Available online at www.sciencedirect.com

ScienceDirect

Future Generation Computer Systems 00 (2020) 1–17

FGCS

A Request Dispatching Method for Efficient Use of Renewable Energy in Fog Computing Environments

Aref Karimiafshar[a], Massoud Reza Hashemi[a], Mohammad Reza Heidarpour[a], Adel N. Toosi[b,*]

[a]*Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 8415683111, Iran*
[b]*Department of Software Systems and Cybersecurity, Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia*

Abstract

In the emerging era of Internet of Things (IoT), fog computing plays a critical role in serving delay-sensitive and location-aware applications. As a result, fog nodes are envisioned to be heavily deployed and form future distributed data centers. Powering fog nodes with green energy sources (such as solar and wind), not only helps in environmental and CO_2 emission control but also paves the way towards a "sustainable IoT technology". However, the downside of green energy is its variation and unpredictability, which needs to be engineered. In this paper, we use the Lyapunov optimization technique to derive algorithms for dynamic dispatching of the users' requests among the nearby fog nodes and remote data centers. The proposed algorithms take into account the time constraints of the requests and maintain the system stability while efficiently utilizes the available green energy sources. Exhaustive simulation results, based on solar radiation data supplied by the Australian Bureau of Meteorology, confirm the efficiency of the proposed algorithms. In particular, in terms of service time, the number of deadline misses and green energy utilization, the proposed algorithms outperform the state-of-the-art alternative up to 6%, 17% and 12%, respectively.

Keywords: Fog Computing, Lyapunov Optimization Technique, Request Dispatching, Renewable Energy, Virtual Queue.

1. Introduction

Fog computing is a new paradigm which brings processing, storage and control to the edge of the network, close to the end devices. As a result, fog computing promotes bandwidth conservation, fast response time, and context-aware applications [1]. Potential benefits of fog computing can be fully harvested through proper resource management which addresses service placement and request dispatching. However, the heterogeneity of the computing nodes and variety in the requests and their requirements in terms of computation, communication and energy make the request dispatching challenging [2, 3].

The fog paradigm introduces a highly distributed platform with a large number of computing nodes. The energy consumption of fog nodes becomes challenging if the nodes are all powered by the centralized power grid [4]. Furthermore, due to the wide distribution of the fog nodes, it is not possible to implement the central, intelligent power management techniques deployed in cloud data centers. In addition, It is impossible to reduce the energy cost by placing the nodes where the cheaper energy is available, because they should be placed somewhere in the user premises. Although the fog paradigm implies a different condition compared with cloud from the energy management point of view, it also brings other new opportunities. Therefore, it requires some new methods of its own based on the fog properties [5, 6].

With respect to cloud, fog nodes consume less power and have smaller footprints [7], which lead them to have a better position in effective utilization of on-site cheap renewable energy (green energy) sources. Recently, renewable energy technology is growing very fast (increasing in performance and efficiency, and decreasing in cost). Therefore, it seems to be feasible and reasonable to utilize on-site renewable sources in fog computing environment [8]. However, the renewable energy is *unpredictable* and *intermittent*. For example, solar energy is available during the day and depends on weather conditions. Therefore, to mitigate the amount of produced energy, fog nodes can bank green energy in batteries or on the grid network (called

*Corresponding author
Email addresses: a.karimiafshar@ec.iut.ac.ir (Aref Karimiafshar), hashemim @cc.iut.ac.ir (Massoud Reza Hashemi), mrheidar@cc.iut.ac.ir (Mohammad Reza Heidarpour), adel.n.toosi@monash.edu (Adel N. Toosi)

net metering). However, both batteries and net metering accompany some problems [9]. In batteries, self-discharge and internal resistance cause energy losses, chemicals that are used are harmful to the environment, and the purchasing and maintaining (P&M) is costly (the cost of P&M can dominate in a solar system). On the other hand, net metering may not be available in every part of the world, and the voltage transformation while feeding the green energy into the power grid also leads to energy losses. Therefore, we focus on fog nodes powered with grid-tied solar systems having no energy storage. Fog nodes use the grid as a 'backup' source of energy when renewable energy is not sufficient.

In this paper, we investigate the problem of managing workload to shape the electricity demands to match the available green energy supply. We state the problem as designing a request dispatching controller. The controller dispatches requests among available computing nodes in fog or cloud tier to jointly minimize service time and maintain green energy utilization and system stability at a satisfactory level.

The controller's optimum decision making process is first formulated as a stochastic optimization problem. The green energy utilization and selection between fog and cloud are presented as the constraints beside the main objective (i.e., service time minimization). We use the idea of virtual queues [10] and turn the satisfaction of these constraints into a pure stability problem. Then Lyapunov Optimization Technique (LOT) is leveraged to develop effective online algorithms with guaranteed performance.

Our main contributions in this paper are summarized as follow:

- Utilizing the on-site renewable energy sources, we introduce a dynamic request dispatching strategy to lessen the burden on the power grid while minimizing the service time and stabilizing the queues.
- Based on the Lyapunov optimization technique and the concept of virtual queues, we introduce easy-to-implement online request dispatching algorithms.
- The proposed methods are evaluated by extensive simulations based on real solar irradiation data.
- Analysis is performed to assess the sensitivity of the proposed method to different conditions.

In the remainder of the paper, the related works are summarized in Section 2, the system model is described in Section 3, Section 4 is devoted to the problem statement. The basics of our methods is explained in Section 5. The proposed methods are introduced in Section 6. Section 7 presents the simulation results and Section 8 provides a further discussion. Finally, the paper is concluded in Section 9.

2. Related Works

In this section, some of the most relevant works are presented and discussed. We classify the prior research works into four subsections and separately discuss them.

2.1. Resource Management in Fog Computing Environment

Resource management and request dispatching in the context of fog computing have been widely studied in the literature [11, 12, 13]. For example, in [11], Ni *et al.* suggested a resource allocation strategy based on priced time Petri nets which considers the price and time cost of task completion. They also take into account the credibility score of both users and fog nodes. Also, Yousefpour *et al.* [12] proposed an offloading policy for minimizing service time. They employ inter fog communication to decrease service time through load sharing. However, these works are based on estimated waiting time, and therefore, the estimation precision may affect the performance of the methods.

Energy consumption is an important metric at the time of dispatching requests. As a result, there have been some studies on energy-efficient resource management and request dispatching [14, 15]. For example, in [14], Deng *et al.* suggested a workload allocation algorithm to minimize power consumption while taking into account constrained service time. Our work is different from these works in two aspects. First, we investigate the problem of request dispatching subject to system stability. Indeed, we jointly minimize service time and maintain green energy utilization and stability at a satisfactory level. Second, utilizing on-site renewable energy source, we investigate the problem of request dispatching to properly deal with intermittency and varying nature of renewable energy sources.

2.2. Renewable Energy in Fog Computing Environment

Renewable energy technology has been grown very fast in the last decade. It is now considered as a low cost, high-performance alternative for providing energy for fog nodes [5]. However, a carefully designed dispatching request strategy will be required to utilize the capacity of fluctuating renewable energy sources properly. Recent work suggested a framework for energy management which handles distributed renewable energy sources [4]. In order to fully utilize renewable energy and improve the Quality of Service (QoS) for constrained service time, the framework supports the cooperation of computing and energy sources. In a similar work, Li *et al.* [8] proposed an analytic model for offloading computation to edge or cloud resources. The model improves QoS while considering the availability of renewable energy sources. Although such works deal with better utilizing renewable energy sources, they do not take into consideration the system stability.

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

Reference	**Technology**	**Problem**	**Lyapunov**	**Objectives**		**Architecture**		**RNE**[5]
				S.T [1]	E.C [2]	Cntr[3]	Dstr[4]	
[4]	Edge Computing	Energy management			✓	✓		✓
[8]	Edge Computing	Computation offloading			✓		✓	✓
[11]	Fog Computing	Resource allocation		✓		✓		
[12]	Fog Computing	Computation offloading		✓			✓	
[14]	Fog Computing	Workload allocation		✓	✓	✓		
[16]	Cloud, Edge and Fog Computing	Deployment options ranking		✓	✓	✓		
[17]	Industrial Fog Computing	Computation offloading		✓	✓	✓		
[18]	Content delivery wireless network	Resource allocation	✓	✓		✓		
[19]	Mobile Cloudlet Platforms	Computation offloading	✓	✓		✓		
[20]	Fog Computing	Computation offloading	✓	✓	✓		✓	
[21]	Homogeneous Fog Networks	Tasking scheduling	✓	✓	✓		✓	
[22]	Industrial Fog computing	Computation offloading	✓	✓	✓	✓		
[23]	Mobile Cloud Computing	Transmission scheduling	✓	✓	✓		✓	
[24]	Mobile Cloud Computing	Computation offloading	✓	✓	✓		✓	
[25]	Mobile Cloud Computing	Scheduling and link selection	✓	✓	✓	✓		
[26]	Mobile Cloud Computing	Resource management and allocation	✓	✓	✓	✓		
[27]	Mobile Cloud Computing	Computation offloading	✓		✓	✓	✓	
This work	Fog Computing	Request dispatching	✓	✓	✓	✓		✓

[1] S.T: Service Time [2] E.C: Energy Consumption [3] Cntr: Centralized [4] Dstr: Distributed [5] RNE: Renewable Energy

2.3. Stochastic Methods for Resource Allocation

There have been extensive studies on resource allocation based on stochastic methods, such as the Markov Decision Processes (MDP) or LOT. MDP is a powerful mathematical framework in a dynamic environment which also is leveraged for resource allocation and power management in the context of fog and edge computing [16, 28, 17]. For example, Kochovski *et al.* [16] introduced a mechanism for automatically ranking candidate deployment options for micro-services. They deployed a Markov-based probabilistic model by taking into account the Non-Functional requirements and usage context. Wang *et al.* [17] investigated task offloading in fog computing by considering both energy consumption and service time. They formulated the problem as a cost-minimization one, and exploited MDP and reinforcement learning to devise dynamic scheduling algorithms. However, MDP-based methods suffer from the curse dimensionality. As it is mentioned in [16], increasing the number of transitions can significantly impact the computational complexity of the MDP-based methods.

Recently, LOT has been adopted in several works as a means to derive resource management strategies in the fog computing environment [18, 19]. In particular, Zhao *et al.* proposed an online node assignment and resource allocation algorithm based on LOT, to improve service time, network throughput and fairness [18]. The request dispatching problem is decomposed into two separate operations, namely node assignment at fog control node and bandwidth allocation at the fog access node, and then, LOT is leveraged to solve each part separately. Similarly, in [19] LOT is utilized to control the request admission and dispatching, purchase required computing service, and resource allocation. While the cost (in terms of service time or price) is the main objective in the above-mentioned works, there also have been studies on energy consumption minimization, such as [20, 21, 22, 23, 24, 25]. For example, in [21], Yang *et al.* suggested a scheduling scheme to minimize energy consumption while reducing service time. Adopting LOT, Yang *et al.* proposed an online algorithm to balance between overall energy consumption and average service time. Furthermore, Chen *et al.* [22] studied the problem of service placement and task admission in a fog network. The problem is formulated as a stochastic optimization problem subject to conditions in the form of long-term constraints. Leveraging LOT, Chen *et al.* developed an online algorithm to minimize average service time while satisfying the battery constraints of fog nodes.

2.4. Focus of the Paper

Our work is very relevant to [26] and [27]. In [27], Zhang *et al.* studied the problem of computation offloading, and proposed an energy-efficient algorithm based on LOT as the solution. The problem is formulated as a stochastic optimization problem to minimize average energy consumption while satisfying both data and energy queue stability. Then, the problem is transformed into more deterministic easy-to-solve subproblems. They consider an execution model consists of both local and Mobile-Edge Computing (MEC) server execution, and introduce an asleep/awake decision to improve the energy harvested by devices. In contrast, we consider a computing ecosystem consisting of computing nodes in the fog and the cloud. Also, in this paper, these are fog nodes, not the end devices, which are capable of harvesting green energy. In other words, we do not deal with decisions of computation offloading at end devices level, but with dispatching of the incoming requests into the controller, among available computing nodes with on-site renewable energy sources. On the other hand, unlike [27], in which energy consumption is minimized, our focus in this paper is to minimize the service time while maintaining the utilization of the fluctuating renewable energy above a predefined threshold. The summary of the comparison between this paper and most relevant works is presented in Table 1.

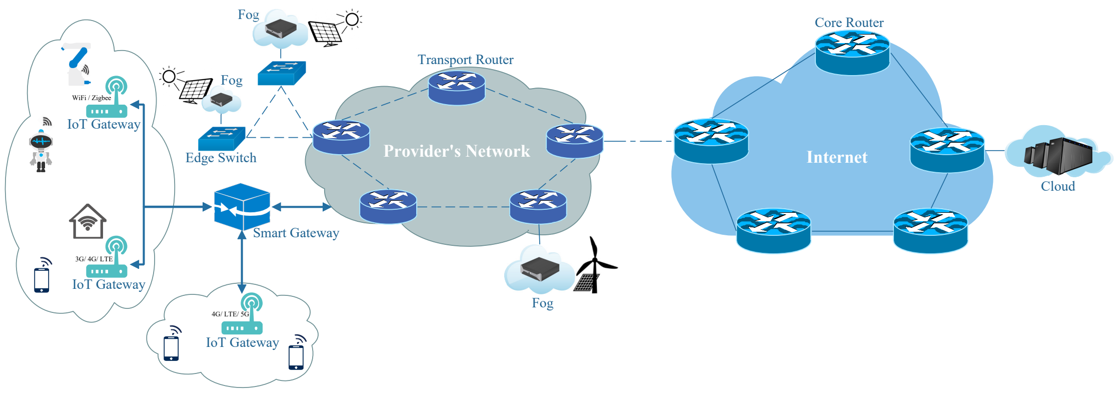


Figure 1: High Level Fog Network Architecture.

3. System Model

We consider a fog environment consisting of N heterogeneous fog nodes, a cloud at a remote data center, IoT nodes, and a request dispatching controller. We assume that the fog nodes can be powered with on-site renewable energy sources, the power grid network, or both. In contrast to fog nodes, the only source of energy for the cloud is the grid.

The fog nodes are highly distributed and heterogeneous. Therefore, they have different amounts of computing capability, and the IoT devices may encounter different delay to access them. The access delay related to each fog node varies depending on its properties and location, and is significantly lower than the cloud access delay, Fig. 1. The cloud is assumed to have more processing power with respect to the fog nodes.

The IoT and fog nodes are divided into some domains (for example, a domain of nodes in a university campus, a factory or shopping center and etc.), which induces a clustering structure of the system. Each cluster of domains is under the control and management of one request dispatching entity (Controller). Requests with different sizes and requirements come into the system from IoT devices or other fog nodes, as shown in Fig. 2. Actually, the IoT devices send their requests to the IoT gateway where the controller can be embedded as a component in the gateway software. Requests' and resources' information are registered in the controller. The controller, based on the available resources, system and network conditions, and requests' demands, makes a decision on how to dispatch the requests among computing nodes. Upon making the decision, the requests are assigned to the most proper computing nodes.

Notation: In this paper, vectors are specified by boldface letter. $\bar{X}$ indicates the average of the variable X. We interchangeably may use $y(t)$ or $\hat{y}(f(t))$ where the latter is to clearly represents the dependency of $y(t)$ to $f(t)$. $\mathbb{E}\{.\}$ denotes the expectation operator. $\limsup$ represents for Limit Superior[1]. Furthermore, the definition of key symbols are summarized in Table 2.

Table 2: THE SUMMARY OF KEY SYMBOLS

Symbol	Definition
t	Index of time slots
$A(t)$	Request arrival rate into the system
i	Index of computing nodes
$R_i(t)$	Request arrival rate into the computing node i
$B_i(t)$	Service rate of the computing node i
Cp_i	Processing capability of the computing node i
$\mathbf{c}(t)$	Vector of decision control
$Q_i(t)$	Queue backlog related to the computing node i
$e_i(t)$	Energy consumption of the computing node i
$e_i^P(t)$	Energy consumed for processing
$e_i^C(t)$	Energy consumed for communication
k	Index of requests
$M_{k,i}^I$	Input data volume of request k
$M_{k,i}^R$	Output data volume of request k
$\eta_{k,i}$	Energy consumed by the computing node i to process the request k
X_k	Processing demand of the request k
$S_k(t)$	Service time of the request k
$u_{k,i}$	Time to upload the request k to the computing node i
$w_{t,i}$	Waiting time of the request k in the queue of computing node i
$p_{k,i}$	Processing time of request k on the computing node i
$d_{k,i}$	Time to send back the result of request k from the computing node i
$L(t)$	Lyapunov function
$\Delta(L(t))$	Drift in Lyapunov function

We look at the system in a time-slotted manner indexed by t, where $t \in \{0, 1, 2, \dots\}$. At each time slot, requests arrive to the system with arrival rate $A(t)$, where $A(t)$ is independent and identically distributed (i.i.d) over time slots and $\mathbb{E}\{A(t)\} = \lambda$. We assume that the system conditions (including the amount of resources, wireless channel condition and etc.) during each time slot remain fixed, but can vary from one time slot to the next one. It is at the beginning of each time slot that the controller performs the request dispatching algorithm.

3.1. Queue Model of Computing Nodes

We model each computing node as an M/M/1 queue. Requests arrive at queue i with arrival rate $R_i(t)$ and are

[1]A well defined limit while $t \to \infty$, for a function $f(t)$ may or may not exist, while the function is relay limited (like function sin). Therefore, $\limsup_{t\to\infty} f(t)$ is defined as the largest value that limit $f(t)$ over any subsequence of t_k that increase to infinity while limit $f(t_k)$ exists.

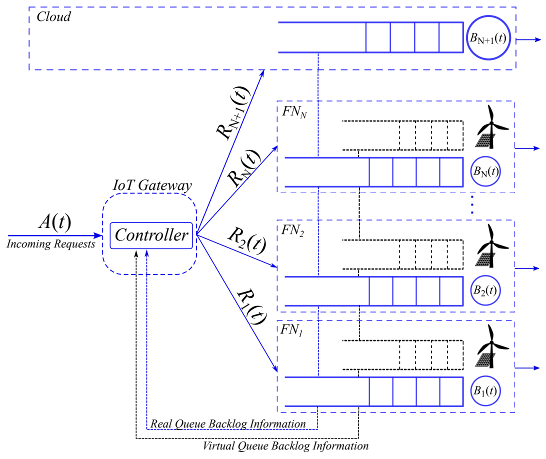


Figure 2: Queue Model of The System.

served with service rate $B_i(t)$, which is related to the computing capability of the nodes[2], Cp_i. The dynamic equation for each queue at time slot t is described by,

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

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. Equation (1) states that from the requests which come into the queue at each time slot and those remained from previous slots as much as the service rate is processed and the remainder is left for the next slot, as the queue backlog.

At each time slot, the controller makes a decision $\mathbf{c}(t)$ to assign requests to the most suitable computing nodes. Where, $\mathbf{c}(t)$ is a vector defined in the form[3] of $\mathbf{c}(t) \triangleq \{(k,i) | k \in \{1, 2, \dots, K^t\} \textit{ and } i \in \{1, 2, \dots, N+1\}\}$, K^t is the number of requests in the current time slot and $N+1$ is the number of computing nodes[4]. Each element of $\mathbf{c}(t)$ indicates which requests are assigned to which computing nodes. Therefore, $R_i(t)$ is a function of decision control $\mathbf{c}(t)$, $R_i(t) = \hat{R}_i(\mathbf{c}(t))$, and satisfies $0 \le R_i(t) \le R_i^{max}$. Furthermore, it is assumed that the rate of request arrival is in the capacity region of the system [10]. In other words, $A(t) = \sum_{i=1}^{N+1} R_i(t)$ and $A(t) \le \sum_{i=1}^{N+1} B_i(t)$ for all time slots, which is performed by admission control.

3.2. Queue Stability

Regarding the queue model of the system in the previous section, we say a queue is *stable* if the average queue backlog is always *finite*. Therefore, we can formally define the queue stability [10] as,

$$\bar{Q}_i(t) = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{|Q_i(t)|\} < \infty \tag{2}$$

Equation (2) requires that not only the long term behavior of the queues to be finite, but also the queues' backlog always remain under some finite value.

3.3. Energy Consumption Models

In this section, we present energy consumption models (ECM) for requests and computing nodes. The request-related ECM indicates the amount of energy which is required for processing a request on a computing node. It is a fine-grained model and depends on the amount of processing that a request needs to be done. The node-related ECM indicates the energy consumption of a specific computing node under the current overall workloads. The node-related ECM is relatively coarse-grained. It is cumulative and related to the utilization of the computing node[5].

3.3.1. Request-related ECM

For each request k and fog node i, we define request energy consumption, $\eta_{k,i}$, which is related to its processing amount, X_k. The request energy consumption determines how much energy is needed to serve the request k on the selected node i. We can calculate the request energy consumption by a quadratic function as in [14],

$$\eta_{k,i} = a_i X_k^2 + b_i X_k + \alpha_i, \tag{3}$$

where $a_i > 0$ and $b_i, \alpha_i \ge 0$ which are defined accordingly for each node.

3.3.2. Node-related ECM

We consider the energy consumption of a computing node at each time slot, $e_i(t)$, as the sum of energy consumed for processing purpose, $e_i^P(t)$, and the energy consumed for communication purpose, $e_i^C(t)$, at that time slot as:

$$e_i(t) = e_i^P(t) + e_i^C(t), \tag{4}$$

where $e_i(t)$, $e_i^P(t)$ and $e_i^C(t)$ are functions of the decision control $\mathbf{c}(t)$. Therefore, we can rewrite (4) in the form of $\hat{e}(\mathbf{c}^i(t)) = \hat{e}^P(\mathbf{c}^i(t)) + \hat{e}^C(\mathbf{c}^i(t))$. In the following we explain how to obtain each part.

Processing ECM: Computing nodes consume energy, whether idle or active. The energy consumption of a computing node for processing purpose at each time slot, $e_i^P(t)$, can be modeled as [22, 29],

$$e_i^P(t) = e_i^{P,I}(t) + e_i^{P,A}\left(\frac{\lambda_i(t)}{\mu_i(t)}\right), \tag{5}$$

[2] $B_i(t)$ is considered a fixed value for each computing node and equal to its computing capability, $B_i(t) = Cp_i$.

[3] We also define $\mathbf{c}^i(t)$ to determine all requests assigned to node i as $\mathbf{c}^i(t) \triangleq \{(k,i) | k \in \{1, 2, \dots, K_i^t\}\}$ where K_i^t denotes the total number of requests assigned to node i. Also, $c_k(t)$ is defined to indicate which node the request k is assigned, as $c_k(t) = (k,i)$.

[4] In $N+1$, N indicates the number of fog nodes and 1 is for the cloud.

[5] The ECM for a computation node is not simply the summation of the energy required to accomplish its assigned tasks. The reason is that in practice, other parameters, such as idle wake-up energy, play important roles in the node's ultimate energy consumption.

where $e_i^{P,I}$ and $e_i^{P,A}$ are idle energy consumption and active energy consumption (when a computing node is processing the workloads) of the i^{th} node, respectively. Considering the queue model of the computing node, $\lambda_i(t) = \mathbb{E}\{R_i(t)\}$ denotes the average requests arrival rate into the queue, $\mu_i(t) = \mathbb{E}\{B_i(t)\}$ denotes the average service rate of the queue, and $(\lambda_i(t)/\mu_i(t)) < 1$ represents the utilization of the computing node.

Communication ECM: Each request to be processed in a computing node needs to transfer the required input data to the selected node. Also, after the request get processed, the results need to be sent back. The sum of the required energy for receiving the input data, $e_i^{C,r}(t)$, and returning back the results, $e_i^{C,s}(t)$, is called communication energy consumption[6], $e_i^C(t) = e_i^{C,r}(t) + e_i^{C,s}(t)$.

Considering ε_i as the amount of energy (Joule) that fog node i consumes to operate each unit of data, we obtain the required energy to receive input data as [30],

$$e_i^{C,r}(t) = \sum_{k=0}^{K_i^t}(\varepsilon_i M_{k,i}^I) \tag{6}$$

and the required energy to transmit the results as [30],

$$e_i^{C,s}(t) = \sum_{k=0}^{K_i^t}((\varepsilon_i + \varsigma_{ij}(t))M_{k,i}^R) \tag{7}$$

where $\varsigma_{ij}(t)$ denotes the amount of energy (Joule) node i consumes to transmit a unit of data to device j. Furthermore, $M_{k,i}^I$ and $M_{k,i}^R$ represent the volume of input and output data (Kbytes) that is needed to be transferred for each request k to/from node i, respectively. Based on Shannon's channel capacity theorem, we can write $\varsigma_{ij}(t)$ in terms of the data-rate, $r_{ij}(t)$, and link's bandwidth, W_{ij}, as [26],

$$\varsigma_{ij}(t) = \beta_{ij}(t)(2^{r_{ij}(t)/W_{ij}} - 1)/r_{ij}(t), \tag{8}$$

where $\beta_{ij}(t)$ depends on the parameters such as noise level, transmission quality and etc. Equation (8) shows that the increase in communication speed requires more power.

4. Problem Statements

In this section, we first deal with expressing the green energy utilization constraint satisfaction by defining two lateral constraints. Then, we go through the formulation of the main problem by defining the objective function and considering related constraints.

[6]The communication energy consists of different parts; the energy is consumed by the computing nodes, network equipment (routers and switches) in the path and IoT devices, but here we just concern about part of it that is consumed by the computing nodes.

4.1. Average Utilization of Green Energy in Fog Nodes

As we assumed that the fog nodes are equipped with on-site renewable energy sources, it is desirable to utilize green energy as much as possible whenever it is available (i.e. green energy is not zero). Because the requests are assumed to be time-constrained, we cannot postpone requests to a later time. On the other hand, the extra energy cannot be stored in a battery to be used later. As a result, an optimal controller will use green energy as much as possible at each time slot. We can categorize the status of energy consumption with respect to the produced green energy for each fog node into two states, as illustrated in Fig. 3. First, the situation where the energy consumption of the node is lower than (or equal to) the green energy produced by the on-site renewable sources, such as e_1 and e_2. Second, the energy consumption is higher than the on-site produced green energy, which leads the node to borrow the remainder energy from the main grid, such as e_3.

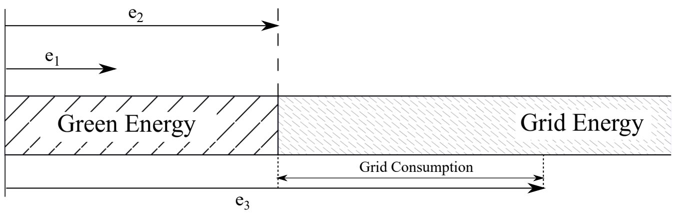


Figure 3: Grid Consumption.

In order to maximize the total utilization of green energy, the requests have to be first assigned to the nodes with the available green energy greater than the energy consumption which best suits the requests demands, if any ones exist. We take this constraint into consideration by defining a utilization function, $U_i(t)$, which is a function of decision control, $U_i(t) = \hat{U}(\mathbf{c}^i(t))$. The green energy-related utilization function is defined based on the description of grid consumption as expressed in Fig. 3, in the form of,

$$\hat{U}(\mathbf{c}^i(t)) = \max[e(\mathbf{c}^i(t)) - e_{pro}^i(t), 0], \tag{9}$$

where, $e_{pro}^i(t)$ denotes the amount of energy produced by the green energy sources at time slot t, and $e(\mathbf{c}^i(t))$ denotes the amount of energy consumed by node i at time slot t under the decision control $\mathbf{c}$(t), which is obtained by (4). Grid (or brown energy) consumption indicates the amount of energy borrowed from the grid after the green energy is completely used. Therefore, our goal is finding a decision ($\mathbf{c}(t)$) which leads to lower amount of grid consumption. Accordingly, it is sufficient here to have the energy consumption for all the fog nodes lower than the produced green energy. For all decisions which satisfy the condition, it is the choice of performance. We present this constraint in the form of long term average grid consumption and aim to keep it below a predetermined threshold

(Υ), $\bar{U}_i(t) = \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{U_i(t)\} \leq \Upsilon$ for some finite value of Υ.

4.2. Fog or Cloud Energy Aware Selection

The controller may deal differently with requests depending on the green energy availability. Whenever there is not any green energy available, it is better to serve some heavy requests at the cloud because cloud computing is utilizing smarter and more efficient energy management, along with the opportunity of utilizing cheaper energy. Therefore, those requests that are best suited to be offloaded to cloud can be sent to a remote data center. The best fit requests are those with large amount of energy consumption and less amount of data communication. So, we define utility function $\nu(t)$ for request k and the selected node i, as $\nu(t) = \hat{\nu}(c_k(t)) = \eta_{k,i}/M^I_{k,i}$, where $\eta_{k,i}$ and $M^I_{k,i}$ denote the request's energy consumption in Joules defined by (3) and the input data volume of request k needed to be transferred to node i in Kbytes, respectively. The value of utility function for the requests that need high amount of energy and have low communication demands are large, these requests are sent to cloud. Accordingly, the value of this function for requests that are best suited to be served in fog is small. So, we have to choose the decision $\mathbf{c}(t)$ in the way to lower the value of utility function for requests assigned to fog nodes. Therefore, we define the long term average utility constraint for the requests that are served by the fog nodes under decision control $\mathbf{c}(t)$, as $\bar{\nu}(t) = \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{\sum_{c_k(t)\in\mathbf{c}^f(t)} \hat{\nu}(c_k(t))\} \leq \Gamma$, where $\mathbf{c}^f(t)$ is a subset of $\mathbf{c}(t)$ and $\mathbf{c}^f(t) = \{(k,i)|k \in \{1,2,\ldots,K^t\}\ and\ i \in \{1,2,\ldots,N\}\}$. Moreover, Γ is a constant with a non-negative finite value.

In some cases, invoking specific services on some fog nodes may pose certain limitations on the placement of other services. For example, in the case of micro-services with fairly long invocation chains, it may be preferred to place requests on fog nodes completely to avoid the prohibitive latency of the cloud data center. Thus, further investigation is required to involve such considerations into the method, which can be appeared in the form of sending invocation chain characteristics along with other parameters to the controller in order to schedule the chain on proper fog nodes.

4.3. Problem Formulation

Towards mathematically describing the problem, we first introduce the service time related objective function. Then considering the aforementioned constraints and the objective function, we present a mathematical definition of the problem.

The service time is calculated from when the request is sent to the selected computing node until the time the results come back to the requesting node. We can ignore the time required to submit a request to the controller and get informed about the controller's decision regarding the place (one of the fog nodes or the cloud) for serving the request, because introducing the requests to the controller is via metadata which leads to a negligible and fixed time for all the requests. So, we define the service time[7] for request k as the sum of the time to upload the request to the computing node, $u_{k,i}$, the time that the request waits in the related queue, $w_{t,i}$, the time needed the request gets served, $p_{k,i}$, and the time to return back the results, $d_{k,i}$, in the form of $S_k(t) = \hat{S}(c_k(t)) = u_{k,i} + w_{t,i} + p_{k,i} + d_{k,i}$. $w_{t,i}$ and $p_{k,i}$ are obtained by X/Cp_i, where X is the queue backlog or the processing amount for the request k, respectively. $u_{k,i}$ and $d_{k,i}$ are obtained by $M_{k,i}/r_{ij}(t)$, where $M_{k,i}$ denotes the volume of input/output data. It is worth noting that the network level metrics (jitter, packet loss, bandwidth, and latency), depending on the application, can be differently affected by the variations of $r_{ij}(t)$, and consequently influence user-experienced QoS [31]. Finally, the total service time in time slot t is obtained by $S(t) = \hat{S}(\mathbf{c}(t)) = \sum_{k=1}^{K^t} \hat{S}(c_k(t))$.

We investigate the request dispatching problem in order to minimize service time while maintaining green energy utilization and queue stability at a satisfactory level. Therefore, considering the service time in the form of long term average, green energy utilization in the form of two aforementioned constraints, and the definition of stability, we can formulate the main problem as,

$$\mathbf{P1:} \min_{\mathbf{c}(t)} \ (\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{\hat{S}(\mathbf{c}(t))\}) \tag{10}$$

$$S.T\ \bar{U}_i(t) \leq \Upsilon \tag{10a}$$

$$\bar{\nu}(t) \leq \Gamma \tag{10b}$$

$$\bar{Q}_i(t) \leq \infty\ for\ i \in \{1,2,\ldots,N+1\} \tag{10c}$$

P1 is a stochastic optimization problem where (10a) induces the grid energy consumption constraint, (10b) is related to the selection between the fog and cloud, and (10c) ensures the stability condition. In order to solve P1, one needs the dynamics of the system to be known in advance, but it is difficult and imprecise if not impossible. So, P1 is a highly challenging problem to solve while its solution depends on the statistical information of system dynamics.

One of the promising methods to deal with such stochastic optimization problems is the Lyapunov Optimization Technique (LOT). LOT leads to an approximation solution to P1 which is adjustable by parameter V within $[O(\frac{1}{V}), O(V)]$ of the optimum solution in terms of service time and green energy utilization, respectively [10]. By increasing the value of V, we put more emphasis on minimizing the service time; while by decreasing it, there is

[7]It is assumed that the central request dispatching controller has complete knowledge about link level information (such as links' delays and data rates) throughout the edge network. In practice, this assumption holds in software-defined network (SDN) architectures [3].

less emphasis on the service time and more on the green energy utilization. Queues' backlogs, processing demand, the volume of input and output data, and the link throughput are the inputs of the algorithm that are determined at run-time. Moreover, the number of fog nodes, the availability of green energy harvesting equipment, and the fog nodes' service rates which depends on their processing capabilities are given parameters at the design time. In the next section, we briefly introduce the fundamental basics of LOT, and then in section VI, we return to our primary problem P1, apply the LOT, and develop our proposed request dispatching algorithms.

5. Queue-stable Optimized Solution

In this section, we first describe the Lyapunov optimization technique. Then we explain the idea of virtual queues which are used to satisfy lateral constraints of an optimization problem.

5.1. Lyapunov Optimization Technique

The Lyapunov method has been widely adopted in the theory of control systems to prove system stability without solving the system dynamic equation [32, 33]. Furthermore, a variation of the Lyapunov method, called Lyapunov Optimization Technique (LOT) is used to jointly stabilize system and minimize a cost function. LOT is also used in computer science to investigate the system queue stability and also the problem of jointly minimizing a sort of penalty and stabilizing the system queues [10]. The main idea of LOT is based on starting from a stable condition and preserving its stability by controlling system changes continuously. In LOT, a positive scalar function called Lyapunov function, $L(t)$, is defined to assess the stability condition. A small value of L indicates that all the queues are working normally, but a large value of L indicates that at least there is a queue which gets congested. Here Lyapunov function is defined as in [10], in the form of a quadratic function:

$$L(t) = \frac{1}{2}\sum_{i=1}^{N+1} Q_i(t)^2 \tag{11}$$

where $Q_i(t)$ denotes the backlog of the i^{th} queue. The Lyapunov function is a scalar measure of congestion in the queues.

We also define drift in Lyapunov function as the difference in Lyapunov function at two consecutive time slots:

$$\Delta(L(t)) = \mathbb{E}\{L(t+1) - L(t)|\mathbf{Q(t)}\}, \tag{12}$$

where, $\mathbf{Q(t)}$ is the vector of all the queues, $\mathbf{Q(t)} \triangleq (Q_1(t), Q_2(t), \ldots, Q_{N+1}(t))$. The drift helps to push Lyapunov function towards a low congestion state, by setting control on the change in Lyapunov function from one slot to the next. If we start from a stable initial condition (finite initial queue state), bounding the drift to a finite value in each time slot leads to a stable condition.

Although, greedily minimizing the drift at each time slot guarantees the queue stability, but it may affect the objective function which has not yet been incorporated. Therefore, we incorporate the objective function by involving penalty function with a scaling factor V into the drift, which now is called Drift plus Penalty (DpP). The penalty is an appropriate function to which the objective function is mapped. Thus, in the LOT, we now try to minimize the DpP in each time slot. We write the DpP expression as:

$$\Delta(L(t)) + V\mathbb{E}\{S(t)|\mathbf{Q(t)}\}, \tag{13}$$

where $\mathbb{E}\{S(t)|\mathbf{Q(t)}\}$ represents the penalty function. In equation (13), the non-negative constant value V relatively determines the importance of stability against the penalty function. For instance, defining $V = 0$ means stabilizing the queue with no consideration of the penalty. On the other hand, increasing V to larger values ($V > 0$) includes the weighted penalty function in the decision.

5.2. Virtual Queue

Besides the real queues that exist in the system, we can also define some new queues called virtual queues. Indeed, we can map some requirements in the system (such as constraints on the power, cost and etc.) to these virtual queues. The virtual queues can be used to satisfy constraints in a long term average representation of the parameters besides the main problem [34]. For example, if we have a constraint in the form of $\bar{x}(t) \leq 0$ in the system, then, by defining an appropriate virtual queue and stabilizing the queue we can satisfy it.

Let's define the virtual queue $H(t)$ with the update equation as:

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

then the following proposition holds.

Proposition 1: Stabilizing the virtual queue $H(t)$ enforces the $\bar{x}(t) \leq 0$ constraint.

Proof: see Appendix A. □

Leveraging the virtual queues helps us to turn the problem of satisfying the constraints into a pure stability problem within LOT.

6. Proposed Request Dispatching Algorithms

In this section we employ the LOT to solve the optimization problem P1. We define the virtual queues Z_i and G related to the constraints (10a) and (10b), respectively.

Corresponding to the constraint (10a) which imposes $\bar{U}_i(t) \leq \Upsilon$, we define $y_i(t) = \hat{y}(\mathbf{c}^i(t)) = \hat{U}(\mathbf{c}^i(t)) - \Upsilon$ to rewrite the constraint in the form of $\bar{y}_i(t) \leq 0$. Now, we can define the virtual queue $Z_i(t)$ with the updating equation

$$Z_i(t+1) = \max[Z_i(t) + y_i(t), 0]. \tag{15}$$

The constant value Υ in the constraint (10a) and in the definition $y_i(t)$ is a predetermined threshold representing the maximum desirable brown energy consumption defined by power management policies.

Furthermore, because of constraint (10b), We need to add another virtual queue. Therefore, with regard to constraint (10b) which states that $\bar{\nu}(t) \leq \Gamma$, we define $f(t) = \hat{f}(\mathbf{c}^f(t)) = \sum_{c_k(t) \in \mathbf{c}^f(t)} \hat{\nu}(c_k(t)) - \Gamma$. Now, we can define the virtual queue $G(t)$ to satisfy the constraint with the following update equation:

$$G(t+1) = \max[G(t) + f(t), 0]. \tag{16}$$

where, in the definition of $f(t)$ (and also in constraint (10b)), the parameter Γ presents the desirable threshold for the proportionality of requests to be sent to the cloud, and is obtained by $\bar{Z}(t) \times \Omega$, where $\bar{Z}(t)$ denotes the average backlog of virtual queues Z_i, for $i \in \{1, 2, \ldots, N\}$, at time slot t and the constant value Ω is a predefined threshold.

The above transformation of constraints into the virtual queues, convert the original problem into a pure stability problem in which the goal is to stabilize the requests real and virtual queues. Therefore, the vector $\boldsymbol{\theta}(t)$ is defined as concatenation vector of real and virtual queues in the form of $\boldsymbol{\theta}(t) \triangleq [\boldsymbol{Q}(t), \boldsymbol{Z}(t), G(t)]$. Accordingly, we can define Lyapunov function as, $L(t) = \frac{1}{2}(\rho_1 \sum_{i=1}^{N+1} Q_i(t)^2 + \rho_2 \sum_{i=1}^{N+1} Z_i(t)^2 + \rho_3 G(t)^2)$ where $\rho_j, j = 1, 2, 3$ are auxiliary variables that help to prioritize different types of queues related to each others. Accordingly, Lyapunov DpP expression based on vector $\boldsymbol{\theta}(t)$ takes the form of $\Delta(L(t)) + V\mathbb{E}\{S(t)|\boldsymbol{\theta}(t)\}$.

Proposition 2: Considering $\mathbf{c}(t)$ as an arbitrary dispatching control decision that is adoptable for all time slots t, the Lyapunov DpP satisfies:

$$\begin{aligned} &\Delta(L(t)) + V\mathbb{E}\{S(t)|\boldsymbol{\theta}(t)\} \leq \Psi + V\mathbb{E}\{\hat{S}(\mathbf{c}(t))|\boldsymbol{\theta}(t)\} \\ &-\rho_1 \sum_{i=1}^{N+1} Q_i(t)B_i(t) + \rho_1 \sum_{i=1}^{N+1} Q_i(t)\mathbb{E}\{R_i(t)|\boldsymbol{\theta}(t)\} \\ &+ \rho_2 \sum_{i=1}^{N+1} Z_i(t)\mathbb{E}\{y_i(t)|\boldsymbol{\theta}(t)\} + \rho_3 G(t)\mathbb{E}\{f(t)|\boldsymbol{\theta}(t)\}, \end{aligned} \tag{17}$$

where Ψ is a constant.

Proof: The proof is presented in Appendix B. □

Inferring from proposition 2 and starting from a stable condition, we can solve the minimization problem P1 by minimizing the upper bound of Lyapunov DpP. Thus, based on proposition 2, the minimization problem P1 is transferred to P2 at each time slot as follow:

$$\textbf{P2:} \min \quad V \sum_{k=1}^{K^t} S_k(t) + \rho_1 \sum_{i=1}^{N+1} Q_i(t)R_i(t) + \rho_2 \sum_{i=1}^{N+1} Z_i(t)y_i(t) + \rho_3 G(t)f(t) \tag{18}$$

Remark 1: Solving P2 in each time slot not only minimizes the service time as the objective function but also enables us to jointly satisfy the lateral constraints and stabilize system queues.

In order to solve P2, we present an algorithm which is called Joint Algorithm (JoA). The algorithm uses only the current system, network and green energy sources' conditions to make a decision. JoA is implemented in the controller. It makes decision control $\mathbf{c}(t)$ at the beginning of each time slot which leads to a condition that minimizes the long term average service time and satisfies stability and green energy-related constraints.

6.1. Joint Algorithm

At each time slot, JoA goes through all the possible combinations of assigning the requests to the computing nodes and choose the best among all, $\mathbf{c}^*(t)$. Thus, all the requests in the current time slot get dispatched at once with overall effect that those requests may have on each others and the queues.

It can be implemented by exhaustive search or branch and bound methods, but in either way, JoA comes with $\mathbf{c}^*(t)$ that induces the minimal value to the expression in (18). Here, we use exhaustive search to find $\mathbf{c}^*(t)$. The proposed algorithm is summarized in Algorithm 1.

Algorithm 1: *Joint Algorithm (JoA)*

```
    Input: List of requests and computing nodes
    Output: Decision control $\mathbf{c}^*(t)$
1:  Initialization
        Initialize the control parameters in the LOT
2:  While $t < t_{end}$, do
    /*Obtaining required parameters and finding the best $\mathbf{c}^*(t)$ */
3:      $I(t)$ = all possible combinations of assigning $K^t$ requests
    to $N+1$ computing nodes
4:      $\mathbf{c}_j(t) = I_j(t),\ j \in \{1, 2, \ldots, |I(t)|\}$
    /* $\mathbf{c}(t) \triangleq \{(k,i)|k \in \{1, 2, \ldots, K^t\}$ and $i \in \{1, 2, \ldots, N+1\}\}$*/
5:      For all $\mathbf{c}_j(t)$ in $I(t)$
6:          $value\_DpP[j] = V\hat{S}(\mathbf{c}_j(t)) + \rho_1(\mathbf{Q}(t)\hat{R}(\mathbf{c}_j(t)))+$
            $\rho_2(\mathbf{Z}(t)y(\mathbf{c}_j(t))) + \rho_3(G(t)f(\mathbf{c}^f(t)))$
        End for
7:      $\mathbf{c}^*(t) = \arg\min_{\mathbf{c}_j(t)}(value\_DpP)$
8:      Dispatch according to $\mathbf{c}^*(t)$
9:      Update the Queues
    End While
```

The algorithm takes the list of arrived requests and existing computing nodes as the input. It returns the decision control $\mathbf{c}^*(t)$ as the output.

Line 1: Initialize the control parameters in the LOT, such as control parameter V. It helps to adjust the system according to the system management policy.

Line 3-4: Provide all the possible combinations of request dispatching and defining $\mathbf{c}_j(t)$ to access each combination in the set of all combinations $I(t)$.

Line 5: Go through all the possible combinations of request dispatching.

Line 6-7: Calculate the value of expression in the minimization problem P2 and find the smallest value and its related decision control $\mathbf{c}^*(t)$ among all $\mathbf{c}(t)$.

Line 8-9: Dispatch requests according to the selected decision control $\mathbf{c}^*(t)$ and update the queues.

The complexity of the solution is of the order $(N+1)^{K^t}$, where $N+1$ denotes the number of computing nodes (in fog and the cloud) and K^t indicates the number of requests that are ready to be dispatched in the current time slot. Therefore, JoA suffers from the growth of dimensionality with the increase in the arrival rate or the number of computing nodes.

In order to deal with the issue of dimensionality, we introduce a Greedy Algorithm (GrA) in the next section. In the greedy solution instead of considering the problem of dispatching all the requests at once, the requests greedily get dispatched one by one.

6.2. Greedy Algorithm

In the greedy solution, requests are assigned one by one. It is like a situation where the controller stops the time, dispatches a request and observes the effects then updates the queues and goes through further requests in the time slot. Finally, the controller issues the decisions as a whole. We assume that time slot t is divided into small enough sub time slots (S-T), which only one request arrives in each S-T. Therefore, the decision control for each S-T ($c_k(t)$) only includes assigning a request to a computing node. The union of $c_k(t)$ for all S-T determines $\mathbf{c}(t)$. GrA is summarized in Algorithm 2.

Algorithm 2: *Greedy Algorithm (GrA)*

```
      Input: List of requests and computing nodes
      Output: Decision control $\mathbf{c}^*(t)$
1:    Initialization
          Initialize the control parameters in the LOT
2:    While $t < t_{end}$, do
      /*Obtaining required parameters and finding the best $\mathbf{c}^*(t)$ */
      /* $c_k(t) \triangleq (k,i)$ */
3:        For $k = 1$ to $K^t$
4:            For $i = 1$ to $N+1$
5:                $value\_DpP[k] = V\hat{S}(\mathbf{c}_k(t)) + \mathbf{Q}(t)\hat{R}(\mathbf{c}_k(t)) +$
                  $\mathbf{Z}(t)y(\mathbf{c}_k(t)) + G(t)f(\mathbf{c}^f(t))$
              End For
6:            $\mathbf{c}_k^*(t) = \arg\min_{\mathbf{c}_k(t)}(value\_DpP)$
7:            Logically update the queues based on the model
          End For
8:        $\mathbf{c}^*(t) = \cup_{k=1}^{K^t} c_k^*(t)$
9:        Dispatch according to $\mathbf{c}^*(t)$
10:       Update the Queues
      End While
```

Line 1: Initialize the control parameters in the LOT, such as control parameter V. It helps to adjust the system according to the system management policy.

Line 3: Moves on the set of arrived requests to find the best computing node to be dispatched to it.

Line 4: Moves on the set of computing nodes to find the best candidate.

Line 5: Calculate the value of expression in minimization problem P2.

Line 6: Find the smallest value and its related decision control $c_k^*(t)$ among all $c_k(t)$.

Line 7: After finding the best computing node for the request of the current round, we assume the request is dispatched to the selected node and logically update the queues and other system model parameters.

Line 8-10: Form the final decision control $\mathbf{c}^*(t)$ as the union of each decision control related to each request in the set of arrived requests, dispatch requests according to it and update the queues.

The complexity of GrA is of the order $K^t(N+1)$. It is a low complexity solution and has a very close performance to JoA. In order to evaluate the effectiveness of the proposed methods and compare our Joint (JoA) and Greedy (GrA) solution to each other, in the next section, we report some simulations.

7. Evaluation and Simulation Results

In this section, we first describe our simulation setup. Then, we present the results of the fixed-parameters simulations, analyzing the effect of control parameter V, and evaluating the scalability of the algorithms. Finally, we go through different scenarios (by changing a specific parameter at each scenario; such as arrival rate, computing and communication demands) to show how the proposed algorithms behave under different conditions.

7.1. Simulation setup

In order to evaluate the proposed methods, we designed a custom simulator in Matlab and simulated a fog network consisting of N fog nodes (three fog nodes [$N = 3$] for fixed-parameter simulations and up to 40 nodes for scalability analysis) and a cloud server at a remote data center based on the system queue model in Fig. 2. The computing nodes are described with their processing capabilities in terms of million instructions per second (MIPS) and the link properties toward requesting devices in terms of transmission rates. The processing capability for the fog nodes are denoted by the uniform distribution $U[50, 700]$, and for the cloud server is set to 3500 MIPS. Also, the transmission rates from the computing nodes towards the requesting nodes are uniformly generated by $U[1, 5]$ Mbps.

It is assumed that the requests arrival rate follows the Poisson process with the average rate of λ. Furthermore, the requests' computing and communication demands follow the exponential distribution process with the average rate of γ_1 and γ_2, respectively. The selected values for model parameters are summarized in Table 3.

Table 3: CONFIGURATION OF SYSTEM MODEL PARAMETERS

λ	γ_1	γ_2	V	Υ	Γ
0.23	0.12	0.35	7e+14	1	10e+5.6

We adopt real data on green energy production by using Australian hourly information about solar radiation [35] in the form of solar Global Horizontal Irradiance (GHI). GHI presents the instantaneous intensity of solar radiation falling on a horizontal surface. The irradiance unit is watts per square meter (W/m^2).

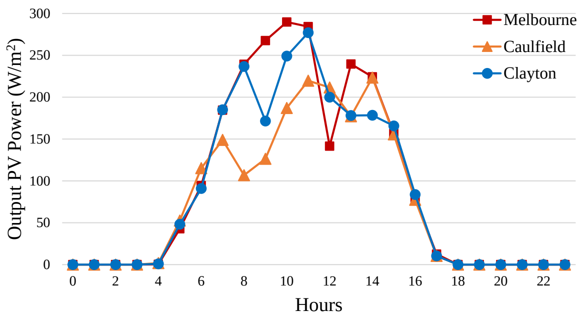


Figure 4: Output PV Power related to the site of each fog node on the 17th of January 2018.

Besides the solar irradiation density, we need the incident angle of solar radiation to the photovoltaic (PV) module to measure the effective solar incident. For a tilted module surface, the perpendicular component of the incident solar radiation to the module surface determines the amount of solar radiation incident on the tilted absorbing surface. We adopt the model in [36] to calculate the effective solar incident on the PV module.

The tilt angle, to achieve the maximum power for a tilted PV module over the course of year, should be equal to the latitude of its location [36]. Each solar panel has an efficiency level, and we assume 30% efficiency for PV modules.

We consider three fog nodes in our fixed-parameter simulation setup. The node F1 is placed in Monash Clayton campus, F2 in Caulfield campus and F3 in the center of Melbourne city. The simulations are performed on information related to the 17th of January 2018. Fig. 4 depicts the output PV power related to each fog node for 24 hours of the selected day in our scenario.

Different performance metrics have been considered to evaluate the efficiency of the proposed method, namely, average queue backlog, average service time, the average number of deadline misses and average brown energy consumption.

As a benchmark, we compare the results to a state-of-the-art request dispatching method, called "workload-aware scheme (WAS)". The WAS method based on communication to computation ratio (CCR) assigns incoming requests to the most proper computing node [4]. Furthermore, to highlight the results, we also compare our methods against the random dispatching method (Rnd). The Rnd method randomly assigns the requests to the computing nodes [21].

7.2. *Simulation Results*

Extensive simulations are performed to assess the efficiency of the proposed methods in terms of stability, service time, deadline miss and green energy utilization for the configuration given by Table 3. In the following, we explain the experiments' results.

7.2.1. *Fixed-Parameters Simulation*

We assume the requests from a domain of IoT nodes come into the system with the average arrival rate $\lambda = 0.23$. The computing and communication demands of the requests follow the exponential distribution with average $\gamma_1 = 0.12$ and $\gamma_2 = 0.35$, respectively. The requests are introduced to the controller and get dispatched to four computing nodes. The experiment is conducted for 100 runs and finally the results are expressed in the form of average of all experiments.

In Fig. 5, the average queue backlog related to each computing node is plotted for different methods. As illustrated in the figure, all the queues in our proposed methods and WAS work nearly to each other, but for example, the queue related to F1 in Rnd scheme has grown dramatically (the queue backlog of 1294 compared to order of 100 in our proposed methods or WAS). In Fig. 6, the average service time (from left) and the average number of deadline misses (from right) are depicted, it is observable that JoA has better performance (in terms of average service time and average number of deadline misses) than others and also GrA has very close performance to JoA (JoA performs 5.7% and 6% better than GrA in terms of average service time and average number of deadline misses, respectively). It can be seen from Fig. 7 that JoA can utilize the on-site green energy sources more efficient than WAS and Rnd. Also, GrA can provide a very close utilization to JoA (JoA performs 1.8% better than GrA).

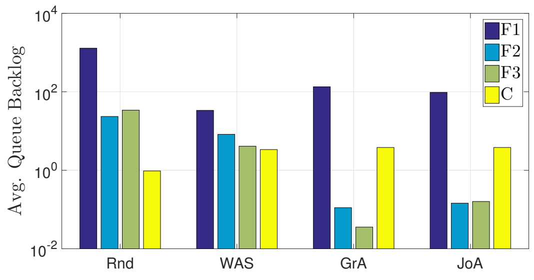


Figure 5: Average Queue Backlog of each node for different methods.

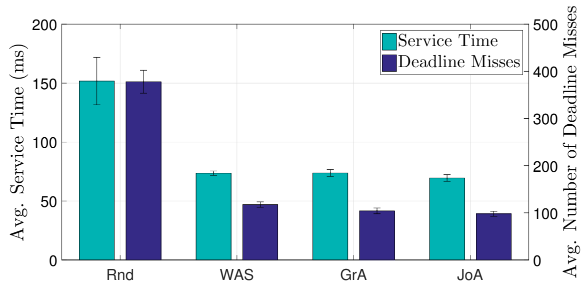


Figure 6: Average Service Time (Left) and Average Number of Deadline Misses (Right) for each method, the error bars show standard deviation.

7.2.2. *The effect of Control Parameter V*

The control parameter V in DpP, relatively determines the importance of stability against service time. Indeed,

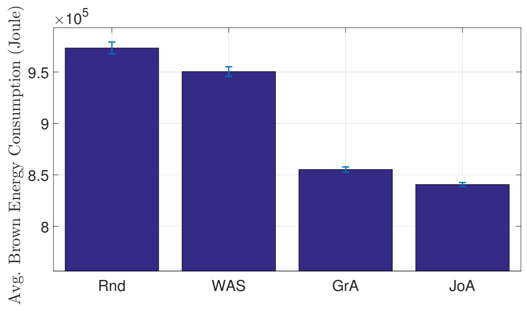


Figure 7: Average Brown Energy Consumption for each method, the error bars show standard deviation.

stabilizing the queues can be in the cost of performance. Therefore, by increasing or decreasing the value of V we determine how it should judge the trade-off between stability and performance. For example, increasing the value of V means we intend to put more emphasis on the performance. In the second scenario, we made changes in the control parameter V to see how the system metrics would change. Fig. 8 and Fig. 9 plot the average of queues' backlog, service time, the number of deadline misses and brown energy consumption when V changes from a small to a large value.

As the value of V increases, more emphasis is put on the service time. Therefore, we see the service time decreases (Fig. 8a). Consequently, the number of deadline misses is decreased (very slight decrease in Fig. 8b). On the other hand, it is observed that more requests are dispatched to close fog nodes and consequently the queues' backlog are increased for these fog nodes (Fig. 9). More requests mean more demands for energy. Considering that in some hours, the green energy is not available or adequate to fulfill the demands, the amount of brown energy consumption is increased accordingly (Fig. 8c).

7.2.3. Scalability

Further, we provide investigation on the scalability of the system by increasing the number of fog nodes. In this scenario, we only involve GrA algorithm in our simulations, because JoA suffers from the growth of dimensionality, as we previously mentioned. Regarding that we tend to evaluate how the system behaves when the system scales, we simultaneously increase the number of fog nodes and the workload injected into the system. Therefore, with respect to the fixed-parameters simulation setting, we consider one domain of IoT nodes that inject requests with the average arrival rate of $\lambda = 0.23$ into the system, for every three fog nodes. In this way, we can easily extend the simulation setup by adding more fog nodes and more domains of IoT nodes at each step (for instance, 5 domains of IoT nodes with 15 fog nodes or 10 domains with 30 fog nodes). Fig. 10a to Fig. 10c depict the behavior of the proposed method by scaling the system in terms of averaged service time, averaged number of deadline misses and averaged brown energy consumption, respectively.

As it is understood from Fig. 10a the average service time slightly increases with scaling the system, because in a large system with more number of fog nodes and with regard to geo-distribution and heterogeneity of the nodes, we may have more available green energy at different time intervals which causes the controller dispatches more requests to such nodes. Consequently, we see it comes with some performance degradation. Therefore, from Fig. 10b, we observe that the number of deadline misses grows accordingly. Furthermore, by increasing the number of fog nodes more energy is consumed, as it can be observed from Fig. 10c. Increasing the number of fog nodes, however, provides the opportunity for better utilization of green energy sources, as it is demonstrated by Fig. 11. Therefore, the optimal fog size can be determined by considering the trade off between service time minimization and green energy utilization. Finally, from Fig. 10 and Fig. 11, we can observe that the proposed method outperforms the baseline schemes under different scales.

7.3. Evaluation under different conditions

In this section, we will provide further investigation to show how the proposed methods work under different conditions. We vary system parameters, such as arrival rate, computing and communication demands, based on the system queue model in Fig. 2, to see the behavior of the proposed methods.

7.3.1. Arrival Rate

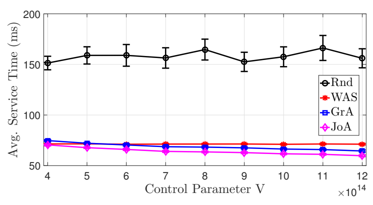
We conduct an experiment to evaluate the behavior of the proposed method against changes in the arrival rate. It enables us to test the proposed methods under more pressure. In this scenario, we increase the arrival rate to inject more workloads into the system and observe the metrics. The experiment contains an average of 24 hours simulation in 100 runs for each arrival rate in the range[8].

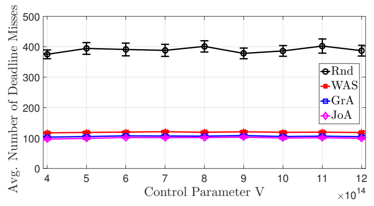
Injecting more workloads into the system causes larger queue backlog (Fig, 12), which leads the requests to wait for a longer time in the queues. Consequently, the requests encounter longer service time (Fig. 13a) which causes more requests to miss their deadlines, Fig. 13b. Furthermore, the energy consumption increases because there are more requests to be served and more demands for energy. Regarding that the green energy is not available all the time or may not be enough at some hours, it gets forced to draw more energy from the main grid which leads to an increment in brown energy consumption, Fig. 13c.

7.3.2. Computing demand

The computing demand determines how much processing a request needs. It is assumed that the computing demand follows the exponential distribution with average γ_1. In a different scenario, we made changes in γ_1 and assessed the behavior of the proposed methods under different processing loads.

[8]From 0.1 to 0.9 by step 0.1, for each step.


(a) Average Service Time


(b) Average Number of Deadline Misses

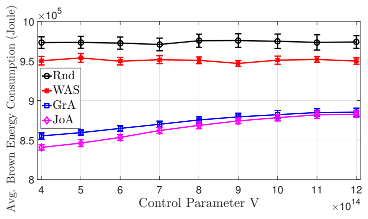
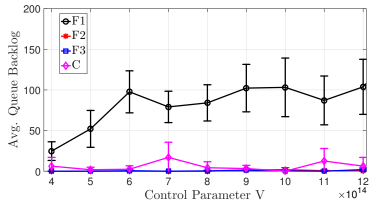

(c) Average Brown Energy Consumption

Figure 8: Behavior of the proposed algorithms with respect to control parameter V in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Brown Energy Consumption.


(a) Average Queue Backlog-JoA

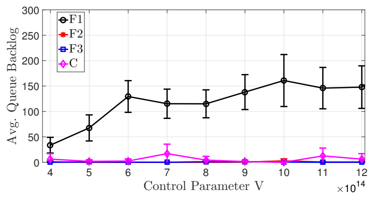

(b) Average Queue Backlog-GrA

Figure 9: Behavior of the proposed algorithms with respect to control parameter V in terms of Average Queue Backlog, in the case of (a) JoA and (b) GrA.

When heavy requests come to the system, more processing demand is injected into the system. Therefore, the queue backlog increases (Fig. 14) and leads to a longer service time (Fig. 15a), which results in more requests to miss their deadlines, Fig. 15b. On the other hand, more energy is consumed by the computing nodes and regarding that part of the consumed energy is provided by the brown energy sources, we see the consumption of brown energy is increased, Fig. 15c.

7.3.3. Communication demand

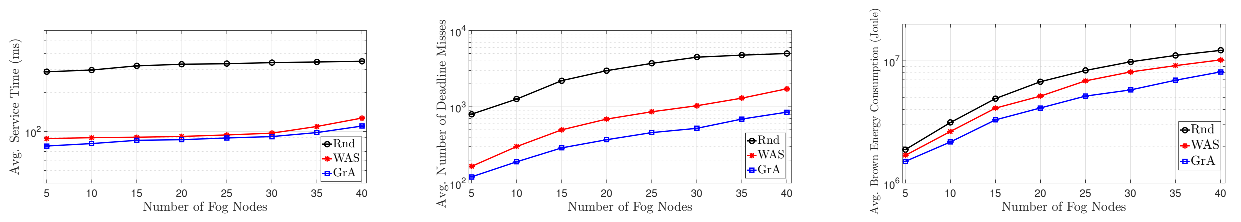
The communication demand determines the amounts of data that a request needs to communicate. It is assumed that the communication demand follows exponential distribution with average γ_2. To see how it can affect the behavior of the proposed methods, we made changes in γ_2 to inject requests with different communication requirements.

The proposed algorithms try to dispatch a request with larger amounts of data communication to the nearest computing nodes to lessen the burden on the network core. Regarding that such nodes are provided with less powerful resources, the service time increases, Fig. 17a. Therefore, the queue backlog increases accordingly, because the requests have to wait for a longer time, Fig. 16. Consequently, more requests miss their deadlines, Fig. 17b. On the other hand, with more requests get dispatched to the nearest nodes the demand for energy is increased, and it goes behind the capability of on-site renewable energy sources which results in an increment in the brown energy consumption, as observed from Fig. 17c.

8. Discussion

Remark 1: In this work, by leveraging the LOT, we proposed two online algorithms, JoA and GrA, as solutions for request dispatching problem. The methods are independent of the system dynamics and work based on the current system conditions and the queues' backlog information. The JoA exhaustively searches the requests' assignment space at each time slot and dispatches all the requests at once (near optimum solution), while GrA greedily dispatches the requests one by one. Request dispatching is a multi-objective problem. Therefore, in order to make a better comparison, we provide a multi-axis graph, the Kiviat diagram in Fig. 18, to show how different schemes perform with respect to each metric and against each other. The values on each axis are normalized with the biggest value. As it can be observed from Fig. 18, JoA outperforms other schemes. In particular, JoA outperforms WAS [4] up to 6%, 17% and 12% in terms of service time, deadline miss and green energy utilization, respectively. Furthermore, it is observed that GrA performs very close to JoA. In comparison to GrA, on average JoA performs 5.8%, 6% and 1.8% better in terms of service time, deadline miss and green energy utilization, respectively. Considering that GrA has much less computational complexity, it may be an attractive solution when scale matters.

Remark 2: The requests that come into the system are defined as some jobs with certain amounts of computing and communication demands. The controller makes its one-off decisions to allocate jobs to fog nodes based on the current situation of the fog nodes, network conditions and tasks' demands. Thus, the system never preempts or replaces the allocated jobs. However, if there exists a mechanism in the system to issue the performance degraded jobs as new ones, then, because in our strategy, we leverage the virtual queues, our ultimate goal turns into keeping the queues stable. Therefore, it may take several time slots for any change to be reflected in the queues' status. This prevents oscillations in our methods as it needs



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

Figure 10: Behavior of GrA with respect to the number of fog nodes in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Brown Energy Consumption.

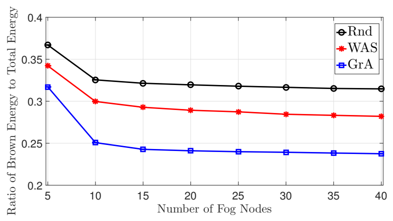


Figure 11: The ratio of Brown Energy Consumption to the Total Energy Consumption with respect to the number of fog nods.

to violate the thresholds.[9]

9. Conclusions and Future Work

There are a large number of computing nodes in the fog paradigm, which are geographically distributed over a wide area. Providing the required energy for this large number of nodes becomes challenging if they are all powered by the main grid. Utilizing renewable energy sources emerges as a feasible solution, which not only reduces the burden on the main power grid but also leads to lower CO$_2$ emission and environmental harms.

Considering the on-site renewable energy sources, we investigated the request dispatching problem to lessen the burden on the main power grid (by better utilizing renewable energy sources) while providing system stability and minimizing service time. The problem is of the typical problems that need dynamics of the system to be known in advance. In this paper, we proposed two online algorithms, one optimum (JoA) and one greedy (GrA), based on the Lyapunov optimization technique and the concept of virtual queues. The algorithms work only based on the current system condition and queues' backlog and do not require statistical information of system dynamics. The simulation results proved the efficiency of the proposed methods. The complexity and performance of the proposed algorithms were studied. The main result is that although JoA outperforms GrA, the performance of GrA remains very close to that of JoA with much less computational complexity. Therefore, GrA provides a better pragmatic solution for large scale systems. Our sensitivity analyses show that the proposed methods keep their efficiency under different conditions.

While our paper presents a comprehensive framework for request dispatching problem subject to the effective utilization of green energy and system stability, there are still many open problems to pursue in this new research direction. For instance, in this paper, we assumed that the controller has the updated information of the queues in the system, but the problem can be investigated under the condition of outdated information. Furthermore, we investigated the stability of the system asymptotically considering the long-term behavior of the system. The study of the transient behavior of the system is an interesting research area. Also, the use of LOT framework can be extended to more domains, to satisfy their special requirements. For example, applying learning schemes in many end-user devices and applications results in a huge surge in processing required for such applications at the edge. The proposed method of virtual queues can be used to meet the requirements of such requests by properly mapping them into the Lyapunov framework. Another future work might be consideration of storage or battery for the fog nodes.

Appendix A. Proof Proposition 1

Proof: Recall if $H(t)$ satisfies (14) then for all $t > 0$ we have:

$$H(t+1) - H(t) \geq x(t). \tag{A.1}$$

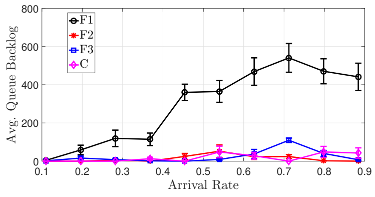
For any t_1 and t_2 such that $0 \leq t_1 < t_2$, summing (A.1) over $\tau \in \{t_1, \ldots, t_2\}$ and using the law of telescoping sums, yields:

$$H(t_2) - H(t_1) \geq \sum_{\tau=t_1}^{t_2-1} x(t). \tag{A.2}$$

Substituting $t_1 = 0$ and $t_2 = t$, and dividing both sides by t, yields:

$$\frac{H(t)}{t} - \frac{H(0)}{t} \geq \frac{1}{t} \sum_{\tau=0}^{t-1} x(\tau). \tag{A.3}$$

[9]The threshold in equation (15) and the definition of $y_i(t)$ (or in equation (16) and the definition $f(t)$) induces that a change in the system condition whenever can affect the queue status that pass its bound.


(a) Average Queue Backlog-JoA

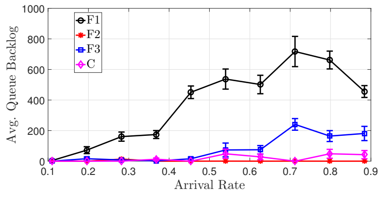
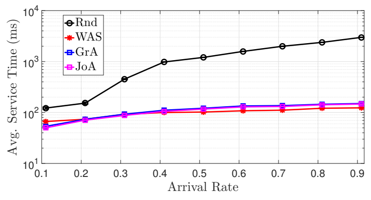
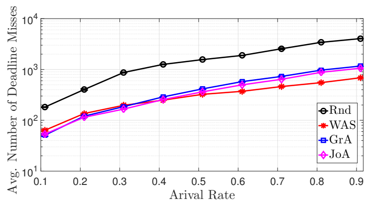

(b) Average Queue Backlog-GrA

Figure 12: Behavior of the proposed algorithms with respect to arrival rate (λ_1) in terms of Average Queue Backlog, in the case of (a) JoA and (b) GrA.


(a) Average Service Time


(b) Average Number of Deadline Misses

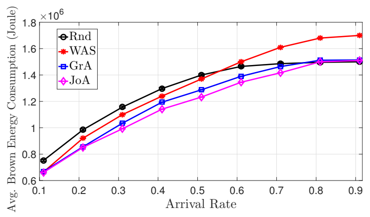
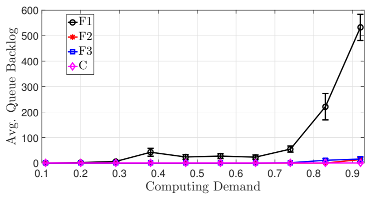

(c) Average Brown Energy Consumption

Figure 13: Behavior of the proposed algorithms with respect to arrival rate (λ_1), in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Brown Energy Consumption.


(a) Average Queue Backlog-JoA

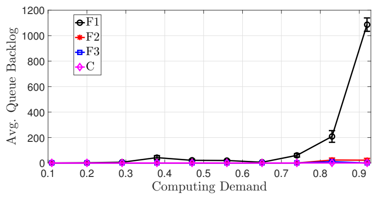
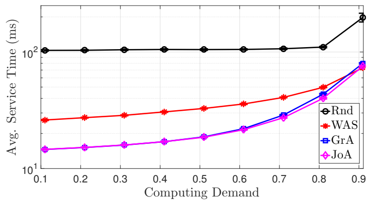
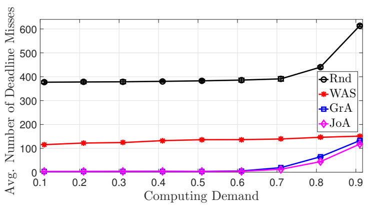

(b) Average Queue Backlog-GrA

Figure 14: Behavior of the proposed algorithms with respect to computing demand ($1-\gamma_1$) in terms of Average Queue Backlog, in the case of (a) JoA and (b) GrA.


(a) Average Service Time


(b) Average Number of Deadline Misses

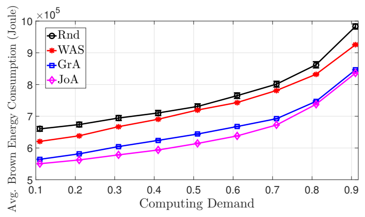

(c) Average Brown Energy Consumption

Figure 15: Behavior of the proposed algorithms with respect to computing demand ($1-\gamma_1$) in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Brown Energy Consumption.

Taking expectation of both sides of above inequality and taking lim sup as $t \to \infty$ yields:

$$\limsup_{t\to\infty} \frac{\mathbb{E}\{H(t)\}}{t} \geq \limsup_{t\to\infty} \bar{x}(t). \tag{A.4}$$

Thus, if $H(t)$ is stable, it means the left-hand-side of (A.4) is 0 and we have:

$$\limsup_{t\to\infty} \bar{x}(t) \leq 0. \tag{A.5}$$

This means the time average constraint $\bar{x}(t)$ is satisfied.

Appendix B. Proof Proposition 2

Proof: First, squaring both sides of the dynamic equation (1), using the fact that $\max[a-b,0]^2 \leq (a-b)^2$ and multiplying both sides by $\frac{1}{2}$, yields:

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

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

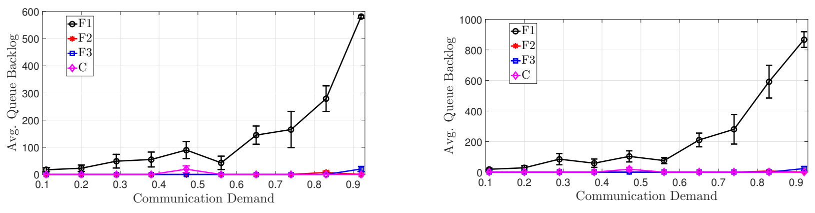
Do similarly for Z_i and G yields:

$$\frac{Z_i(t+1)^2 - Z_i(t)^2}{2} \leq \frac{y_i(t)^2}{2} + Z_i(t)y_i(t), \tag{B.2}$$

and

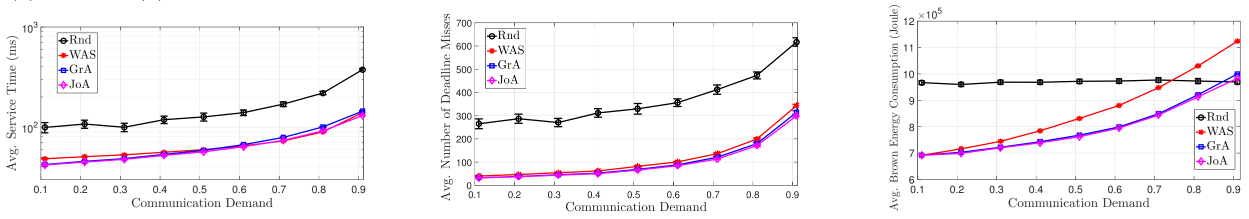
$$\frac{G(t+1)^2 - G(t)^2}{2} \leq \frac{f(t)^2}{2} + G(t)f(t). \tag{B.3}$$

Then, multiplying by ρ_1, ρ_2 and ρ_3 both sides of above three equations, taking conditional expectation with respect to $\boldsymbol{\theta}(t)$ and summing over $i \in \{1, 2, \ldots, N\}$ and



(a) Average Queue Backlog-JoA (b) Average Queue Backlog-GrA

Figure 16: Behavior of the proposed algorithms with respect to communication demand $(1-\gamma_2)$ in terms of Average Queue Backlog, in the case of (a) JoA and (b) GrA.



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

Figure 17: Behavior of the proposed algorithms with respect to communication demand $(1-\gamma_2)$ in terms of (a) Average Service Time, (b) Average Number of Deadline Misses, and (c) Average Brown Energy Consumption.

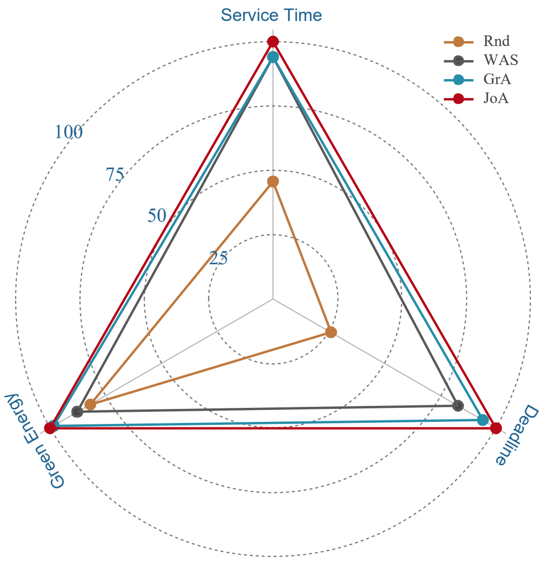


Figure 18: Overall comparison of the proposed methods.

$k \in \{1, 2, \ldots, K^t\}$ gives an upper bound on $\Delta(L(t))$ as follows:

$$\Delta(L(t)) \\ \leq \Psi + \rho_1 \sum_{i=1}^{N+1} Q_i(t)\mathbb{E}\{R_i(t)|\boldsymbol{\theta}(t)\} - \rho_1 \sum_{i=1}^{N+1} Q_i(t)B_i(t) \\ + \rho_2 \sum_{i=1}^{N+1} Z_i(t)\mathbb{E}\{y_i(t)|\boldsymbol{\theta}(t)\} + \rho_3 G(t)\mathbb{E}\{f(t)|\boldsymbol{\theta}(t)\}, \tag{B.4}$$

where $\Psi \geq \frac{1}{2}(\rho_1 \sum_{i=1}^{N+1} \mathbb{E}\{R_i(t)^2 + B_i(t)^2|\boldsymbol{\theta}(t)\} + \rho_2 \sum_{i=1}^{N+1} \mathbb{E}\{y_i(t)|\boldsymbol{\theta}(t)\} + \rho_3\mathbb{E}\{f(t)|\boldsymbol{\theta}(t)\} - \rho_1 \sum_{i=1}^{N+1}\mathbb{E}\{\tilde{B}_i(t) R_i(t)|\boldsymbol{\theta}(t)\})$.

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

References

[1] M. Chiang, T. Zhang, Fog and iot: An overview of research opportunities, IEEE Internet of Things Journal 3 (6) (2016) 854–864.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao, A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications, IEEE Internet of Things Journal 4 (5) (2017) 1125–1142.

[3] M. Mukherjee, L. Shu, D. Wang, Survey of fog computing: Fundamental, network applications, and research challenges, IEEE Communications Surveys & Tutorials 20 (3) (2018) 1826–1857.

[4] W. Li, T. Yang, F. C. Delicato, P. F. Pires, Z. Tari, S. U. Khan, A. Y. Zomaya, On enabling sustainable edge computing with renewable energy resources, IEEE Communications Magazine 56 (5) (2018) 94–101.

[5] F. Jalali, S. Khodadustan, C. Gray, K. Hinton, F. Suits, Greening iot with fog: A survey, in: 2017 IEEE International Conference on Edge Computing (EDGE), IEEE, 2017, pp. 25–31.

[6] Y. Li, A.-C. Orgerie, I. Rodero, B. L. Amersho, M. Parashar, J.-M. Menaud, End-to-end energy models for edge cloud-based iot platforms: Application to data stream analysis in iot, Future Generation Computer Systems 87 (2018) 667–678.

[7] E. K. Markakis, K. Karras, N. Zotos, A. Sideris, T. Moysiadis, A. Corsaro, G. Alexiou, C. Skianis, G. Mastorakis, C. X. Mavromoustakis, et al., Exegesis: Extreme edge resource harvesting for a virtualized fog environment, IEEE Communications Magazine 55 (7) (2017) 173–179.

[8] Y. Li, A.-C. Orgerie, I. Rodero, M. Parashar, J.-M. Menaud, Leveraging renewable energy in edge clouds for data stream analysis in iot, in: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), IEEE, 2017, pp. 186–195.

[9] Í. Goiri, R. Beauchea, K. Le, T. D. Nguyen, M. E. Haque, J. Guitart, J. Torres, R. Bianchini, Greenslot: scheduling energy consumption in green datacenters, in: SC'11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2011, pp. 1–11.

[10] M. J. Neely, Stochastic network optimization with application to communication and queueing systems, Synthesis Lectures on Communication Networks 3 (1) (2010) 1–211.

[11] L. Ni, J. Zhang, C. Jiang, C. Yan, K. Yu, Resource allocation strategy in fog computing based on priced timed petri nets, ieee internet of things journal 4 (5) (2017) 1216–1228.

[12] A. Yousefpour, G. Ishigaki, R. Gour, J. P. Jue, On reducing iot service delay via fog offloading, IEEE Internet of Things Journal 5 (2) (2018) 998–1010.

[13] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, 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 4 (5) (2017) 1204–1215.

[14] R. Deng, R. Lu, C. Lai, T. H. Luan, H. Liang, Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption, IEEE Internet of Things Journal 3 (6) (2016) 1171–1181.

[15] P. G. V. Naranjo, E. Baccarelli, 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 74 (6) (2018) 2470–2507.

[16] P. Kochovski, P. D. Drobintsev, V. Stankovski, Formal quality of service assurances, ranking and verification of cloud deployment options with a probabilistic model checking method, Information and Software Technology 109 (2019) 14–25.

[17] Y. Wang, K. Wang, H. Huang, T. Miyazaki, S. Guo, Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications, IEEE Transactions on Industrial Informatics 15 (2) (2018) 976–986.

[18] S. Zhao, Y. Yang, Z. Shao, X. Yang, H. Qian, C.-X. Wang, Femos: Fog-enabled multitier operations scheduling in dynamic wireless networks, IEEE Internet of Things Journal 5 (2) (2018) 1169–1183.

[19] W. Fang, X. Yao, X. Zhao, J. Yin, N. Xiong, A stochastic control approach to maximize profit on service provisioning for mobile cloudlet platforms, IEEE Transactions on Systems, Man, and Cybernetics: Systems 48 (4) (2016) 522–534.

[20] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, A. Paulraj, Distributed online optimization of fog computing for selfish devices with out-of-date information, IEEE Transactions on Wireless Communications 17 (11) (2018) 7704–7717.

[21] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, J. Wang, Debts: Delay energy balanced task scheduling in homogeneous fog networks, IEEE Internet of Things Journal 5 (3) (2018) 2094–2106.

[22] L. Chen, P. Zhou, L. Gao, J. Xu, Adaptive fog configuration for the industrial internet of things, IEEE Transactions on Industrial Informatics 14 (10) (2018) 4656–4664.

[23] W. Fang, X. Yin, Y. An, N. Xiong, Q. Guo, J. Li, Optimal scheduling for data transmission between mobile devices and cloud, Information Sciences 301 (2015) 169–180.

[24] J. Kwak, Y. Kim, J. Lee, S. Chong, Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems, IEEE Journal on Selected Areas in Communications 33 (12) (2015) 2510–2523.

[25] H. Wu, K. Wolter, Dynamic transmission scheduling and link selection in mobile cloud computing, in: International Conference on Analytical and Stochastic Modeling Techniques and Applications, Springer, 2014, pp. 61–79.

[26] H. Zhang, Z. Chen, J. Wu, Y. Deng, Y. Xiao, K. Liu, M. Li, Energy-efficient online resource management and allocation optimization in multi-user multi-task mobile-edge computing systems with hybrid energy harvesting, Sensors 18 (9) (2018) 3140.

[27] G. Zhang, Y. Chen, Z. Shen, L. Wang, Distributed energy management for multi-user mobile-edge computing systems with energy harvesting devices and qos constraints, IEEE Internet of Things Journal.

[28] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, P. D. Drobintsev, Trust management in a blockchain based fog computing platform with trustless smart oracles, Future Generation Computer Systems 101 (2019) 747–759.

[29] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, R. S. Tucker, Fog computing may help to save energy in cloud computing, IEEE Journal on Selected Areas in Communications 34 (5) (2016) 1728–1739.

[30] Y. Sahni, J. Cao, S. Zhang, L. Yang, Edge mesh: A new paradigm to enable distributed intelligence in internet of things, IEEE access 5 (2017) 16441–16458.

[31] U. Paščinski, J. Trnkoczy, V. Stankovski, M. Cigale, S. Gec, Qos-aware orchestration of network intensive software utilities within software defined data centres, Journal of Grid Computing 16 (1) (2018) 85–112.

[32] G. C. Walsh, H. Ye, L. G. Bushnell, Stability analysis of networked control systems, IEEE transactions on control systems technology 10 (3) (2002) 438–446.

[33] C. Pukdeboon, A review of fundamentals of lyapunov theory, The Journal of Applied Science 10 (2) (2011) 55–61.

[34] M. J. Neely, Energy optimal control for time-varying wireless networks, IEEE transactions on Information Theory 52 (7) (2006) 2915–2934.

[35] Bureau of meteorology.
URL `http://www.bom.gov.au/climate/how/newproducts/IDCJAD0111.shtml`

[36] Photovoltaic education network.
URL `http://pveducation.org/`