IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 May 2023, accepted 4 July 2023, date of publication 14 July 2023, date of current version 21 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3295496

RESEARCH ARTICLE

Energy-Aware Optimum Offloading Strategies in Fog-Cloud Architectures: A Lyapunov Based Scheme

NECO VILLEGAS[1], LUIS DIEZ[1], IDOIA DE LA IGLESIA[2], MARCO GONZÁLEZ-HIERRO[2], AND RAMÓN AGÜERO[1], (Senior Member, IEEE)

[1]Communications Engineering Department, University of Cantabria, 39005 Santander, Spain
[2]IoT and Digital Platforms Department, Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA), 20500 Arrasate/Mondragón, Spain

Corresponding author: Neco Villegas (villegasn@unican.es)

This work was supported by the Spanish Government (Ministerio de Economía y Competitividad, Fondo Europeo de Desarrollo Regional, MINECO-FEDER) by means of the Project SITED: Semantically-enabled Interoperable Trustworthy Enriched Data-spaces under Grant PID2021-125725OB-I00.

ABSTRACT We introduce offloading policies for fog-cloud architectures that consider different performance parameters. We design and develop a three-tier platform, using virtualization techniques, which can be used to deploy different scenarios, with nodes having distinct features, mimicking fog and cloud characteristics. We then exploit Lyapunov's control theory to introduce offloading policies that balance energy consumption at fog nodes and monetary cost of using the cloud. The proposed scheme is able to find a trade-off between these two parameters, while ensuring system stability and so delay requirements. We compare our algorithm with baseline solutions (adapted round-robin), and the results evince that it is able to yield better performance, even under high loads and stringent energy requirements. By tweaking the algorithm operational parameters, we show that it is able to adapt its behavior to different goals, and we assess its performance under realistic configurations.

INDEX TERMS Fog, cloud, offloading, Lyapunov, energy, modeling.

I. INTRODUCTION

Today, the number of cloud services is continuously increasing, especially those offered by key players such as Microsoft Azure, AWS, and Google Cloud. Their demand has been fostered, among other applications, by the surge of Internet of Things (IoT) and Industrial IoT (IIoT) services. In fact, this demand is rather likely to keep this increasing trend during the coming years. At the same time, we have witnessed a vast deployment of 5G networks, whose underlying technologies bring several advantages, for instance in terms of latency, availability, or reliability. Hence, many stakeholders and verticals that were constrained by their strict requirements see now an opportunity to deploy different IoT and IIoT services [1]. Indeed, together with the massive deployment of 5G cellular networks, the number of IoT connections has already reached 14.6 billion, and it is expected to exceed 30 billion by 2027.[1]

As a result, there is a growing interest in the strong integration of IoT with cloud computing, as this combination brings many new opportunities. However, the expected strong increase of IoT and IIoT services, their limited energy and processing capabilities, and their deployment in new sectors with more stringent requirements lead to situations where cloud computing may not be affordable in terms of delay, energy consumption or price, among other aspects. As an alternative, fog computing has emerged as an extension of cloud computing, providing high-performance computing services for IoT applications by offloading tasks to a geographically nearby fog node instead of a remote cloud [2]. The implementation of offloading mechanisms in distributed

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Merlino.

[1]https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook

computing systems would depend on their particular functionalities. As an example, in the case of the Kubernetes framework, and those based on it, the *load-balancer* service distributes external requests between multiple servers.

One of the main differences between cloud and fog computing is the scale of hardware components. Cloud computing provides high availability of computing resources with relatively high power consumption (data centers), while fog computing provides moderate availability of computing resources with lower power consumption (small servers, routers, switches, gateways...). Although both options, cloud and fog computing, can be used independently, we claim it is not needed to choose one, since fog and cloud complement each other, and the cooperation between them leads to an optimal use of resources, enhancing the capabilities of the system in terms of energy efficiency, cost reduction, and data processing, aggregation and storage. An orchestrator could handle this cooperation between cloud and fog [3].

In this sense, services may experience poor performance, increasing their latency due to queuing and computation delays when large numbers of tasks simultaneously arrive at a resource-constrained fog node. For this reason, cooperation between fog nodes and the cloud might bring several benefits. This combination leads to three-tier (IoT-fog-cloud) architectures, which leverage advantages from the two approaches, reducing service delay and energy consumption. In addition, the strong dependency on cloud service providers is alleviated, thus leading to (monetary) cost reductions. In order to consolidate these potential advantages, it becomes necessary to use a workload allocation scheme in an IoT–fog–cloud cooperation system, which would yield optimal performance for different scenarios and under heterogeneous requirements.

Most of the concepts discussed above are also applicable to edge-computing. Although fog and edge computing both move the computing tasks closer to end-nodes, these solutions are not identical. OpenFog Consortium (OPC) establishes a clear distinction, since fog computing follows a hierarchical paradigm, providing computing, networking, storage, control, and acceleration anywhere, from cloud to things, while edge computing tends to be limited to computing at the edge. In this work, we will use the term fog computing, but everything that will be discussed here is also applicable to edge.

Along with all the advantages of fog computing, there are some challenges to be tackled for computation task offloading mechanisms. The first one is the amount of workload to offload from fog to cloud instances, considering a potentially broad range of performance parameters. Another question to address concerns the place where timely decisions regarding the offloading take place. In this paper, we focus on the former, and we propose an adaptive algorithm that jointly considers delay, together with energy consumption at the fog, and monetary cost of the cloud.

The main contributions of this work are briefly summarized below:

1) Based on a three-tier IoT–fog–cloud architecture, we propose a workload dynamic allocation scheme that considers the energy consumption and monetary cost in a random and uncontrolled environment.
2) We tackle the resulting stochastic optimization problem by applying Lyapunov's control theory, so that it boils down to a queue system stabilization problem. The resulting optimization problem can be then solved with the drift-plus-penalty algorithm, which corresponds to a sequence of integer linear programming (ILP) problems.
3) We perform a thorough analysis of the proposed scheme in different scenarios, and under heterogeneous conditions. The behavior of the proposed scheme is also compared with benchmark solutions.

The rest of the document is structured as follows. In Section II, we discuss existing works related to the combination of IoT, fog, and cloud, as well as other workload offloading algorithms, pointing out how our proposal differs from them. In Section III, we describe the system model and the solution proposed for the offloading algorithm. Then, Section IV describes the platform that was deployed for the evaluation, while Section V discusses the performance of the proposed solution. Finally, Section VI concludes the paper, summarizing its main outcomes, and providing an outlook of our future work.

II. RELATED WORK

The combination of IoT and IIoT with fog and cloud computing has recently attracted the attention of the scientific community from different angles. With a global perspective, some works have proposed suitable architectures. For instance, the authors in [1] focus on an architectural overview of Industry 4.0, based on IoT-fog-cloud integrated solutions, identifying a number of use cases and emerging challenges. Similarly, Mouradian et al. present in [2] a survey of fog computing, establishing a common and concise set of evaluation criteria that embraces both architectures and algorithms. The authors of [4] show an integrated architectural model for combining Mobile Edge Computing (MEC) and fog computing for 5G networks. They propose to dynamically orchestrate all functions and needed resources at 5G nodes, without assuming any predefined configuration. Although these works share the same application scenario as ours, their scope is more at the architecture level, while our main interest is on the processing offloading logic, and algorithmic solutions to yield optimum behaviors.

Other existing works have focused more specifically on offloading approaches. Sengupta et al. propose in [5] a solution to secure a three-tier architecture as well as an offloading technique to enforce security features. An alternative architectural solution is presented in [6], where the authors analyze a hierarchical fog deployment, comparing it with flat topologies. The authors use queuing theory to analyze the system performance, and they consider different types of requests,

TABLE 1. Parameters and performance metrics considered in recent literature for similar scenarios.

	Algorithm	Delay	Energy	Cost	Stability
[16]	Distributed optimization based on ADMM.	✓	✓	✗	✗
[17]	Lyapunov-based stochastic prog.	✓	✗	✗	✓
[19]	Lyapunov-based stochastic prog.	✓	✓	✗	✓
[20]	Predictive offloading.	✓	✓	✗	✓
Ours	Lyapunov-based stochastic prog.	✓	✓	✓	✓

proposing offloading solutions to tackle high loads. The authors of [7] discuss a novel SDN/NFV architecture that embraces edge and cloud, to optimize computation tasks. SDN is also used in [8] to introduce an algorithm that selects the optimum access and computation points.

Another interesting group of works focuses on energy consumption. For instance, computation offloading in a fog-cloud environment is tackled, from an analytical perspective, in [9], where the authors seek to optimize the power consumption using the accelerated gradient algorithm. Similarly, deep reinforcement learning is exploited in [10] to optimize energy consumption in fog architectures, and Iqbal and Buhnova analyze in [11] the energy efficiency gains brought by fog computing in smart buildings scenarios.

Besides, several techniques have been applied to fog-cloud computing sharing. For instance, game theory is used in [12] to implement resource allocation in a three-tier IoT-fog-cloud architecture, while a novel auction scheme is adopted in [13]. Peralta et al. use network coding in [14] to minimize the download time from a fog-cloud architecture, while meta-heuristic optimization is used in [15] to enhance task scheduling over these scenarios. In [16], the authors work with a distributed Alternating Direction Method of Multiplier (ADMM) in order to solve a workload offloading problem that considers the quality-of-experience (QoE) and power efficiency as performance parameters. Xiaoting et al. propose in [17] a drift plus computing cost based on Lyapunov's optimization to effectively offload applications to achieve the trade-off between offloading cost and system performance in edge computing for IoT. In [18], an offline computational offloading strategy using a Markov decision process is proposed. The authors used the available bandwidth as a constraint for MEC in Vehicular Networks (VNs). Another algorithm based on Lyapunov's optimization is developed in [19]. It works online, without requiring future information, and it reduces computation latency while keeping a low energy consumption. In the same line, in [20] a predictive offloading and resource allocation scheme is proposed for multi-tier fog computing systems.

In contrast to these works, our proposal focuses not only on energy consumption, but also on monetary cost as well, while maintaining queue stability and so reducing latency. In addition, as other solutions using Lyapunov's theory, ours also takes into account the temporal evolution of the scenario, where random events can take place. In Table 1 we compare our proposal with similar approaches from the literature, at least in their goals. We have selected those works that assume random (uncontrolled) environments and propose techniques to provide instantaneous adaptation. The comparison is in terms of the decision algorithm, and the performance parameters it considers. We use the following metrics:

- Delay: it refers to the delay suffered by computation tasks or services, from the moment they are generated until they are fully processed.
- Energy consumption, which is mostly due to the processing. It is typically considered for fog nodes, which have more limited capabilities.
- Monetary cost: it corresponds to the cost of using the processing capacity of the cloud. We consider a "pay as you go" model, as this is what most providers (Amazon, Microsoft, IBM, Google, etc) offer.
- Stability of the memory queues in the overall system.

As can be observed, our solution is the only one that jointly considers energy (at fog nodes) and monetary cost (at cloud nodes), while keeping system stability. We can thus conclude that the work presented herewith complements and broadens the available state-of-the-art related to the distribution of computing tasks in fog/cloud deployments.

TABLE 2. System model symbols and variables.

Notation	Description
N	number of processing points, such as CPUs at the fog and cloud instances
M	number of independent applications generating services to be processed
$a_m(t)$	arrivals at the m-th application queue at time slot t, measured in number of packets
$b_m(t)$	departures from the m-th application queue at time slot t, measured in number of packets
$Q_m(t)$	queue backlog of the m-th application at time slot t
$\alpha_{m,n}(t)$	Decision for the m-th application and the n-th CPU at time slot t, measured in number of packets
$\alpha(t)$	$M \times N$ matrix of decision variables
$\mathcal{A}(t)$	admissible decision set at time slot t
$\omega_n(t)$	transfer rate of n-th processing option in slot t. It reflects varying available processing capacity
$g_m(t)$	processing complexity of services of the n-th application at time slot t
$C(t)$	monetary cost of the usage of cloud at time slot t
$k_n(t)$	generic cost of using the n-th processing point at slot time t
$E_n(t)$	energy cost of the n-th processing point at time slot t
E_nth	average energy threshold of the n-th processing point
$G_n(t)$	virtual queue related to energy consumption of the n-th processing point at slot t
$\hat{\cdot}$	it is used to indicate an arbitrary function that yields a variable $\cdot$.

III. SYSTEM MODEL

This section describes the stochastic system model and the control policy design using Lyapunov's theory. Table 2 summarizes the symbols used in the proposed model and their meaning. We use the term service to refer to chunks of bytes over which we need to apply certain computing process.

We consider a computing system composed of fog and cloud nodes with different processing capabilities. In this scenario, multiple user applications generate services, comprising a number of packets, which are sent to fog nodes. Then, the services can be either computed locally (at the fog nodes) or offloaded to the cloud. The system also includes an orchestrator, or master node, which takes offloading deci-

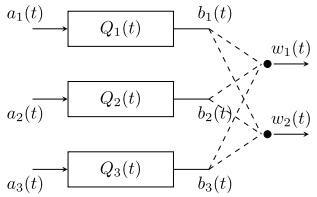


FIGURE 1. System model with 3 applications and 2 CPUs.

sions, depending on the particular implemented policy and system state.

Let M be the number of applications that generate services. We assume slotted time, and each application generates one service at every slot. In turn, the number of packets per service follows a certain random distribution, so that the service size is random. The packets of the generated services are locally stored in application queues. We assume the scenario has N processing alternatives, including local processors (alternatives $\{1, \ldots, N-1\}$) and one cloud (alternative N). At every slot the master node establishes the amount of data of each application to be processed at either of the two possibilities: local processing or at the cloud, to satisfy some constraints. The workload offloading policy must ensure that the application queues remain stable, in order to avoid long queuing delays. In this scenario, we apply Lyapunov's control theory, which has been extensively used in stochastic optimization to guarantee system stability.

Let $a_m(t)$ be the amount of packets that arrive at the m application queue, $m \in \{1, \ldots, M\}$, at slot t, and $b_m(t)$ the number of packets drained from that queue as a consequence of the policy that is enforced. The queue dynamics are given by Eq. (1), where $Q_m(t+1)$ is the queue backlog of the m-th application queue at time t.

$$Q_m(t+1) = \max[Q_m(t) - b_m(t), 0] + a_m(t) \tag{1}$$

We aim to ensure mean rate stability of the application queues, which is defined in Definition 1.

Definition 1 (Mean Rate Stability): A queue is mean rate stable if

$$\lim_{T\to\infty} \frac{1}{T}\,\mathbb{E}\{Q(t)\} = 0 \tag{2}$$

where $Q(t)$ is the length of the queue at time t, and $\mathbb{E}$ is the expectation operator.

Let $\alpha(t)$ be a MxN matrix such that the element $\alpha_{m,n}(t)$ corresponds to the amount of data of the m-th application that is allocated to the n-th processor in slot t. In each slot, we make an $\alpha(t)$ decision within a set $\mathcal{A}(t)$ of possible choices. Besides, it is assumed that there is transfer data rate for each processing option, which dictates how many bytes can be accepted in a given slot. Then, we can define $b_m(t)$ according to Eq. (3), where $\omega(t)$ is the transfer rate of each processor in slot t, in bytes per slot. In general, we assume that the transfer rate varies over time, following an arbitrary distribution. As can be observed, the amount of data drained for each application is a function of the decision and the transfer rate of the processors. Figure 1 shows an example of the system model with 3 applications and 2 processors. It is worth noting that the transfer rate can be influenced by both the computation power of the CPU and the communication capacity between the application queue and the processor. For instance, in local processing, the transfer rate would be dominated by the computation capacity, while in the case of remote processing (data to be sent to the cloud), it would be bounded by the communication capacity.

$$b_m(t) = \hat{b}(\alpha(t), w_1(t), w_2(t), \ldots) = \hat{b}(\alpha(t), \omega(t)) \tag{3}$$

In order to avoid assigning non-existent packets, in each slot we impose that the total amount of bytes allocated from an application queue i at slot t does not exceed the bytes in the corresponding queue, as stated in Eq. (4).

In addition, we assure that the assignment does not exceed the transfer capacity with the constraint defined in Eq. (5), where $g_i(t)$ denotes a generic scaling factor. For instance, in the case of transfer rate limited by the computation capacity, this parameter would be related to the computation complexity of the service. This way, more complex services would yield slower computation times for the same number of bytes, which is represented by scaling the number of bytes, while keeping constant the computation capacity.

$$\sum_{j=1}^{N} \alpha_{ij}(t) \le Q_i(t) \quad \forall i \in \{1, \ldots, M\}, \forall t \tag{4}$$

$$\sum_{i=1}^{M} g_i(t) \cdot \alpha_{ij}(t) \le \omega_j(t) \quad \forall t, \forall j \tag{5}$$

Now we can re-write the departure $b_m(t)$, as shown in Eq. (6).

$$b_m(t) = \hat{b}(\alpha(t), \omega(t)) = \sum_{j=1}^{N} \alpha_{m,j}(t) \tag{6}$$

We also consider the penalties of using the different processing alternatives. We use the symbol $k_i(t)$ to denote the cost of using the i-th processing alternative at time slot t. Penalties are defined for the cloud and local fog CPUs in different ways. For cloud processing we aim to minimize the monetary cost, which is related to the required computation power, as defined in Eq. (7):

$$C(t) = \hat{C}\left(\sum_{i=1}^{M} g_i(t) \cdot k_N(t) \cdot \alpha_{i,N}(t)\right) \tag{7}$$

where $g_i(t)$ corresponds, as mentioned earlier, to the computation complexity of services generated by application i, the processing alternative N corresponds to the cloud, and so $k_N(t)$ is the monetary cost of using the cloud at slot t. As can be seen, the cloud cost is proportional to the amount of traffic forwarded to the cloud instance, scaled by the computation complexity. In general, we assume that both the computational complexity of the services of each application

and the cloud fee can vary over time, following arbitrary random distributions.

On the other hand, in the case of local processing in the fog, we will focus on energy consumption. In this case, we do not seek to minimize it, but to ensure that, on average, it remains below a certain value. This would be needed, for instance, for battery-driven devices which can be recharged periodically. We define the energy constraint in Eq. 8 as follows:

$$E_j(t) = \sum_{i=1}^{M} g_i(t) \cdot k_j \cdot \alpha_{i,j}(t) \quad \forall j \neq N \tag{8}$$

where k_j holds for a general mapping between the number of bytes to be processed, scaled by the processing complexity, and the energy required for that processing. Opposed to the cloud, the cost associated with the energy (k_j) would mostly depend on the processor hardware characteristics, so we assume it does not vary: $k_j(t) = k_j \ \forall t \ \forall j \neq N$. In both cases, we account for the penalties over time, so that we use their time-average-expectations $\overline{C}$ and $\overline{E}$, which are defined as follows:

$$\overline{C} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}\{C(t)\} \tag{9}$$

$$\overline{E} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}\{E(t)\} \tag{10}$$

Altogether, we want a control policy that minimizes the following optimization problem::

Problem 1:

$$\min_{\alpha(t)} \quad \overline{C} \tag{11}$$

$$\textbf{s.t.} \quad \overline{E}_j \leq E_j^{Th} \quad \forall j \in \{1, \ldots, N-1\} \tag{12}$$

$$\alpha(t) \in \mathcal{A}(t) \tag{13}$$

where E_j^{Th} is the energy threshold defined for each processor of a fog node, and $\mathcal{A}(t)$ holds for the set of constraints defined in Eq. (4) and (5) in every slot. Using the stochastic optimization framework developed in [21], inequalities related to energy consumption limitation can be converted to virtual queues, alike the application queues defined above. Then, the update of the virtual queue associated with the energy of fog processor j, G_j, is defined as:

$$G_j(t+1) = max\{G_j(t) + (E_j(t) - E_j^{Th}), 0\} \tag{14}$$

The virtual queue is introduced as a strong method for ensuring that the required average energy consumption constraint is satisfied. Thus, we can define the set of queues (of applications and the virtual queue) as $\Theta(t)$. The Lyapunov's function $L(\Theta(t))$ and the drift $\Delta(\Theta(t))$ are defined as shown below:

$$L(\Theta(t)) = \frac{1}{2}\left(\sum_{j=1}^{N} G_j(t) + \sum_{i=1}^{M} Q_i(t)\right) \tag{15}$$

$$\Delta(\Theta(t)) = \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\} \tag{16}$$

The solution to Problem 1 is the drift-plus-penalty algorithm. At each slot t, the state of the queues is observed, and a decision that solves Problem 2 is made, where V is a positive weighting factor that establishes the trade-off between the drift and the penalty. This is an integer linear programming (ILP) problem, which can be solved using existing tools. The complete process is depicted in Algorithm 1.

Problem 2:

$$\min_{\alpha(t)} \quad V \cdot C(t) + \sum_{i=1}^{M} Q_i(t)[a_i(t) - b_i(t)] + \tag{17}$$

$$\sum_{j=1}^{N} G_j(t)(E_j(t) - E_j^{Th}) \tag{18}$$

$$\textbf{s.t.} \quad \sum_{j=1}^{N} \alpha_{ij}(t) \leq Q_i(t) \quad \forall i \in \{1, \ldots, M\}, \forall t \tag{19}$$

$$\sum_{i=1}^{M} g_i(t) \cdot \alpha_{ij}(t) \leq w_j(t) \quad \forall t, \forall j \tag{20}$$

Algorithm 1 Offloading Decision Based on Drift-Plus-Penalty

1: **Initialization:**
Set V and E_{th}.
2: **Repeat:**
1) Observe $a(t) = [a_1(t), \ldots, a_M(t)]$, $Q(t) = [Q_1(t), \ldots, Q_M(t)]$ and $\omega = [\omega_1, \ldots, \omega_N]$.
2) Choose decisions $\alpha(t)$ to minimize Problem 2 applying solver *scipy.optimize.milp*.
3) Update $G(t) = [G_1, \ldots, G_N]$ according to Eq. (14).

IV. EVALUATION PLATFORM

There exist several alternatives to deploy and manage fog and cloud instances. Most of the big technological companies provide cloud services, such as AWS, Azure, Linode, etc. In parallel, there are alternatives that allow the deployment of proprietary and self-managed fog/cloud instances, both commercial (e.g. VMware) or open source (e.g. OpenStak, Apache CloudStack, Proxmox). However, these technologies are not designed for testing or evaluating the performance of orchestration solutions, but to manage running services. In this sense, there is a need for frameworks that fill the gap between analytical evaluation and planning, to address the evaluation of the expected performance under controlled environments. There are some related works where three-tier architecture platforms have been developed, such as [22], [23], and [24], but they had some limitations for our purposes. Hence, we opted to develop our own framework, tailored to the analysis of computation sharing/offloading in fog/cloud environments.

The developed platform is illustrated in Figure 2. It embraces three types of elements that mimic fog, cloud, and a master node. Fog nodes generate independent synthetic traffic flows belonging to different applications, and using

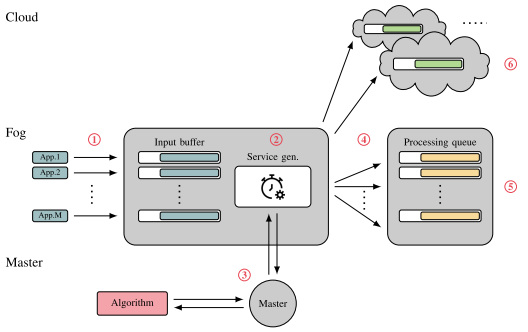


FIGURE 2. General view of the fog/cloud platform.

configurable random distributions (i.e. Poisson, uniform, Lognormal, etc.). The generated traffic of each application is stored in an input buffer, illustrated in Figure 2 as step 1. From the generated traffic, fog nodes define services (processing tasks) that embrace a number of packets. Service generation, depicted in Figure 2 with step 2, is also configurable, using random distributions. When services are defined, the fog node queries the master about which node the corresponding data has to be sent to be processed, and the master node takes a decision according to the implemented algorithm (step 3). The fog node then sends the service data to either local processors or remote cloud instances (step 4). Finally, performance logs are generated (steps 5 and 6) for each service and for monitoring temporal evolution of all devices' states. It is worth noting that the platform implements a generic interface to communicate with the master node, which is independent of the decision algorithm adopted. In addition, the algorithm of the master node is implemented as a plugin, which can be modified without further changes, thus allowing the comparison of different decision schemes under the same circumstances.

In order to ensure a scalable, lightweight platform, where multiple nodes can be deployed without overloading the host machine, all nodes have been containerized using Docker. Containerization allows users to quickly deploy multiple customized containers in the same host. Each container is an isolated software unit that packages code and its dependencies, allowing it to run, regardless of the underlying host. Docker executes container images that are lightweight, standalone, executable packages of software that include everything needed to run an application (i.e. code, runtime, system tools, system libraries, and settings). Images can be configured with the so-called Dockerfile to execute customized applications in isolated containers. Docker runs over the host operating system, and it acts as an abstraction layer for the applications.

In the case of the evaluation platform described herewith, each of the containers works as a fog, cloud, or master node, which are built from different customized Docker images. In all cases, the images use Ubuntu 22.04 as their base operating system, and we make the required modifications over it. The functionalities of master, fog, and cloud nodes within Docker containers have been developed in Python programming language. In the following, we describe the functionality of each component in more detail:

- **Fog node.** It implements logic to mimic the behavior of a real node, and it comprises a set of concurrent functionalities using multi-threading. The first functionality deals with the emulation of service creation and the generation of data belonging to those services. In this sense, instead of receiving traffic from real IoT devices, the tool emulates such traffic generation. Then, before storing the packets in the arrival buffer, we add a header that allows tracking and appropriately processing all of them, either locally or remotely. As for the traffic processing, instead of treating each packet individually, a service model has been implemented. We define a service as a chunk of related packets. Later, depending on the information provided by the master node, the services are transferred from the arrival buffer to the processing queue at the fog node, or sent to a cloud node, to be remotely processed. The last functionality of a fog node is to locally process traffic.
- **Cloud node.** Inside these nodes a single Python program is executed with a simple functionality, to process incoming services. Each cloud has a receiver that stores all packets belonging to services sent to the cloud. This receiver is implemented as an independent thread that is continuously running throughout the whole experiment. Finally, following the same logic as in fog nodes, a processor with a configurable processing rate has been implemented in the cloud node, as another independent thread.
- **Master node.** This node enforces offloading policies using state information gathered from the fog and cloud nodes and other system parameters. The algorithm hosted in the master node establishes the particular processing sharing policy to evaluate, and it would thus depend on the evaluation goals. The platform implements a minimum set of control packets that are exchanged between fog and master nodes. On the one hand, when deciding where to execute a service, the information about the fog node state (queue occupancy, processing rate, etc.) and about the current service (i.e. size) is sent to the master node. Using that information, and the corresponding algorithm, the master node answers, indicating how to proceed, i.e. whether to use local processing or the cloud node to forward the task to. In this sense, the deployment of different processing policies would boil down to the selection of the appropriate algorithm, leading to a very flexible experiment configuration.

By deploying our own platform, we aim at having the flexibility to closely inspect the practical aspects of different solutions. We thus ensure that our theoretical findings are aligned with their applicability in real systems. Furthermore, this control allows us to customize the platform to our specific needs and research goals, and to conduct

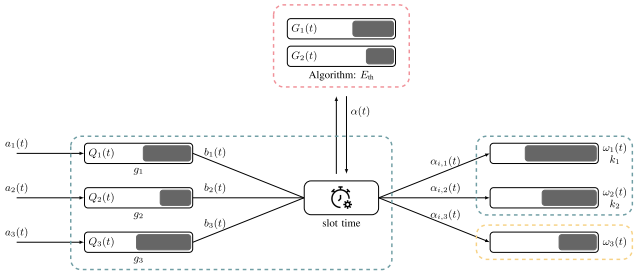


FIGURE 3. The fog/cloud system model with application queues (Q_m), virtual energy queue (G_n) and processor queues.

experiments and simulations that precisely match our requirements. In any case, and exploiting the modular design of our architecture, moving the Docker containers that implement the cloud functionality to real deployments could be done rather straightforwardly.

V. RESULTS

This section discusses the performance analysis of our proposal. First, we study the properties of our algorithm in 2 synthetic scenarios, with and without cloud processing alternative. The former scenario focuses on the trade-off between energy constraints and monetary cost, while the latter configuration pays special attention to the impact that energy limitation may have on the incoming application traffic. Then, we evaluate the proposed scheme in a more realistic scenario, in terms of processing capacity and traffic generation.

In these 3 setups, the performance of our solution is compared with that observed with a simple round-robin algorithm, which is thus used as a benchmark. The round-robin policy allocates packets in equal portions and in circular order, handling all of them without any priority whatsoever. Once all the capacity of a processor that can be used in a slot is used up, this policy moves on to the next processor. It does not, therefore, consider energy consumption or monetary cost. Moreover, we implement it in a way that changes the initial application in every slot, in order to avoid prioritizing one application. Packets that cannot be processed in the fog due to lack of capacity are sent to the cloud. In addition, a modified round-robin algorithm is also used that ensures that the energy threshold is respected in every slot.

Figure 3 depicts the system setup. In a nutshell, it has a fog node with 3 applications and 2 processors, one cloud node, and a master node that runs the algorithms. The base configuration details that we used for the first two setups are depicted in Table 3. As can be observed, in all cases, we run executions elapsing 1000 slots of 1 second each. At every slot, the applications generate a service consisting of a random number of packets. Table 3 shows the aggregated average application rate, while the particular rate for each application will be specified in each scenario. As can be observed in Table 3 some arbitrary random variables are defined as constants, so that the results can be better understood.

TABLE 3. Simulation setup.

Parameter	Value
Simulated slots	1000
Slot time	1 s
Processing capacity of a CPU at a fog node	1 kB/s
Processing capacity at a cloud node	100 kB/s
Packet length	200 + 12 bytes
Average total traffic rate	Poisson $[6, \ldots, 30]$ pkt/s
Service rate	1 serv/s
Fog energy factor (k_j)	1
Cloud cost (k_N)	1
Application complexity (g_m)	1
Energy threshold	$[1, \ldots, 5]$
#Applications	3
#Fog CPUs	2

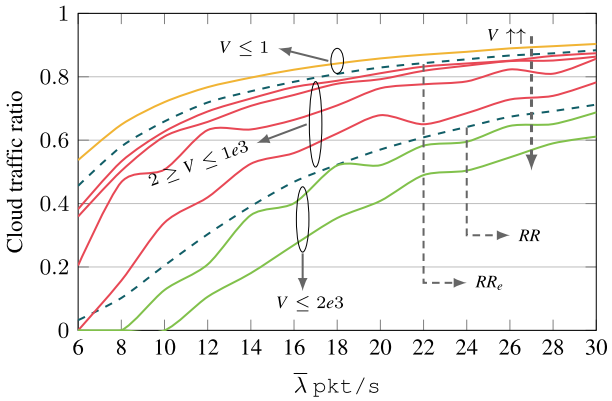


FIGURE 4. Cloud usage evolution vs. total traffic rate.

A. FOG AND CLOUD COLLABORATION ANALYSIS

With the first setup, we aim to analyze the processing balance between fog and cloud nodes upon different configurations. It is worth remarking that the processing capacity of the cloud node, considering the traffic generated by the services, can be considered infinite.

First, we evaluate the impact of the V parameter over the computation offloading. It is worth recalling that such parameter adjusts the trade-off between energy consumption and monetary cost. Figure 4 shows the ratio of traffic sent to the cloud node upon different values of the aggregated average traffic rate. In this setup, we fix the energy threshold value, E_{th}, to 2. In addition, simulations are carried out for different values of V. The figure also shows, with dashed lines, the results obtained when using the round-robin (RR) and an energy-aware round-robin variant (RR_e). The former consumes the entire processing capacity of the fog, sending the surplus to the cloud. The latter performs a round-robin selection between the fog processors without exceeding the energy threshold, sending the rest of the packets to the cloud. For each configuration, Figure 4 shows the cloud usage for experiments lasting 1000 slots.

As expected, we can observe that a higher traffic rate leads to more traffic processed in the cloud. In addition, as the value of V increases, we see a decreasing trend in cloud usage, as expected. We can also identify 3 different operation regions of the proposed solution, as we vary V, delimited by the round-robin algorithms.

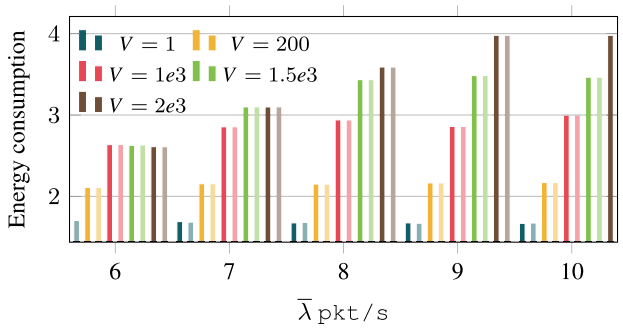


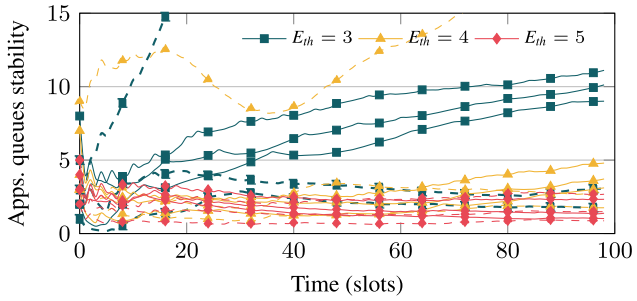
FIGURE 5. Average energy consumption vs. total traffic rate.

In the first region, we can see that there is more traffic sent to the cloud than with the RR_e policy. This happens with values of V below 1, where the monetary cost is given very little importance. This results in not using the maximum processing capacity of the fog, even if it does not reach the energy threshold. The second region corresponds to values of V between 2 and 1000, and the observed performance lies between the two round-robin versions. In this region, higher values of V significantly reduce the use of the cloud. In turn, it leads to an eventual saturation of the fog processors. In order to fulfill the energy threshold, the proposed solution keeps traffic in the application queues and balances the decisions to ensure system stability, considering application queues, energy, and cost.

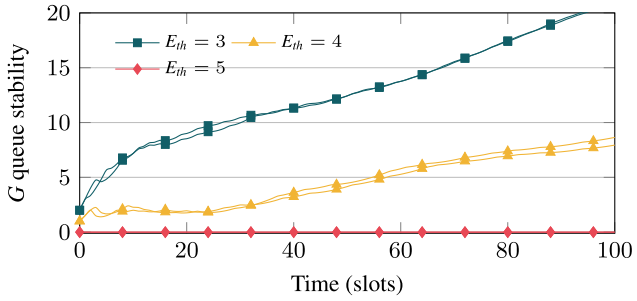
We further analyze this effect by studying how the different configurations impact the energy performance indicator. In Figure 5 we represent, with a bar plot, the average energy consumption yielded by the proposed solutions for different configurations of the aggregated traffic rate and for various V values. It is worth remarking that in all cases the energy threshold is set to 2 and that the aggregated processing capacity at the edge node is 4 pkt/s.

As can be seen, when V is within the first region observed in Figure 4 ($V = 1$), the average energy consumption is well below the threshold, regardless of the traffic rate. As we increase the value of V we can observe that the proposed scheme is not able to keep the energy below the threshold, due to the high cost of using the cloud instance. In addition, the results evince that the impact of V also varies with the traffic rate. In this sense, the energy consumption saturates with $V = 1e3$ for the lowest traffic rate (6 pkt/s), while this saturation value increases for higher rates. We can conclude that moderate V values, with low traffic, enforce the processing of all services at the fog, since the virtual energy queue does not grow much. On the other hand, with higher rates, the virtual queue increases, and some services are therefore eventually sent to the cloud, unless the value of V is also increased.

This first set of results validates the behavior of the proposed scheme, showing that it is able to balance the computing load, considering different parameters (energy, cost, and application queues). Furthermore, it can be configured to foster different behaviors, thanks to the V configuration parameter.



(a) Application queues. Round-robin policy is represented with dashed lines.



(b) Virtual energy queue.

FIGURE 6. Queues evolution vs. energy threshold.

B. FOG PERFORMANCE ANALYSIS

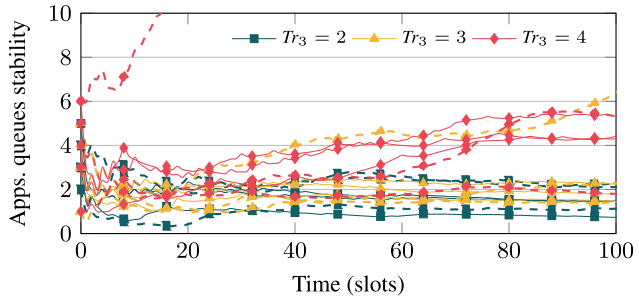
We now focus on the impact that different configurations have on the energy and applications queues at the fog. As we have seen in the previous section, low V values would avoid overloading fog nodes, since they enforce many services to be sent to the cloud. Having that in mind, we consider a setup where the cloud is not available, which would be in fact alike having a high V. Without the cloud, we can assess more critical configurations in terms of processing capacity, allowing a closer look at the stability of application queues.

In this case, the average traffic rate is set to 7 packets per slot. Specifically, the first, second, and third applications generate 1, 2, and 4 packets per slot, respectively. Figure 6 shows the temporal queue stability of application and energy queues, using Eq. 2, for different values of the energy threshold. It is worth noting that large thresholds lead to not imposing any limit on energy consumption. For the sake of visibility, we show the results for the first 100 slots.

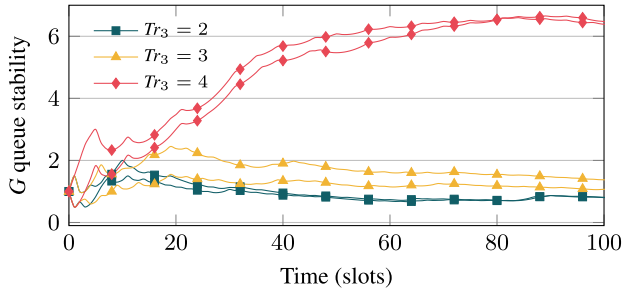
Figure 6a illustrates the application queue stability yielded by our scheme and that obtained with energy-limited round-robin (dashed lines). The different colors correspond to the stability of the different application queues.

In general, we can observe that the proposed scheme manages to keep the stability of all application queues regardless of the limits set on energy consumption. As can be observed, when using round-robin with the energy threshold set to 3 and 4, there is one highly unstable queue, which corresponds to the application with the higher rate, while the others show quite low values.

On the other hand, our scheme is able to adapt to the application rates, as evinced by the fact that all queues show



(a) Application queues. Round Robin policy is represented with dashed lines



(b) Virtual energy queue

FIGURE 7. Queues evolution vs. traffic rate of one application.

similar values. When a threshold of 3 is used, stability cannot be guaranteed, but with softer energy restrictions, stability is reached quickly. Figure 6b shows the stability of the energy virtual queue (G_j) of each CPU of the fog node. The results show a trend similar to that seen for the application queues. As we relax the energy constraint, the proposed scheme is able to stabilize both types of queues, while it penalizes those queues with more stringent requirements.

We now study the impact of the traffic rate on the queues. Figure 7 uses a queue stability representation similar to the previous ones. In this case, we set the energy threshold to 3.5, and we run experiments for different values of the traffic rate of the third application, while the others do not change (1 and 2 `pkt/s`, respectively). The results show that the round-robin policy is not able to stabilize the queue corresponding to the third application as we increase its rate. As can be seen, when the third application is configured with 4 packets per second, the stability of the corresponding queue exceeds the graph limits in slot 15, and it shows a growing trend even when the application rate is set to 3 `pkt/s`. In contrast, the proposed algorithm is able to keep the application queues stable, even at the highest rate. As can be observed, when the rate of the third application is set to 4 `pkt/s`, the queue tends to a stable value within the analyzed interval. At the same time, we can observe that the proposed scheme is also able to stabilize the energy queues, as shown in Figure 7b.

C. REALISTIC SETUP

We now analyze the performance of the proposed scheme using realistic configuration parameters. In particular, we adjust the application traffic distributions and device capabilities as shown in Table 4. These values are taken

TABLE 4. Simulation setup for a realistic environment.

Parameter	Value
Simulated slots	1000
Slot time	1 s
Processing capacity of a CPU at a fog node	100 kB/s
Processing capacity at a cloud node	10^6 kB/s
Energy threshold	$[75, \ldots, 120]$
Traffic distribution (for each app)	$Lognormal(4.5, 0.8)$

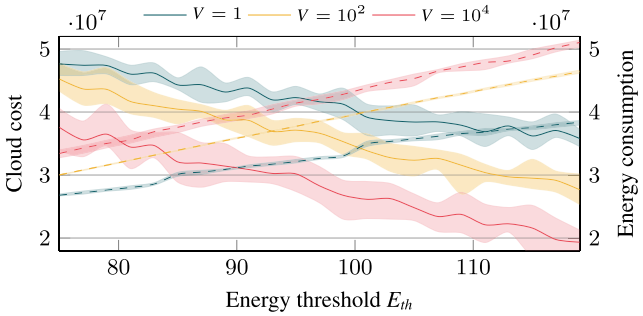


FIGURE 8. Cloud cost and energy consumption vs. E_{th} for different V values.

from [25], where the authors collected a two-week input-output workload traces of 2500 nodes from AliCloud, one of the largest cloud providers in Asia. Using the collected data, the authors characterized the distribution of different parameters of workloads, such as inter-arrival time, or service size.

The results obtained in [25] revealed that the best-fitting option for traffic distribution is lognormal. In particular, the expected value and variance of the underlying normal distribution are set to 4.5 and 0.8, respectively. In turn, the corresponding model shows an average traffic rate of 125 `pkt/s`.[2] We also scale the capacity of the fog node according to the new traffic rates, so that it can cope, on average, with the traffic generated by the three applications. Thus, the energy threshold would play an important role in the simulation results in this scenario.

Under this configuration, we aim to represent optimal configurations of the algorithm, according to the affordable monetary cost and energy consumption limitations. In this sense, Figure 8 shows a two-axis representation of the monetary cost (left axis) and energy consumption (right axis) with solid and dashed lines, respectively. The lines represent the average value obtained from 30 independent experiments, each lasting 1000 slots. Along with the average values we also represent the maximum and minimum obtained during the simulations with the shaded background. We show the results as we increase the value of the energy threshold E_{th}, and for different values of the V parameter.

As expected, when relaxing the energy threshold the cost decreases, since more data is processed at the fog, while the energy consumption grows. The intersections correspond

[2]The expected value of the lognormal distribution $\mathcal{X}$ is given by $\mathbb{E}\{\mathcal{X}\} = \exp(\mu + \frac{\sigma^2}{2})$, where μ and σ^2 are the expected value and variance of the normal distribution $\mathcal{N}$, so that $ln(\mathcal{X}) \sim \mathcal{N}(\mu, \sigma^2)$.

TABLE 5. Intersection points for different V values.

V	1	10	10^2	10^3	10^4
E_{th}	80.13	91.61	92.96	108.61	113.54
Cost ($\cdot 10^6$)	35.33	37.05	36.96	37.07	37.36

to points (configurations) at which the two costs are alike. In this sense, the proposed scheme allows establishing configurations (V and E_{th}) that equal energy and monetary costs. As can be observed, for the considered scenario, E_{th} needs to be set between 90 and 115 for the whole range of V values. In Table 5 we indicate the intersection points of more V values, which were not included in Figure 8. Similar analysis can be done for different relationships between energy consumption and cloud cost, or by fixing the energy threshold instead of the cloud cost.

VI. CONCLUSION

In this work, we present a novel multi-objective algorithm to offload computation load between fog and cloud tiers. The proposed solution jointly considers metrics related to applications, fog nodes, and cloud instances. In a nutshell, it is able to find a trade-off between application queues, energy consumption in fog nodes, and monetary cost associated with cloud usage. On the one hand, it ensures that the processing delay is upper bounded, while keeping the average energy consumption below a configured threshold. On top of that, when cloud instances are also available, the algorithm can be configured to also consider the monetary cost of the cloud, along with the aforementioned parameters.

We first propose a generic system model that assumes arbitrary varying traffic patterns, available computation capacity and cloud cost. Then, we formulate a stochastic optimization problem and use Lyapunov's theory to cast it to a temporal sequence of Integer Linear Programming (ILP) problems, which can be easily solved with existing tools. The proposed scheme is afterwards applied to a variety of fog/cloud scenarios, to validate its performance under different system configurations. The results evince that the proposed scheme is able to balance the use of fog and cloud instances, ensuring that the energy consumption remains below the configured threshold. We have also shown that it is possible to tweak the behavior of the proposed solution to give more or less relevance to cloud cost or energy consumption limitation. We also analyzed the response of the proposed solution to balance traffic queues and energy consumption. We have observed that it is able to adapt to unbalanced traffic loads, ensuring system stability even under high loads or more stringent energy constraints. Finally, we have broadened our analysis with a more realistic setup, showing that it can be configured to operate with a given desired response, for instance ensuring that the energy consumption and cloud monetary cost are alike.

In our future work, we will extend the model in different ways. First, we will analyze the performance when adding more complex functions (for instance, logarithmic) to the optimization problem, to foster different trade-offs between cost and energy limitations. In addition, we will analyze the possibility to take into account the occupancy of the processor queues within the system model, so that it will evolve to a network of connected queues, where back-pressure algorithms can be applied. We will also analyze the applicability of delay-based back-pressure, to foster very low latency system responses.

Furthermore, besides comparing the performance of the proposed offloading mechanism against alternative algorithms, considering not only energy and cost, but also other performance metrics, such as delay or occupancy, we will broaden the comparison to other optimization frameworks, such as AI and ML.

REFERENCES

[1] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial Internet of Things and industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4674–4682, Oct. 2018.

[2] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.

[3] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for Internet of Things services," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 16–24, Mar. 2017. [Online]. Available: https://www.scopus.com

[4] P. Bellavista, L. Foschini, and D. Scotece, "Converging mobile edge computing, fog computing, and IoT quality requirements," in *Proc. IEEE 5th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2017, pp. 313–320.

[5] J. Sengupta, S. Ruj, and S. D. Bit, "A secure fog-based architecture for industrial Internet of Things and industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2316–2324, Apr. 2021.

[6] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial IoT data scheduling based on hierarchical fog computing: A key for enabling smart factory," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4590–4602, Oct. 2018.

[7] S. Garg, K. Kaur, G. Kaddoum, and S. Guo, "SDN-NFV-aided edge-cloud interplay for 5G-envisioned energy internet ecosystem," *IEEE Netw.*, vol. 35, no. 1, pp. 356–364, Jan. 2021.

[8] I. Ahammad, M. A. R. Khan, and Z. U. Salehin, "QoS performance enhancement policy through combining fog and SDN," *Simul. Model. Pract. Theory*, vol. 109, May 2021, Art. no. 102292. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X21000216

[9] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-optimal dynamic computation offloading for industrial IoT in fog computing," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 566–576, Jun. 2020.

[10] Y. Ren, Y. Sun, and M. Peng, "Deep reinforcement learning based computation offloading in fog enabled industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4978–4987, Jul. 2021.

[11] D. Iqbal and B. Buhnova, "Fog based energy efficient process framework for smart building," in *Proc. Eval. Assessment Softw. Eng. (EASE)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 387–393, doi: 10.1145/3463274.3463364.

[12] Y. Jie, C. Guo, K.-R. Choo, C. Z. Liu, and M. Li, "Game-theoretic resource allocation for fog-based industrial Internet of Things environment," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3041–3052, Apr. 2020.

[13] A. Aggarwal, N. Kumar, D. P. Vidyarthi, and R. Buyya, "Fog-integrated cloud architecture enabled multi-attribute combinatorial reverse auctioning framework," *Simul. Model. Pract. Theory*, vol. 109, May 2021, Art. no. 102307. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X21000307

[14] G. Peralta, P. Garrido, J. Bilbao, R. Agüero, and P. M. Crespo, "Fog to cloud and network coded based architecture: Minimizing data download time for smart mobility," *Simul. Model. Pract. Theory*, vol. 101, May 2020, Art. no. 102034. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X19301650

[15] H. Singh, S. Tyagi, P. Kumar, S. S. Gill, and R. Buyya, "Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions," *Simul. Model. Pract. Theory*, vol. 111, Sep. 2021, Art. no. 102353. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X21000678
[16] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, May 2017, pp. 1–9.
[17] X. Duan, F. Xu, and Y. Sun, "Research on offloading strategy in edge computing of Internet of Things," in *Proc. Int. Conf. Comput. Netw., Electron. Autom. (ICCNEA)*, Sep. 2020, pp. 206–210.
[18] Y. Qi, L. Tian, Y. Zhou, and J. Yuan, "Mobile edge computing-assisted admission control in vehicular networks: The convergence of communication and computation," *IEEE Veh. Technol. Mag.*, vol. 14, no. 1, pp. 37–44, Mar. 2019.
[19] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 207–215.
[20] X. Gao, X. Huang, S. Bian, Z. Shao, and Y. Yang, "PORA: Predictive offloading and resource allocation in dynamic fog computing systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 72–87, Jan. 2020.
[21] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*, (Synthesis Lectures on Communication Networks). San Rafael, CA, USA: Morgan & Claypool, 2010, doi: 10.2200/S00271ED1V01Y201006CNT007.
[22] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "IFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw., Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2509
[23] A. Kertesz, T. Pflanzner, and T. Gyimothy, "A mobile IoT device simulator for IoT-fog-cloud systems," *J. Grid Comput.*, vol. 17, no. 3, pp. 529–551, Sep. 2019.
[24] I. Lera, C. Guerrero, and C. Juiz, "YAFS: A simulator for IoT scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91745–91758, 2019.
[25] Z. Ren, W. Shi, J. Wan, F. Cao, and J. Lin, "Realistic and scalable benchmarking cloud file systems: Practices and lessons from AliCloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 11, pp. 3272–3285, Nov. 2017.

NECO VILLEGAS received the B.Sc. degree (Hons.) in telecommunications engineering from the University of Cantabria, in 2021, where he is currently pursuing the M.Sc. degree. He has been a Researcher with the Communications Engineering Department, University of Cantabria, since 2022. His research interests include edge/cloud computing, task offloading, and Lyapunov's stability theory.

LUIS DIEZ received the M.Sc. and Ph.D. degrees from the University of Cantabria, in 2013 and 2018, respectively. He is currently an Assistant Professor with the Communications Engineering Department, University of Cantabria. As for teaching, he has supervised 30 B.Sc. and M.Sc. thesis. He teaches in courses related to cellular networks, network dimensioning, and service management. He has been involved in different international and industrial research projects. His research interests include future network architectures, resource management in wireless heterogeneous networks, and the IoT solutions and services. He has published more than 55 scientific and technical papers in those areas. He has served as a TPC member and a reviewer for a number of international conferences and journals.

IDOIA DE LA IGLESIA received the degree in telecommunications engineering from the University of Deusto, in 2013, the master's degree in transport systems from the University of the Basque Country, in 2014, and the Ph.D. degree in vehicular communications from the University of Deusto, in January 2019. Throughout her research career, she was an Intern in prestigious centers, such as Nokia Bell Labs or NEC Laboratories Europe. During the last years, she has been involved in different projects related to the IoT and edge computing in different business sectors. She has been with the Ikerlan Technology Research Centre, since 2018. Since November 2022, she has been leading the IoT and Digital Platforms Team with the Ikerlan Technology Research Centre. She received the Best Academic Record and Best Final Project Award from the University of Deusto.

MARCO GONZÁLEZ-HIERRO received the M.Sc. degree in telecommunications engineering from the University of the Basque Country (UPV/EHU). He has been a Researcher with the Ikerlan Technology Research Centre, since 2016, and has extensive experience as a Project Manager and a Software Developer. He is currently the Head of the Information and Communications Technologies (ICT) Department, leading over 60 researchers focused on the IoT, edge-to-cloud continuum, and machine learning/artificial intelligence research with over 3M€ portfolio in research and development projects for companies. He also has expertise in edge/cloud systems architecture, mobile connectivity, and cybersecurity. He has participated in numerous technology transfer projects for industry, mainly in the energy, railway, and elevation sectors.

RAMÓN AGÜERO (Senior Member, IEEE) received the M.Sc. degree (Hons.) in telecommunications engineering from the University of Cantabria, in 2001, and the Ph.D. degree (Hons.), in 2008. Since 2016, he has been the Head of the IT Area (Deputy CIO), University of Cantabria. He is currently a Professor with the Communications Engineering Department, University of Cantabria. He has supervised five Ph.D. and more than 70 B.Sc. and M.Sc. thesis. He is the main instructor in courses dealing with networks, and traffic modeling, both at B.Sc. and M.Sc. levels. His research interests include future network architectures, especially regarding the (wireless) access part of the network and its management, multihop (mesh) networks, and network coding. He has published more than 200 scientific papers in such areas. He is a regular TPC member and a reviewer of various related conferences and journals. He serves on the editorial board for IEEE Communication Letters (Senior Editor, since 2019), IEEE Open Journal of the Communications Society, *Wireless Networks* (Springer), and *Mobile Information Systems* (Hindawi).