An Energy-Conservative Dispatcher for Fog-Enabled IIoT Systems: When Stability and Timeliness Matter

Aref Karimiafshar, Massoud Reza Hashemi, Mohammad Reza Heidarpour, and Adel N. Toosi, *Member, IEEE*

Abstract—The deployment of fog computing resources in industrial internet of things (IIoT) is essential to support time-sensitive applications. To utilize resources efficiently, a brand-new request dispatcher is required to sit between the IIoT devices and the pool of fog resources. The need for such a dispatcher stems from the challenges specific to these systems. Firstly, fog-enabled IIoT systems are highly dynamic and distributed. Second, fog nodes are typically power and resource limited. Finally, many IIoT applications feature critical time-sensitivity, referred to as timeliness, and cannot tolerate response delay beyond a specific threshold. This paper proposes an efficient dispatching algorithm to minimize energy consumption and deadline misses while keeping the system stability at a satisfactory level. We leverage Lyapunov Optimization technique to tackle the problem and handle the system dynamics. We perform extensive simulations to verify the effectiveness of the proposed method and provide sensitivity, scalability and model parameter analysis. The simulation results prove the superiority of the proposed method over the state-of-the-art method up to 22% and 10% in terms of average deadline misses and energy consumption, respectively. Further, we perform practical experiments to prove the validity of the proposed method in a real testbed.

Index Terms—Industrial Internet of Things, Industry 4.0, Edge Computing, Fog Computing, Timeliness, Stability, Lyapunov Optimization Technique.

1 Introduction

Fog computing (FC) in some aspects differs from cloud computing, but perhaps the main distinguishing aspect is agility [1]. Bringing computation resources to the edge of the network makes it possible to lower the response time [2]. Thus, time-sensitive requests can be served at the network edge [3], which also leads to another advantage, that is, making the network core less busy [4].

In Industrial Internet of Things (IIoT), these are the critical-mission sensors and actuators connecting to the Internet. As a result, IIoT applications enforce pressing demands on the stability of the system (even under stressed conditions), preserving time constraints and energy conservation [5]. Processing the requests at the premises of users, near the source of data and where they may be consumed, helps them meet their time constraints. Thus, in various aspects, FC can help IIoT [6]. Although structural changes (i.e., introducing a fog layer) is necessary, it is not profitable unless accompanied with a *dispatching mechanism* to place each request on a proper computing resource (fog/cloud). An efficient request dispatching strategy makes decisions based on available resources, network conditions and request requirements, and therefore, it can lead to a lower service time and lower energy consumption, among others [7].

This paper proposes a novel request dispatching (or resource allocation) algorithm for fog-enabled IIoT applications. There are essential features specific to fog-enabled IIoT systems that urge the need for brand new resource allocation mechanisms. Specifically, fog-enabled IIoT systems are characterized by the following features:

Timeliness: In IIoT, *timeliness* has significant importance, as many applications impose a deadline for the time between sending their requests and receiving the corresponding responses. For example, the obstacle detection (OD) for autonomous vehicles (AV) requires the timeliness of service to be effective. Consider a scenario in which suddenly an obstacle (human being, large object or something harmful) appears on the road when an AV is driving on the highway. Here, the OD application should recognize the type of obstacle and its velocity to make a suitable decision, such as immediate collision avoidance or lane change maneuver. This scenario is possible if the time needed for data communication and processing does not exceed the tolerable delay. Indeed, if the AV drives at a speed of 200 Km/h, it would cover a distance of 50 m in about 900 ms. Therefore, the OD application has a deadline of 90 ms [8], [9]. Similarly, the monitoring application for robotic arms in the automated industry (for instance: using robotic arms for movement of objects) also requires a response time less than 100 ms [10]. These are time-critical applications where timeliness does matter, and deadline awareness is essential [8].

Various applications may behave differently if the deadline is missed. This difference can be modeled by considering different costs (or utilization loss) for various applica-

- *A. Karimiafshar, M. Hashemi, and M. Heidarpour are with the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 8415683111, Iran. E-mail: a.karimiafshar@ec.iut.ac.ir, {hashemim, mrheidar} @iut.ac.ir.*
- *A. N. Toosi is with the Department of Software Systems and Cybersecurity, Faculty of Information Systems, Monash University, Clayton, VIC 3800, Australia. E-mail: adel.n.toosi@monash.edu.*

tions as the response time exceeds their deadlines. However, the number of *deadline misses* is a key performance indicator in such systems [11], [12], [13]. It is worth noting that there is a difference between the minimization of service time and the number of deadline misses. For example, consider a system containing three nodes with different processing capabilities ($F1 < F2 < F3$), and a task "A" with a 35ms deadline arriving into the system. Also assume that execution of task A on $F1$, $F2$, and $F3$ lasts for 40ms, 33ms, and 25ms, respectively. As a result, a service time minimization strategy forces us to execute task A on $F3$. But, if instead, we focus on minimizing the number of deadline misses, a degree of freedom shows up as there is no difference between $F2$ and $F3$. Consequently, the degrees of freedom of this kind may be leveraged to better cope with stressed situations.

Stability: IIoT applications are categorized as mission-critical, and therefore, any instability in the system may lead to catastrophic consequences. Stability in the form of bounded-input bounded-output (BIBO) for the system under consideration means that the backlogs of task queues for individual fog nodes (FNs) do not grow unboundedly as long as the average computational demand of time-sensitive tasks arriving into the system is less than the total computation capability of all FNs, excluding the cloud [14]. Thus, for practical scenarios with limited buffering (queuing) capacity, keeping backlogs as small as possible would be a design goal of any stable request dispatcher. Furthermore, IIoT heavily relies on machine-to-infrastructure or machine-to-machine communications, and therefore, requires the system to be robust, stable and secure [6].

Dispersion and heterogeneity: Fog nodes can be dispersed in various geographical locations and heterogeneous in their electrical/computational power and persistency characteristics. As a result, dispatching algorithms devised originally for cloud computing can not be directly reused for fog computing systems. Specifically, using methods such as virtual machine consolidation and turning on/off the servers is not suitable in fog computing systems.

Energy: In contrast to the cloud, FNs can be power-limited devices that share their resources intermittently. As a result, special care is needed for conservative energy consumption [15], [16].

In this paper, based on a comprehensive review of the related works (Section 2), we consider a three-layer IIoT system model (Section 3) consisting of industrial equipment at the bottom (as end devices), FNs at the middle (at the edge of the network) and the cloud at the top (remote data centers accessed through the Internet). We, then (in Section 4), investigate the request dispatching among the FNs and the cloud in such a way to *minimize total energy* consumption subject to preserving *timeliness* and system *stability*. The stochastic and dynamic nature of the system parameters (such as the requests' arrival rate and attributes) makes the optimal solution intricate and too heavy for practical deployment. As a result, we resort to the so-called "Lyapunov Optimization Technique (LOT)" to devise a greedy suboptimum dispatching algorithm that is stable and works independently of the dynamics of the system parameters (Section 5). In our problem formulation, we consider minimizing energy as the objective function and convert stability and timeliness conditions, respectively, into queues and *virtual queues* [14]. As a result, while LOT minimizes the energy, it also guarantees the stability of our proposed algorithm in terms of the number of queued requests and the deadline misses. Extensive simulations and practical experiments are conducted (in Section 6) to evaluate the effectiveness of the proposed method under different conditions.

The main contributions of the paper are summarized as follows:

- Besides system stability and energy minimization, we stress on and grant special attention to *timeliness* as an essential service requirement in many IIoT applications.
- Based on LOT and virtual queues, a dynamic request dispatching algorithm is proposed, which jointly minimizes energy consumption and adaptively addresses system stability and timeliness. The proposed method also incorporates a novel mechanism to proactively choose and send some requests to the cloud, which helps the system provide more space for future requests and makes it more robust against dynamic system conditions.
- We further demonstrate the linear time-complexity of the proposed algorithm and prove its stability with respect to the number of queued requests and deadline misses.
- Based on the simulation experiments, we perform the sensitivity, scalability, and model parameter analysis.
- A prototyping platform is implemented to validate the proposed method.

2 Related Works

2.1 Request Dispatching in FC Environment

There have been extensive studies on computation offloading, resource management and request dispatching in the context of FC [17], [18]. For example, Ni et al. [19], considering the requests' completion time and price as the objectives, proposed a resource allocation strategy based on priced timed Petri nets. Deng et al. [20] investigated the workload allocation problem considering the tradeoff between power consumption and transmission delay in FC. Meng et al. [21] investigated the delay-constrained computation offloading problem in FC, considering both energy consumption and time constraint of the tasks. Similarly, Wang et al. [22] considered computation offloading in wireless powered Mobile-Edge Computing (MEC) by jointly optimizing the frequency of the computing nodes, number of offloaded bits and transmission energy beamforming. By considering both partial and binary offloading modes, [23] introduces an energy-efficient MEC design that aims to minimize the total energy consumption subject to computational latency constraints. Using a software-defined network, Zeng et al. [7] introduced a task scheduling and resource management method to minimize request completion time. Also, Gu et al. in [24] proposed a cost-efficient task distribution in fog-enabled medical cyber-physical systems. In [25], Skarlat et al. presented a conceptual framework for resource provisioning in FC, considering heterogeneity in applications and resources. In [26], Zhang et al., using

game theoric approaches, addressed the problem of resource allocation in a distributed fashion. Such works deal with resource management and request dispatching in FC by taking different objectives into account and through various methods. However, their works are so general, and they do not consider the specific requirements in IIoT, such as the real-time nature of requests in IIoT applications.

2.2 Request Dispatching in IIoT

In the context of IIoT, Chekired et al. [27] proposed a hierarchical architecture of FC and a workload assignment algorithm for offloading peak loads over higher tiers. They divided the requests into low and high priority requests and used two priority queuing models. Finally, using mixed nonlinear integer programming, they proposed a branch and bound solution to the problem. Mishra et al. [28] investigated the scheduling of requests among fog servers using metaheuristic techniques. They considered the tradeoff between energy consumption and makespan. Furthermore, Li et al. [29] proposed an implementation architecture for fog-enabled IIoT. They also introduced a resource partitioning scheme based on service popularity by exploiting Zipf's law. Moreover, Shi et al. [5] introduced a load balancing strategy based on a genetic algorithm in an IIoT environment. These works dealt with the resource scheduling problem in the context of IIoT. However, they did not focus on the time-sensitivity of requests and system stability as we do in this paper. Furthermore, we use the LOT in the category of stochastic methods, which completely fits into the stochastic nature of the problem.

2.3 Stochastic Methods of Request Dispatching

There are some works that try to solve the resource allocation and computation offloading problem by leveraging stochastic methods, such as [30], [31], [32], [34], [35]. For example, Wang *et al*. [30], based on the Markov decision processes (MDP) and reinforcement learning framework, introduced a dynamic reinforcement learning scheduling algorithm and a deep dynamic scheduling algorithm to solve the computation offloading problem. However, such MDP-based methods, by increasing the number of transitions, suffer from the curse of dimensionality.

Besides, Kwak et al. [31] adopted LOT to minimize the time-averaged energy consumption for given delay constraints. Also, Zhao et al. [32] considering a heterogeneous delivery wireless network solved the problem of node assignments at the control tier and allocating resources at the access tier based on LOT. They dealt with service hosting, resource allocation and computation offloading in the context of FC, considering service time minimization and reducing energy consumption. Different from the above works, which mainly focused on system performance, in this work, we focus on the system timeliness and try to dispatch requests so that more requests meet their deadlines while minimizing the energy consumption of the computing nodes.

In recent works [15], [33] and [16] are the most relevant works to our work in this paper. Yang et al. [15] investigated the problem of dynamic task scheduling considering the tradeoff between energy consumption and service time.

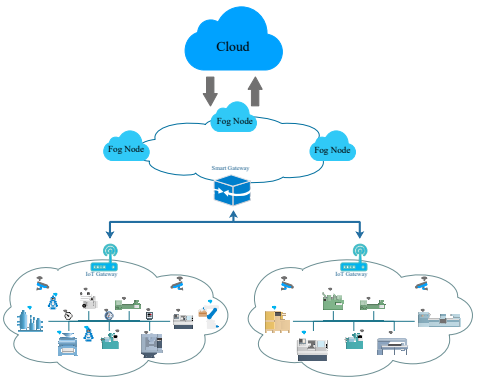


Fig. 1: High level fog network architecture.

They adopted LOT and proposed an algorithm as the solution to minimize the service time. In [33], we investigated the request dispatching problem in a fog environment to match energy consumption with the green energy profile. Leveraging LOT, we proposed an online algorithm to simultaneously dispatch requests among available computing nodes and adjust frequency and modulation levels. However, we did not consider time-sensitivity of the requests and system timeliness. In the concept of timeliness, we are concerned with meeting deadlines instead of minimizing the average service time. Indeed, the requests must be served before the deadline. Therefore, it has a binary nature; as requests either may pass the deadline or not. Primarily, the Lyapunov framework is used to deal with analog variables, but in the current work, we face the challenge of dealing with binary variables while using the Lyapunov framework. This challenge is new and has not been considered before in the literature. Furthermore, despite our previous work that we tried to match the energy consumption with the green energy profile, in this work, we have focused on reducing the energy consumption. Finally, the problem is investigated specifically with regard to the characteristics of IIoT and Industry 4.0.

Chen et al. [16] investigated task admission and service hosting in a fog environment to minimize average service time while satisfying battery energy constraints. They formulated the problem as a mixed-integer nonlinear stochastic optimization problem. Based on LOT, the long-term formulation of the problem is relaxed and then is converted to a sequence of easily solvable per-slot sub-problems that depend only on currently available system information. Finally, they proposed an online distributed algorithm as the solution. The studies mentioned above focused on resource allocation and task scheduling in a fog environment and used LOT to derive algorithms to minimize the service time. However, they considered service time minimization as the main objective and did not focus on the system timeliness as we do in this paper. It is noteworthy that minimizing the service time does not necessarily lead to timeliness. Focusing on service time optimization will not remove the outliers that are unacceptable to time-critical applications such as autonomous vehicles. Finally, Table 1 presents the summary of the most relevant works.

TABLE 1: THE COMPARISON BETWEEN THIS WORK AND THE MOST RELEVANT WORKS

Reference	Technology	Problem	Lyapunov (stability)	Objectives S.T [1]	Objectives D.M [2]	IIoT
[19]	Fog computing	Resource allocation		✓		
[20]	Fog computing	Workload allocation		✓		
[21]	Fog computing	Computation offloading			✓	
[7]	Software-defined fog network	Resource management		✓		
[22]	Mobile edge computing	Computation offloading		✓		
[23]	Mobile edge computing	Computation offloading		✓		
[24]	Mobile edge computing	Resource management		✓		
[25]	Fog computing	Resource provisioning		✓		
[26]	Cloud, edge and fog computing	Resource allocation		✓		
[27]	Industrial fog computing	Workload assignment		✓		✓
[28]	Fog computing	Service allocation		✓		✓
[29]	Fog computing	Resource partitioning		✓	✓	✓
[5]	Fog computing	Load balancing		✓		✓
[30]	Fog computing	Computation offloading		✓		✓
[31]	Mobile cloud computing	Resource allocation	✓		✓	
[32]	Content delivery wireless network	Resource allocation	✓	✓		
[15]	Homogeneous fog networks	Task scheduling	✓	✓		
[33]	Fog computing	Request dispatching	✓	✓		
[16]	Industrial fog computing	Computation offloading	✓	✓		✓
This work	Fog computing	Request dispatching	✓	✓	✓	✓

[1] S.T: Service Time (or Makespan [28]) [2] D.M: Deadline Miss (or Delay Constraint)

3 System Model

We consider a fog network composed of N FNs and a cloud at a remote data center. In IIoT, there are many IoT devices running applications with different requirements in terms of data rate and time constraints. Because of the limited resources, IoT devices may need to be served by external resources. We assume that the IoT devices send their requests to a gateway with a controller which decides to redirect requests to the best-suited computing node among available ones (preferably one of the FNs or, if necessary, the cloud), as illustrated in Fig. 1. Once the scalability matters, the IoT devices and FNs may be grouped into clusters (domains) with dedicated controllers, and all requests in one cluster are dispatched via the cluster's controller. The optimum dispatching algorithm requires information about the attributes of the requests, the conditions and capabilities of FNs, and the bandwidth of the network links.

For the sake of readability, in the following, we first introduce the notations used throughout the paper.

Notations: The vectors are specified in bold face letters, such as $\mathbf{Q}$. We use $\mathbb{E}\{.\}$ to specify the expectation operator. In order to show that parameter f is a function of x, we use $\hat{f}(x)$. Also, $\limsup$ denotes Limit Superior. Furthermore, a summary of key symbols are presented in Table 2.

The controller maintains a queue model of the computing nodes and updates the model parameters continuously. Fig. 2 presents the queue model of the system. We assume a time-slotted dispatching strategy, with slots indexed by $t \in \{0, 1, 2, \dots\}$. $A(t)$ denotes the arrival rate into the controller at time slot t in the unit of instructions per second. It is assumed that $A(t)$ is independent and identically distributed (i.i.d.) over time slots. $R_i(t)$ denotes the amount of requests assigned to the i^{th} computing node at time slot t, and we can write $A(t) = \sum_{i=1}^{N+1} R_i(t)$. The requests go through the queues to get served. The queues' backlog is one of the important information used by the controller to decide about how to dispatch the requests. The dispatching decision at time t is denoted by $\boldsymbol{\alpha}(t)$ which defines a mapping between K^t, the set of requests arrived at time slot t, and FNs (indexed by $i = 1, \cdots, N$) and the cloud (indexed by $i = N + 1$) as $\boldsymbol{\alpha}(t) = \{(k, i) | k \in K^t \text{ and } i \in \{1, 2, \dots, N + 1\}\}$.

TABLE 2: THE SUMMARY OF KEY SYMBOLS

Symbol	Definition
t	Index of time slots
$A(t)$	Request arrival rate into the system at time slot t
i	Index of computing nodes
$R_i(t)$	Request arrival rate into the computing node i at time slot t
$B_i(t)$	Service rate of the computing node i at time slot t
K^t	Set of requests arrived at time slot t
$\boldsymbol{\alpha}(t)$	Vector of decision control at time slot t
$Q_i(t)$	Queue backlog related to the computing node i at time slot t
k	Index of requests
$r_i(t)$	Transmission rate of the computing node i at time slot t
M_k	Volume of data need to be transmitted by request k
S_k	Processing demand of the request k
$e_i^c(t)$	Computation energy consumption of the computing node i at time slot t
$e_i^{tr}(t)$	Communication energy consumption of the computing node i at time slot t
$e_i(t)$	Total energy consumption of the computing node i at time slot t
$\psi_k(t)$	Service time of the request k
$L(t)$	Lyapunov function
$\Delta(L(t))$	Drift in Lyapunov function

3.1 Queue Dynamic

As illustrated in Fig. 2, there are $N + 1$ queues related to computing nodes (the FNs and the cloud) in the system. After dispatching requests to computing nodes by the controller, they wait in the corresponding queues for their turns to be served. The dynamics of the i^{th} queue is obtained by,

$$Q_i(t + 1) = \max[Q_i(t) - B_i(t), 0] + R_i(t), \qquad (1)$$

for $t \in \{0, 1, 2, \dots\}$, where $Q_i(t)$ denotes the queue backlog of the i^{th} computing node at time slot t, and $B_i(t)$ denotes the service rate of the computing node. Moreover, for $B_i(t)$ we can write $B_i(t) = f_i/\varphi$ where f_i is the CPU frequency and φ is the average number of cycles per instruction [20].

3.2 Computation and Communication Models

Different computing nodes may have different processing power. They are also generally dispersed across the network. Therefore, assigning a request to one computing node or another may lead to varying amounts of computation and

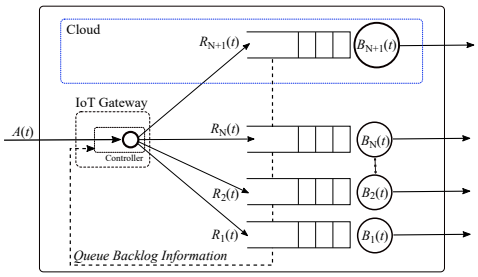


Fig. 2: Queue model of the system.

communication delay and consumed energy. The delays and energy consumption are variant and depend on the system conditions and workload. In the following, we present the delay and energy consumption models.

3.2.1 Delay

Computation Delay: The computation delay refers to the delay which is incurred by serving a request in a computing node at the fog or cloud tier. Following the model in [16], the computation delay of the request k at the computing node i is obtained by,

$$\tau_{k,i}(t) = \frac{S_k}{B_i(t)}, \tag{2}$$

where S_k denotes the processing demand of the request k in the form of Million Instructions (MI).

Communication Delay: The communication delay refers to the delay which is incurred during forward/backward transmission of the data between IoT devices and the computing node in the fog or cloud. Similar to the model in [20], the communication delay for the request k and the computing node i is obtained by,

$$D_{k,i}(t) = \frac{M_k}{r_i(t)}, \tag{3}$$

where $r_i(t)$ denotes the transmission rate, at time slot t, of the link from the computing node i to the IoT device originating the request k, and M_k represents the volume of data needed to be transmitted.

3.2.2 Energy Consumption

The computing node i consumes energy $e_i(t)$ either for computation, denoted by $e_i^c(t)$, or communication, denoted by $e_i^{tr}(t)$. Therefore, the total energy consumption can be written as $e(t) = \hat{e}(\boldsymbol{\alpha}(t)) = \sum_{i=1}^{N+1}(e_i^c(t) + e_i^{tr}(t))$. In the following, we present the computation and communication energy model.

Computation Energy: Power consumption in a processing device has two parts, static and dynamic. Static part, $P_{i,s}^c$, is consumed even if the device is idle, while the dynamic part, $P_{i,d}^c$, is proportional to the utilization of the resources [36].

$$p_i^c(t) = p_{i,s}^c + p_{i,d}^c(t) = p_{i,s}^c + (p_{i,max}^c - p_{i,s}^c)\vartheta_i(t), \tag{4}$$

where $p_{i,max}^c$ represents the maximum power that the computing node i can dissipate, and $\vartheta_i(t) \leq 1$ is the resource utilization. Thus, the computation energy e_i^c can be obtained by $e_i^c(t) = p_i^c(t)\tau_i(t)$, where $\tau_i(t) = \sum_{k \in K_i^t} \tau_{k,i}(t)$. Furthermore, K_i^t denotes the set of requests assigned to the computing node i at time slot t.

Communication Energy: The communication energy is a function of transmission rate, and can be written as [37]:

$$e_i^{tr}(t) = \sum_{k \in K_i^t} (\beta_i x_i (2^{r_i(t)/x_i} - 1) + p_{i,s}^{tr})(D_{k,i}(t)), \tag{5}$$

where x_i is the fixed symbol rate, $\beta_i(t)$ is a parameter determined by the noise level and the transmission quality, and $p_{i,s}^{tr}$ is the transmitter's statics (modulation-independent) power consumption.

4 Problem Statement

In this section, we mathematically formulate the problem of request dispatching. First, we present the constraints related to timeliness, load balancing, and stability. Then, we introduce an optimum dispatching strategy as the solution to a constrained stochastic and non-linear optimization problem in which the averaged energy consumption is minimized subject to the timeliness and system stability constraints.

4.1 Problem Constraints

4.1.1 Timeliness Constraint

In IIoT applications (e.g., industry 4.0), processing the requests in a timely manner according to the context has significant importance. Delays beyond the specified deadline may lead the system to an unstable situation. So, the total service (response) time[1] for each request k, denoted by ψ_k, should be preferably less than its related delay threshold, Γ_k. The total service time for request k assigned to the computing node i under allocation decision $\boldsymbol{\alpha}(t)$, is obtained by $\psi_k(t) = \hat{\psi}_k(\boldsymbol{\alpha}(t)) = u_{k,i}(t) + w_{k,i}(t) + \tau_{k,i}(t) + d_{k,i}(t)$, where $u_{k,i}(t)$ is the delay incurred during uploading the input data of the request k from the IoT device to the computing node i, and $w_{k,i}(t)$, $\tau_{k,i}(t)$, and $d_{k,i}(t)$, are, respectively, the related waiting (queuing) delay, computation delay, and the delay of transmitting the results back to the IoT device. The parameters $\tau_{k,i}(t)$ is obtained by (2), $u_{k,i}(t)$ and $d_{k,i}(t)$ are obtained by (3), and the waiting delay is obtained by $w_{k,i}(t) = Q_i(t)/B_i(t)$ at time slot t.

We define $\varphi_k(t) = \hat{\varphi}_k(\boldsymbol{\alpha}(t)) = \max\{\rho(\hat{\psi}_k(\boldsymbol{\alpha}(t)) - \Gamma_k), 0\}$ as an indicator of how far the service time is from the Γ_k threshold [2], and parameter ρ is used to determine how much violating a deadline may cost to an application. We try to keep the expected value of $\varphi_k(t)$ below a throttling threshold, C_D by enforcing,

$$\bar{\varphi} = \limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{\varphi(t)\} \leq C_D,$$

where $\varphi(t)$ is defined as $\varphi(t) = \sum_{k=1}^{K^t} \varphi_k(t)$. The throttling threshold C_D provides control over the number of deadline misses and can be tuned based on the IIoT application under consideration. A large value of C_D causes more deadline misses to be allowed in the system.

1. We consider the time from when the requests are sent to a computing node until the time the results are sent back to the IoT devices as the total service time.

2. We can also substitute Γ_k with the average of the deadlines, the upper/lower bound of the deadlines or any approximation of them.

4.1.2 Load Balancing Constraint

Dispatching the incoming requests evenly, as much as possible, among the computing nodes is the best practice for *fair* and *stable* resource sharing on average, whenever we do not have the knowledge of system dynamics [5].

The backlog of each queue in the system is an indicator of the load at that node. Due to heterogeneity in the computing nodes, load balancing should be performed according to the nodes' capabilities. Therefore, after normalizing the queues' backlog with respect to nodes' capabilities, we impose the load balancing constraint in the form of,

$$\bar{q_i} = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{Q_i(t)/\nu_i(t)\} = Q_{avg}(t),$$

for $i \in \{1, 2, \cdots, N+1\}$, and the average of all the real queues $Q_{avg}(t)$ for each time slot. While $\nu_i(t)$ helps to normalize the backlog of the queue i, and is obtained by $\nu_i(t) = B_i(t)/B_{max}(t)$, where $B_{max}(t) = \max_{i=1,2,\cdots,N+1} B_i(t)$.

4.1.3 Selection between Fog and Cloud Constraint

Although it is desirable to serve as many requests as possible at the fog tier, in our method, some are opportunistically chosen to be sent to the cloud to provide more space for other existing or future ones that may have more stringent delay constraints. In this case, the requests that are best suited to the cloud in terms of computation and communication are sent to the cloud. These are computation-intensive requests with the least amount of data transmission to avoid making the network core busy.

In other words, as "computationally-intensive" tasks may block FNs (with restricted resources) from timely serving other in queue tasks, they represent good candidates to be offloaded to the cloud. However, the computation cost is not the only factor to assess whether a task should be sent to the cloud or not. Another important factor is the "communication cost", as it affects the delay and the energy required for the communication too. Therefore, any decision criterion for cloud offloading may mix these two factors in a suitable form. In this paper, we consider a utility function in the form of $U_k(t) = S_k/M_k$. If the value of this function for a request is greater than a predefined threshold, the request is a good candidate to be offloaded to the cloud. The threshold value depends on the queue backlog and changes accordingly. So, we define the long term average utility of requests which is served by the FNs as

$$\bar{U} = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{U_k(t)\} \leq C_L$$

for $k \in K_F^t$, where K_F^t denotes the set of requests assigned to the FNs at time slot t. Moreover, C_L is a threshold for suitability of a request to be sent to the cloud. Indeed, we dynamically tune the suitability of the requests for sending to the cloud by taking the current status of the queues' backlog into consideration in the form of $C_L \times Q_{i_\epsilon}(t) = C_\gamma$ for $i \in \{1, 2, \ldots, N\}$, with some finite constant value C_γ which is set based on the system management policy. A large value of C_γ indicates that a request must have high computing demands or low communication requirements to be sent to cloud. We define $Q_{i_\epsilon}(t) = Q_i(t) + \epsilon$ for $\epsilon > 0$, therefore considering $Q_{i_\epsilon}(t) > 0$, we can write $C_L = Q_{i_\epsilon}^{-1}(t) \times C_\gamma$.

4.1.4 Stability

In the queue model of the system, stability refers to a condition that all the queues work normally and we do not have any congested queue. Mathematically speaking, the queue of the computing node i is stable [14] if,

$$\bar{Q_i} = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{|Q_i(t)|\} < \infty. \tag{6}$$

Equation (6) not only considers the long term behavior of the system, but also requires the queues' backlogs to be always finite.

4.2 Problem Formulation

Now we are in the stage to present our optimum dispatching strategy which minimizes the total consumed energy subject to timeliness and stability conditions. The proposed optimum dispatching strategy is the solution to the following stochastic constrained optimization problem:

$$\textbf{P1:} \min_{\boldsymbol{\alpha}(t)} \ (\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{\hat{e}(\boldsymbol{\alpha}(t))\}) \tag{7}$$

$$S.T\ \bar{\varphi} \leq C_D \tag{7a}$$

$$\bar{q_i} = Q_{avg}(t) \tag{7b}$$

$$\bar{U} \leq C_L \tag{7c}$$

$$\bar{Q_i} < \infty \ for\ i \in \{1, 2, \ldots, N+1\}. \tag{7d}$$

The problem in (7) minimizes the long term average of energy consumption subject to the conditions (7a) to (7d). Condition (7a) presents the timeliness constraint by enforcing the requests to be dispatched in the right place and served before their delay thresholds, (7b) is related to the load balancing constraint. Condition (7c) is for the selection between the fog and cloud, by serving the proper requests in the FNs and opportunistically sending some requests to the cloud, and (7d) is for preserving the stability of queues.

P1 is a stochastic optimization problem with various stochastic parameters. In each time slot, it needs to make decisions on choosing the most proper nodes in the fog or cloud tier to minimize energy consumption while providing system stability and timeliness, which requires the knowledge of the system dynamics to be known in advance. However, obtaining such knowledge of the system is not easy. Therefore, **P1** is a highly challenging problem. Thus, in this paper, we leverage LOT to derive an approximate algorithm that works without a priori knowledge of future system dynamics.

5 Proposed Method

In this section, we stand on the concrete "Lyapunov stability theory" to derive a solution for the main problem **P1**. Specifically, we first cast **P1** into the Lyapunov optimization framework. Then, we leverage the virtual queue concept and turn the constraints into a pure stability problem. Finally, an online request dispatching algorithm is presented using the so-called Lyapunov Optimization Technique (LOT). Although

the proposed algorithm is sub-optimum, it has provable performance guarantees and outperforms the state-of-the-art methods, as demonstrated in Section 6.

5.1 Compiling P1 into LOT Language

In control systems theory, the Lyapunov function has been widely used to prove the stability of systems. On the other hand, problems in which, along with stability, the *performance* is also important, can benefit from a variant of the Lyapunov function, which is called the Lyapunov Optimization Technique (LOT). Here, we use LOT to jointly stabilize system queues and minimize energy consumption as the penalty. In LOT, a positive scalar function called the Lyapunov function, $L(t)$, is defined. $L(t)$ represents the status of congestion in the system's queues. We use the Lyapunov function as defined in [14], in the form of

$$L(t) = \frac{1}{2}\sum_{i=1}^{N+1} Q_i(t)^2, \tag{8}$$

recalling that $Q_i(t)$ is the backlog of the i^{th} queue. A small value of $L(t)$ indicates that all the queues work normally, but a large value for $L(t)$ means that at least one of the queues is congested, which leads the system into an unstable situation. The LOT builds on the fact that if the initial state of the queues is bounded, "greedily" minimizing

$$\Delta(L(t)) = \mathbb{E}\{L(t+1) - L(t)|Q_1(t), Q_2(t), \ldots, Q_{N+1}(t)\}, \tag{9}$$

at each time slot ensures the stability of the system queues. The function $\Delta(L(t))$ is called *drift* in the Lyapunov function and as mentioned plays a central role in LOT.

The next step in LOT is to incorporate a *penalty* term in addition to the drift. The purpose of the penalty is to bring, besides the stability constraints, other performance metrics into the scene. Therefore, instead of minimizing the pure drift, LOT considers minimizing a hybrid term of drift plus penalty in the form of

$$\Delta(L(t)) + V\mathbb{E}\{P(t)|\mathbf{Q(t)}\}, \tag{10}$$

where $P(t)$ is the penalty function, and V is a scaling factor to adjust the relative importance of the stability and performance with respect to each other. As minimizing energy is the objective function in **P1**, in our case, $P(t)$ is an appropriate representation of the energy consumption. While **Q(t)** is the vector of all the real queues and defined as, $\boldsymbol{Q(t)} \triangleq [Q_1(t), Q_2(t), \ldots, Q_{N+1}(t)]$.

One final remark about LOT that remains is how to deal with problems that contain other constraints besides the stability. The answer to this question is the concept of *virtual queues* which we ought to deploy in order to thoroughly tackle **P1** using the LOT strategy.

Specifically, virtual queues can be utilized to model time-average constraints within the LOT. Regarding the constraints defined in previous sections, we use virtual queues to turn the satisfying constraints problem into a pure stability one within the LOT. Related to each of the constraints, we define a virtual queue. Each queue has its own updating equation, which is specifically defined for that queue. Let suppose that we are successful in modeling various (non-stability) constraints as M virtual queues, $\tilde{\mathcal{Q}}_j, j = 1, \cdots, M$, then all we need, in order to use LOT, is just to generalize the Lyapunov function as

$$L(t) = \frac{1}{2}\left(\sum_{i=1}^{N+1} Q_i(t)^2 + \sum_{j=1}^{M} \tilde{\mathcal{Q}}_j(t)^2\right) \tag{11}$$

to include both real (stability) and virtual (non-stability) queues. In the following we derive the virtual queues corresponding to different constraints in **P1**.

– *Constraint (7a)*: We define Z, a virtual queue that is related to the delay constraint, discussed in the previous sections. Virtual queue Z is updated according to

$$Z(t+1) = \max[Z(t) + y(t), 0], \tag{12}$$

where $y(t) = \hat{y}(\boldsymbol{\alpha}(t)) = \hat{\varphi}(\boldsymbol{\alpha}(t)) - C_D$.

– *Constraint (7b)*: Related to the load balancing constraint, we define virtual queues H_i, $i = 1 \cdots, N+1$, with the updating equation of

$$H_i(t+1) = H_i(t) + g_i(t), \tag{13}$$

where $g_i(t) = \hat{g}_i(\boldsymbol{\alpha}(t)) = \frac{Q_i(t)}{\nu_i(t)} - Q_{avg}(t)$.

– *Constraint (7c)*: To choose between the fog and the cloud, we define the virtual queue G. The update equation for the virtual queue G is expressed by,

$$G(t+1) = \max[G(t) + f(t), 0], \tag{14}$$

where $f(t) = \hat{f}(\boldsymbol{\alpha}(t)) = \sum_{k\in K^t} \hat{U}_k(\boldsymbol{\alpha}(t)) - C_L$.

5.2 Online Algorithm

Now we are in the stage of solving **P1** using LOT. First we define $\boldsymbol{\theta}(t)$ to be a vector of concatenation of all the real queues $\boldsymbol{Q}(t)$ and virtual queues $Z(t)$, $\boldsymbol{H}(t) = [H_i(t), \cdots, H_{N+1}(t)]$, and $G(t)$ as $\boldsymbol{\theta}(t) \triangleq [\boldsymbol{Q}(t), Z(t), \boldsymbol{H}(t), G(t)]$. Accordingly, the generalized Lyapunov function, for the problem **P1**, that takes into account real and virtual queues, can be written as

$$L_{\boldsymbol{\theta}}(t) = \frac{1}{2}\Big(\sum_{i=1}^{N+1} Q_i(t)^2 + Z(t)^2 + \sum_{i=1}^{N+1} H_i(t)^2 + G(t)^2\Big), \tag{15}$$

and the drift takes the form of

$$\Delta(L_{\boldsymbol{\theta}}(t)) = \mathbb{E}\{L_{\boldsymbol{\theta}}(t+1) - L_{\boldsymbol{\theta}}(t)|\boldsymbol{\theta}(t)\} \tag{16}$$

On the other hand, the objective function in **P1** leads us to consider $P(t) = \mathbb{E}\{e(t)|\boldsymbol{\theta}(t)\}$ as the penalty. Therefore, the drift-plus-penalty corresponding to **P1** is given by

$$\Delta(L_{\boldsymbol{\theta}}(t)) + V\mathbb{E}\{e(t)|\boldsymbol{\theta}(t)\} \tag{17}$$

which can be upper-bounded as (see appendix A in the supplementary material for the proof)

$$\begin{aligned}
&\Delta(L_{\boldsymbol{\theta}}(t)) + V\mathbb{E}\{e(t)|\boldsymbol{\theta}(t)\} \leq \\
&\Upsilon + V\mathbb{E}\{\hat{e}(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\} - \sum_{i=1}^{N+1} Q_i(t)B_i(t) \\
&+ \sum_{i=1}^{N+1} Q_i(t)\mathbb{E}\{\hat{R}_i(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\} + Z(t)\mathbb{E}\{\hat{y}(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\} \\
&+ \sum_{i=1}^{N+1} H_i(t)\mathbb{E}\{\hat{g}_i(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\} + G(t)\mathbb{E}\{\hat{f}(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\},
\end{aligned} \tag{18}$$

where Υ is a constant.

To stabilize all the queues and minimize the penalty, we need to minimize the right hand side of inequality (18). Therefore, we obtain the minimization problem **P2**, given in (19), as an LOT approach to approximately solve **P1**.

$$\begin{aligned}\textbf{P2:}\min_{\boldsymbol{\alpha}(t)}\ & V\mathbb{E}\{\hat{e}(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\}+\sum_{i=1}^{N+1}Q_i(t)\mathbb{E}\{\hat{R}_i(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\}\\ &+Z(t)\mathbb{E}\{\hat{y}(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\}+\sum_{i=1}^{N+1}H_i(t)\mathbb{E}\{\hat{g}_i(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\}\\ &+G(t)\mathbb{E}\{\hat{f}(\boldsymbol{\alpha}(t))|\boldsymbol{\theta}(t)\}.\end{aligned} \tag{19}$$

In order to solve the long term minimization problem **P2**, we can opportunistically minimize **P3** at every time slot [14].

$$\begin{aligned}\textbf{P3:}\min_{\boldsymbol{\alpha}(t)}\quad & V\hat{e}(\boldsymbol{\alpha}(t))+\sum_{i=1}^{N+1}Q_i(t)\hat{R}_i(\boldsymbol{\alpha}(t))+Z(t)\hat{y}(\boldsymbol{\alpha}(t))\\ &+\sum_{i=1}^{N+1}H_i(t)\hat{g}_i(\boldsymbol{\alpha}(t))+G(t)\hat{f}(\boldsymbol{\alpha}(t)).\end{aligned} \tag{20}$$

To efficiently solve **P3**, we propose a greedy algorithm which starts from the first task received during time slot t, dispatches this task to the node which minimizes the objective function in **P3**, and one-by-one, repeats this procedure for all other tasks received during time slot t.

Algorithm 1: *LGA*

1: **Initialization**
   Initialize the model parameters in the LOT
2: **Foreach** t **Do**
   */*Obtaining required parameters and finding the best* $\boldsymbol{\alpha}(t)$ **/*
3: **For** $k=1$ **to** K^t
4: **For** $i=1$ **to** $N+1$
5: $\boldsymbol{\alpha}_k(t)\triangleq i$
6: $\hat{y}(\boldsymbol{\alpha}_k(t))=\max\{\rho(\hat{\psi}(\boldsymbol{\alpha}_k(t))-\Gamma_k),0\}-C_D$
8: $\hat{f}(\boldsymbol{\alpha}_k(t))=\hat{U}(\boldsymbol{\alpha}_k(t))-C_L$
9: $value\_DpP[i]=V\hat{e}(\boldsymbol{\alpha}_k(t))+Q_i(t)(\frac{H_i(t)}{\nu_i(t)}+1)$
$+Z(t)\hat{y}(\boldsymbol{\alpha}_k(t))+G(t)\hat{f}(\boldsymbol{\alpha}_k(t))$
End For
10: $\boldsymbol{\alpha}^*_k(t)=\arg\min_{\boldsymbol{\alpha}_k(t)}(value\_DpP)$
11: *Logically update the queues based on the model*
End For
12: $\boldsymbol{\alpha}^*(t)=\cup_{k=1}^{K^t}\boldsymbol{\alpha}^*_k(t)$
13: *Dispatch according to* $\boldsymbol{\alpha}^*(t)$
14: *Update the Queues*
End While

The proposed algorithm, Lyapunov greedy algorithm (LGA), is summarized in Algorithm 1, and explained as follows.

Line 1: Initialize the required parameters in the LOT, such as the controlling parameter V.

Line 3-9: For all the requests go through the available computing nodes and calculate the value of the expression in (20).

Line 10-11: Find the computing node which leads to the least value for the expression in (20) to obtain $\boldsymbol{\alpha}^*_k(t)$. Logically update the queue based on $\boldsymbol{\alpha}^*_k(t)$ and go through further requests.

Line 12-14: Form the whole allocation decision, $\boldsymbol{\alpha}^*(t)$, by union of all $\boldsymbol{\alpha}^*_k(t)$. Dispatch the requests and update the queues.

Complexity Analysis: LGA greedily minimizes the expression in (20) at each time slot. Regrading the fact that requests are dispatched at the beginning of each time slot, for each request (Line 3) LGA goes through all the computing nodes (Line 4) and selects the best choice. Thus, the complexity of LGA is of the order $|K^t|\times(N+1)$, where $(N+1)$ indicates the number of computing nodes and $|K^t|$ denotes the number of requests arrived during time slot t into the system. Therefore, LGA is greedy in nature with linear computational complexity (with regard to $|K^t|$ and N) and can efficiently work in practice.

Optimality Analysis: Although LGA is sub-optimum, but it guarantees a bound on the achievable performance. Specifically, let $\overline{e^*(t)}$ denote the time-averaged total energy consumption achieved by LGA, and e^{opt} be the optimal value of the total energy consumption (i.e., the exact solution to the problem **P1**), then, Theorem 1 shows that the gap between $\overline{e^*(t)}$ and e^{opt} closes as the scaling factor V grows.

Theorem 1: The achievable value of energy consumption by LGA, $\overline{e^*(t)}$, is within $\frac{\Upsilon}{V}$ of e^{opt}, i.e,

$$\overline{e^*(t)}\leq e^{opt}+\frac{\Upsilon}{V}$$

where Υ is a constant.

Proof: Please refer to Appendix B in the supplementary material. □

Remark 1: Theorem 1 demonstrates that under LGA, $\overline{e^*(t)}$ is inversely proportional to V (with sufficient large value of V, it can asymptotically converge to e^{opt}). Also, it can be shown that the queue backlogs are linear to V, as revealed in Theorem 2.

Theorem 2: Under LGA, for the average overall queue backlogs, $\overline{Q(t)}$, the following holds:

$$\overline{Q(t)}\leq\frac{V[e^{opt}-\overline{e^*(t)}]}{\varepsilon}+\frac{\Upsilon}{\varepsilon}$$

where ε is a constant.

Proof: Please refer to Appendix B in the supplementary material. □

Remark 2: Theorems 1 and 2 together explain an $[O(\frac{1}{V}),O(V)]$ tradeoff between energy consumption and the overall queue backlogs.

6 Performance Evaluation

The performance of the proposed method is evaluated through simulation and practical experiments. We have performed extensive simulations using a custom-designed simulator in Matlab. In this section, first, we describe the simulation setup. Then, we present the results for the fixed-parameters simulation and sensitivity and scalability analysis. At the end of the section, we describe our implementation of a prototype in a real-world environment and present the results for the fixed-parameters experiments.

6.1 Simulation

6.1.1 Simulation Setup

Similar to the scenarios in [5], [20], [38], we have built a fog environment consisting of 3 FNs, a cloud server at a remote data center and 20 IoT devices. The IoT devices

are connected to FNs based on a random topology. The links between IoT devices and FNs in the random-generated topology are characterized by their transmission rates uniformly generated by $U[2, 10]$Mbps, which may vary from a time slot to the next one. However, the transmission rates are assumed to be fixed during each time slot. We describe each FN by its computing capability in the unit of "Million Instructions Per Second" (MIPS) randomly generated using the uniform distribution over [1500 2700] interval. Also, the cloud server is described by its computing capability, which is assumed to be 3600 MIPS. The transmission rate to the cloud is assumed to be fixed and equal to 2 Mbps. The IoT devices generate requests following a Poisson process with different average rates during the day (λ_d) and night (λ_n).

It is assumed that the computing and communication demands of requests are exponential random variables with the average rates γ_1 and γ_2, respectively. Table 3 summarizes the system model's parameters and those required in the proposed algorithm.

TABLE 3: CONFIGURATION OF SYSTEM MODEL PARAMETERS

λ_d	λ_n	γ_1 (MI)	γ_2 (KB)	V	C_D	C_γ
0.2	0.1	0.5	0.5	10^7	1	5×10^5

We run the simulation for 1000 time slots. The results are reported in the form of an average of 100 experiments (iterations).

We compare our proposed method to the state-of-the-art method in [16], called "Adaptive Fog Configuration (AFC)". AFC introduces a strategy for the placement of IoT devices' requests on FNs or the cloud. AFC similarly takes the service time, which consists of computation and communication delay, as the primary objective. To jointly minimize the averaged service time and satisfy the long-term energy budget constraint, it adopts the Lyapunov optimization framework. AFC introduces an online algorithm by converting the formulated problem in the long term presentation form into a sequence of per-slot sub-problems. Furthermore, to highlight the performance gain of the proposed method, we also compare our results against two baseline approaches, "Rnd" and "SJQ". Where Rnd uniformly dispatches the requests among the available computing nodes, and SJQ dispatches the request to the computing node with the shortest job queue.

The comparison is made between different methods in terms of the average service time, average number of deadline misses, average energy consumption, and average queue backlog.

6.1.2 Fixed-parameters Simulation

In this section, based on the configuration specified in Table 3, we provide an overall comparison between the proposed method, AFC, and the baseline approaches.

Fig. 3 shows the average service time (left-hand bars) and the average number of deadline misses (right-hand bars) for various methods. As it is observed from Fig. 3, LGA outperforms AFC, SJQ and Rnd up to 5%, 11% and 17% in terms of average service time, and up to 22%, 64% and 73% in terms of the average number of deadline misses, respectively.

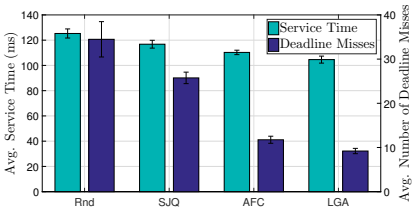


Fig. 3: Average Service Time and Average Number of Deadline Misses for various methods.

Furthermore, as observed in Fig. 4, LGA shows lower energy consumption than other methods. In particular, LGA on average shows 10% lower energy consumption than AFC.

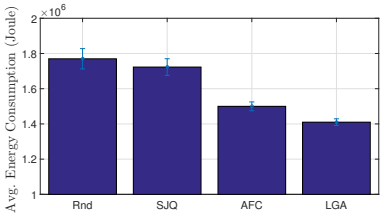


Fig. 4: Average Energy Consumption under each of the selected request dispatching methods.

Fig. 5 depicts the average queue backlog of the three FNs (the queue backlog for the cloud is nil). It can be observed that LGA and SJQ show a stable behavior (with the total queue backlog of 90 and 150 MI, respectively) while the queue backlog in AFC and Rnd method grow dramatically (with the total queue backlog more than 1000 and 5000 MI, respectively) which can lead to an unstable situation. Instability traces in Rnd is not out of expectation as it dispatches requests with the same probability among all the available computing nodes while their processing power is not the same. Furthermore, AFC takes the service time as the primary objective into its consideration and does not involve the queue backlog in its formulation. Therefore in this method, the requests are dispatched wherever they can be served faster. Consequently, it can lead to a situation where some FNs encounter large queue backlog and others encounter small or zero queue backlog. On the other hand, although LGA takes the service time into its consideration, it also involves the queue backlog in its formulation. Therefore, the requests are dispatched more evenly among the FNs. But yet, because of the deadline constraint, the requests can not be dispatched to some FNs. Thus, the queue backlog of some FNs is very small or zero.

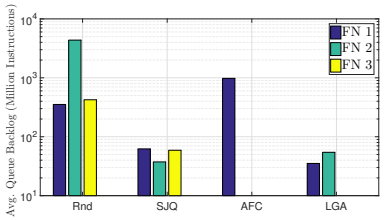
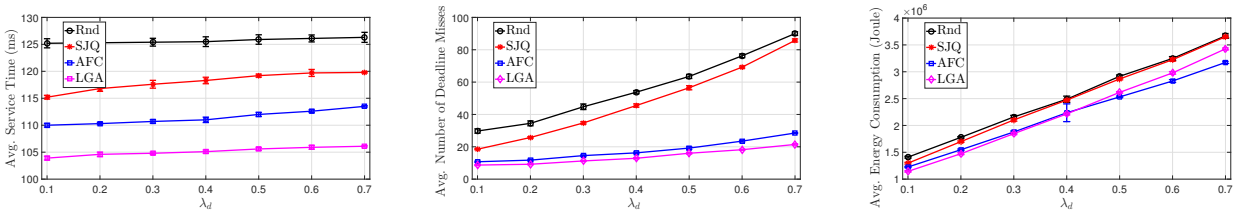
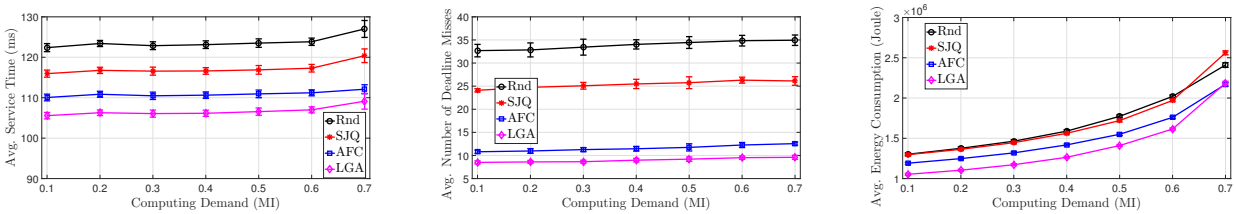


Fig. 5: Average Queue Backlog for each fog node under each of the selected request dispatching methods.



(a) Average Service Time (b) Average Number of Deadline Misses (c) Average Energy Consumption

Fig. 6: Behavior of the proposed online algorithm (LGA) in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Energy Consumption with respect to Arrival Rate (λ_d).



(a) Average Service Time (b) Average Number of Deadline Misses (c) Average Energy Consumption

Fig. 7: Behavior of the proposed online algorithm (LGA) in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Energy Consumption with respect to Computing Demand ($1 - \gamma_1$).

6.1.3 Sensitivity Analysis

This section provides further analysis on how the system parameters can affect the performance and algorithm effectiveness. Any changes in the request arrival rate, computing demand and the amount of data needed to be transmitted may affect the effectiveness of the proposed algorithm. We make changes in these parameters to perform sensitivity analysis and determine how different algorithms react to the changes in the system model parameters.

In this series of experiments, considering the system configuration described in Table 3, we change one of the parameters at each experiment while keeping others intact.

A. Arrival Rate: By increasing the average request arrival rate (λ_d), more workloads are injected into the system. Thus, the computing nodes have more requests to serve, which leads to a growth in the queue backlog and, consequently, the service time and the number of deadline misses as evidenced by Fig. 6(a) and Fig. 6(b), respectively. Furthermore, any increase in the number of requests leads to more energy consumption, as illustrated in Fig. 6(c).

Furthermore, LGA is less sensitive to the request arrival rate in terms of the number of deadline misses. In particular, LGA shows 59.45% growth in the number of deadline misses while its competitive, AFC, shows 62.3% growth by increasing the average arrival rate from $\lambda_d = 0.1$ to $\lambda_d = 0.7$.

B. Computing Demand: As the computing demands of the requests are increased, it takes longer to serve the requests as seen in Fig. 7(a). Consequently, the number of deadline misses and energy consumption increase as observed from Fig. 7(b) and Fig. 7(c), respectively. Also, Fig. 7 shows the superiority of the proposed method over other methods under different computation demands.

Moreover, LGA shows less sensitivity to computing demands of the requests than AFC when we change γ_1 from 0.1 to 0.7. In particular, LGA shows 21.5%, and AFC shows 24.1% growth in the number of deadline misses.

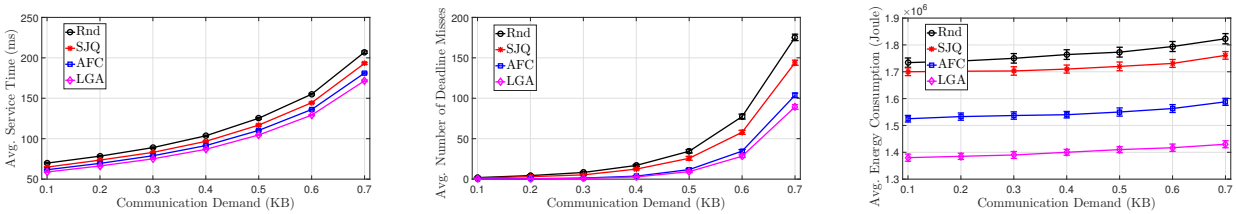
C. Communication Demand: Based on the definition of communication delay, in (3), and service time, in section 4, any increment in the volume of data communication leads to longer service time, as it is observed from Fig. 8(a). Consequently, we see the number of deadline misses also increases, Fig. 8(b). On the other hand, based on the communication models in (3) and (5), more communication demand results in more energy consumption as evidenced by Fig. 8(c). Therefore, overall we see LGA outperforms other methods under different communication demands and shows less sensitivity to the volume of data communication. In particular, by changing γ_2 from 0.3 to 0.7, LGA shows 96% growth while AFC shows 98% growth in the number of deadline misses.

The relation between system timeliness and service time can also be investigated through Fig. 6 to Fig. 8. For example, in Fig. 8, although service time is growing from 58 to 75 by increasing the communication demand from $\gamma_2 = 0.1$ to $\gamma_2 = 0.3$, but the number of deadline misses does not change in this interval. It is justified based on the notion of deadline because a service time up to $\gamma_2 = 0.3$ is lower than the deadline threshold, and its changes have no impact on the deadline misses and consequently system timeliness.

Remark 3: Service time of a request can only affect the timeliness if it exceeds the deadline threshold. If service time is lower than the threshold, further reduction in the service time has no impact on the timelines.

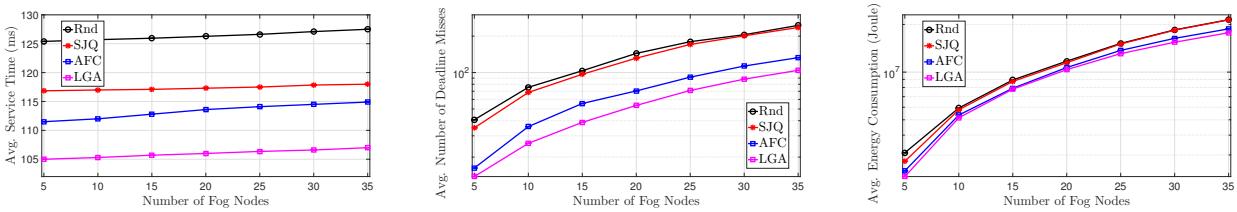
6.1.4 Scalability Analysis

In this section, we provide further simulations to evaluate the scalability of the proposed method. Considering the aforementioned fixed-parameters simulation setup, we design a scenario in which the system is scaled at each step by increasing the number of FNs and IoT devices. We start with 5 FNs and 33 IoT devices and scale the system by adding five more FNs and relativity scale IoT devices



(a) Average Service Time (b) Average Number of Deadline Misses (c) Average Energy Consumption

Fig. 8: Behavior of the proposed online algorithm (LGA) in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Energy Consumption with respect to Communication Demand ($1 - \gamma_2$).



(a) Average Service Time (b) Average Number of Deadline Misses (c) Average Energy Consumption

Fig. 9: Behavior of the proposed online algorithm (LGA) in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Energy Consumption as the system scaled with more fog nodes and IoT devices.

at each step (for example, 10 FNs and 66 IoT devices or 20 FNs and 132 IoT devices). The simulation is performed for 1000 time slots. The experiment is repeated 100 times and averaged results are reported. Fig. 9(a) to Fig. 9(c) show the simulation results for average service time, the average number of deadline misses and average energy consumption, respectively. As it is observed from Fig. 9(a), the average service time slightly increases as the system scales up, for example, from 125 ms with 5 FNs and 33 IoT devices to 127 ms with 35 FNs and 231 IoT devices. Also, the average number of deadline misses increases as we scale the system, as it is observed from Fig. 9(b), but the ratio of deadline misses to the injected requests do not change considerably. For example, it changes from 2.76 with 5 FNs to 2.98 with 35 FNs. The increases in the average service time and the average number of deadline misses happen because as the system is scaled, we have more FNs in the system, which results in the requests to be dispatched to FNs that are farther from the device that originates the request as the best choice at the time. On the other hand, with an increase in the number of FNs, more energy will be consumed, as it is observed from Fig. 9(c). But overall, these simulations prove the superiority of the proposed method over the baseline schemes under different scales of the system.

Remark 4: In practice, the FNs are divided into some domains and sections. There may exist hundreds or thousands of such domains in a clustered way (in each cluster, a group of domains operate under the control of a dispatching entity). Therefore, the dispatching algorithm separately runs on each controller.

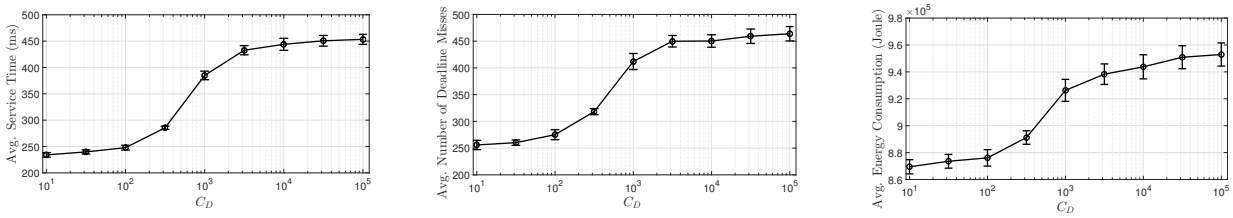
6.1.5 Model Parameters Analysis

This section is devoted to the analysis of system model parameters (i.e., C_D, C_L and C_γ). Regarding that C_L is a function of C_γ, we provide analysis for one of them (just for C_γ). Based on the aforementioned setup, we conduct some experiments to evaluate the effect of changes in C_D and C_γ on the system behavior. In order to better reflect the effect of these parameters, we design a scenario in which the requests arriving into the system are network intensive (i.e. $\gamma_2 = 0.9$).

C_D provides control on the throttling threshold for the permissible number of deadline misses. Thus, changing the value of C_D can affect the system behavior. Fig. 10(a) to Fig. 10(c) show the simulation results in terms of average service time, average number of deadline misses and average energy consumption, respectively. For small values of C_D, more requests are dispatched to the nearer nodes, which leads to a lower service time, and a lower number of deadline misses as it is observed from Fig. 10(a) and Fig. 10(b), respectively. On the other hand, specific to our case that the requests are network intensive, dispatching the requests to nearer nodes also leads to lower energy consumption, as it is understood from Fig. 10(c).

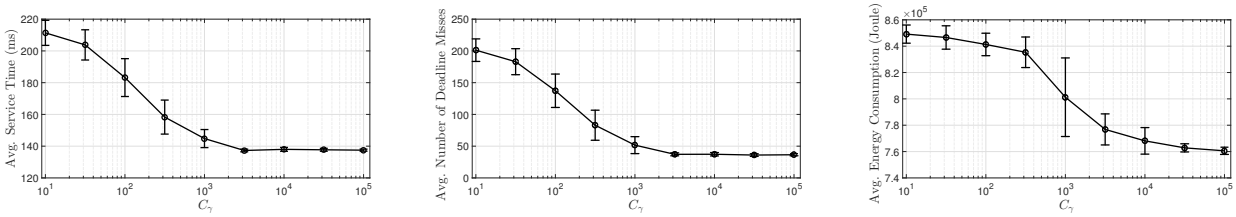
As C_γ is the threshold for the proportionality of requests to be sent to the cloud, changing the value of C_γ can affect dispatching the requests. Fig. 11(a) to Fig. 11(c) show the simulation results for the average service time, the average number of deadline misses and the average energy consumption, respectively. A small value of C_γ means that the requests which do not have considerable computing demand are dispatched to the cloud. Therefore, the service time and the number of deadline misses increase, as it is observed from Fig. 11(a) and Fig. 11(b), respectively. Dispatching the requests to the cloud also leads to a greater amount of energy consumption, as it is understood from Fig. 11(c).

Finally, it is notable that in practice in real environment, tuning the model parameters are performed based on the management policy of the system. For example, tuning C_D is performed based on the percentage of permissible deadline misses in the system.



(a) Average Service Time (b) Average Number of Deadline Misses (c) Average Energy Consumption

Fig. 10: Behavior of the proposed online algorithm (LGA) in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Energy Consumption with respect to model parameter C_D.



(a) Average Service Time (b) Average Number of Deadline Misses (c) Average Energy Consumption

Fig. 11: Behavior of the proposed online algorithm (LGA) in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Energy Consumption with respect to model parameter C_γ.

6.2 Prototyping Platform

To further validate the effectiveness of the proposed method, we constructed a real testbed prototype. Fig. 12 illustrates the testbed and its hardware configuration. We considered a scenario with three FNs (a group of Raspberry Pis) and a cloud server (a desktop computer). The detailed specifications of the computing nodes are listed in Table 4. The computing nodes are connected with different network delays to a desktop computer (PC) which performs the responsibilities and activities of the controller. A pool of tasks (each in the form of a hash problem) is prepared on the controller. Based on a Poisson process, the tasks are injected into the system ($\lambda_d = 0.5$). Furthermore, we leveraged the "tc", a Linux utility program, to configure the kernel packet scheduler and emulate the delay in the link between the controller (the edge) and the cloud. An accurate usb powermeter[3] is used to measure the energy consumption of the FNs. The model parameters are set based on Table 3. The source code of our implementation is available on GitHub[4].

Fig. 12: An illustration of the real test-bed prototype.

We conducted practical experiments on the prepared testbed to compare the performance of our proposed method, LGA, with baseline approaches. For each experiment, the performance of different methods was logged for 1000 tasks, and the results are reported in the form of an average of 10 repetitions. Fig. 13 shows the average service time (left-hand bars), and the average number of deadline misses (right-hand bars) for each method. As it can be observed, LGA outperforms the other methods. In particular, LGA surpasses AFC by 23% and 58% in terms of the average service time and the average number of deadline misses, respectively.

3. UT658A USB Tester,https://www.uni-trend.com/html/product/tyyq/Borescope/UT658/UT658A.html
4. https://github.com/fgcmp-lab/lga

TABLE 4: THE SPECIFICATIONS OF THE REAL EXPERIMENTAL SETUP

	FN 1	FN 2	FN 3	Cloud
Model	Raspberry Pi2	Raspberry Pi2	Raspberry Pi3	PC
CPU	Cortex-A7	Cortex-A7	Cortex-A53	Intel Core i7
Clock	900MHz	900MHz	1.2GHz	3.6GHz
RAM	1GB	1GB	1GB	8GB
Delay	0.82ms	0.88ms	0.34ms	320ms

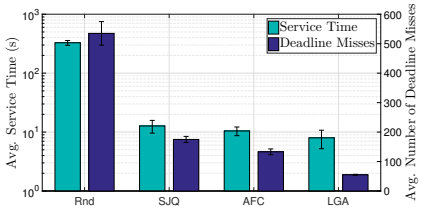


Fig. 13: The Average Service Time and the Average Number of Deadline Misses for various methods in practical experiments.

Furthermore, as observed in Fig. 14, LGA demonstrates a lower average total energy consumption (across different computing nodes) than the other methods. Specifically, in contrast to AFC, LGA consumes 5% less energy.

Finally, Fig. 15, depicts the average queue backlog for the FNs. It is observed that the queue backlog grows dramatically when the Rnd method is used. On the other hand, in SJQ, the tasks are evenly distributed among the FNs.

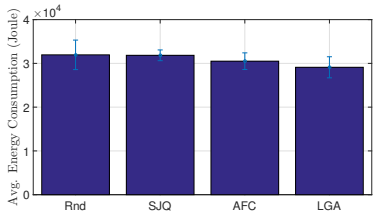


Fig. 14: Average Energy Consumption under each of the selected request dispatching methods in practical experiments.

Moreover, for the case of AFC, the queue backlogs for FN1 and FN2 have almost the same sizes as those for the SJQ. But AFC manages to lower the backlog of the FN3 down to a level that is not observable in Fig 15. Moving further in this direction, LGA successfully makes the backlogs of two FNs (i.e., FN2 and FN3) insignificant and outperforms all the other approaches in terms of the queue backlogs.

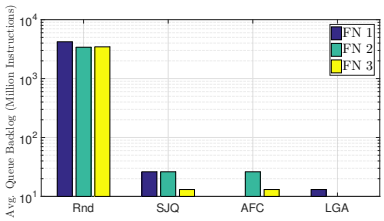


Fig. 15: Average Queue Backlog for each fog node under each of the selected request dispatching methods in practical experiments.

6.3 Discussions

To evaluate the performance of the proposed method, we conducted both simulation and practical experiments. The simulation environment preliminary is set up to provide more efficient control on the system parameters. This enables us to perform sensitivity, scalability and model parameters analysis. On the other hand, the prototyping platform provides a sense of how our proposed method would act in a real environment. Considering the high level of abstraction in the simulated environment, on one hand, and the real application tasks in the prototype, on the other hand, we witness some degree of discrepancies between the simulation and practical results. However, the results are consistent when we study the relative performance of different methods. In particular,

- Both simulation and practical results agree on the supremacy of the LGA method.
- Our findings from the practical experiments are aligned with what simulation predicts.

7 Conclusions and Future Work

Fog enabled IoT is a three-layer structured platform that aims to provide a mechanism to facilitate time-sensitive IoT applications. However, to be fully effective, this mechanism needs to be coupled with intelligent strategies (policies) that determine how the system resources should be used. In this paper, we focused on request dispatching strategy among the fog nodes and the remote cloud in the context of industrial IoT (IIoT) applications. We identified timeliness, stability, and energy consumption as important measures to be carefully monitored by any dispatching method in the IIoT context. We first formulated the problem as a stochastic non-linear optimization that minimizes energy subject to timeliness and stability constraints. Then, we leveraged Lyapunov Optimization Technique (LOT) to derive an efficient greedy algorithm, *LGA*, to solve the dispatching optimization problem. We proved that LGA drives the system such that timeliness and stability constraints are satisfied. Moreover, we conducted extensive numerical experiments in both simulation and the real-world testbed to showcase the performance of our proposed algorithm against baseline and alternative methods. Numerical results verified that LGA outperforms others in the average service time, the number of deadline misses, and energy consumption. In this paper, we focused on the IIoT application. Considering other IoT usage scenarios, such as intelligent transportation, smart city, and smart grids, with their own specific quality of service demands and conditions, draws an exciting road map for future work.

References

[1] M. Ali, N. Riaz, M. I. Ashraf, S. Qaisar, and M. Naeem, "Joint cloudlet selection and latency minimization in fog networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4055–4063, 2018.

[2] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis, "A cooperative fog approach for effective workload balancing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 36–45, 2017.

[3] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2015.

[4] S. El Kafhali and K. Salah, "Efficient and dynamic scaling of fog nodes for iot devices," *The Journal of Supercomputing*, vol. 73, no. 12, pp. 5261–5284, 2017.

[5] C. Shi, Z. Ren, K. Yang, C. Chen, H. Zhang, Y. Xiao, and X. Hou, "Ultra-low latency cloud-fog computing for industrial internet of things," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[6] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018.

[7] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016.

[8] A. Chebaane, A. Khelil, and N. Suri, "Time-critical fog computing for vehicular networks," *Fog Computing: Theory and Practice*, pp. 431–458, 2020.

[9] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, "Delay-sensitive task offloading in the 802.11 p-based vehicular fog computing systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 773–785, 2019.

[10] T. Lins and R. A. R. Oliveira, "Cyber-physical production systems retrofitting in context of industry 4.0," *Computers & industrial engineering*, vol. 139, p. 106193, 2020.

[11] C. Tang, M. Hao, X. Wei, and W. Chen, "Energy-aware task scheduling in mobile cloud computing," *Distributed and Parallel Databases*, vol. 36, no. 3, pp. 529–553, 2018.

[12] M. Gorlatova, H. Inaltekin, and M. Chiang, "Characterizing task completion latencies in fog computing," *arXiv preprint arXiv:1811.02638*, 2018.

[13] Z. Zhou, P. Liu, Z. Chang, C. Xu, and Y. Zhang, "Energy-efficient workload offloading and power control in vehicular edge computing," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2018, pp. 191–196.

[14] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[15] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "Debts: Delay energy balanced task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2094–2106, 2018.
[16] L. Chen, P. Zhou, L. Gao, and J. Xu, "Adaptive fog configuration for the industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4656–4664, 2018.
[17] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, 2019.
[18] 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 Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2017.
[19] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed petri nets," *ieee internet of things journal*, vol. 4, no. 5, pp. 1216–1228, 2017.
[20] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.
[21] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21 355–21 367, 2017.
[22] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2017.
[23] F. Wang, J. Xu, and Z. Ding, "Multi-antenna noma for computation offloading in multiuser mobile edge computing systems," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 2450–2463, 2018.
[24] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 1, pp. 108–119, 2015.
[25] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized iot service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427–443, 2017.
[26] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier iot fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, 2017.
[27] 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 Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4590–4602, 2018.
[28] S. K. Mishra, D. Puthal, J. J. Rodrigues, B. Sahoo, and E. Dutkiewicz, "Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4497–4506, 2018.
[29] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart resources partitioning for fog computing-enabled industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4702–4711, 2018.
[30] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 976–986, 2018.
[31] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.
[32] S. Zhao, Y. Yang, Z. Shao, X. Yang, H. Qian, and C.-X. Wang, "Femos: Fog-enabled multitier operations scheduling in dynamic wireless networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1169–1183, 2018.
[33] A. Karimiafshar, M. R. Hashemi, M. R. Heidarpour, and A. N. Toosi, "Effective utilization of renewable energy sources in fog computing environment via frequency and modulation level scaling," *IEEE Internet of Things Journal*, 2020.
with hybrid energy harvesting," *Sensors*, vol. 18, no. 9, p. 3140, 2018.
[34] H. Zhang, Z. Chen, J. Wu, Y. Deng, Y. Xiao, K. Liu, and M. Li, "Energy-efficient online resource management and allocation optimization in multi-user multi-task mobile-edge computing systems
[35] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Distributed online optimization of fog computing for selfish devices with out-of-date information," *IEEE Transactions on Wireless Communications*, vol. 17, no. 11, pp. 7704–7717, 2018.
[36] W. Li, I. Santos, F. C. Delicato, P. F. Pires, L. Pirmez, W. Wei, H. Song, A. Zomaya, and S. Khan, "System modelling and performance evaluation of a three-tier cloud of things," *Future Generation Computer Systems*, vol. 70, pp. 104–125, 2017.
[37] B. Zhang, R. Simon, and H. Aydin, "Energy management for time-critical energy harvesting wireless sensor networks," in *Symposium on Self-Stabilizing Systems*. Springer, 2010, pp. 236–251.
[38] P. G. V. Naranjo, E. Baccarelli, and M. Scarpiniti, "Design and energy-efficient resource management of virtualized networked fog architectures for the real-time support of iot applications," *The Journal of Supercomputing*, vol. 74, no. 6, pp. 2470–2507, 2018.

Aref Karimiafshar received M.Sc. and PhD degrees in computer engineering from Isfahan University of Technology (IUT), Iran, in 2013 and 2020, respectively. He is currently a Postdoctoral Research Fellow in the Department of Electrical and Computer Engineering at IUT. His current research interests include operating systems, cloud computing, edge computing, and Internet of Things. Currently, he is working on resource management in cloud/edge computing and integrating renewable energy.

Massoud Reza Hashemi received a PhD degree in electrical and computer engineering from the University of Toronto, Canada, in 1998. From 1998 to 1999, he was a Postdoctoral Fellow with the University of Toronto. He was a founding member and the Lead Systems Architect with AcceLight Networks in 1999, where he developed some of the key system elements of a multi-terabit multiservice core switch. Since 2003, he has been with the Isfahan University of Technology, where he is currently an Associate Professor. As the head of the university IT center from 2005 to 2013, he restructured and consolidated the foundations of IT in the campus, the university campus network, and the university data center. From 2016 to 2017, he was a Visiting Scholar with the University of Toronto on sabbatical leave. His current research interests include Software-Defined Networks, Cyber-physical Systems, IoT and Fog Computing.

Mohammad Reza Heidarpour received B.Sc. and M.Sc. degrees in electronics and communication engineering from Isfahan University of Technology (IUT), Iran, in 2006 and 2008, respectively, and the PhD degree in electrical engineering from the University of Waterloo (UW), Ontario, Canada, in 2013. He has done postdoctoral research at the Coding and Signal Transmission (CST) laboratory in UW from May 2013 to November 2015 and at IUT from December 2016 to October 2017. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at IUT. Dr Heidarpour's research interests include broad areas of Wireless Communication, Data Networks, and Network Algorithms.

Adel N. Toosi is a lecturer at the Department of Software Systems and Cybersecurity, Faculty of Information Technology, Monash University, Australia. Before joining Monash, Dr Toosi was a Postdoctoral Research Fellow at the University of Melbourne from 2015 to 2018. He received his PhD degree in 2015 from the School of Computing and Information Systems at the University of Melbourne. Dr Toosi's research interests include scheduling and resource provisioning in Cloud/Fog/Edge Computing environments, Internet of Things, Software-Defined Networking, Green Computing and Energy Efficiency. Currently, he is working on building sustainable Edge/Fog computing environments. For further information, please visit his homepage: http://adelnadjarantoosi.info.

Supplementary Material for An Energy-Conservative Dispatcher for Fog-Enabled IIoT Systems: When Stability and Timeliness Matter

Aref Karimiafshar, Massoud Reza Hashemi, Mohammad Reza Heidarpour, and Adel N. Toosi, *Member, IEEE*

APPENDIX A
PROOF OF INEQUALITY (18):

Proof: Using $\max[a-b,0]^2 \leq (a-b)^2$, from (1), we can write

$$\frac{Q_i(t+1)^2 - Q_i(t)^2}{2} \leq \frac{B_i(t)^2 + R_i(t)^2}{2} - \tilde{B}_i(t)R_i(t) - Q_i(t)[B_i(t) - R_i(t)], \quad (1)$$

where $\tilde{B}_i(t) \triangleq \min[Q_i(t), B_i(t)]$.

Similarly, from the updating rule of the virtual queues $Z(t)$, $\boldsymbol{H}(t)$ and $G(t)$ we can write

$$\frac{Z(t+1)^2 - Z(t)^2}{2} \leq \frac{y(t)^2}{2} + Z(t)y(t), \quad (2)$$

$$\frac{H_i(t+1)^2 - H_i(t)^2}{2} \leq \frac{g_i(t)^2}{2} + H_i(t)g_i(t), \quad (3)$$

and

$$\frac{G(t+1)^2 - G(t)^2}{2} \leq \frac{f(t)^2}{2} + G(t)f(t). \quad (4)$$

Using the upper-bounds (1) to (4), we have an upper-bound for $\Delta(L_{\boldsymbol{\theta}}(t))$, given by

$$\begin{aligned}\Delta(L_{\boldsymbol{\theta}}(t)) \leq \Upsilon + \sum_{i=1}^{N+1} Q_i(t)\mathbb{E}\{R_i(t)|\boldsymbol{\theta}(t)\} - \sum_{i=1}^{N+1} Q_i(t)B_i(t) \\ + Z(t)\mathbb{E}\{y(t)|\boldsymbol{\theta}(t)\} + \sum_{i=1}^{N+1} H_i(t)\mathbb{E}\{g_i(t)|\boldsymbol{\theta}(t)\} \\ + G(t)\mathbb{E}\{f(t)|\boldsymbol{\theta}(t)\},\end{aligned} \quad (5)$$

where we have maintained significant terms, i.e., those containing queues or virtual queues (and thus can be very large), and replaced all the other by a constant finite upper-bound Υ (which does exist as we assume system parameters, such as request arrival rate and processing capacity of FNs, are all bounded parameters).

Finally, adding $V\mathbb{E}\{e(t)|\boldsymbol{\theta}(t)\}$ to both sides proves the results. □

APPENDIX B
PROOF OF THEOREMS 1 AND 2

Proof: Assuming the original problem **P1** is feasible, there exists a decision that is independent of the queue backlogs, which based on (18) and doing some manipulations, yields (ε is a non-negative value) [1]:

$$\Delta(L_{\boldsymbol{\theta}}(t)) + V\mathbb{E}\{e^*(t)|\boldsymbol{\theta}(t) \leq \Upsilon + Ve^{opt} - \varepsilon\overline{Q}(t), \quad (6)$$

where $\overline{Q}(t) \triangleq \frac{1}{2N+4}(\sum_{i=1}^{N+1} Q_i(t) + Z(t) + \sum_{i=1}^{N+1} H_i(t) + G(t))$, representing overall backlogs of the queues in the system.

Taking expectation of both sides of (6), yields:

$$\begin{aligned}\mathbb{E}\{L(\boldsymbol{\theta}(t+1))\} - \mathbb{E}\{L(\boldsymbol{\theta}(t))\} + V\mathbb{E}\{e^*(t)\} \\ \leq \Upsilon + Ve^{opt} - \varepsilon\mathbb{E}\{\overline{Q}(t)\},\end{aligned} \quad (7)$$

by summing (7) over $t \in \{0,1,2,...,T-1\}$, we have

$$\begin{aligned}\mathbb{E}\{L(\boldsymbol{\theta}(T))\} - \mathbb{E}\{L(\boldsymbol{\theta}(0))\} + V\sum_{t=0}^{T-1}\mathbb{E}\{e^*(t)\} \\ \leq \Upsilon + Ve^{opt} - \varepsilon\sum_{t=0}^{T-1}\mathbb{E}\{\overline{Q}(t)\},\end{aligned} \quad (8)$$

By rearranging the terms in (8), dividing by VT and taking $\limsup$ when $T \to \infty$ yields:

$$\overline{e^*(t)} \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{e^*(t)\} \leq e^{opt} + \frac{\Upsilon}{V}, \quad (9)$$

Theorem 1 is proved. □

by rearranging the terms in (8), dividing by εT and taking $\limsup$ when $T \to \infty$ yields:

$$\overline{Q}(t) \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{\overline{Q}(t)\} \leq \frac{V[e^{opt} - \overline{e^*(t)}]}{\varepsilon} + \frac{\Upsilon}{\varepsilon}, \quad (10)$$

Theorem 2 is proved. □

REFERENCES

[1] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

Supplementary Material for An Energy-Conservative Dispatcher for Fog-Enabled IIoT Systems: When Stability and Timeliness Matter

Aref Karimiafshar, Massoud Reza Hashemi, Mohammad Reza Heidarpour, and Adel N. Toosi, *Member, IEEE*

APPENDIX A
PROOF OF INEQUALITY (18):

Proof: Using $\max[a-b,0]^2 \leq (a-b)^2$, from (1), we can write

$$\frac{Q_i(t+1)^2 - Q_i(t)^2}{2} \leq \frac{B_i(t)^2 + R_i(t)^2}{2} - \tilde{B}_i(t)R_i(t) - Q_i(t)[B_i(t) - R_i(t)], \quad (1)$$

where $\tilde{B}_i(t) \triangleq \min[Q_i(t), B_i(t)]$.

Similarly, from the updating rule of the virtual queues $Z(t)$, $\boldsymbol{H}(t)$ and $G(t)$ we can write

$$\frac{Z(t+1)^2 - Z(t)^2}{2} \leq \frac{y(t)^2}{2} + Z(t)y(t), \quad (2)$$

$$\frac{H_i(t+1)^2 - H_i(t)^2}{2} \leq \frac{g_i(t)^2}{2} + H_i(t)g_i(t), \quad (3)$$

and

$$\frac{G(t+1)^2 - G(t)^2}{2} \leq \frac{f(t)^2}{2} + G(t)f(t). \quad (4)$$

Using the upper-bounds (1) to (4), we have an upper-bound for $\Delta(L_{\boldsymbol{\theta}}(t))$, given by

$$\begin{aligned}\Delta(L_{\boldsymbol{\theta}}(t)) \leq \Upsilon &+ \sum_{i=1}^{N+1} Q_i(t)\mathbb{E}\{R_i(t)|\boldsymbol{\theta}(t)\} - \sum_{i=1}^{N+1} Q_i(t)B_i(t) \\ &+ Z(t)\mathbb{E}\{y(t)|\boldsymbol{\theta}(t)\} + \sum_{i=1}^{N+1} H_i(t)\mathbb{E}\{g_i(t)|\boldsymbol{\theta}(t)\} \\ &+ G(t)\mathbb{E}\{f(t)|\boldsymbol{\theta}(t)\},\end{aligned} \quad (5)$$

where we have maintained significant terms, i.e., those containing queues or virtual queues (and thus can be very large), and replaced all the other by a constant finite upper-bound Υ (which does exist as we assume system parameters, such as request arrival rate and processing capacity of FNs, are all bounded parameters).

Finally, adding $V\mathbb{E}\{e(t)|\boldsymbol{\theta}(t)\}$ to both sides proves the results. □

APPENDIX B
PROOF OF THEOREMS 1 AND 2

Proof: Assuming the original problem **P1** is feasible, there exists a decision that is independent of the queue backlogs, which based on (18) and doing some manipulations, yields (ε is a non-negative value) [1]:

$$\Delta(L_{\boldsymbol{\theta}}(t)) + V\mathbb{E}\{e^*(t)|\boldsymbol{\theta}(t) \leq \Upsilon + Ve^{opt} - \varepsilon\overline{Q}(t), \quad (6)$$

where $\overline{Q}(t) \triangleq \frac{1}{2N+4}(\sum_{i=1}^{N+1} Q_i(t) + Z(t) + \sum_{i=1}^{N+1} H_i(t) + G(t))$, representing overall backlogs of the queues in the system.

Taking expectation of both sides of (6), yields:

$$\begin{aligned}&\mathbb{E}\{L(\boldsymbol{\theta}(t+1))\} - \mathbb{E}\{L(\boldsymbol{\theta}(t))\} + V\mathbb{E}\{e^*(t)\} \\ &\leq \Upsilon + Ve^{opt} - \varepsilon\mathbb{E}\{\overline{Q}(t)\},\end{aligned} \quad (7)$$

by summing (7) over $t \in \{0,1,2,...,T-1\}$, we have

$$\begin{aligned}&\mathbb{E}\{L(\boldsymbol{\theta}(T))\} - \mathbb{E}\{L(\boldsymbol{\theta}(0))\} + V\sum_{t=0}^{T-1}\mathbb{E}\{e^*(t)\} \\ &\leq \Upsilon + Ve^{opt} - \varepsilon\sum_{t=0}^{T-1}\mathbb{E}\{\overline{Q}(t)\},\end{aligned} \quad (8)$$

By rearranging the terms in (8), dividing by VT and taking $\limsup$ when $T \to \infty$ yields:

$$\overline{e^*(t)} \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{e^*(t)\} \leq e^{opt} + \frac{\Upsilon}{V}, \quad (9)$$

Theorem 1 is proved. □

by rearranging the terms in (8), dividing by εT and taking $\limsup$ when $T \to \infty$ yields:

$$\overline{Q}(t) \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{\overline{Q}(t)\} \leq \frac{V[e^{opt} - \overline{e^*(t)}]}{\varepsilon} + \frac{\Upsilon}{\varepsilon}, \quad (10)$$

Theorem 2 is proved. □

REFERENCES

[1] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.