by the ACR of Fig.1b is declined. Thus, we have: \( 0 \leq \lambda(t) \leq a(t) \), so that: \( 1 - (\lambda(t)/a(t)) \), is the fraction of the input traffic that is rejected at slot \( t \). Furthermore, we consider (time-slotted) \( \text{G/G/1}/N_I \) and \( \text{G/G/1}/N_O \) fluid systems for modeling the input and output queues of Fig.1b, respectively. Due to the admission control, both queues are loss-free and they implement the FIFO service discipline.

Let \( s(t) \in \mathbb{R}_+ \) and \( q(t) \in \mathbb{R}_+ \) be the numbers of IUs stored by the input and output queues of Fig.1b at the beginning of slot \( t \). Furthermore, let \( L_{tot} \) (IU/slot) be the size of the incoming workload that: (i) is drained from the input queue at the beginning of slot \( t \); and, (ii) is injected into the output queue at the end of slot \( t \). Finally, let \( r(t) \) (IU/slot) be the workload that is drained from the output queue and transmitted over the vehicular TCP/IP connection of Fig.1b during slot \( t \). Hence, the time evolutions of the backlogs \( \{s(t) \in \mathbb{R}_+^+, t \geq 0\} \) and \( \{q(t) \in \mathbb{R}_+^+, t \geq 0\} \) of the input and output queues are dictated by the following Lindley’s equations, respectively:

\[
s(t+1) = s(t) - L_{tot}, t \geq 0, \quad (1)
\]

\[
q(t+1) = q(t) - r(t) + L_{tot}, t \geq 0. \quad (2)
\]

2.2 The considered virtualized Fog architecture

Let \( N_S \) be the number of physical servers which equip the Fog node of Fig.1b, and let \( M_{max}(k), k = 1, \ldots, N_S \) be the maximum number of VMs that can be hosted by

1. The meaning of an IU is application dependent. It may represent a bit, byte, segment or even an overall large-size application task (for example, a large image). We anticipate that, in the carried out tests of Section 6, IUs are understood as Mbit.
the $k$-th physical server, so that: $M \doteq \sum_{k=1}^{N_S} M_{\text{max}}(k)$, is resulting maximum number of VMs hosted by the Fog node. In principle, each VM operates as a virtual server, which is capable to process $f_i$ IUs per second (e.g., $i$ is the VM index). Depending on the size $L(i)$ (IU) of the task to be currently processed by $VM(i)$, the corresponding processing rate $f_i$ may be adaptively scaled at run-time through DVFS techniques. In our framework, $f_i$ may assume values over the interval $[0, f_i^{\text{max}}]$, where $f_i^{\text{max}}$ (IU/slot) is the maximum processing rate of $VM(i)$.

Being $L_{\text{tot}}$ the overall size of the current workload to be processed, let $L(i) \geq 0$, $i = 1, \ldots, M$, be the size of the task that the Adaptive Load Dispatcher of Fig.1b assigns to $VM(i)$, so that, the following constraint: $\sum_{i=1}^{M} L(i) = L_{\text{tot}}$, guarantees that the overall workload is partitioned into (at the most) $M$ parallel tasks. Hence, the set of attributes which characterize each VM is:

$$\{ \Phi(\eta_i), \Delta, \mathcal{E}_{c,\text{idle}}(i), \mathcal{E}_{c,\text{max}}(i), f_i^{\text{max}} \}, \quad i = 1, 2, \ldots, M,$$  

(3)

where $\Delta$ is the maximum per-slot and per-VM allowed processing time (in seconds); $\mathcal{E}_{c,\text{idle}}(i)$ (Joule) is the static energy consumed by $VM(i)$ in the idle state; $\mathcal{E}_{c,\text{max}}(i)$ (Joule) is the maximum energy consumed by $VM(i)$; $f_i^{\text{max}}$ is the maximum processing rate of $VM(i)$; and $\Phi(\eta_i)$ is the utilization function of $VM(i)$, which is typically evaluated as in [20]: $\Phi(\eta_i) \doteq \left( \frac{f_i}{f_i^{\text{max}}} \right)^2$, with $\eta_i \doteq \frac{f_i}{f_i^{\text{max}}}$.
It is shown in [20] that there is a linear relationship between the CPU utilization $\Phi(\eta_t)$ by $V M(i)$ and the corresponding computing energy consumption $E_c(i)$, so that we can write:

$$E_c(i) = E_c^{ide}(i) + (f_i / f_i^{max})^2 (E_c^{max}(i) - E_c^{ide}(i)). \tag{4}$$

Furthermore, we note that switching from the processing rate: $f_i(t-1)$ (e.g., the processing rate of $V M(i)$ at the $(t-1)$-th slot) to: $f_i(t)$ (e.g., the processing rate of $V M(i)$ at the $t$-th slot) entails an energy overhead of $E_{dyn}(i,t)$ [20]. Although, the actual behavior of the function: $E_{dyn}(i,t)$ depends on the adopted DVFS technique, a quite common model is the following quadratic one [20]:

$$E_{dyn}(i,t) = k_c (f_i(t) - f_i(t-1))^2. \tag{5}$$

In Eq. (5), $k_c$ (Joule/($Hz^2$)) denotes the reconfiguration cost induced by an unit-size rate switching and it is typically limited up to few hundreds of $mJ$’s per ($MHz^2$) [20].

**Remark 1 - Discrete DVFS**

Before proceeding, a remark on the allowed spectrum of the per-VM processing rates is in order. Specifically, we note that actual VMs are generally instantaneously atop physical computing cores which offer, indeed, only a finite set: $O = \{ \tilde{\eta}^0, \tilde{\eta}^1, \ldots, \tilde{\eta}^{Q-1} \} \subseteq f^{max}$ of discrete processing rates. Hence, in order to deal with both continuous and discrete reconfigurable physical computing infrastructures, we borrow the approach formerly developed, for example, in [21], [22]. Specifically, after indicating by: $B = \{ \tilde{\eta}^0, \tilde{\eta}^1, \ldots, \tilde{\eta}^{Q-1} \}$ the discrete values of $\eta$ which correspond to the set $O$, we build up a Virtual Energy Consumption curve $\Phi(\eta)$ by resorting to a piecewise linear interpolation of the allowed $O$ operating points: $\{ \tilde{\eta}^{(j)}, \Phi(\tilde{\eta}^{(j)}) \} ; j = 0, \ldots , (Q-1)$. Obviously, we may use it as the true energy consumption curve for the resource provisioning. Unfortunately, being the virtual curve of continuous type, it is no longer guaranteed that the resulting optimally scheduled processing rates are still discrete valued. However, as also explicitly noted in [21], [22], any point: $(\eta^*, \Phi(\eta^*))$, with $\tilde{\eta}^j < \eta^* < \tilde{\eta}^{j+1}$, on the virtual curve may be actually attained by time-averaging over $\Delta$ secs the corresponding surrounding vertex points: $(\tilde{\eta}^j, \Phi(\tilde{\eta}^j))$ and $(\tilde{\eta}^{j+1}, \Phi(\tilde{\eta}^{j+1}))$. Due to the piecewise linear behavior of the virtual curve, as in [21], [22], it is guaranteed that the average energy cost of the discrete DVFS system equates that of the corresponding actual one over each time interval of duration $\Delta$. □

### 2.4 Intra-Fog communication

We assume that the $i$-th virtual end-to-end connection (e.g., the $i$-th virtual link) of the Fog platform of Fig.1b is bidirectional and symmetric. Furthermore, in order to guarantee reliable (e.g., loss and error-free) transport of data, as in [23], [24], we assume that the TCP New Reno protocol is implemented for managing the intra-Fog end-to-end transport connections. Hence, under steady-state operating conditions, the power $P_{i}^{net}(t)$ (measured in ($Watt$)) drained by the $i$-th end-to-end connection may be evaluated as in [25]:

$$P_i^{net}(t) = \Omega_i (RTT_i R_i(t))^2, \quad i = 1, \ldots, M, \tag{6}$$

where $RTT_i$ is the average round-trip-time of the $i$-th intra-Fog connection, $R_i(t)$ ($IU$/slot) is the corresponding communication rate at slot $t$, and $\Omega_i$ ($Watt$) is the power consumption of the connection when the product: (round-trip-time) by (communication-rate) is unit-valued. Hence, after indicating by $L_i(t)$ ($IU$/slot) the size of the task assigned to $V M(i)$ at slot $t$, the corresponding communication energy $E_{LAN}(i,t)$ (Joule) wasted by the $i$-th virtual link is:

$$E_{LAN}(i,t) = P_i^{net}(t)L_i(t)/R_i(t) \equiv (2\Omega_i/(T_S-\Delta))(RTT_i L_i(t))^2. \tag{7}$$

In Eq. (7), the last relationship follows from the fact that, by design, the allowed per-slot transmission time is: $(T_S - \Delta)$, so that the corresponding communication rate equates: $R_i(t) = (2L_i(t)/(T_S - \Delta))$. At this regard, we also note that, in practical application scenarios, the maximum per-slot aggregate traffic conveyed by the intra-Fog LAN of Fig.1b is generally limited up to an assigned value: $R_{max}$ ($IU$/slot), which depends, in turn, on the employed (typically, Ethernet-type) switch technology [23], [24]. As a consequence, the following constraint must hold:

$$\sum_{i=1}^{M} R_i(t) = \sum_{i=1}^{M} (2L_i(t)/(T_S - \Delta)) \leq R_{max}. \tag{8}$$

### 2.5 Rate-vs.-energy tradeoff in TCP/IP vehicular connections

Goal of the output queue of Fig.1b is to effectively cope with the fading and mobility-induced random (possibly, unpredictable) fluctuations of the bandwidth of the corresponding single-hop downlink connection (see the RSU-to-vehicle green ray at the bottom of Fig.1b). At this regard, we note that, from an application point of view, there are three main reasons for resorting to the TCP/IP protocol, in order to implement this connection. First, it guarantees, by design, reliable (e.g., loss and error-free) transport of data [4]. Second, it allows the vehicular client to perform flow control, in order to adaptively match the transmission rate of the serving RSU to the actual (typically, time-varying) operating conditions of the used mobile device. Third, being connection-oriented, the TCP protocol allows the exploitation of the (aforementioned) location awareness of the Fog node, in order to measure the instantaneous state of each sustained RSU-to-vehicle connection. Hence, without loss of generality, in the sequel we focus on the case of a single RSU-to-vehicle connection (see Fig.1a). The goal is to save energy by timely adapting the transmission rate $r(t)$ ($IU$/slot) of the supported TCP connection. In fact, $r(t)$ depends on both the transmit energy $E_W(t)$ and the connection state $\sigma(t)$ as in [17], [25]:

$$r(t) = \sigma(t)(E_W(t))^{1/2}, \quad (IU$/slot) \tag{9}$$

where the state of the supported vehicular connection at slot $t$ is defined as in [17], [25]:

$$\sigma(t) \equiv (K_0(z(t))^{1/2}/RTW(t), \quad t \geq 1, \tag{10}$$

so that:

$$E_W(t) \equiv E_W(r(t), \sigma(t)) = (r(t)/\sigma(t))^2. \tag{11}$$
In Eq. (10), \( z(t) \) is the mobility function of the vehicular client served by the considered TCP/IP mobile connection. It may be modeled as a time-correlated log-distributed sequence [26]: \( z(t) \overset{d}{=} a_0 \cdot 10^{0.1z(t)} \), where \( a_0 = 0.9738 \), and \( \{z(t), t \geq 1\} \) is a time-correlated stationary zero-mean unit-variance Markov random sequence, whose probability density function is evenly distributed over the interval \([-\sqrt{3}, \sqrt{3}]\) [26]. The (positive) constant \( K_0 \) in (10) captures the performance of the FEC-based error-recovery system implemented at the Physical layer of Fig.1b [25], while \( RTT_W(t) \) is the round-trip-time of the connection. It is iteratively calculated through the Jacobson’s formula:

\[
RTT_W(t) = 0.75 \cdot RTT_W(t-1) + 0.25 \cdot \Delta_{IP}(t),
\]

where \( \Delta_{IP}(t) \) is the instantaneous packet-delay (in multiple of the slot period) measured at the Transport layer of the considered connection. Furthermore, in micro-cellular land-mobile applications, the corresponding correlation coefficient: \( h \overset{d}{=} E\{z(t)z(t-1)\} \) that describes the time-correlation of the fading-induced fluctuations of the state of the connection may be evaluated as in [26]:

\[
h = (0.82)^{\frac{\bar{v}}{100}},
\]

where \( \bar{v} \) (m/s) is the average speed of the served vehicular client of Fig.1a. Hence, after indicating by \( \Delta_{IP}^{\text{max}} \) the maximum instantaneous delay of the connection, from Eq. (12), it follows that: \( RTT_W(t) \leq \Delta_{IP}^{\text{max}} \), so that the corresponding state in (10) is lower bounded as in:

\[
\sigma(t) \geq \sigma_{\text{min}} \overset{d}{=} K_0 \left( a_0 10^{-0.15} \right)^{1/2} / \Delta_{IP}^{\text{max}}.
\]

Overall, from the outset, it follows that the resulting total communication-plus-computing energy: \( E_{\text{tot}}(t) \) consumed at slot \( t \) by the Fog node of Fig.1b is given by the following final expression:

\[
E_{\text{tot}}(t) = \sum_{i \in \mathcal{S}(t)} E_{\text{cpu}}(i,t) + E_{\text{mem}}(i,t) + E_{\text{LAN}}(i,t) + E_W(t).
\]

### 3 Resource Reconfiguration

Let \( \mathcal{S}(t) \) (resp., \( \overline{\mathcal{S}}(t) \)) be the set of turned-ON (resp., turned-OFF) VMs at slot \( t \) (see Fig.1b). The proposed scheduler interleaves reconfiguration and consolidation slots. Specifically, at each reconfiguration slot, the scheduler does not change the sets \( \mathcal{S}(t) \) and \( \overline{\mathcal{S}}(t) \) of the turned ON/OFF VMs (and, then, it does not change the corresponding sets of turned ON/OFF physical servers). However, it performs the following reconfiguration actions (see Fig.1b): (i) control of the input and output flows; and, (ii) assignment of the tasks and processing rates to the turned-ON VMs. In the remaining part of this section, we deal with the resource reconfiguration problem, while resource consolidation is afforded in Section 4.

Hence, after indicating by:

\[
L_{\text{tot}}(t) = \sum_{i \in \mathcal{S}(t)} L_i(t), \tag{16}
\]

the size of the workload which is drained by the input queue of Fig.1b, the resource reconfiguration problem to be afforded at slot \( t \) is formulated as follows:

\[
\min_A \quad \theta E_\sigma \{ E_{\text{tot}}(t) \} - (1 - \theta) E_\sigma \{ r(t) \}, \tag{17.1}
\]

subjected to:

\[
E_\sigma \{ E_W(\sigma, r) \} \leq E_{\text{ave}}, \tag{17.2}
0 \leq \lambda(t) \leq a(t), \tag{17.3}
0 \leq s(t + 1) \leq N_t, \tag{17.4}
0 \leq q(t + 1) \leq N_o, \tag{17.5}
\min \{q(t); r_{\text{max}}\} \leq r(t) \leq \theta \min \{q(t); r_{\text{max}}\}, \tag{17.6}
0 \leq f_i(t) \leq f_i^{\text{max}}, \quad i \in \mathcal{S}(t), \tag{17.7}
0 \leq L_i(t) \leq L_i^{\text{max}} \equiv \Delta_{IP}^{\text{max}} \leq \Delta, \quad i \in \mathcal{S}(t), \tag{17.8}
L_{\text{tot}}(t) \leq \min \{s(t); (T_S - \Delta)(R_{\text{max}}/2)\}. \tag{17.9}
\]

In Eq. (17.1), \( A \overset{d}{=} \{ L_i(t), f_i(t), i \in \mathcal{S}(t); \lambda(t), r(t) \} \) is the set of variables to be reconfigured at slot \( t \) and \( \theta \in [0,1] \) is an (assigned) weight factor, that tunes the desired tradeoff between average consumed energy and wireless transmission rate.

Regarding the constraints in Eqs. (17.2)-(17.10), some remarks are in order. The constraint in (17.2) is typically dictated by spectral compatibility and energy budget issues [2]. It limits up to: \( E_{\text{ave}} \) (Joule) the per-slot average energy available for transmitting over the single-hop mobile connection of Fig.1a (see the RSU-to-vehicle green ray at the bottom of Fig.1a). The constraint in (17.3) accounts for the (aforementioned) admission control performed by the access router of Fig.1b, while Eqs. (17.4) and (17.5) account for the finite sizes of the input and output buffers, respectively. Eq. (17.6) guarantees that the size \( r(t) \) of the workload drained from the output queue of Fig.1b does not exceed the corresponding queue backlog \( q(t) \), and also forces \( r(t) \) to fall into a desired range: \( [r_{\text{min}}, r_{\text{max}}] \) of transmission rates. Furthermore, Eq. (17.7) (resp., Eq. (17.8)) bounds the maximum processing rate (resp., workload) of \( \mathcal{S}(i) \), while Eq. (17.9) is a hard limit on the corresponding per-slot and per-VM processing time. Finally, Eq. (17.10) guarantees that the size \( L_{\text{tot}}(t) \) of the workload drained by the input queue does not exceed the corresponding queue backlog \( s(t) \), while being also compliant with the constraint in (8) on the maximum aggregate rate of the intra-Fog LAN.

Before proceeding, we explicitly point out that \( r_{\text{max}} \) (IU/slot) in (17.6) is the maximum instantaneous rate conveyed by the mobile connection of Fig.1b, and its value is typically dictated by the WiFi technology actually employed for implementing the single-hop wireless link of Fig.1a [8]. Furthermore, the corresponding \( r_{\text{min}} \) (IU/slot) in (17.6) fixes the minimum instantaneous rate to be guaranteed, and, then, it plays the role of a hard QoS requirement, that is enforced by the supported application. From Eqs. (9), (14) and (17.2), it follows that feasible values of \( r_{\text{min}} \) are limited up to: \( \sigma_{\text{min}} \sqrt{E_{\text{ave}} \cdot \text{IU}/\text{slot}} \).
3.1 Feasibility and QoS guarantees

Regarding the feasibility of the resource reconfiguration problem in (17.1)-(17.10), the following formal result holds.

**Proposition 1. Feasibility conditions.**

Under the reported framework, the following four inequalities:

\[ E_W(r_{\min}, r_{\min}) \leq E_{\text{ave}}, \]  
(18.1)

\[ \sum_{i=1}^{M} I_i^{\max} = \sum_{i=1}^{M} i^{\max} \Delta \geq r_{\min}, \]  
(18.2)

\[ (R_{\max}/2) (T_S - \Delta) \geq N_I, \]  
(18.3)

\[ a(t) \geq r_{\min}, \forall t \geq 0, \]  
(18.4)

guarantee that the resource reconfiguration problem in (17.1)-(17.10) is feasible. \( \square \)

Hence, since the reported conditions assure that the reconfiguration problem admits the solution, we pass to consider the corresponding QoS properties. At this regard, we point out that a combined exploitation of the constraints in (17.4)-(17.6) leads to the following hard bounds on the resulting communication-plus-computing delay and rate-jitter of the solution of the afforded resource reconfiguration problem.

**Proposition 2. Hard QoS guarantees.**

Let the feasibility conditions of Proposition 1 be met. Let \( T_{QI}, T_{SI}, T_{QO}, \) and \( T_{SO} \) be the random variables that measure the wireless random queue delay of the input queue, the service time of the input queue, the queue delay of the output queue and the service time of the output queue, respectively. Thus, the following QoS guarantees hold:

a) the random total delay: \( T_{\text{tot}} \triangleq T_{QI} + T_{SI} + T_{QO} + T_{SO} \) (in multiple of the slot period) induced by the Fog platform of Fig.1b is limited (in a hard way) up to:

\[ T_{\text{tot}} \leq ((N_I + N_O)/r_{\min}) + 2; \]  
(19.1)

b) the instantaneous jitter affecting the wireless transmit rate is upper bounded as in:

\[ |r(t_1) - r(t_2)| \leq (r_{\max} - r_{\min}), \text{for any } t_1 \text{ and } t_2; \]  
(19.2)

c) the corresponding average jitter: \( \sigma_r \triangleq \sqrt{E \{ (r(t) - \nu_r)^2 \}} \) is limited as in:

\[ \sigma_r \leq \sqrt{(r_{\max})^2 - (r_{\min})^2}. \]  
(19.3)

The reported QoS guarantees lead to the conclusion that the NetFC platform of Fig.1b is capable, indeed, to support real-time (i.e., delay and jitter-sensitive) applications, while meeting the bound in (17.2) on the consumed wireless energy.

3.2 Solution of the resource reconfiguration problem

In order to present the solution: \( \{ f_i^*(t), L_i^*(t), i \in S(t); L_{\text{tot}}^*(t), r^*(t) \} \), of the stated resource reconfiguration problem, let us introduce the following dummy position:

\[ \psi(t) \triangleq \min \{ s(t); \max \{ r_{\min}; \tau^*(t - 1) \} \}, \]  
(20)

where: \( \tau^*(t) \triangleq (1/\tau)(\sum_{\tau=0}^{t-1} r^*(\tau)) \), is the average wireless transmit rate up to \( (t - 1) \)-th slot. Hence, the following result holds for the optimal resource reconfiguration.

**Proposition 3. Solution of the resource reconfiguration problem.**

Under the feasibility conditions in (18.1)-(18.4), for the solution of the resource reconfiguration problem in (17.1)-(17.10), we have that:

a) the optimal total workload \( L_{\text{tot}}^*(t) \) to be processed is scheduled as in:

\[ L_{\text{tot}}^*(t) = \begin{cases} 0, & \text{for } q(t) > N_O + r_{\min} - r_{\max}; \\
 \psi(t), & \text{for } q(t) \leq N_O + r_{\min} - r_{\max}; \end{cases} \]  
(21.1)

b) the optimal admitted input traffic \( \lambda^*(t) \) is scheduled as in:

\[ \lambda^*(t) = \min \{ a(t); N_I - s(t) + L_{\text{tot}}^*(t) \}; \]  
(21.2)

c) the optimal transmission rate \( r^*(t) \) over the mobile connection is scheduled as in:

\[ r^*(t) = \left( \frac{(1 - \theta) \sigma(t)^2}{2 \zeta^*(t)} \right)^{\min \{ r_{\max}; q(t) \}}, \]  
(21.3)

where \( \zeta^*(t) \) is the nonnegative Lagrange multiplier of the constraint in (17.2). It may be adaptively updated according to the following gradient-based projection iteration:

\[ \zeta^*(t) = [\zeta^*(t - 1) - \alpha (E_{\text{ave}} - E_W (\sigma(t), r^*(t - 1)))], \]  
(21.4)

where \( \alpha \) is a suitable nonnegative step-size;

d) for \( i \in S(t) \), the optimal processing rate \( f_i^*(t) \) and task size \( L_i^*(t) \) of the i-th VM are scheduled as in:

\[ f_i^*(t) = \left[ \frac{k_e f_i^*(t - 1) + (\Delta \nu_i^*(t)/20)}{k_c + (\frac{(1 - \theta) \sigma(t)}{2 \zeta^*(t)})} \right] \]  
(21.5)

and

\[ L_i^*(t) = \min \left\{ \frac{(T_S - \Delta)}{4 \Omega_i (R_{\text{RTT}})^2 \theta_0} \frac{L_{\text{max}}}{\bar{L}_{\text{max}}}; \left| \Delta f_i^*(t)/\bar{L}_{\text{max}} \right| \right\}. \]  
(21.6)

where the (nonnegative) Lagrange multiplier \( \nu_i^*(t) \) of the i-th constraint in (17.9) is given by:

\[ \nu_i^*(t) = \left[ \mu^*(t) - \frac{4 \Omega_i (R_{\text{RTT}})^2 L_i^*(t)}{(T_S - \Delta)} \right]^+. \]  
(21.7)

Finally, the Lagrange multiplier \( \mu^*(t) \) of the constraint in (16) is the (unique nonnegative) root of the following algebraic equation:

\[ \sum_{i \in S(t)} L_i^*(\mu(t)) = L_{\text{tot}}^*(t), \]  
(21.8)

where \( L_i^*(\cdot) \) is given by (21.6), with \( \mu^*(t) \) replaced by the dummy variable \( \mu(t) \). \( \square \)

Regarding the tuning of the step-size \( \alpha \) in (21.4), we anticipate that, for values of the average speed \( \bar{v} \) in (13) ranging from: 30 \( (Km/h) \), to: 150 \( (Km/h) \), the following setting: \( \alpha = 10^{-2} \), allows the iteration in (21.4) to converge to its steady-state value within 20-25 slots, with a final accuracy within 1\%.
4 Resource Consolidation

Let $T_{CN} \triangleq \{t_0, t_1, \ldots \}$ indicate the set of the consolidation slot indexes. By design, at each consolidation slot $t \in T_{CN}$, the scheduler updates the sets $S(t)$ and $\mathcal{S}(t)$ of the turned ON/OFF VMs and, then, turns ON/OFF the corresponding physical servers. Afterwards, on the basis of these updated sets, the scheduler performs again the resource reconfiguration actions previously described in Section 3. In our framework, resource consolidation is performed according to the following three considerations.

First, as in [27], the set $T_{CN}$ of the consolidation slots is adaptively built up on the basis of the average utilization of the currently turned-ON VMs. For this purpose, after indicating by $|S(\tau)|$ the number of VMs that are turned ON at slot $\tau$, the following average utilization coefficient:

$$\mathcal{U}(\tau) \triangleq \frac{1}{|S(\tau)|} \left( \sum_{i \in S(t)} \frac{L_i(\tau)}{L_{\text{max}}^i} \right),$$

is evaluated at every slot $\tau \geq 0$. Afterwards, as in [27], the following $l$-out-of-$m$ decision rule is applied: the current slot $t$ is flagged as a consolidation slot if at least $l$ out of the $m$ most recently measured values of $\mathcal{U}(\tau)$ as the predicted utilization for slot $t$ fall out of a target interval $\mathcal{I} \triangleq [T_{lL}, T_{lU}]$. The interval $\mathcal{I}$ is the set of the desired utilization values and its end-points: $T_{lL}$ and $T_{lU}$ are the lower and upper thresholds on the desired average VM utilization.

Second, in our framework, a turned-OFF physical server is turned ON if at least one of the hosted VMs is turned ON. Furthermore, a turned-OFF VM is turned ON if its processing rate is no longer vanishing. Hence, at each consolidation slot $t \in T_{CN}$, the energy consumption of the $i$-th VM may be modeled as in [28]:

$$\mathcal{E}_c(i, t) = \mathcal{E}_c^{\text{idle}}(i) - (f_i(t) - f_{i}^{\text{max}})^2(\mathcal{E}_c^{\text{max}}(i) - \mathcal{E}_c^{\text{idle}}(i)),$$

where $-1(x)$ is the unit-size step-function. The first term in (23) accounts for the decrement (resp., increment) of the static computing energy that is experienced by turning ON (resp., OFF) the $i$-th VM.

Third, turning ON a VM induces a latency of $T_{ON}$ seconds [27]. Hence, at each consolidation slot $t \in T_{CN}$, the following constraints must be also enforced:

$$T_{ON} f_i(t) + \left( \frac{L_i(t)}{L_{i}^{\text{max}}(t)} \right) \leq \Delta, \quad \forall i \in \mathcal{S}(t-1),$$

(24.1)

$$\left( \frac{L_i(t)}{f_i(t)} \right) \leq \Delta, \quad \forall i \in \mathcal{S}(t-1),$$

(24.2)

$$\sum_{i=1}^{M} L_i(t) \leq \min \{ s(t); (T_{S} - \Delta)(R_{\text{max}}/2) \}.$$  

(24.3)

The constraint in (24.1) accounts for the (aforementioned) latency, while the constraint in (24.2) enforces vanishing workload at vanishing processing rate, and, then, it guarantees that no workload is processed by $VM(i)$ when it is turning OFF. Finally, Eq. (24.3) assures that the overall workload to be processed by all available VMs does not exceed the maximum feasible one.

On the basis of the above three remarks, the resource consolidation problem is formulated as follows: minimize the objective function in (17.1) (with $\mathcal{E}_c(i, t)$ in (15) given by (23)), under the sets of constraints in Eqs. (17.2)-(17.8) and Eqs. (24.1)-(24.3). The set $A$ of the variables to be optimized is the same one of Section 3.

5 Adaptive Implementation and Complexity of the Proposed Scheduler

In order to effectively cope with the random (possibly, unpredictable) time fluctuations of the input traffic and state of the mobile connection of Fig.1a, we proceed to develop an adaptive implementation of the scheduler, which computes on-the-fly the solutions of both the resource reconfiguration and consolidation problems. For this purpose, we resort to the primal-dual algorithm recently revised in [29].

Generally speaking, it applies gradient-based iterations, in order to adaptively update both the primal variables (i.e., the variables to be optimized) and the dual variables (i.e., the Lagrange multipliers associated to the constraints) of the considered constrained optimization problem. In our framework, the proposed resource scheduler runs primal-dual iterations: (i) at the beginning of each reconfiguration slot, in order to compute the solution $\mu^*(t)$ of the nonlinear algebraic equation in (21.8); and, (ii) at the beginning of each consolidation slot, in order to numerically compute the set $\mathcal{A}^*$ of the optimal consolidated resources.

For this purpose, after indicating by $x(t) \in A$ a (scalar) resource variable to be updated at slot $t$ and by $\mathcal{L}(t)$ the Lagrangian function of the considered optimization problem, the $n$-th primal-dual iteration assumes the following general form [29]:

$$x(n) = \left[ x(n-1) - g_s(n-1) \frac{\partial}{\partial x} \mathcal{L}(n-1)(t) \right]_+, \quad n \geq 1,$$

(25)

where $n = 1, 2, \ldots$, is a discrete iteration index which runs at the beginning of the considered slot $t$. Furthermore, according to [29], in Eq. (25), we have that: (i) $\nabla_x \mathcal{L}(n-1)(t)$ is the scalar derivative of the considered Lagrangian function $\mathcal{L}(t)$, which is done with respect to the variable $x$ and is evaluated at the resource configuration setting attained at the previous iteration $(n-1)$; (ii) $g_s(n-1)(t)$ is a suitable $n$-varying (e.g., adaptive) step-size. According to [29], it is set as in: $g_s(n-1)(t) = 0.5(x(n-1)(t))^2$; and, (iii) the projection in (25) accounts for the nonnegative values of all involved variables.

Just as an illustrative example, at the beginning of each reconfiguration slot $t$, the root $\mu^*(t)$ of the (nonlinear) algebraic equation in (21.8) is computed by the scheduler by running the following iterations in the $n$-index:

$$\mu^{(n)}(t) = \left[ \mu^{(n-1)}(t) - 0.5 \left( \mu^{(n-1)}(t) \right)^2 \times \left( \sum_{i \in S(t)} L_i^{(n-1)}(t) - L_{i\text{tot}}^*(t) \right) \right]^+, \quad n \geq 1,$$

(26)

2. Consider that, due to the presence of the step-size function in (23), the (previously stated) resource consolidation problem resists, indeed, closed-form solution.
where \( L_{\text{tot}}^{(t)} \) is given by Eq. (21.1), and \( L^{(n-1)}(t) \) is given by Eq. (21.6), with \( \mu^*(t) \) and \( f^*(t) \) replaced by \( \mu^{(n-1)}(t) \) and \( f^{(n-1)}_n(t) \), respectively.

Algorithm 1 reports a pseudo-code for the overall adaptive implementation of the proposed resource scheduler.

Algorithm 1: A pseudo-code of the proposed adaptive resource scheduler

```
1: for \( t \geq 1 \) do
2:   Apply the \( t\)-out-\( m \) decision rule and flag \( t \) as reconfiguration or consolidation slot;
3:   if \( t \) is a consolidation slot, then
4:     Perform resource consolidation by running the iterations in Eq. (25);
5:     Update the sets \( S(t) \) and \( \overline{S}(t) \);
6:   end if
7:   if \( t \) is a reconfiguration slot, then
8:     Evaluate Eqs. (21.1)-(21.4);
9:     Compute \( \mu^*(t) \) through Eq. (26);
10:    Evaluate Eqs. (21.5)-(21.7);
11:   end if
12:   Update \( s(t+1) \) and \( q(t+1) \) through Eqs. (1) and (2), respectively;
13:   Compute \( \overline{U}(t) \) through Eq. (22), with \( \tau \) replaced by \( t \);
14: end for
```

Regarding the resulting implementation complexity, two main remarks are in order.

First, since the iterations in (25) are carried out at the beginning of each slot, the iteration index \( n \) must run faster than the slot time \( T_S \). Hence, the actual time duration: \( T_I(s) \) of each \( n \)-indexed iteration must be so small to allow the iterations in (25) to converge within a limited fraction of the slot time. At this regard, the formal results of Theorem 3.3 of [29] assure the asymptotic convergence of the iterations in (25). Furthermore, we have numerically ascertained that, at least in the tests of Section 6, about 10-15 \( n \)-indexed steps suffice, in order to attain the convergence of (25) with a final accuracy within 1%. Hence, in the carried out tests, we pose: \( T_I = T_S/250 \).

Second, since each VM updates the size of the task to be processed and the corresponding processing rate, the total number of variables to be updated at each slot scales up linearly with the number \( M \) of the available VMs. Therefore, the total per-slot implementation complexity of the scheduler of Algorithm 1 is of the order of \( O(M) \). Furthermore, since the pursued primal-dual approach allows the distributed implementation of the solutions of the afforded reconfiguration and consolidation problems [29], each VM may locally update its task size and processing rate through local measurements. As a consequence, the per-VM implementation complexity of the proposed scheduler does not depend on the number \( M \) of the available VMs, that is, it is of the order of \( O(1) \).

Overall, the above remarks lead to the conclusion that the implementation of the proposed scheduler is adaptive and distributed, while the resulting per-slot execution time is limited up to about 15 iteration periods, regardless of the (possibly, large) size the considered Fog node.

5.1 Implementation aspects of the intra-Fog Virtualization layer

Main tasks of the intra-Fog Virtualization layer of Fig.1b are twofold. First, it must guarantee that the demands for the per-VM processing rates \( \{f^*_i\} \) and per-link communication rates \( \{R^*_i\} \) done by the Middleware layer are mapped onto adequate CPU cycles and channel bandwidths at the underlying Physical layer. Second, it must profile at runtime the maximum and idle energies actually consumed by the running VMs.

Regarding the first task, we note that QoS mapping of the demands for processing and communication rates may be actually performed by equipping the Virtualization layer of Fig.1b with a per-VM queue system that implements (in software) the so-called mClock and SecondNet schedulers [30], [31]. Interestingly, Table 1 of [30] points out that the mClock scheduler is capable to guarantee hard (e.g., absolute) reservation of CPU cycles on a per-VM basis by adaptively managing the computing power of the underlying DVFS-enabled physical cores (see Algorithm 1 of [30] for the code of the mClock scheduler). Likewise, the SecondNet network scheduler in [31] provides bandwidth-guaranteed virtualized Ethernet-type contention-free links atop any set of TCP-based (possibly, multi-hop) end-to-end connections. For this purpose, SecondNet implements a suitable Port-Switching based Source Routing algorithm, that may directly work at the Middleware layer of Fig.1b (see Section 3 of [31]).

Regarding the (aforementioned) second task, the maximum and idle energies consumed by each VM may be profiled at runtime by resorting to the so-called Joulemeter tool [32]. It is a software tool that operates at the Virtualization layer of Fig.1b and is capable to provide the same energy metering functionality for VMs as currently exists in hardware for physical servers. For this purpose, joulemeter uses hypervisor-observable hardware power states to track the VM energy usage on each hardware component (see Section 5 of [32] for a detailed description of the joulemeter tool). Interestingly enough, the field trials reported in [32] support the general conclusion that the maximum and idle energies wasted by a running VM are proportional to the corresponding maximum and idle energies wasted by the hosting physical server, with the scaling factor being given by the fraction of the physical cores actually used by the VM.

6 Test Results and Performance Comparisons

This section presents the tested energy performance of the proposed scheduler under a set of synthetic and real-world input arrival traces, and, then, compares it with the corresponding ones of the DVFS-based scheduler in [25], Lyapunov-based scheduler in [11], static scheduler in [22] and NetDC scheduler in [10].

6.1 Simulated Fog setup

The simulations are carried out by exploiting the numerical software of the MATLAB platform. They emulate the energy consumptions of 5 homogeneous quad-core Dell PowerEdge
servers, which are equipped with 3.06 GHz Intel Xeon CPU and 4 GB of RAM. All the emulated servers are connected through commodity Giga Ethernet Network Interface Cards (NICs) and host up to 4 VMs (see Table 1).

In order to measure the actual delay performance of the proposed adaptive scheduler, we resort to a per-IU average total delay performance index (measured in multiple of the slot period), formally defined as in:

$$T_{\text{tot}}^\ast = \lim_\tau \frac{1}{\tau} \sum_{\tau=1}^{\tau} s^\ast(\tau) + \lim_\tau \frac{1}{\tau} \sum_{\tau=1}^{\tau} \lambda^\ast(\tau) + 2. \quad (27)$$

In Eq. (27), the first ratio is the average delay: $T_{\text{QO}}^\ast = \pi^\ast / \lambda^\ast$ induced by the input buffer of Fig.1b, the second ratio is the average delay: $T_{\text{QO}}^\ast = \pi^\ast / \lambda^\ast$, and the third ratio is the overall corresponding average service times.

### 6.2 Simulated vehicular setup

A three-line highway vehicular environment is simulated, where $N_{\text{vc}}$ vehicles proceed without performing abrupt U-turns. They move along the highway of Fig.1a with speeds uniformly ranging from: $v_{\text{min}}$ to $v_{\text{max}}$, so that: $\bar{v} = (v_{\text{max}} - v_{\text{min}})/2$, is the resulting per-vehicle average speed. Uniformly spaced RSUs serve the VCs and the simulated RSU-to-vehicle radio propagation accounts for free-space loss with exponent 4 and log-normal fading with standard deviation 8 [8], [19].

Since no any assumption has been introduced about the statistical behavior of the state in (10) of the considered RSU-to-vehicle mobile connection, the optimality of the proposed scheduler of Algorithm 1 is retained, regardless of the mobility patterns actually followed by the vehicles. However, the performed simulations refer to a highway environment, and, then, it is reasonable to assume that: (i) at each slot time, each vehicle moves to the next spatial cluster or remains in the currently occupied cluster; and, (ii) being the vehicle speeds evenly distributed and subject to random spatial fluctuations, the sojourn times of the vehicles in each cluster are geometrically distributed r.v.’s. These considerations lead, in turn, to adopt the so-called Markovian random walk model with random positioning, in order to simulate the vehicle mobility. According to this model, per-vehicle transitions among adjacent spatial clusters are represented by a Markovian spatial chain, in which each state corresponds to one spatial cluster with radius $R_a$ (see Fig.1a). Furthermore, the per-vehicle inter-cluster transition probability $\beta_{\text{vc}}$ and per-cluster average number of served vehicles $N_{\text{vc}}$ may be numerically evaluated as in [8], [19]:

$$\beta_{\text{vc}} = \bar{v}/v_{\text{max}}, \quad \text{and}, \quad N_{\text{vc}} = A_{\text{jam}} (1 - \beta_{\text{vc}})(\pi/2)(R_a)^2, \quad (28)$$

where $A_{\text{jam}}$ (vehicle/m$^2$) is the maximum spatial density of vehicles under traffic congestion phenomena.

### 6.3 Performance results

In the first test scenario, we run the proposed scheduler and evaluate the resulting average total consumed energy $E_{\text{tot}}^\ast$ under a synthetic (e.g., independent and identically distributed) input arrival random sequence. Fig.2 reports $E_{\text{tot}}^\ast$ for various values of $M$, and its examination points out that: (i) $E_{\text{tot}}^\ast$ decreases for increasing $M$; and, (ii) $E_{\text{tot}}^\ast$ increases for increasing values of the average wireless transmission rate $\pi^\ast$. The second set of simulations in Fig.3 presents the total average consumed energies $E_{\text{tot}}^\ast$ at $M = 10$, 15 and 20 for various values of $E_{\text{ave}}$. Fig.3 points out that, by increasing the number of VMs, $E_{\text{tot}}^\ast$ still decreases. However, the interesting point is that, even for increasing values of the available $E_{\text{ave}}$, the admission control performed by the proposed scheduler attempts to reduce the resulting $E_{\text{tot}}^\ast$ to the minimum (see the quasi-flat behavior of the plots of Fig.3 at medium/large values of $E_{\text{ave}}$).

### 6.3.1 Fraction of accepted requests-vs.-induced delay

The plots of Figs.4 and 5 report the (numerically evaluated) average fraction: $F_A^\ast = \lim_{\tau \to \infty} (1/\tau) \sum_{\tau=1}^{\tau} \lambda^\ast(\tau)/\lambda(\tau)$, of

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta = 0.8$ (s)</td>
</tr>
<tr>
<td>$N_t = N_S = 50$ (Mbit)</td>
</tr>
<tr>
<td>$k_s &lt; 0.5$, 0.05 (J/(MHz)^2)</td>
</tr>
<tr>
<td>$f_{\text{max}} = 10$ (Mbit/slot)</td>
</tr>
<tr>
<td>$\tau_{\text{max}} = \alpha_{\text{max}} = 8$ (Mbit/slot)</td>
</tr>
<tr>
<td>$\tau_{\text{min}} = 1.5$ (Mbit/slot)</td>
</tr>
<tr>
<td>$\theta = 0.5$</td>
</tr>
<tr>
<td>$R_{\text{max}} = 1000$ (Mbit/slot)</td>
</tr>
<tr>
<td>$T_S = 1$ (s)</td>
</tr>
<tr>
<td>$E_{\text{avg}}^\ast = 4$ (Joule)</td>
</tr>
<tr>
<td>$N_S = 5$</td>
</tr>
<tr>
<td>$M_{\text{max}} = 4$</td>
</tr>
<tr>
<td>$\alpha = 10^{-2}$</td>
</tr>
<tr>
<td>$\beta_{\text{vc}} = 100/130$</td>
</tr>
<tr>
<td>$N_{\text{vc}} = 40/3$</td>
</tr>
<tr>
<td>$\Omega = 0.5$ (Watt)</td>
</tr>
<tr>
<td>$h = 0.95$</td>
</tr>
<tr>
<td>$v = 100$ (Km/h)</td>
</tr>
<tr>
<td>$E_{\text{ave}} = 0.75$ (Joule)</td>
</tr>
<tr>
<td>$v_{\text{max}} = 130$ (Km/h)</td>
</tr>
<tr>
<td>$R_a = 250$ (m)</td>
</tr>
</tbody>
</table>

#### Table 1: Default values of the main simulated parameters.
the input traffic that is actually admitted by the proposed scheduler and the corresponding average total delay $T_{\text{tot}}^*$ in (27). These plots refer to the case of $k_c = 0.05 \ (J/(MHz)^2)$, $M = 10$ and $h = 0.95$. They allow us to gain insight about the tradeoff between admission capability and induced delay of the proposed scheduler. Specifically, two main conclusions may be drawn from the plots of Figs.4 and 5. First, the average fraction $F_A^*$ of the admitted traffic increases for increasing values of the storage capacity $N_I = N_O$ of the input/output buffers, and/or increasing values of $E_{\text{ave}}$ (see Figs.4 and 5). Second, according to the Little law, the corresponding average total delay suffered by the admitted traffic increases for increasing values of $N_I = N_O$ (see Fig.4), while it (quickly) decreases for increasing values of $E_{\text{ave}}$ (see Fig.5).

**6.4 Mobility effects**

Goal of this set of numerical tests is to acquire insight about the effects induced by the vehicle mobility on the resulting average total delay $T_{\text{tot}}^*$ in (27), when the target performance is assigned in terms of average wireless transmission rate $\tau^*$. For this purpose, the application scenario of Table 1 has been still considered at $M = 10$ and $k_c = 0.05 \ (J/(MHz)^2)$. The corresponding numerically evaluated performance is reported in Table 2. An examination of these results leads to two main conclusions. First, the minimum size $N_I = N_O$ of the input/output buffers required to meet the target values $\tau^*$ and $E_{\text{tot}}^*$ (quickly) increases for increasing values of $h$ (e.g., for decreasing values of the average vehicle speed $\bar{v}$). This is due to the fact that larger values of $h$ increase the coherence time of the sequence $\{\sigma(t)\}$ of the connection state (see Eqs. (10) and (13)), so that $\{\sigma(t)\}$ tends to be “less ergodic”. Thus, in order to offset this effect without penalizing the energy performance, the scheduler requires more buffering capacity, that, in turn, penalizes the resulting delay performance (see the last column of Table 2). Second, at least in the simulated scenarios, the requested buffer size $N_I = N_O$ tends to scale as: $O \left( -1/\log(h) \right)$ for $h \geq 0.87$.

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

In order to test the consolidation capability of the proposed scheduler when the arrival sequence $\{a(t)\}$ exhibits time-correlation, we have considered the arrival trace of Fig.6. It reports the real-world trace in Fig.2 of [28] and refers to the I/O workload taken from four RAID volumes of an enterprise storage cluster in Microsoft (see Section IV.A of [28]). In order to maintain the peak workload fixed at 8 (Mbit/slot), each arrival of Fig.6 carries out a traffic of 0.206 (Mbit).

### 6.5.1 Adaptive consolidation performance

The numerical trials of this sub-section aim at testing both the actual convergence to the steady-state and corresponding convergence speed of the proposed iterative consolidation algorithm of Section 5. For this purpose, we have evaluated and compared its performance against the optimal one. By design, at each consolidation instant $t$, the optimal consolidation scheduler computes the optimal sub-set of the consolidated VMs by performing an exhaustive search over
TABLE 2: Buffer size-vs.-delay tradeoff at target $\tau^*$ and $E_{tot}^*$ for various values of the mobility-induced correlation coefficient $h$. PMR:= Peak-to-Mean Ratio of the input traffic.

<table>
<thead>
<tr>
<th>Workload Types</th>
<th>$h$</th>
<th>$\tau^*$ (Mbit/slot)</th>
<th>$E_{tot}^*$ (Joule)</th>
<th>$N_1 = N_O$ (Mbit)</th>
<th>$T_{tot}^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic workload at $\pi = 7.5$ (Mbit/s) and PMR = 1.25</td>
<td>0.870</td>
<td>6</td>
<td>36</td>
<td>4</td>
<td>4.3</td>
</tr>
<tr>
<td>Real-world workload of [28]</td>
<td>0.967</td>
<td>6</td>
<td>36</td>
<td>7</td>
<td>6.2</td>
</tr>
<tr>
<td></td>
<td>0.997</td>
<td>6</td>
<td>36</td>
<td>24</td>
<td>7.3</td>
</tr>
</tbody>
</table>

Fig. 6: Sampled trace of an I/O arrival sequence from an enterprise cluster in Microsoft [28]. The corresponding PMR and time-correlation coefficient are 2.49 and 0.85, respectively.

In this sub-section, we compare the energy performance of the proposed adaptive scheduler against the corresponding ones of some state-of-the-art schedulers, e.g., the GRADient-based Iterative Scheduler (GRADIS) in [25], the Lyapunov-based scheduler (Lyap) in [11], the static scheduler in [22], and the NetDC scheduler in [10]. Goal of this last set of

TABLE 3: Average energy loss (in percent) of the proposed consolidation algorithm against the optimal one. The application scenario of Section 6.5.1 is considered at $M = 10$ and $E_{ave} = 0.5$ (Joule).

<table>
<thead>
<tr>
<th>$k_e$</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 (mJ/(MHz)^2)</td>
<td>1.2%</td>
<td>0.8%</td>
<td>0.5%</td>
</tr>
<tr>
<td>5 (mJ/(MHz)^2)</td>
<td>0.87%</td>
<td>0.7%</td>
<td>0.3%</td>
</tr>
</tbody>
</table>

An examination of the results of Table 3 leads to three main conclusions. First, the actual rate of occurrence of consolidation events increases by passing from Case 1 to Case 3. This induces, in turn, a reduction in the performance penalty suffered by the proposed consolidation algorithm, whose chances of convergence to the optimal consolidated configurations increase, indeed, for increasing rate of the occurrence of the consolidation events. Second, the energy penalty suffered by the proposed consolidation algorithm is nearly negligible and, at least in the carried out tests, it remains, indeed, limited up to 1.2%, even at high values of $k_e$. Third, since the rate of the occurrence of VM underutilization phenomena increases for increasing values of $k_e$ and too frequent occurrence of VM underutilization phenomena tends to increase the resource reconfiguration actions to be carried out at the consolidation instants, the energy penalty suffered by the proposed consolidation algorithm tends to somewhat increases for growing values of $k_e$.

The plots of Fig.7 report the corresponding (numerically evaluated) time-behaviors of the average utilization $\overline{U}(.)$ in (22) at $k_e = 5$ (mJ/(MHz)^2). A comparison of the plots of

Fig. 7: Time-behaviors of the numerically evaluated average utilization $\overline{U}(.)$ under the application scenario of Section 6.5.1 at $M = 10$ and $E_{ave} = 0.5$ (Joule). The horizontal dotted lines mark the upper/lower utilization thresholds of Cases 1, 2 and 3.

Fig.7 with the arrival trace of Fig.6 confirms that the proposed consolidation algorithm is capable to track the abrupt changes of the input arrival sequence by scaling up/down the number of turned ON VMs, while guaranteeing that the average utilization in (22) falls into the desired target intervals.
numerical tests is to (numerically) evaluate and compare the reductions in the overall average energy consumption: \( E_e + E_{dyn} + E_{LAN} \), induced by the adaptive resource scaling and VM consolidation performed by the proposed scheduler. This is motivated by the fact that current data centers usually rely on static resource provisioning, so that, by design, a fixed number of VMs constantly runs at the maximum processing rate \( f_{max} \) in order to provide the computing capacity needed to satisfy the peak input traffic \( a_{max} \). Although the resulting static scheduler does not suffer by the energy costs arising from the adaptive resource scaling and consolidation, it induces overbooking of the used resources. Hence, the average energy consumption: \( E^{\text{(STS)}} \triangleq E_e^{\text{(STS)}} + E_{dyn}^{\text{(STS)}} + E_{LAN}^{\text{(STS)}} \), of the Static Scheduler (STS) provides a basic performance benchmark [22].

In order to carry out fair energy comparisons, in the tests of this sub-section, we have implemented the proposed scheduler by directly setting: \( a(t) = \lambda(t) = \nu(t) \), together with \( \theta = 1 \) and \( \sigma(t) = \infty \). In so doing, the objective function in (17.1) to be minimized by the proposed scheduler reduces to the summation of the computing and communication energies consumed by the Fog infrastructure of Fig.1b. Hence, on the basis of the carried out numerical tests, we have experienced that: (i) the average energy reduction of the proposed scheduler over the STS is of about 47% at \( k_e = 500 \) (mJ/(MHzs)), while it increases up to about 56% at \( k_e = 5 \) (mJ/(MHzs)); and, (ii) at least in the carried numerical tests, a fraction of about 35%–40% of these energy savings is provided by the performed consolidation actions. These results confirm that the proposed consolidation algorithm provides an effective means for leveraging the sudden time-variations exhibited by the input traffic.

Finally, in order to evaluate the energy reduction induced by scaling up/down the intra-Fog processing and communication rates, we have also tested the schedulers proposed in [10], [11], [25]. Table 4 reports the obtained energy consumptions averaged over the corresponding average numbers of actually turned-ON VMs. According to Table 4, the average energy savings of the proposed scheduler over the NetDC scheduler in [10], the Lyapunov-based scheduler in [11] and the GRADIS scheduler in [25] approach about 60%, 20% and 33% at medium/large values of \( M \). This confirms that the proposed scheduler is capable to effectively adapt to the time-varying fluctuations of the input traffic.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we developed a cognitive computing-inspired scheduler for the joint adaptive tuning of the: (i) input traffic; (ii) output traffic; and, (iii) resource reconfiguration and consolidation, of virtualized Fog platforms that support vehicular TCP/IP connections. The overall goal is the energy-saving support of QoS-demanding computing-intensive delay-sensitive I2V services. Remarkable features of the developed joint scheduler are that: (i) its implementation is distributed and adaptive; (ii) it minimizes the energy consumed by the overall NetFC platform for computing, intra-Fog communication and wireless transmission over the vehicular TCP/IP connection; and, (iii) despite the unpredictable time-varying nature of the states of underlying TCP/IP vehicular connections, it is capable to provide hard QoS guarantees, in terms of minimum per-connection instantaneous wireless transmission rate, maximum instantaneous rate-jitter and maximum queuing-plus-computing delay. Actual performance of the proposed scheduler has been numerically tested under both synthetic and real-world input traffic, various mobility conditions and settings of the networked Fog platform. This work can be extended in some directions of potential interest. Just as an example, closed networked multi-tier computing infrastructures may be considered for the support of delay-tolerant session-based services [33]. Since, in this application scenario, intra-slot traffic arrivals could be allowed, live migration of VMs could be also forecast for attaining additional energy savings. Optimizing live migration of VMs without resorting to exhaustive NP-hard numerical approaches could be an interesting topic for future work.

REFERENCES

TABLE 4: Average computing-plus-communication energy consumptions under the application scenario of Section 6.5.2.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>11.5</td>
<td>16.7</td>
<td>15.7</td>
<td>12.7</td>
</tr>
<tr>
<td>8</td>
<td>8.2</td>
<td>15.2</td>
<td>14.1</td>
<td>9.8</td>
</tr>
<tr>
<td>10</td>
<td>6.8</td>
<td>13</td>
<td>13.8</td>
<td>7.4</td>
</tr>
<tr>
<td>12</td>
<td>5.5</td>
<td>11.5</td>
<td>13.7</td>
<td>6.5</td>
</tr>
<tr>
<td>14</td>
<td>5.1</td>
<td>9.7</td>
<td>13.5</td>
<td>6.0</td>
</tr>
<tr>
<td>16</td>
<td>4.9</td>
<td>8.2</td>
<td>13.3</td>
<td>5.6</td>
</tr>
<tr>
<td>18</td>
<td>4.2</td>
<td>7.1</td>
<td>13.2</td>
<td>5.4</td>
</tr>
<tr>
<td>20</td>
<td>3.8</td>
<td>6.8</td>
<td>13.0</td>
<td>5.2</td>
</tr>
</tbody>
</table>


Mohammad Shojafor is currently last year Ph.D. student in information and communication engineering at DIET Dept. of the Sapienza University of Rome. He received his MSc in software engineering in Qazvin Islamic Azad University, Qazvin, Iran in 2010. Also, he received his BSc. in computer engineering-software major in Iran university science and technology, Tehran, Iran in 2006. His current research focuses on distributed computing, wireless communications, and mathematical/AI optimization. Mohammad was a programmer and analyzer in exploration directorate section at National Iranian Oil Company (NIOC) and Tidewater ltd. Co. in Iran from 2008-2013, respectively.

Nicola Cordeschi is a Researcher in Information and Communication Engineering at the DIET Dept. of Sapienza University of Rome. He received the five-year Laurea degree (summa cum laude) in Communication Engineering and the Ph.D. degree in Information and Communication Engineering from the Sapienza University of Rome in 2004 and 2008, respectively. His research activity focuses on data communication, distributed computing and design/optimization of QoS transmission systems for wireless/ wired multimedia applications.

Enzo Baccarelli is Full Professor in Information and Communication Engineering at the DIET Dept. of Sapienza University of Rome. He received the Laurea degree in Electronic Engineering and the Ph.D. degree in Communication Theory and Systems, both from Sapienza University. In 1995, he received the Post-Doctorate degree in Information Theory and Applications from the INFOCOM Dept. of Sapienza University of Rome. His current research focuses on data networks, distributed computing and adaptive optimization.