Multimedia Tools and Applications (2019) 78:24639–24655
https://doi.org/10.1007/s11042-018-7051-9

A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments

Georgios L. Stavrinides[1] · Helen D. Karatza[1]

Received: 18 September 2018 / Revised: 24 November 2018 / Accepted: 7 December 2018 /
Published online: 20 December 2018

Abstract

In this paper, we propose a hybrid fog and cloud-aware heuristic for the dynamic scheduling of multiple real-time Internet of Things (IoT) workflows in a three-tiered architecture. In contrast to traditional approaches where the main processing of IoT jobs is performed in the fog layer, our approach attempts to schedule computationally demanding tasks with low communication requirements in the cloud and communication intensive tasks with low computational demands in the fog, utilizing possible gaps in the schedule of the fog and cloud virtual machines. Furthermore, during the scheduling process, our approach takes into account the communication cost incurred by the transfer of data from the sensors and devices in the IoT layer to the fog layer. The performance of the proposed heuristic is evaluated and compared via simulation to a baseline cloud-unaware strategy, under different cases of workload. The simulation results reveal that the proposed scheduling heuristic provides on average 76.69% lower deadline miss ratio, compared to the baseline policy. However, this is achieved at a significant monetary cost, due to the usage of cloud resources.

Keywords Internet of Things · Fog computing · Cloud computing · Real-time workflows · Scheduling

1 Introduction

As the *Internet of Things (IoT)* continues to encompass a wide spectrum of devices and sensors, an unprecedented volume and variety of data is generated at staggering speeds, often requiring processing within strict time constraints, in a *real-time* manner [7, 10, 23]. Healthcare and traffic monitoring IoT devices and sensors are examples of such time-critical cases [9, 19, 30]. Often the IoT data are processed by real-time jobs, consisting of tasks with precedence constraints among them, forming a *workflow*, where the output data of a task

✉ Georgios L. Stavrinides
gstavrin@csd.auth.gr

Helen D. Karatza
karatza@csd.auth.gr

[1] Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

are used as input by other tasks. A task in the workflow without any parent tasks is called an *entry task*, whereas a task without any child tasks is called an *exit task* [5, 25–27].

Due to the explosive growth of the IoT, *fog computing* emerged as a new paradigm, complementing cloud computing. The fog extends the cloud to the network edge, close to where the IoT data are generated, instead of sending vast amounts of IoT data to the cloud, as physical proximity affects end-to-end latency [2, 6, 11]. Typically, only selected data are sent to the cloud, for example for further historical analysis. The heterogeneous computational resources in the fog can be virtualized, as in the case of cloud computing. Consequently, a fog node can be a virtual machine (VM) [16]. However, the number and computational capacity of the resources in a fog platform are typically limited, compared to those in a cloud environment.

1.1 Motivation

As the volume, variety and velocity of the generated IoT data continue to increase, their real-time processing requires resources with higher computational capacity than those traditionally found in a fog environment. On the other hand, the computational capacity of the cloud is virtually unlimited, but entails a higher communication latency, as well as monetary cost [17].

Consequently, the workload operating on IoT data should be appropriately distributed to resources in both the fog and cloud layers, taking into account the computational and communication characteristics of each individual job [14]. The workload orchestration in such a framework requires the utilization of an effective and dynamic fog and cloud-aware scheduling heuristic. This is especially crucial in time-critical environments, such as traffic control systems [13].

1.2 Contribution

Towards this direction, in this paper we propose a hybrid fog and cloud-aware heuristic for the dynamic scheduling of multiple real-time IoT workflows in a three-tiered architecture. In contrast to traditional approaches where the main processing of IoT jobs is performed in the fog layer, our approach attempts to schedule computationally demanding tasks with low communication requirements in the cloud and communication intensive tasks with low computational demands in the fog, utilizing possible gaps in the schedule of the fog and cloud VMs. Furthermore, during the scheduling process, our approach takes into account the communication cost incurred by the transfer of data from the sensors and devices in the IoT layer to the VMs in the fog layer.

The remainder of the paper is organized as follows: Section 2 provides an overview of related literature. Section 3 presents the system and workload models, as well as the employed cloud pricing scheme. Section 4 describes the proposed scheduling heuristic, while Section 5 gives a description of the performance metrics, the experimental setup and analyzes the simulation results. Finally, Section 6 summarizes and concludes the paper.

2 Related work

One of the most well-known workflow scheduling techniques in heterogeneous environments, is the *Heterogeneous Earliest Finish Time (HEFT)* strategy, proposed by Topcuoglu et al. [29]. It consists of a task selection phase and a processor selection phase. During the task selection phase, tasks are prioritized according to their position in the workflow graph and the task with the highest priority is selected. Subsequently, in the processor selection phase, the selected task is assigned to the processor that can provide it with the earliest finish time, utilizing idle time slots in the processor's schedule. In [1], Arabnejad and Barbosa propose the *Predict Earliest Finish Time (PEFT)* strategy, which is essentially an enhanced version of the HEFT policy. It introduces a look ahead feature based on an optimistic cost table. The authors show that their proposed approach outperforms HEFT in terms of the scheduling length ratio metric.

Jiang et al. in [12], present a novel clustering algorithm, the *Path Clustering Heuristic with Distributed Gap Search (PCH-DGS)*, for the scheduling of multiple workflows in a heterogeneous cloud. Their proposed method tries to insert each group of tasks into the first available schedule gap in a processor's schedule. The tasks of a workflow are partitioned into groups in an attempt to minimize the communication cost between them. In case the time gap cannot accommodate all of the tasks of the group, the rest of the group's tasks are inserted into the next available gap in the schedule of the same or other processor in the system, in a recursive manner. Even though all of the above algorithms are suitable for scheduling workflows in a heterogeneous environment, however, they do not consider the characteristics of a fog and cloud architecture. More importantly, they are static and they do not take into account any timing constraints.

Scheduling in hybrid fog and cloud environments has been attracting more and more attention [4, 20, 28]. A workload allocation approach in a fog-cloud architecture is proposed in [8] by Deng et al. The authors investigate the tradeoff between power consumption and transmission delay in the two-tiered architecture. Their approach attempts to determine the optimal workload allocation between the fog and cloud layers, based on these two factors. Based on simulation experiments and analytical solutions, it is shown that by sacrificing modest computational resources in order to save communication bandwidth and reduce transmission latency through the proposed approach, fog computing can significantly improve the performance of cloud computing. However, the proposed approach cannot be applied to workflow applications, as data dependencies and precedence constraints are not considered between the tasks of the workload.

In [15], Nan et al. propose an online algorithm, called *Unit-slot Optimization*, for scheduling applications in a three-tiered architecture, consisting of an IoT layer, a fog layer and a cloud layer. The fog nodes do not involve any monetary cost for processing the applications, but have limited computational capacity. On the other hand, the cloud nodes are more computationally capable, but involve monetary cost. A portion of the applications that arrive at the fog layer are offloaded to the cloud layer in an attempt to find a balance between the average application response time and the average monetary cost. The proposed approach dynamically adjusts the tradeoff between these two factors, based on the technique of Lyapunov optimization. It is shown that the proposed approach can provide cost-effective processing, while guaranteeing average response time. However, even though the response time of the applications is considered in this work, no real-time constraints (i.e. deadlines) are taken into account. Furthermore, the proposed policy is not suitable for scheduling workflow applications, as no inter-task dependencies are considered.

On the other hand, a first attempt to workflow scheduling based on collaboration between cloud and fog computing is presented by Pham et al. in [18]. The major objective of the proposed heuristic, *Cost-Makespan aware Scheduling (CMaS)*, is to achieve a tradeoff between the application time and the cost of the use of cloud resources, under user-defined constraints. Even though this approach is both fog and cloud-aware and is suitable for real-time workflows, utilizing idle time slots during the scheduling process, however, it exhibits the following drawbacks:

– It is static and thus not practically suitable for the dynamic nature of IoT applications.
– It only considers a single workflow for scheduling.
– During the scheduling process, it does not take into account the communication cost incurred by the transfer of data from the IoT layer.

On the contrary, the fog and cloud-aware heuristic proposed in this paper is suitable for the dynamic scheduling of multiple real-time workflows, utilizing possible schedule gaps. Furthermore, during the scheduling process, it takes into account the communication cost incurred by the transfer of data from the sensors and devices in the IoT layer to the VMs in the fog layer.

3 Problem formulation

3.1 System model

The three-tiered environment under study is depicted in Fig. 1. The IoT layer consists of sensors and devices that transmit data through a WiFi or cellular (4G/LTE) network to the fog. The fog layer consists of a set of fog nodes. Specifically, the fog environment has an

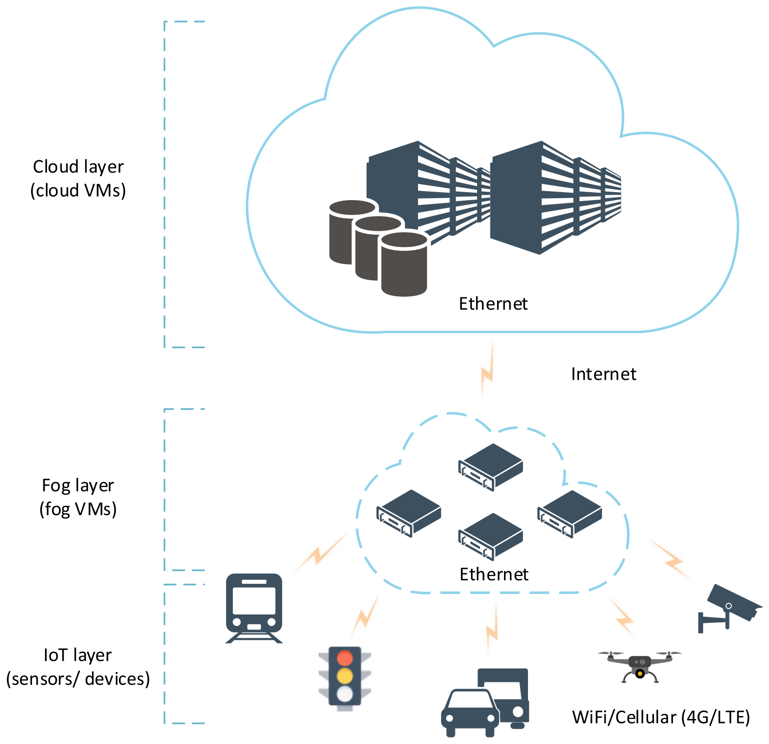

Fig. 1 The IoT, fog and cloud layers of the architecture under study

underlying infrastructure that consists of a set $\mathcal{H}^{\text{fog}} = \{host_1^{\text{fog}}, ..., host_h^{\text{fog}}\}$ of h^{fog} physical hosts with heterogeneous processors, connected via Ethernet. Each host $host_i^{\text{fog}}$ has a multi-core processor that consists of c_i^{fog} identical cores. There is a set $\mathcal{V}^{\text{fog}} = \{vm_1^{\text{fog}}, ..., vm_v^{\text{fog}}\}$ of v^{fog} VMs in the fog, where each VM is assigned a virtual CPU (vCPU).

The cloud layer consists of a set of reserved dedicated cloud nodes. Specifically, there is a set $\mathcal{H}^{\text{cloud}} = \{host_1^{\text{cloud}}, ..., host_h^{\text{cloud}}\}$ of h^{cloud} physical hosts with heterogeneous processors, connected via Ethernet. Each host $host_i^{\text{cloud}}$ has a multi-core processor that consists of c_i^{cloud} identical cores. There is a set $\mathcal{V}^{\text{cloud}} = \{vm_1^{\text{cloud}}, ..., vm_v^{\text{cloud}}\}$ of v^{cloud} VMs in the cloud, where each VM is assigned a virtual CPU (vCPU). The vCPU of a fog or cloud VM corresponds to a physical core of the respective host. The operating frequency f_i of a VM vm_i corresponds to the operating frequency of its assigned physical core. All of the cores in the fog and cloud layers require the same number of clock cycles per instruction.

The data transfer rate between the IoT devices and sensors and the fog layer is denoted by l^{IoT} and is uniformly distributed in the range $\left[\overline{l^{\text{IoT}}} \cdot \left(1 - L^{\text{IoT}}/2\right), \overline{l^{\text{IoT}}} \cdot \left(1 + L^{\text{IoT}}/2\right)\right]$, where L^{IoT} is the heterogeneity degree of the network that connects the IoT layer and the fog layer, whereas $\overline{l^{\text{IoT}}}$ is the mean data transfer rate between the two layers.

The VMs in the fog and cloud layers are fully connected by a virtual network that connects the two layers over the Internet (e.g. through a site-to-site VPN). The data transfer rate between two fog VMs vm_i^{fog} and vm_j^{fog} is denoted by l_{ij}^{fog} and is uniformly distributed in the range $\left[\overline{l^{\text{fog}}} \cdot \left(1 - L^{\text{fog}}/2\right), \overline{l^{\text{fog}}} \cdot \left(1 + L^{\text{fog}}/2\right)\right]$, where L^{fog} is the heterogeneity degree of the virtual network in the fog layer, whereas $\overline{l^{\text{fog}}}$ is the mean data transfer rate of the respective communication links.

On the other hand, the data transfer rate between two cloud VMs vm_i^{cloud} and vm_j^{cloud} is denoted by l_{ij}^{cloud} and is uniformly distributed in the range $\left[\overline{l^{\text{cloud}}} \cdot \left(1 - L^{\text{cloud}}/2\right), \overline{l^{\text{cloud}}} \cdot \left(1 + L^{\text{cloud}}/2\right)\right]$, where L^{cloud} is the heterogeneity degree of the virtual network in the cloud layer, whereas $\overline{l^{\text{cloud}}}$ is the mean data transfer rate of the respective communication links. Finally, the data transfer rate between a fog VM vm_i^{fog} and a cloud VM vm_j^{cloud} is denoted by l_{ij}^{inter} and is uniformly distributed in the range $\left[\overline{l^{\text{inter}}} \cdot \left(1 - L^{\text{inter}}/2\right), \overline{l^{\text{inter}}} \cdot \left(1 + L^{\text{inter}}/2\right)\right]$, where L^{inter} is the heterogeneity degree of the virtual network that connects the two layers, whereas $\overline{l^{\text{inter}}}$ is the mean data transfer rate of the respective communication links. It is noted that the superscript indicators in the variables names are used in order to differentiate between the variables corresponding to each layer. There is a fog and cloud-aware central scheduler running on a dedicated host in the fog layer that is responsible for scheduling the tasks to the VMs in the fog and the cloud.

3.2 Workload model

The data generated and transmitted to the fog by the devices and sensors of the IoT layer, are processed by multiple real-time workflow jobs, which arrive dynamically at the central scheduler in a Poisson stream with rate λ. Each workflow job is represented by a *directed acyclic graph (DAG)* $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of the nodes of the graph and $\mathcal{E}$ is the set of the directed edges between the nodes. Each node represents a component task n_i of the workflow, whereas a directed edge e_{ij} between two tasks n_i and n_j represents the data that must be transferred from task n_i to task n_j. The component tasks of a workflow are not preemptible, as preemption of real-time tasks may lead to performance degradation [3, 22, 24]. In the rest of the paper, the terms job, workflow and DAG are used interchangeably.

Each task n_i has a weight w_i that denotes its *computational volume*, i.e. the number of clock cycles required to execute the instructions of the particular task. The computational volume of each task is exponentially distributed with mean $\overline{w}$. The *computational cost* of the task n_i on a VM vm_j is given by:

$$Comp(n_i, vm_j) = w_i / f_j \tag{1}$$

where f_j is the operating frequency of VM vm_j.

Each edge e_{ij} between two tasks n_i and n_j has a weight z_{ij} that represents its *communication volume*, i.e. the number of GB of data needed to be transferred between the two tasks. The communication volume of each edge is exponentially distributed with mean $\overline{z}$. The *communication cost* of the edge e_{ij} is incurred when data are transferred from task n_i (scheduled on VM vm_m) to task n_j (scheduled on VM vm_n) and is defined as:

$$Comm\left((n_i, vm_m), (n_j, vm_n)\right) = z_{ij} / l_{mn} \tag{2}$$

where l_{mn} is the data transfer rate of the communication link between the VMs vm_m and vm_n, which may belong to the same layer, i.e. fog (l_{mn}^{fog}) or cloud (l_{mn}^{cloud}), or different layers, i.e. one in fog and one in cloud (l_{mn}^{inter}). In case both tasks n_i and n_j are scheduled on the same VM or on VMs that run on the same physical host, the communication cost of the edge e_{ij} is considered negligible.

Each entry task of a workflow requires input data that may vary in size. The *input data size* d_i of an entry task n_i is exponentially distributed with mean $\overline{d}$. The communication cost incurred by the transfer of input data from the IoT layer to a task n_i scheduled on a fog VM vm_m, is given by:

$$Comm\left(n_i, vm_m^{\text{fog}}\right) = d_i / l^{\text{IoT}} \tag{3}$$

where l^{IoT} is the data transfer rate between the IoT and the fog layer. In case the input data are required to be transferred from the IoT layer to the cloud layer (i.e. to a cloud VM vm_n), they are first uploaded to the fog layer and then forwarded to the cloud layer, incurring an additional overhead. Hence, the communication cost in this case is given by:

$$Comm\left(n_i, vm_n^{\text{cloud}}\right) = d_i \cdot \left(1/l^{\text{IoT}} + 1/l^{\text{inter}}\right) \tag{4}$$

The length of a path in the graph is the sum of the computational and communication costs of all of the tasks and edges, respectively, on the path, including the input data communication cost of the respective entry task on the particular path. The *critical path length* CPL is the length of the longest path in the graph. Each real-time job has an end-to-end *firm deadline* D within which all of its component tasks must finish execution. It is defined as:

$$D = A + RD \tag{5}$$

where A is the *arrival time* of the workflow and RD is its *relative deadline*, which is uniformly distributed in the range $[CPL, 2CPL]$. In the time-critical environment under study, the deadline of each job must be met, otherwise its results would be useless. Therefore, in such a case, the job is considered lost.

The *communication to computation ratio* CCR of a workflow is the ratio of its average communication cost to its average computational cost on the target system and is given by:

$$CCR = \frac{\sum_{e_{ij} \in \mathcal{E}} \overline{Comm(e_{ij})}}{\sum_{n_i \in \mathcal{N}} \overline{Comp(n_i)}} \tag{6}$$

where $\mathcal{N}$ and $\mathcal{E}$ are the sets of the nodes and the edges of the workflow, respectively. $\overline{Comm(e_{ij})}$ is the average communication cost of the edge e_{ij} over all of the communication links in the system, whereas $\overline{Comp(n_i)}$ is the average computational cost of the task n_i over all of the VMs in the system. An example of a workflow job is shown in Fig. 2.

3.3 Cloud pricing scheme

As mentioned earlier, the cloud layer consists of reserved dedicated physical hosts, each of which is charged at an effective average hourly rate C_{host}. Furthermore, the data that are transferred in and out of the cloud are charged per TB at a rate C_{data}.

4 Hybrid fog and cloud-aware scheduling heuristic

A hybrid heuristic is employed that schedules a ready task of a workflow to a fog or a cloud VM, depending on the task's potential communication and computational cost. Specifically, in contrast to traditional approaches where the main processing of IoT jobs is performed in the fog layer, our approach attempts to schedule computationally demanding tasks with low communication requirements in the cloud (which has resources with greater computational capacity than the fog, but higher communication latency between the cloud VMs and the

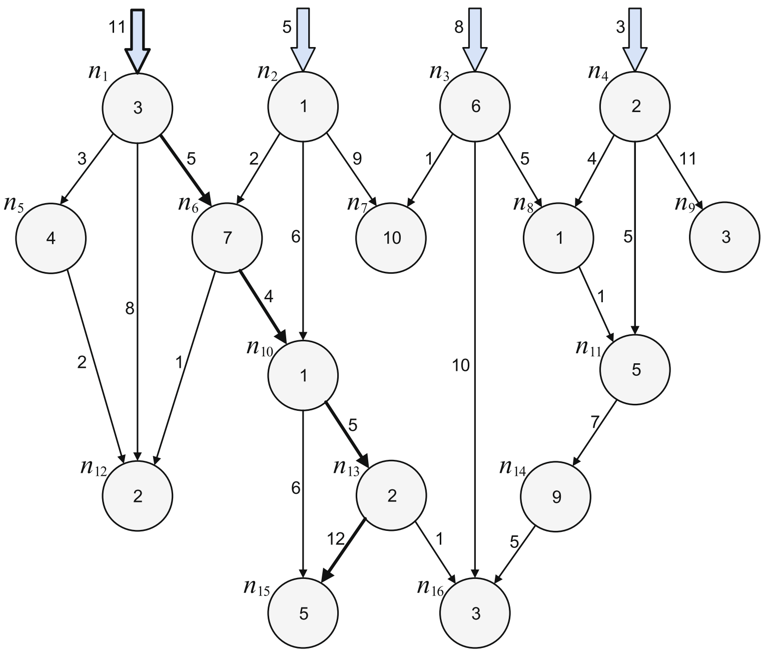

Fig. 2 An IoT data-processing workflow represented as a directed acyclic graph with four entry tasks and five exit tasks. The number in each node denotes the average computational cost of the represented task. The number on each edge denotes the average communication cost between the two tasks that it connects. The blue arrows pointing to the entry tasks of the graph indicate the average communication cost incurred by the transfer of the required input data from the IoT devices. The critical path of the graph is depicted with thick arrows

IoT layer) and communication intensive tasks with low computational demands in the fog (which has limited computational resources, but lower communication latency between the fog VMs and the IoT layer, compared to the cloud), utilizing possible gaps in the schedule of the fog and cloud VMs. Furthermore, during the scheduling process, our approach takes into account the communication cost incurred by the transfer of data from the sensors and devices in the IoT layer to the VMs in the fog layer. The proposed scheduling strategy consists of two phases: (a) a *task selection phase* and (b) a *VM selection phase*.

4.1 Task selection phase

Tasks are prioritized according to their job's deadline. The task that its job has the earliest deadline, has the highest priority. Consequently, tasks are prioritized according to the *Earliest Deadline First (EDF)* policy. In case two or more ready tasks have the same priority, the task with the highest average computational cost is selected first.

4.2 VM selection phase

Once a task is selected by the scheduler, it is allocated to the VM that can provide it with the earliest *estimated finish time* EFT. All VMs in the fog and cloud layers are considered. The EFT of a ready task n_i on a VM vm_k is given by

$$\begin{aligned} EFT(n_i, vm_k) = \\ \max\{t_{\text{data}}(n_i, vm_k), t_{\text{idle}}(n_i, vm_k)\} + \\ Comp(n_i, vm_k) \end{aligned} \tag{7}$$

where $t_{\text{data}}(n_i, vm_k)$ is the time at which all input data of task n_i will be available on VM vm_k, whereas $t_{\text{idle}}(n_i, vm_k)$ is the time at which vm_k will be able to execute task n_i.

In order to calculate the term $t_{\text{idle}}(n_i, vm_k)$, we must determine the position that task n_i would be placed in the queue of VM vm_k. Firstly, we find the initial position at which the ready task n_i would be placed in the VM's queue, according to its priority. Subsequently, we check whether a schedule gap exists that can be utilized by the task, as follows:

- **Step 1:** In case all of the required input data of the ready task n_i are available on VM vm_k (i.e. $t_{\text{data}}(n_i, vm_k) = t_{\text{current}}$, where t_{current} is the current time), we check whether a schedule gap exists. A schedule gap is formed when the VM is idle and the task n_q placed at the head of the queue is still in the process of receiving its required input data from other hosts. The capacity g of the schedule gap is calculated as:

$$g = t_{\text{data}}(n_q, vm_k) - t_{\text{current}} \tag{8}$$

  where $t_{\text{data}}(n_q, vm_k)$ is the time at which all of the required input data of task n_q will be received.
- **Step 2:** If a schedule gap exists, we try to fill it in with the ready task n_i:

$$g \geq w_i / f_k \tag{9}$$

  where w_i is the communication volume of task n_i and f_k is the operating frequency of VM vm_k. In case the ready task n_i cannot be placed into a schedule gap or a schedule gap does not exist, the position of task n_i in vm_k's queue is determined only by its priority.

The pseudocode corresponding to the above procedure is shown in Algorithm 1. The first step of the procedure is described in lines 5-10, whereas the second step is described in lines 12-16. The utilization of schedule gaps is also performed in the same manner for a task waiting in a queue when all of its input data become available on its assigned VM. Furthermore, it is also performed for tasks that are waiting for service in a queue and either the task in service completes execution or a task is discarded from the queue because its job's deadline has been reached. In the last two cases, eligible tasks are considered according to their priority.

We refer to our proposed scheduling heuristic as *Hybrid-EDF*. For comparison purposes and in order to examine the same heuristic, but in a cloud-unaware setting, an alternative version of our proposed approach was considered in our experiments, *Fog-EDF*. This alternative baseline approach only considers VMs in the fog layer during the VM selection phase. That is, no cloud VMs are utilized for the processing of the workflows.

Algorithm 1 Schedule gap utilization.

```
Input: A ready task $n_i$ and a VM $vm_k$.
Output: Decision on whether a gap in $vm_k$'s schedule can be utilized by task $n_i$.
 1: /* Initialization of variables*/
 2: $gapExists \leftarrow$ false
 3: $gapIsUtilized \leftarrow$ false
 4: /* Step 1: check if a schedule gap exists */
 5: if $t_{\text{data}}(n_i, vm_k) = t_{\text{current}}$ then
 6:     if $vm_k$ is idle and $t_{\text{data}}(n_q, vm_k) > t_{\text{current}}$ then
 7:         $gapExists \leftarrow$ true
 8:         $g \leftarrow t_{\text{data}}(n_q, vm_k) - t_{\text{current}}$
 9:     end if
10: end if
11: /* Step 2: check if the task fits into the gap */
12: if $gapExists$ then
13:     if $g \geq w_i/f_k$ then
14:         $gapIsUtilized \leftarrow$ true
15:     end if
16: end if
17: return $gapIsUtilized$
```

5 Performance evaluation

5.1 Performance metrics

The following metrics were employed for the evaluation of the performance of our proposed scheduling heuristic, Hybrid-EDF, and its comparison to the baseline strategy, Fog-EDF:

- *Deadline Miss Ratio*, which is the ratio of the number of jobs that did not finish their execution within their deadline (and thus lost), over the number of all of the jobs that arrived at the central scheduler of the fog layer, during the observed time period.
- *Percentage of Tasks Executed on Cloud*, which is the percentage of tasks of completed jobs that were executed on cloud VMs, during the observed time period.
- *Total Monetary Cost*, which is the total monetary cost (in US dollars) for the utilization of the resources of the cloud layer. This cost concerns the reserved dedicated hosts in the cloud layer, as well as the data transfers in and out of the cloud, during the observed time period.

5.2 Experimental setup

The performance of the system was evaluated by conducting a series of simulation runs using the independent replications method. Due to the complexity of the system and the workload model under study and in order to have full control on all of the required parameters, we implemented our own discrete-event simulation program in C++, tailored to the specific requirements of the particular problem. The hosts in the fog and cloud layers were based on real-world processors (one processor per host). The fog layer consisted of a small number of hosts with low to moderate computational capacity. Specifically, the fog layer consisted of 1 small (in terms of operating frequency) processor, modeled after the Intel Xeon Bronze 3106 processor, 1 medium processor, modeled after the Intel Xeon Silver 4108 processor, and 1 large processor, modeled after the Intel Xeon Silver 4109T processor. On the other hand, the cloud layer consisted of a larger number of hosts with greater computational capacity than those in the fog layer. Specifically, the cloud layer consisted of 5 small processors, modeled after the Intel Xeon Platinum 8158 processor, 10 medium processors, modeled after the Intel Xeon Gold 6134 processor, and 10 large processors, modeled after the Intel Xeon Platinum 8156 processor. The operating frequency and the number of cores of each processor are shown in Table 1.

In order to directly control the workload parameters and obtain unbiased, general results, not applicable only to particular workload traces, synthetic workload was used. The workflows were generated randomly, using our own random DAG generator, as described in [21]. Each generated workflow was a weakly connected graph, having a path between any pair of tasks, without taking into account the direction of the edges. There was at least one entry and one exit task in each generated workflow. We conducted three sets of experiments: (a) for computationally intensive ($CCR = 0.5$), (b) moderate ($CCR = 1$) and (c) communication intensive ($CCR = 2$) workflows.

For computationally intensive workflows ($CCR = 0.5$), the mean computational volume of the tasks was selected to be equal to $\overline{w} = 1.1 \cdot 10^{12}$ clock cycles, so that on average, a task would take 10 minutes to execute on a fog VM. For moderate and communication intensive workflows ($CCR = \{1, 2\}$), the mean computational volume of the tasks was selected to be equal to $\overline{w} = 0.55 \cdot 10^{12}$ clock cycles, so that on average, a task would take half the time (i.e. 5 minutes) to execute on a fog VM, compared to the computationally intensive case.

For each CCR and mean computational volume $\overline{w}$, the mean communication volume $\overline{z}$ was calculated from (6). In order for the system to be stable, the job arrival rate was chosen to be $\lambda = 0.002$. As we wanted to examine data-intensive workflows, the mean input data size of the entry tasks was chosen to be $\overline{d} = 100$ GB. In order to be in line with the prices of real-world cloud vendors, such as Amazon Web Services and Google Cloud Platform, the reserved dedicated cloud host effective average (for compute optimized hosts) hourly rate was chosen to be $C_{host} = \$1$ per host, whereas the data transfer rate for transfers in and out of the cloud was chosen to be $C_{data} = \$1$ per TB. The heterogeneity degree of the networks in all layers was chosen to be equal to $L = 0.5$, since most modern networks feature moderate heterogeneity. All of the input parameters of the simulation model are shown in Table 1.

We ran 30 replications of the simulation with different seeds of random numbers, for each set of input parameters. Each replication was terminated when 10^4 workflows had been completed. We found by experimentation that this simulation run length was sufficiently long enough to minimize the effects of warm-up time. For every mean value, a 95% confidence interval was calculated. The half-widths of all of the confidence intervals were less

Table 1 Simulation input parameters

Parameter	Value
System Model Parameters	
IoT layer:	
Mean data transfer rate (IoT-fog)	$\overline{l^{\text{IoT}}} = 50$ Mbps
Network heterogeneity degree (IoT-fog)	$L^{\text{IoT}} = 0.5$
Fog layer:	
Number of small hosts	$h_{\text{small}}^{\text{fog}} = 1$
Number of small host processor cores	$c_{\text{small}}^{\text{fog}} = 8$
Small host core operating frequency	$f_{\text{small}}^{\text{fog}} = 1.7$ GHz
Number of medium hosts	$h_{\text{medium}}^{\text{fog}} = 1$
Number of medium host processor cores	$c_{\text{medium}}^{\text{fog}} = 8$
Medium host core operating frequency	$f_{\text{medium}}^{\text{fog}} = 1.8$ GHz
Number of large hosts	$h_{\text{large}}^{\text{fog}} = 1$
Number of large host processor cores	$c_{\text{large}}^{\text{fog}} = 8$
Large host core operating frequency	$f_{\text{large}}^{\text{fog}} = 2.0$ GHz
Number of VMs	$v^{\text{fog}} = 24$
Mean data transfer rate (fog)	$\overline{l^{\text{fog}}} = 1$ Gbps
Network heterogeneity degree (fog)	$L^{\text{fog}} = 0.5$
Mean data transfer rate (fog-cloud)	$\overline{l^{\text{inter}}} = 100$ Mbps
Network heterogeneity degree (fog-cloud)	$L^{\text{inter}} = 0.5$
Cloud layer:	
Number of small hosts	$h_{\text{small}}^{\text{cloud}} = 5$
Number of small host processor cores	$c_{\text{small}}^{\text{cloud}} = 12$
Small host core operating frequency	$f_{\text{small}}^{\text{cloud}} = 3.0$ GHz
Number of medium hosts	$h_{\text{medium}}^{\text{cloud}} = 10$
Number of medium host processor cores	$c_{\text{medium}}^{\text{cloud}} = 8$
Medium host core operating frequency	$f_{\text{medium}}^{\text{cloud}} = 3.2$ GHz
Number of large hosts	$h_{\text{large}}^{\text{cloud}} = 10$
Number of large host processor cores	$c_{\text{large}}^{\text{cloud}} = 4$
Large host core operating frequency	$f_{\text{large}}^{\text{cloud}} = 3.6$ GHz
Number of VMs	$v^{\text{cloud}} = 180$
Mean data transfer rate (cloud)	$\overline{l^{\text{cloud}}} = 1$ Gbps
Network heterogeneity degree (cloud)	$L^{\text{cloud}} = 0.5$
Reserved dedicated host average hourly rate	$C_{host} = \$1$ per host
In-out data transfer rate	$C_{data} = \$1$ per TB
Workload Model Parameters	
Number of completed DAGs	10^4
DAG arrival rate	$\lambda = 0.002$
Number of tasks per DAG	$n \sim U[8, 64]$
Mean entry task input data size	$\overline{d} = 100$ GB
DAG communication to computation ratio	$CCR = \{0.5, 1, 2\}$
Mean task computational volume	$\overline{w} = 0.55 \cdot 10^{12}$ clock cycles for

Table 1 (continued)

Parameter	Value
$CCR = \{1, 2\}$ and $\overline{w} = 1.1 \cdot 10^{12}$ clock cycles for $CCR = 0.5$	
Other Model Parameters	
Scheduling heuristics	Hybrid-EDF, Fog-EDF

than 5% of their respective mean values. Furthermore, in order to evaluate whether the differences between the mean values obtained by each scheduling method were statistically significant, a 95% confidence interval was calculated for the difference between each pair of mean values. The calculated confidence intervals did not include 0 and thus the differences in the results between the employed scheduling policies were statistically significant.

5.3 Simulation results

The simulation results reveal that, in terms of the deadline miss ratio metric, the proposed fog and cloud-aware scheduling heuristic, Hybrid-EDF, outperforms the cloud-unaware baseline strategy, Fog-EDF, for all cases of workload. This is shown in Fig. 3. It is noted that in this case a logarithmic scale is used for the deadline miss ratio values, as they are highly skewed. In the case of computationally intensive ($CCR = 0.5$), moderate ($CCR = 1$) and communication intensive ($CCR = 2$) workflows, Hybrid-EDF yields a much lower deadline miss ratio than Fog-EDF. Especially in the case of computationally intensive workflows, the difference is significant. Specifically, Hybrid-EDF yields a deadline miss ratio equal to 1.51%, whereas Fog-EDF yields a deadline miss ratio equal to 85.76%.

This is due to the fact that the proposed scheduling heuristic attempts to assign computationally demanding tasks with low communication requirements to VMs in the cloud, where there is a larger number of VMs with greater computational capacity than in the fog layer. On the other hand, it tries to schedule communication intensive tasks with low

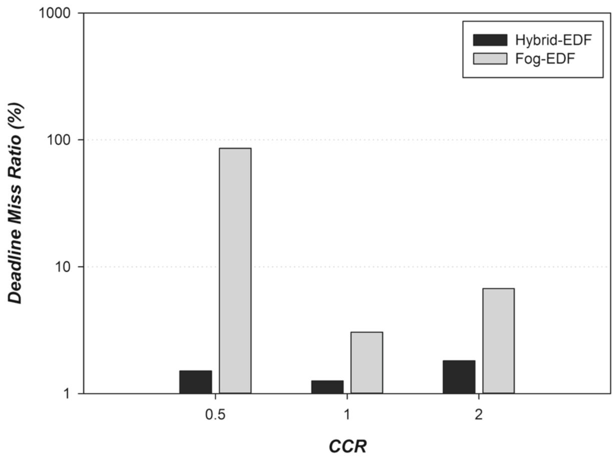

Fig. 3 Deadline Miss Ratio (%) for computationally intensive, moderate and communication intensive workflows (Hybrid-EDF & Fog-EDF)

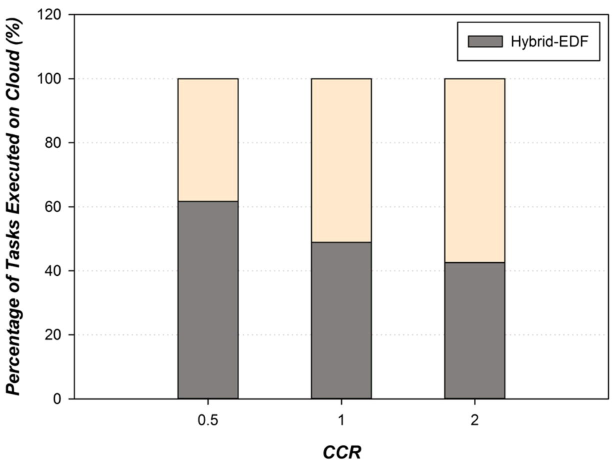

Fig. 4 Percentage of Tasks Executed on Cloud (%) for computationally intensive, moderate and communication intensive workflows (Hybrid-EDF)

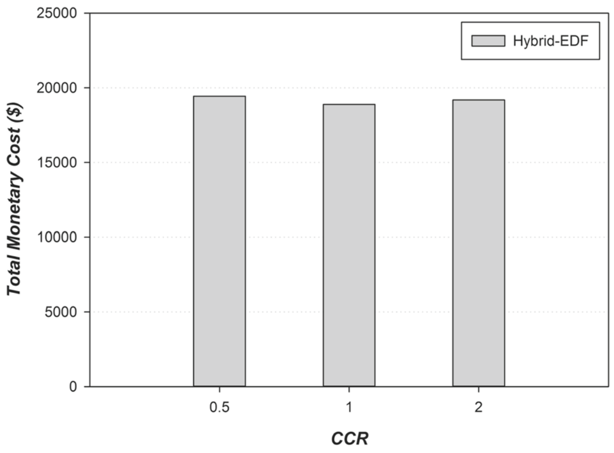

Fig. 5 Total Monetary Cost ($) for computationally intensive, moderate and communication intensive workflows (Hybrid-EDF)

Table 2 Cloud resource usage statistics (Hybrid-EDF)

	CCR		
Parameter	0.5	1	2
Number of physical hosts used	24/25	23/25	25/25
Number of VMs used	173/180	173/180	178/180
Total data transferred	1.5 PB	1.0 PB	1.2 PB
Average data transferred per job	314.76 GB	210.74 GB	253.12 GB

Table 3 Simulation results summary

	Hybrid-EDF	Fog-EDF
CCR	Deadline Miss Ratio (%)	
0.5	1.51	85.76
1	1.26	3.05
2	1.81	6.74
CCR	Percentage of Tasks Executed on Cloud (%)	
0.5	61.65	0.00
1	48.88	0.00
2	42.55	0.00
CCR	Total Monetary Cost ($)	
0.5	19,441.61	0.00
1	18,884.21	0.00
2	19,189.14	0.00

computational demands to fog VMs, which are closer to the IoT sources of the generated data, in an attempt to minimize the incurred communication cost. Overall, Hybrid-EDF provides on average 76.69% lower deadline miss ratio, compared to the baseline policy, Fog-EDF, under all cases of workload.

The scheduling decisions of the proposed strategy are clearly shown in Fig. 4, where it is apparent that for computationally intensive workflows, the majority (61.65%) of the tasks are scheduled on cloud VMs. In the case of moderate workflows, about half (48.88%) of the tasks are scheduled in the cloud and the other half in the fog. On the other hand, in the case of communication intensive workflows, the majority (57.45%) of the tasks are assigned to fog VMs. However, the impressive results of the proposed scheduling heuristic come at a significant monetary cost, as shown in Fig. 5. Specifically, for the simulated duration of about a month, the total monetary cost incurred by the data transfers in and out of the cloud and the reserved dedicated hosts in the cloud, was $19,441.61 in the case of computationally intensive workflows, $18,884.21 in the case of moderate workflows and $19,189.14 in the case of communication intensive workflows.

Thus, even though the proposed Hybrid-EDF policy outperforms the baseline Fog-EDF strategy, it requires a significant monetary cost in order to effectively utilize the cloud resources (as shown in Table 2), in addition to the fog resources, which are free. The simulation results are summarized in Table 3.

6 Conclusions and future directions

In this paper, we proposed a hybrid fog and cloud-aware heuristic, Hybrid-EDF, for the dynamic scheduling of multiple real-time IoT workflows in a three-tiered architecture. In contrast to traditional approaches where the main processing of IoT jobs is performed in the fog layer, our approach attempts to schedule computationally demanding tasks with low communication requirements in the cloud and communication intensive tasks with low computational demands in the fog, utilizing possible gaps in the schedule of the fog and cloud VMs. Furthermore, our approach takes into account during the scheduling process the communication cost incurred by the transfer of data from the sensors and devices in the IoT layer to the VMs in the fog layer.

The performance of the proposed heuristic was evaluated and compared to a baseline cloud-unaware strategy, Fog-EDF, via a series of simulation experiments, for computationally intensive, moderate and communication intensive workflows. The simulation results reveal that Hybrid-EDF outperforms Fog-EDF in the framework under study, providing on average 76.69% lower deadline miss ratio. However, this comes at a significant monetary cost, due to the usage of cloud resources. In an attempt to minimize the monetary cost, we plan to apply our approach in architectures where the cloud layer consists of on-demand multi-tenant VMs, instead of reserved dedicated hosts.

References

1. Arabnejad H, Barbosa JG (2014) List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans Parallel Distrib Syst 25(3):682–694. https://doi.org/10.1109/TPDS.2013.57
2. Bittencourt LF, Diaz-Montes J, Buyya R, Rana OF, Parashar M (2017) Mobility-aware application scheduling in fog computing. IEEE Cloud Comput 4(2):26–35. https://doi.org/10.1109/MCC.2017.27
3. Buttazzo GC (2011) Hard real-time computing systems: predictable scheduling algorithms and applications, 3rd edn. Springer, New York. https://doi.org/10.1007/978-1-4614-0676-1
4. Chen Y (2018) Service-oriented computing and system integration: software, IoT, big data, and AI as services, 6th edn. Kendall Hunt Publishing, Dubuque
5. Chen Y, Tsai WT (2015) Service-oriented computing and web software integration: from principles to development, 5th edn. Kendall Hunt Publishing, Dubuque
6. Cisco (2015) Fog computing and the Internet of Things: extend the cloud to where the things are. Tech. Rep. C11-734435-00, Cisco Systems, Inc
7. Dastjerdi AV, Buyya R (2016) Fog computing: helping the Internet of Things realize its potential. Computer 49(8):112–116. https://doi.org/10.1109/MC.2016.245
8. Deng R, Lu R, Lai C, Luan TH, Liang H (2016) Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet Things J 3(6):1171–1181. https://doi.org/10.1109/JIOT.2016.2565516
9. Gia TN, Jiang M, Rahmani A, Westerlund T, Liljeberg P, Tenhunen H (2015) Fog computing in healthcare Internet of Things: a case study on ECG feature extraction. In: Proceedings of the 13th IEEE international conference on pervasive intelligence and computing (PICom'15), pp 356–363. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.51
10. Hao Z, Novak E, Yi S, Li Q (2017) Challenges and software architecture for fog computing. IEEE Internet Comput 21(2):44–53. https://doi.org/10.1109/MIC.2017.26
11. Jararweh Y, Doulat A, AlQudah O, Ahmed E, Al-Ayyoub M, Benkhelifa E (2016) The future of mobile cloud computing: integrating cloudlets and mobile edge computing. In: Proceedings of the 23rd international conference on telecommunications (ICT'16), pp 1–5. https://doi.org/10.1109/ICT.2016.7500486
12. Jiang HJ, Huang KC, Chang HY, Gu DS, Shih PJ (2011) Scheduling concurrent workflows in HPC cloud through exploiting schedule gaps. In: Proceedings of the 11th international conference on algorithms and architectures for parallel processing (ICA3PP'11), pp 282–293. https://doi.org/10.1007/978-3-642-24650-0_24
13. Liu J, Li J, Zhang L, Dai F, Zhang Y, Meng X, Shen J (2018) Secure intelligent traffic light control using fog computing. Futur Gener Comput Syst 78(2):817–824. https://doi.org/10.1016/j.future.2017.02.017
14. Masip-Bruin X, Marín-Tordera E, Tashakor G, Jukan A, Ren G (2016) Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. IEEE Wirel Commun 23(5):120–128. https://doi.org/10.1109/MWC.2016.7721750
15. Nan Y, Li W, Bao W, Delicato FC, Pires PF, Zomaya AY (2016) Cost-effective processing for delay-sensitive applications in Cloud of Things systems. In: Proceedings of the IEEE 15th international symposium on network computing and applications (NCA'15), pp 162–169. https://doi.org/10.1109/NCA.2016.7778612
16. OpenFog (2016) OpenFog architecture overview. Tech. Rep. OPFWP001.0216, OpenFog consortium architecture working group

17. Pham XQ, Huh EN (2016) Towards task scheduling in a cloud-fog computing system. In: Proceedings of the 18th Asia-Pacific network operations and management symposium (APNOMS'16), pp 1–4. https://doi.org/10.1109/APNOMS.2016.7737240
18. Pham XQ, Man ND, Tri NDT, Thai NQ, Huh EN (2017) A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. Int J Distrib Sens Netw 13(11):1–16. https://doi.org/10.1177/1550147717742073
19. Rahmani AM, Gia TN, Negash B, Anzanpour A, Azimi I, Jiang M, Liljeberg P (2018) Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: a fog computing approach. Futur Gener Comput Syst 78(2):641–658. https://doi.org/10.1016/j.future.2017.02.014
20. Shah-Mansouri H, Wong VWS (2018) Hierarchical fog-cloud computing for IoT systems: a computation offloading game. IEEE Internet of Things 5(4):3246–3257. https://doi.org/10.1109/JIOT.2018.2838022
21. Stavrinides GL, Karatza HD (2014) Scheduling real-time jobs in distributed systems - simulation and performance analysis. In: Proceedings of the 1st international workshop on sustainable ultrascale computing systems (NESUS'14), pp 13–18
22. Stavrinides GL, Karatza HD (2017) The effect of workload computational demand variability on the performance of a SaaS cloud with a multi-tier SLA. In: Proceedings of the IEEE 5th international conference on future Internet of Things and cloud (FiCloud'17), pp 10–17. https://doi.org/10.1109/FiCloud.2017.26
23. Stavrinides GL, Karatza HD (2017) The impact of data locality on the performance of a SaaS cloud with real-time data-intensive applications. In: Proceedings of the 21st IEEE/ACM international symposium on distributed simulation and real time applications (DS-RT'17), pp 1–8. https://doi.org/10.1109/DISTRA.2017.8167683
24. Stavrinides GL, Karatza HD (2017) Simulation-based performance evaluation of an energy-aware heuristic for the scheduling of HPC applications in large-scale distributed systems. In: Proceedings of the 8th ACM/SPEC international conference on performance engineering (ICPE'17), 3rd international workshop on energy-aware simulation (ENERGY-SIM'17), pp 49–54. https://doi.org/10.1145/3053600.3053611
25. Stavrinides GL, Karatza HD (2018) Energy-aware scheduling of real-time workflow applications in clouds utilizing DVFS and approximate computations. In: Proceedings of the IEEE 6th international conference on future Internet of Things and cloud (FiCloud'18), pp 33–40. https://doi.org/10.1109/FiCloud.2018.00013
26. Stavrinides GL, Karatza HD (2018) The impact of workload variability on the energy efficiency of large-scale heterogeneous distributed systems. Simul Model Pract Theory 89:135–143. https://doi.org/10.1016/j.simpat.2018.09.013
27. Stavrinides GL, Karatza HD (2018) Scheduling data-intensive workloads in large-scale distributed systems: trends and challenges, Studies in big data, vol 36, chap 2, 1st edn. Springer, Cham, pp 19–43. https://doi.org/10.1007/978-3-319-73767-6_2
28. Taneja M, Davy A (2017) Resource aware placement of IoT application modules in fog-cloud computing paradigm. In: Proceedings of the 2017 IFIP/IEEE symposium on integrated network and service management (IM'17), pp 1222–1228. https://doi.org/10.23919/INM.2017.7987464
29. Topcuoglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274. https://doi.org/10.1109/71.993206
30. Wen Z, Yang R, Garraghan P, Lin T, Xu J, Rovatsos M (2017) Fog orchestration for Internet of Things services. IEEE Internet Comput 21(2):16–24. https://doi.org/10.1109/MIC.2017.36

Georgios L. Stavrinides received the Ph.D. degree in Computer Science from Aristotle University of Thessaloniki, Greece in 2014. He is currently a postdoctoral researcher in the Parallel and Distributed Systems Group (PDSG), at the Department of Informatics of the Aristotle University of Thessaloniki, Greece, under the supervision of Professor Emeritus Helen D. Karatza. His research interests include: scheduling algorithms, real-time distributed systems, fog and cloud computing, modeling, simulation and performance evaluation of large-scale distributed systems.

Helen D. Karatza is a Professor Emeritus in the Department of Informatics at the Aristotle University of Thessaloniki, Greece. Her research interests include: computer systems modeling and simulation, performance evaluation, grid and cloud computing, energy efficiency in large-scale distributed systems, real-time distributed systems, resource allocation and scheduling. She is the Editor-in-Chief of the Elsevier journal “Simulation Modelling Practice and Theory” and Senior Associate Editor of the “Journal of Systems and Software” of Elsevier. She has been Guest Editor of special issues in multiple international journals.