Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 6215454, 17 pages
https://doi.org/10.1155/2019/6215454

Research Article

Task Placement on Fog Computing Made Efficient for IoT Application Provision

Minh-Quang Tran, Duy Tai Nguyen, Van An Le, Duc Hai Nguyen, and Tran Vu Pham

Ho Chi Minh City University of Technology, VNU-HCM, Vietnam

Correspondence should be addressed to Minh-Quang Tran; quangtran@hcmut.edu.vn

Received 23 July 2018; Revised 23 November 2018; Accepted 20 December 2018; Published 10 January 2019

Guest Editor: Antonio Moschitta

Fog computing is one of the promising technologies for realizing global-scale Internet of Things (IoT) applications as it allows moving compute and storage resources closer to IoT devices, where data is generated, in order to solve the limitations in cloud-based technologies such as communication delay, network load, energy consumption, and operational cost. However, this technology is still in its infancy stage containing essential research challenges. For instance, what is a suitable fog computing scheme where effective service provision models can be deployed is still an open question. This paper proposes a novel multitier fog computing architecture that supports IoT service provisioning. Concretely, a solid service placement mechanism that optimizes service decentralization on fog landscape leveraging context-aware information such as *location, response time, and resource consumption of services* has been devised. The proposed approach optimally utilizes virtual resources available on the network edges to improve the performance of IoT services in terms of response time, energy, and cost reduction. The experimental results from both simulated data and use cases from service deployments in real-world applications, namely, the intelligent transportation system (ITS) in Ho Chi Minh City, show the effectiveness of the proposed solution in terms of maximizing fog device utilization while reducing latency, energy consumption, network load, and operational cost. The results confirm the robustness of the proposed scheme revealing its capability to maximize the IoT potential.

1. Introduction

The Internet of Things (IoT) has emerged as a revolutionary technology that offers a fully connected "smart" world that accelerates the 4th industrial revolution where thousands or millions of things in the physical world are connected with each other. These things share data and services to specify, monitor, and manage the physical world thereby smart city, healthcare, agriculture services, and applications are enabled to transform the way we work, play, and live, improving the quality of life and the human civilization.

Realization of IoT services in a large scale, however, is hindered due to the constraints of IoT devices (embedded on everyday objects such as consumer goods, enduring products, vehicles, utility components, sensors, and other physical devices) in terms of computing resources, memory capacity, energy, and bandwidth limitations. Many of these issues could be resolved by employing the Cloud-Assisted Internet of Things or Cloud-of-Things (CoT) technology as it offers large-scaled and on-demand networked computing resources to manage, store, process, and share IoT data and services [1].

Nevertheless, the CoT paradigm is facing increasing difficulties to handle Big Data generated by IoT services associated with beyond billions of connected devices. As these devices are frequently (e.g., in every second or even shorter intervals) generating data, a large amount of data is generated every moment (exabytes of data per day). If every IoT captured data pattern is transferred to data centers (DCs) on the cloud for processing and storage, and then another large amount of information is returned to users or to actuators on the physical world, a huge volume of traffic is pumped into the network making it congested or malfunctioned. Obviously, this process challenges systems' performance and robustness in terms of ensuring low latency

and network bandwidth consumption, optimal utilization of computational resources, and scalability.

In fact, most of IoT data and services are generated and consumed by local users. Therefore, to cope with the aforementioned issues, a recent trend is to devise effective edge computing infrastructures, termed as *Edge-of-Things (EoT) computing*, *edge computing,* or *fog computing* [2]. Fog computing allows moving compute and storage resources closer to IoT devices where data is generated. Fog computing devices could be smart gateways or routers deployed at the network edge, local PCs, and even mobile devices such as smartphones or tablets carried by users that can offer computing and storage capabilities. These devices play their own roles of determining what data should be stored or processed locally (for low latency and saving network bandwidth) and what needs to be sent to the cloud for further analysis. It is clear that EoT complements CoT paradigm in terms of providing high scalability, low delay, and location awareness and leveraging local resources which are available at the network edges.

Although the benefit of fog computing in the IoTs is clear and the basic ideas of this computing paradigm have been well stated in various researches [3, 4], there still lacks systematical modeling for practical fog computing frameworks and effective service placement approaches to maximize the utilization of existing devices on the fog landscape while satisfying application response times and reducing energy and operation cost. In this article, we propose a novel approach to **task placement on fog computing made efficient for IoT application provision**. The main contributions of this work are summarized as follows.

(i) We propose a systematical fog computing framework consisting of multiple intelligent tiers that effectively support IoT service decentralization.

(ii) We devise an efficient task (service) placement approach to optimizing service decentralization on fog computing landscape leveraging context-aware information such as *location, compute and storage capacities of fog devices, and expected deadline of an application and,* hence, maximize the utilization of fog devices that are available at the network edge, and minimize the latency, energy consumption, and cost.

(iii) We conduct a thorough feasibility and performance analysis with various simulations. The results provide a comprehensive understanding of the effectiveness of the proposed approach in terms of maximizing the utilization of fog devices while reducing latency, energy consumption, and network load. In addition, the experimental results with service deployment in real-world applications that we have built for a smart city ecosystem in Ho Chi Minh City such as the intelligent transportation system (ITS) show the feasibility of the proposed solution. These results demonstrate the robustness of the proposed scheme revealing its capability to maximize the IoT potential.

The rest of the paper is organized as follows. Section 2 reviews related work. The overall architecture and problem definition are described in Section 3. The proposed service placement mechanism is presented in Section 4. Section 5 describes the evaluation of the proposed approach while Section 6 concludes the paper.

2. Related Work

Most of the existing IoT related projects assume the availability of centralized data centers based on a cloud-centric approach [5]. A typical example is described in [6] which addresses necessary components of a cloud-centric IoT architecture. The authors proposed a federation between a private cloud (e.g., Aneka their own system) and a public cloud (e.g., Microsoft Azure cloud) to efficiently handle sensing data from wireless sensor networks (WSNs). In the networking aspects this approach focused on access networks while ignoring the core networks. Consequently, it could not satisfy the effectiveness required in IoT as global services are mainly computed on the cloud, and transmitted and managed over the core networks.

There are several works to reinforce the shortage of cloud-centric IoT approaches by employing localization of computing resource [2, 3, 7]. The work in [7] describes a Locality-Based Utility Computing (LUC) Operating System. These utilities are distributed over the network backbones, such as local servers connecting to routers, aiming to provide the computing resources. The motivation of this work is similar to our present paper which is, instead of building bigger and bigger data centers and oversized networks, we should bring resources near to the network edges or users, where the IoT data is created. Unfortunately, the work in [7] had not utilized context-aware information such as location awareness and other related information to improve its effectiveness but keeps location awareness for future work.

Deploying miniclouds or private clouds on the network edges to handle IoT data processing and service provisioning has extracted numerous researches [8, 9]. Having small clouds or microdata centers at the edges can help to efficiently deliver services closer to users, hence mitigating traffic bombing on the network and reducing communication cost. In order to direct research to a more standardized approach, Cisco proposed the **fog computing** concept in a seminal study in [2]. Fog computing is a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional cloud computing data centers, typically but not exclusively located at the edge of network.

Several studies such as [8, 10] discuss the challenges, potential applications, and benefits of fog computing. It is considered a complementary technique to cloud computing that provides the missing links in the cloud-to-thing continuum in the IoT paradigm [11, 12]. Studies in [13] analyze the essential roles of fog computing for extending continuous links of IoT data and services from the cloud to the network edges. Meanwhile, studies in [14] propose architectural imperatives for fog computing and analyze use cases, requirements, and architectural techniques for fog-enabled IoT networks.

Although there are several proposals that help researches on fog computing converge, to some extent, to standardization. Fog computing is still in its infancy stage containing inherent difficulties that need to be thoroughly investigated. For instance, what is a suitable fog computing scheme where effective and efficient service provision models can be deployed is still an open challenge.

There are numerous studies in the area of resource provisioning in distributed environments such as in cloud computing [15, 16] and mobile cloud computing [17, 18]. Although service provisioning problems in fog computing share similar concepts and research issues with virtual machine placement problems in edge networks as discussed in [19, 20], the existing approaches cannot be directly applied to fog computing. One of the reasons is that fog landscapes are usually more volatile compared to those of cloud environments, hence more context-aware information around fog landscapes and Things should be utilized to effectively adopt with the dynamic change of large-scaled IoT environments.

In order to resolve the difficulties discussed above, several recent researches have been dedicated to resource allocation problems in edge and fog computing by investigating various imperatives [14]. A study in [21] proposes a fog computing platform where software modules are dynamically deployed on end devices (Things), while the study in [22] introduces a model that effectively allocates computing resources on network edges to process local and regional data. In another direction such as in [23], the authors introduce solutions for QoS-aware service allocation in fog computing as a basic optimization problem. More recent work in [24, 25] investigates a conceptual framework for the service provisioning problem in fog computing. These studies propose interesting concepts of fog cells as software modules running on IoT nodes and orchestrating models of these cells for handling services.

Our proposed fog computing scheme in this present paper is closely related to Cisco's Fog computing model [2]. One of the major differences in our work compared to existing work discussed above is that the computing resources in our newly proposed approach are more flexibly designed and allocated/distributed in accordance with context-aware information, specifically *location, network condition, type of service, and quality of the service (QoS, in terms of expected response time)*. Concretely, location identifies the location of IoT data sources or sinks which help to effectively determine where on the fog landscape a particular service should be deployed, in accordance with network conditions and the constraint of the service type. This context is useful for optimally distributing services on the fog, in terms of maximizing virtualized resources available along network edge, while satisfying the expected response time of applications, another context required by users/consumers.

There are also references on the passive microdatacenter, that is, rechargeable datacenter (e.g., using solar energy) at a very small scale [26]. This datacenter concept can be extended and applied in our proposed architecture where not only energy but also communication delay can be reduced by reasonably deploying compute and storage services such as computational power, data organization and indexing, and functional computing on these datacenters. However, as mentioned before, the existing work on fog computing has not effectively utilized context-aware information for optimizing IoT resources to maximize its potential [7].

In this present work, we thoroughly extend our previous study on task placement on fog landscapes [27]. Concretely, we analyze context-aware information that is necessary for effectively provisioning fog services. As a result, resource allocation is effectively conducted improving the quality of services such as reducing communication latency and mitigating network load, while saving energy and deployment costs significantly. In addition, we provide a thorough analysis on experimental results with service deployment in real-world applications for a smart city ecosystem in Ho Chi Minh City such as the ITS to confirm the effectiveness and the robustness of the proposed scheme.

3. Context-Aware Multitier Fog Computing Scheme

This section presents the proposed context-aware multitier fog computing architecture for IoTs where necessary context information and concepts related to task placement in the proposed scheme are described.

3.1. Context-Aware IoT Service Provision. As discussed, context information could be useful for IoT service provision; meanwhile context-aware solutions for effectively utilizing IoT resources have not been thoroughly addressed in the existing technologies [7]. This section describes fundamental context used in the proposed multitier fog computing scheme.

The notion of *context* has been observed in numerous areas including linguistics, knowledge discovery, artificial intelligent, information retrieval, reasoning, and theory of communications [28, 29]. As a high level of abstraction, *context* is defined as *"that which surrounds, and gives meaning to, something else."* In this definition *"something"* can be an artifact, a building, a person, a computer system, or even an assertion in logic as *"context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"* [30].

In the IoT environment, context includes but is not limited to *location, network condition, and type of service, quality of service*. In this work, context information used for service placement is described as follows.

(i) Location: identifies the location of IoT devices/users where IoT data is generated or consumed. This context is more useful for service placement when it is associated with network topology and computation scheme (e.g., fog/edge computing scheme) in the proposed multitier model presented in the next section.

(ii) Network Condition: describes the current network condition such as topology and resource (computation, storage,...) available at each node and communication quality (delay and packet loss) between nodes.

(iii) Type of Service: specifies service's features such as *sensing, actuating, computing, and storing* which define the possibility of deploying a particular service on a specific node/device based on its resource availability (e.g., a storing service cannot be conducted at a light-weight sensor without storage capacity).

(iv) Quality of a Service (QoS): describes the expected QoS when users/consumers receive a service upon their request. This work utilizes the *response time* as the indicator for QoS.

Obviously, the above context information is relevant for optimization of data/service placement, allocating resources (computing, storage, and communications) in the IoT where there are a large number of devices and a huge amount of complicated services. However, it could be difficult to apply this concept of context-aware IoT service provision in the conventional Internet architecture. To realize this approach, we proposed a new computing scheme, namely, the multitier fog computing architecture presented in the next section.

It is worth to be noticed that the context is dynamically changed, specifically in the IoT environment. How to effectively deal with dynamic context complying with the requirements of applications, especially new arrival ones, is an open challenge. It is not always good if the service placement solutions immediately change upon any change of the context and requirements as the service placement algorithms need to back-and-forth scan the environment condition. On the other hand, if the context information is not affected in time, the context could be useless. In order to overcome this dilemma, the proposed service placement method evaluates context and resolves the service placement problems in appropriate time period, namely, *placement turn* as presented in Section 4.3.1. Here, the duration of a turn is dynamically estimated by the response time of all tasks assigned in the current turn at a particular node. New arrival nodes will be stored in a waiting queue which is processed in the next turn with updated context information.

3.2. Overall Architecture. The natural features of an IoT system are its complicated connections between a huge number of devices while the provision of data and services is specific to application domains (e.g., healthcare, agriculture, traffic,...). IoT devices are distributed almost everywhere in the physical world to which data and services are mostly generated and consumed by local users. In some other cases these services are consumed by global users via cloud computing paradigm. In general, sensor systems, distributed service computing elements are connected with each other via intermediate connection elements (e.g., local servers) and global computing elements on the cloud in order to resolve the challenges on scalability, flexibility, and domain specific on IoT. However, in which way these distributed systems can be efficiently managed in the current infrastructures, specifically the current Internet which is not originally designed to support for the distributed computing with high flexibility in the IoT environments, is an essential research question.

In this article, we firstly introduce a novel **multitier architecture for the IoT**, namely, the **3-tier architecture**, and then we propose a novel approach to **task placement on fog computing made efficient for IoT service provision**.

In the 3-tier architecture, IoT elements such as sensors, mobile phones, laptops, vehicles, base stations, local servers, network connection, and management elements are connected in a multitier distributed scheme consisting of different levels of intelligence: *device/group-of-devices tier, regional tier*, and *global tier* as depicted in Figure 1 and described as follows.

(i) *Device/Group-of-Device Tier* includes IoT distributed services (DSs). This tier manages distributed services generated by things such as wireless sensors, vehicles, and mobile devices connected with each other via ad-hoc or P2P modes. These services can be useful for local users such as mobile users surrounding IoT devices in the DS sites. Examples for these services include advertisements sent to mobile users when they pass a favorite restaurant at a department store, a warning on overspeed sent to a driver, and so on.

(ii) *Regional Tier* consists of IoT services that are computed at a fog colony on the fog landscape. Each fog colony consists of a fog orchestration node serving as a service fog endpoint (SFE) and several fog cells. SFEs provide services/contents that could not be found from DSs. SFEs also serve as intermediate processing nodes used for data preprocessing or data integrations, for example, before being forwarded to DCs on the cloud for further computations. This scheme not only mitigates communication and computation costs but also helps to reduce latency of local-based services for local users. In addition, it enables us to provide services which are best fit with the local context. For instance, in a smart IoT-based heart disease monitoring system [31], the system collects patients' ECG and heart beat signals (via wearable devices) to detect abnormality (e.g., heart attacks) and provide healthcare services. The device itself can detect certain abnormalities based on some thresholds on ECG or heart beat signals. However, more sophisticated detections such as those based on machine learning approaches using historical data could not be conducted at IoT devices because of their computation and storage limitations. The proposed scheme can introduce appropriate regional services at suitable clinics, in terms of specialty and capacity of analyzing patients' data (context about service types), which are close to users/patients (context about location). Regional services at the recommended clinic analyze patients' ECG and heart beat signals to advice for on-site treatments and prepare necessary equipment as well as medical practitioners for registering patients when they are hospitalized at the clinic.

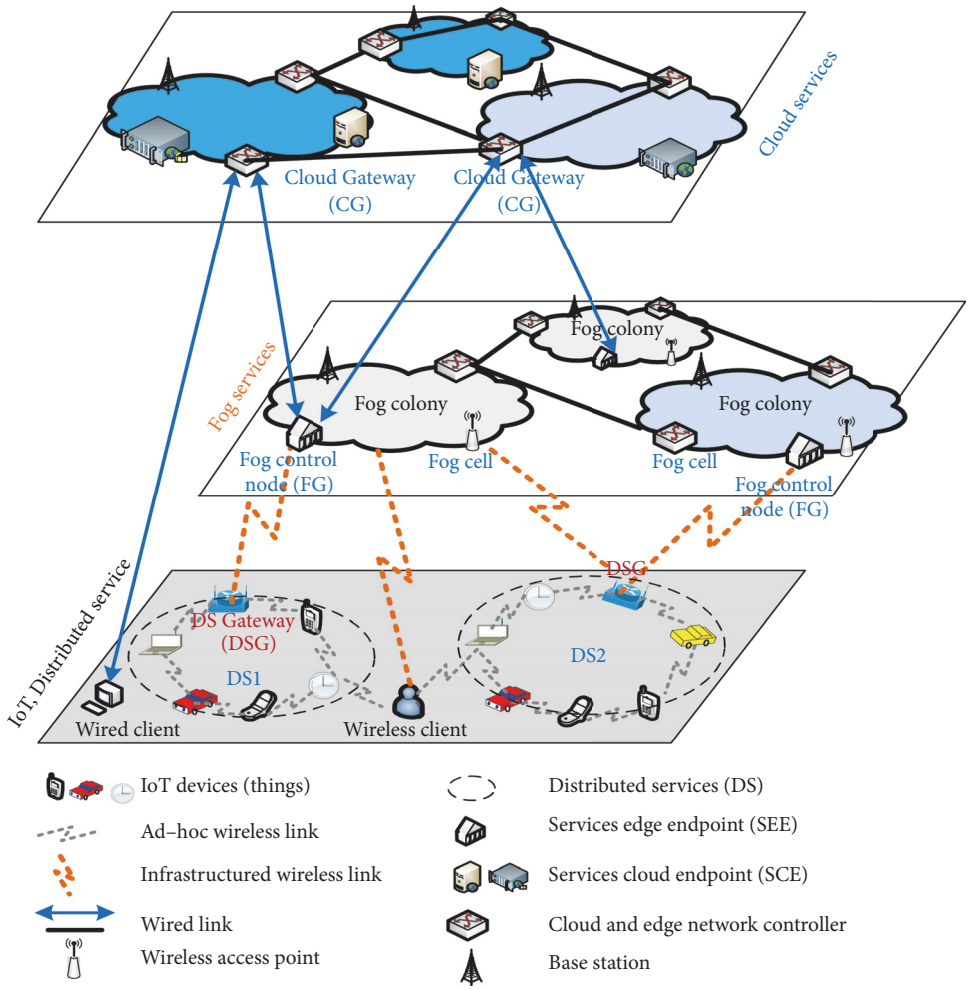


Figure 1: A multitier architecture for the Internet of Things.

(iii) *Global Tier* provides global IoT services or cloud services which are computed/integrated at centralized DCs or service cloud endpoints (SCEs). SCEs collect data from multiple SFEs or even from multiple DSs and provide global services to global users. These global services are adequate with common context to a specific application. For example, in a healthcare system, in accordance with flu symptoms such as high fever, headache, nausea, etc., reported by a user, besides introducing appropriate clinics nearby (by SFEs) as mentioned before, this data is also sent to a preventive healthcare center for further analysis. If this is a transmissive flu, the center will immediately provide related information and instructions to the community to prevent the flu spreading. This is a global service computed on the cloud (at SCEs) using global data such as *medical dictionary, descriptive data about epidemic, and data supplied by users*.

3.3. Applications and Services in the 3-Tier Architecture. In order to understand the usage of the proposed 3-tier architecture, we clarify concepts related to IoT applications and services as follows.

(1) *An IoT Application* is a concrete application on an IoT environment that provides data, information, or actuating functions to the requesting clients from the Internet. An application is a set of services (or tasks) described as follows.

(2) *A Service* is considered as the smallest component that processes a concrete task. A task can be classified in one of the following types: *sensing, actuating, computing, and storing*. The task type will limit the possibility to deploy it on a particular node/device (e.g., a storing service cannot be conducted at a light-weight sensor without storage capacity). In this article, the terms **service** and **task** are used interchangeably.

(3) *Service Provider* is any device in the 3-tier architecture that provides the execution of a task upon a corresponding request from a client. For example, IoT devices (Things) are service providers for sensing or actuating services, whereas providers for computing

services can be IoT devices, fog devices, and DCs on the cloud, and providers for storing services could be fog devices and DCs on the cloud.

(4) *Client* is a user application that issues application requests.

We assume that the code for the available IoT applications and services is already loaded to the corresponding devices (providers) in the 3-tier architecture. The architecture is not aware of computation details done by applications or services. It just knows what type of providers needed to satisfy client requests (defined by sensing actuation and context properties), the overlay topology of the providers, and the resources required by each service with the constraint of service's response time. Assuming that applications and services are predefined and registered to SCEs or SFEs which are front-end points for clients to request particular applications. The procedure generated when a client, *c*, requests an application, *a,* is described as follows:

(i) *c* asks its associated SCE (or even an SFE on its closest fog instance).

(ii) The involved SCE or SFE will identify the tasks (services) associated with the requested application and recognize corresponding providers by running the *task placement method* to optimize the utilization of virtualized resources in the fog.

(iii) The involved providers (i.e., devices where tasks are deployed) execute the corresponding tasks and provide the results (processed data/information, informing of the completion of an actuating/storing task,...) to the service front-end (i.e., SCE or SFE) for integration and then return the integrated results to the client *c*.

4. Task Placement on the Fog Landscape

An inherent issue in fog computing is how to optimize the utilization of virtualized resources on the fog landscape in order to not only mitigate the response time of requested applications but also reduce energy consumption and other operation costs. This section proposes a novel method for maximizing such available resources, in accordance with the current context, when deploying tasks in the proposed 3-tier architecture.

4.1. Problem Definition and System Notations. Given a set of applications $A_1, A_2, \ldots, A_m$ each of which is constrained with a deadline D_{A_k}, suppose that each application is composed of independent tasks (i.e., tasks can be executed simultaneously) (In IoT, the degree of dependencies between tasks could be complicated, we defer those issues to the future work. This work focuses on devising a 3-tier fog computing architecture combining with context information for task placement on the fog landscape.), $A_k = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Let $J = \bigcup_{i=1}^{m} A_k = \{\tau_1, \tau_2, \ldots, \tau_N\}$ be the set of tasks resulting from the decomposition of all applications that need to be deployed on the 3-tier network. This work aims at deploying the tasks mentioned above on the fog landscape and the cloud based on the current context.

As presented in Section 3.1, the four main contexts, namely, *location, network condition including network topology, nodes' resource availability and communication quality, service type, and QoS* (in terms of expected response time), are taken into account in the proposed task/service placement solution. In addition, the task placement approach must satisfy the two criteria as follows.

C1 (hard criteria): No application misses its deadline as described in equation (1), where R_{A_k} is the response time of A_k. Here, the context of expected response time and estimated execution time of tasks are taken into account:

$$R_{A_k} \leq D_{A_k}, \quad \forall A_k \tag{1}$$

C2 (soft/optimal criteria): The number of tasks deployed on the fog landscape is maximized.

In order to properly form and solve the optimized service placement problem, we need to devise estimation metrics for the fog landscape w.r.t the network architecture proposed in Section 3. We describe the functional components, their resource capability, communication delays, and energy consumption in the fog computing paradigm. Table 1 shows the notations and descriptions of terms used for task placement modeling in this paper.

4.2. Task Placement Modeling. This subsection proposes a practical task placement model on the fog landscape to maximize the utilization of already available virtualized resources at the network edges, reducing latency and energy consumption. It is worth to be noted that fog colony is the basic entity of fog landscape, each of which consists of a set of computational devices denoted as fog cells or fog nodes, f. Each fog colony is managed by a fog orchestration node, F, which is a fog cell with more powerful and extended functionality for *managing resources* and *controlling the task placement and execution*. Upon receiving application requests from clients, the corresponding fog orchestration node F is responsible for generating and deploying tasks over the system in accordance with two criteria (C1 and C2) mentioned in Section 4.1. Given a task τ_i, node F will determine to place it on one of the four places:

(i) On itself (i.e., on F)

(ii) On a fog cell on the colony managed by F, that is, any $f \in Res(F)$, where $Res(F)$ denotes a set of fog cells in F's colony that comply with service type required by τ_i

(iii) On its neighbor colony controlled by the orchestration node N (the details of task management and execution are delegated to N)

(iv) On the cloud denoted as R.

It is worth to be noted that the orchestration node F provides context information about network, capacity (CPU, RAM, communication quality, and computation type) of each fog cell in its colony, and the context about its neighbor colony's capacity at the placement time.

Let $x_{\tau_i}^f, x_{\tau_i}^F, x_{\tau_i}^N, x_{\tau_i}^R \in \{0, 1\}$ be binary variables telling whether the task τ_i is deployed on a fog node f ($x_{\tau_i}^f = 1$),

Table 1: Notation and description of terms in the proposed task placement model.

Notation \ Parameter	Notation	Description
Fog landscape	DS	Distributed services
	F	Fog orchestration node
	f	a Fog cell (node)
	f_i^k	Fog cell ith belongs to Fog colony k controlled by the fog orchestration node F^k
	N^k	The set of fog neighbors of F^k
	p	Node p $\in F$
	d^f	Link delay between a fog cell f and its colony's orchestration node F
	d^N	Link delay between a fog orchestration node F and its neighbor N
	d^R	Link delay between a fog orchestration node F and the cloud
	P_{max}^P	Maximum power consumption of node p in the fog landscape
	P_{idle}^P	Power consumption of node p when it is idle
Cloud	P_{idle}^R	Power consumption of a node on the cloud when it is idle
	w	Power consumption per instruction when executing tasks on the cloud
Application	A	Set of applications to be executed
	A_k	The application k
	τ_i	The task i
	J	The set of all tasks that need to be executed
	R_{A_k}	Response time of the application A_k
	D_{A_k}	Deadline of the application A_k
	w_{τ_i}	Deployment time of the task τ_i
	m_{τ_i}	Execution time (makespan) of the task τ_i
	d_{τ_i}	The communication time of the task τ_i
	a_i	The size (MIPS) of the task τ_i
	s_i	Memory required by the task τ_i
	D_i	The total amount of data exchanged when running the task τ_i

on the fog orchestration node F ($x_{\tau_i}^F = 1$), on the neighbor colony ($x_{\tau_i}^N = 1$), or on the cloud ($x_{\tau_i}^R = 1$). Since a task is deployed only once, the constraint in (2) is held:

$$x_{\tau_i}^f + x_{\tau_i}^F + x_{\tau_i}^N + x_{\tau_i}^R = 1, \quad \forall \tau_i \tag{2}$$

Since our purposed method is to maximize the number of tasks assigned on the fog landscape, with a given fog colony orchestrated by F, the objective function is formed in (3):

$$\max \sum_{i}^{N} \left(\alpha \sum_{f}^{Res(F)} x_{\tau_i}^f + \beta x_{\tau_i}^F + \gamma x_{\tau_i}^N \right) \tag{3}$$

where $0 \leq \alpha, \beta, \gamma \leq 1$ are coefficients defining the priority of task deployment on different types of computational entities (fog cell f, fog orchestration node F, neighbor colony N, or on the cloud R, respectively). This prioritizing helps to mitigate the computation time of the solver. These parameters could be determined by examining historical data or can be heuristically selected based on an intuition that a task should be tried at a fog cell or at the fog orchestration node before being propagated to the neighbor colony, and propagating to the cloud is the last choice. In this work, we set $\alpha = \beta = 1, \gamma = 0.5$ for evaluation without losing the generality of the proposed approach while presenting the utilization of location context (i.e., resources close to IoT data sources are utilized first). In addition, Res(F) describes context about network condition (topology, resource availability at each node, etc.) which is periodically updated at the beginning of a task placement turn.

Resolving the objective function in equation (3) provides an optimal placement plan that maximizes the number of tasks deployed on the fog landscape (i.e., near to data sources and better utilize the available virtualized resources). This plan satisfies the QoS constraint presented in equation (1) where every application is completed before a predefined deadline D_{A_k} under the available virtualized resources on the fog landscape. This means that, for a task τ_i which is planned to be deployed on a node p (fog cell, fog orchestration node), p must satisfy resources required by τ_i and all the tasks composing the application A_k must be completed before the application's deadline to satisfy the global constraint in equation (1).

Therefore, the hard constraint in this problem solving is that available resources in the deployment node such as computation power (CPU) and storage (memory) capacity must be adequate to process the requested tasks in time (i.e., the application consisting of tasks will be completed before its deadline). In addition, the priority to assign a task locally on the considering fog colony is significantly higher than that of assigning such a task to a neighbor colony or to the cloud. This can be seen as a soft constraint to maximize the utilization

of fog devices revealing the reduction of communication latency, energy consumption, and operational cost.

4.3. Response Time Estimation. As discussed before, the task placement plan provided by the proposed model must satisfy the hard constraint on the application response time (i.e., $R_{A_k} \leq D_{A_k}$ presented in equation (1)). The difficulty here is that how to appropriately model or estimate the response time of an application which consists of multiple tasks being deployed at different locations. This subsection addresses this issue by thoroughly estimating the time expended for application execution w.r.t the related resource constraints such as CPU power and memory capacity of available fog devices.

4.3.1. Estimating Response Time of a Task and an Application. Accomplishing a task τ_i requires four steps: *task submission, deployment, execution*, and *result return*. Therefore, the *response time* r_{τ_i} of such a task is calculated as follows:

$$r_{\tau_i} = w_{\tau_i} + m_{\tau_i} + d_{\tau_i} \tag{4}$$

where

(i) w_{τ_i} is the *deployment time* in which data and compute resources needed by the task are prepared.

(ii) m_{τ_i} is the *execution time* (or makespan time) in which the task actually utilizes resources on the deploying node for execution.

(iii) d_{τ_i} is the *communication time* consisting of (a) *task submission time* which is the time it takes to move necessary information from the fog orchestration node to the node where the task will be deployed and (b) *result return time* which is the time it takes to return the result to the fog orchestration node and release unused resources.

We assume that resources on the cloud are unlimited; hence when a task τ_i is submitted to the cloud, it is executed and finished immediately. Therefore, if a task is assigned to the cloud, its response time is composed of only the communication time (i.e., $r_{\tau_i} = d_{\tau_i}$).

Estimating the response time of a task running on a fog cell, in contrast, strongly depends on how the fog orchestration node distributes tasks to other nodes and the mechanisms each node uses to schedule task deployment and execution. We briefly describe those as follows.

Each fog node p runs tasks in multiple turns, namely, *1, 2,…, M*. At the beginning of a turn, the node loads *all* tasks assigned to it and deploys all of them. Once a task is deployed successfully, the node uses a part of its computation power to execute the task. Right after the task has been finished, the result will then be transferred back to its corresponding fog control node. When every task has been done, the node releases all resources and marks the current turn as *"finish."* After that, the node moves to the next turn, loads new tasks, and executes them, if there is any assignment.

It should be noted that a task must be completed within a *single* turn. It cannot be propagated across multiple turns. In addition, a node only deploys tasks at the beginning of a turn. Therefore, if a task is assigned to a node, this node does not start the task immediately but waits for the current turn to finish (tasks are put on the node's waiting queue). More formally, a node deploys new tasks if and only if:

(i) there exists at least one task in its waiting queue, and

(ii) all tasks in the previous turn have been finished and the node is ready for releasing resources for deploying new tasks.

Since a node loads all tasks assigned to it in each turn and tasks are executed concurrently, and all fog nodes are controlled (i.e., they can be synchronized) by the fog orchestration node, the response time of an application r_{A_k} can be estimated as follows:

$$r_{A_k} = \max_{\tau_i \in A_k} \left(r_{\tau_i}\right) \mathrm{m} = \max_{\tau_i \in A_k} \left(w_{\tau_i} + m_{\tau_i} + d_{\tau_i}\right) \tag{5}$$

Given this mechanism, the next subsection presents in detail the estimation of each time component, namely, $w_{\tau_i}, m_{\tau_i}, d_{\tau_i}$, based on the available CPU, memory, and communication resources of the destination node (where the task will be deployed on) and the corresponding resource requirement from the task.

4.3.2. Estimating Deployment Time ($w_{\tau i}$). In this work we assume that if a task is deployed on the cloud R, it can be deployed immediately as the resource on the cloud is always available on demand. Therefore, the deployment $w_{\tau i}$ of the task τ_i is accountable when τ_i is deployed on the fog landscape. This time is calculated in equation (6):

$$w_{\tau i} = T_N x_{\tau_i}^N + T_F x_{\tau_i}^F + T_f x_{\tau_i}^f \tag{6}$$

where

(i) T_N is the time the task spends on the waiting queue of the neighbor fog colony N, if it is assigned to the neighbor colony (i.e., $x_{\tau_i}^N = 1$)

T_F is the time the task spends on the waiting queue of the fog orchestration node F, if it is assigned to the fog orchestration node (i.e., $x_{\tau_i}^F = 1$)

(ii) T_f is the time the task spends on the waiting queue of a fog node f, if it is assigned to this node (i.e., $x_{\tau_i}^f = 1$).

In order to estimate T_N, the fog orchestration node F examines the historical data collected from N during previous execution turns. Supposing that at node N, m turns have passed, then T_N is calculated in equation (7):

$$T_N = \overline{T}_m - T_m^p \tag{7}$$

where

(i) T_m^p is the amount of time passed after the turn m had finished (i.e., the fog node in N has run its current turn for T_m^p time unit (s); see Figure 2 for more details).

(ii) $\overline{T}_m$ is the average duration (i.e., the total time a node spends on a turn for deploying, executing tasks and releasing resources) of a turn calculated from the previous m turns and is calculated by the following equation:

$$\overline{T}_m = \lambda T_m + (1 - \lambda)\, \overline{T}_{m-1}, \quad \lambda \in [0, 1] \tag{8}$$

where T_m is the waiting length (duration) of the turn m. λ reflects the sensitivity of the approximation. If λ approaches 1,

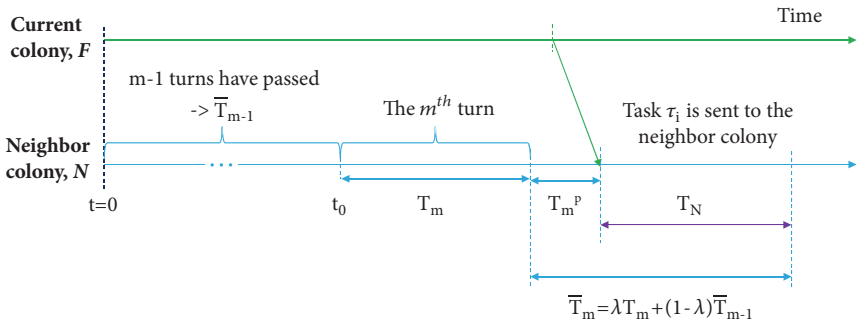


Figure 2: Estimation of the task's waiting time at the neighbor fog colony N, T_N.

the approximation tends to rely much on the recent duration of the last turn. On the other hand, if λ approaches 0, the approximation tends to rely on the previous approximation on historical data.

Obviously, T_F and T_f can also be calculated using this method. However, tasks are given beforehand, expected to be executed in a single turn, and there is no new task or application generated during the execution; it is reasonable to set $T_F = T_f = 0$. Consequently, equation (6) can be rewritten as in

$$w_{\tau i} = T_N x_{\tau_i}^N \tag{9}$$

4.3.3. Estimating Execution Time (m_{τ_i}). Execution time or makespan time $m_{\tau i}$ is the time required to execute the task τ_i given the CPU power of the node where the task is deployed. This time is calculated in the following equation:

$$m_{\tau_i} = x_{\tau_i}^F m_{\tau_i}^F + \sum_{f \in Res(F)} x_{\tau_i}^f m_{\tau_i}^f + x_{\tau_i}^N m_{\tau_i}^N + x_{\tau_i}^R m_{\tau_i}^R \tag{10}$$

where

$m_{\tau_i}^F, m_{\tau_i}^f, m_{\tau_i}^N$, and $m_{\tau_i}^R$ are the execution times if the task τ_i is deployed at the fog orchestration node F, fog node f under the control of F (e.g., $f \in Res(F)$), the neighbor fog colony N, and the cloud R, respectively. These times are calculated as follows.

Let p be the selected node where the task will be deployed; the execution time $m_{\tau i}$ is calculated in accordance with *compute capacity* (i.e., CPU power) of p, denoted as C^p, which is the maximum number of (million) instructions per second (or MIPS) that this node can spend on running tasks (other compute capacity which has to be used for other operations such as transferring data is set aside and will not be considered in here). It should be noted that if p is a node on the considering colony (i.e., F, f), the C^p is available to the considering orchestration node F. If p represents the cloud (i.e., R), in this case the task τ_i must be deployed on the cloud due to the time constraint; the cloud-based compute resource is required adequately to complete the task on time. Therefore, the orchestration node F is not responsible for estimating this resource. As a result, the execution time in equation (10) can be rewritten as in the following equation:

$$m_{\tau_i} = x_{\tau_i}^F m_{\tau_i}^F + \sum_{f \in Res(F)} x_{\tau_i}^f m_{\tau_i}^f + x_{\tau_i}^N m_{\tau_i}^N \tag{11}$$

Besides the *compute capacity* C^p, execution time of a task is also affected by the task size, denoted as $|\tau_i|$, which is measured by the number of (million) instructions to be executed (denoted as MI). This size is directly calculated by the fog orchestration node F and can be used as an explicit input for computation resource requirement. Our target is to reduce the execution time of the task as much as possible; thus the computation capacity of the node must be fully used. To do so, assume that at every turn m the node p has a set of tasks to execute $J_m^p = \{\tau_1, \tau_2, \ldots, \tau_n \mid x_{\tau_i}^p = 1\}$. We assign the computation capacity C^p proportionally to each task in accordance with tasks' sizes. More formally, the amount of computation capacity assigned to task τ_i is presented in the following equation:

$$C_{\tau_i}^p = \frac{|\tau_i|}{\sum_{\tau_j \in J_m^p} |\tau_j|} C^p \tag{12}$$

Consequently, the execution time of task τ_i is calculated in the following equation:

$$m_{\tau_i}^p = \frac{|\tau_i|}{C_{\tau_i}^p} \tag{13}$$

Since compute resource is proportionally divided to tasks based on their sizes, all tasks in the same turn (deployed on the same node) will have the same execution time regardless of their sizes. From (12) and (13) the execution time of a task can be rewritten in

$$\forall \tau_i \in J_m^p : m_{\tau_i}^p = \frac{\sum_{\tau_j \in J_m^p} |\tau_j|}{C^p} \tag{14}$$

Since a fog colony executes all tasks assigned to it within a single turn, we can clarify the execution time of a task when it is deployed at the orchestration fog node F, at a regular fog node f, and when it is propagated to the neighbor colony N as follows.

(i) Estimating the Execution Time $m_{\tau_i}^F$ and $m_{\tau_i}^f$, When τ_i Is Deployed on the Considering Colony. According to equation

(14), $m^F_{\tau_i}$ and $m^f_{\tau_i}$ can be clarified as in equations (15) and (16), respectively:

$$m^F_{\tau_i} = \frac{\sum_{\tau_j \in J} x^F_{\tau_i} \left|\tau_j\right|}{C^F} \tag{15}$$

$$\forall f \in Res\,(F) : m^f_{\tau_i} = \frac{\sum_{\tau_j \in J} x^f_{\tau_i} \left|\tau_j\right|}{C^f} \tag{16}$$

(ii) Estimating the Execution Time $m^N_{\tau_i}$ When the Task τ_i Is Deployed on the Neighbor Colony N. Obviously, $m^N_{\tau_i}$ cannot be directly calculated as the same way as the calculation for $m^F_{\tau_i}$ and $m^f_{\tau_i}$ mentioned above since the orchestration node F does not have information about other tasks that are currently running on the neighbor colony N. In order to overcome this difficulty, we proposed to estimate $m^N_{\tau_i}$ through historical data from previous executions on N. Suppose that there were k tasks executed on N previously, the $m^N_{\tau_i}$ can be estimated in

$$m^N_{\tau_i} = \overline{m^N_k} \tag{17}$$

where $\overline{m^N_k}$ is the average execution time of tasks from the 1st task to the kth task and is defined in

$$m^N_{\tau_i} = \mu m^N_k + (1-\mu)\,\overline{m^N_{k-1}}, \quad \mu \in [0,1] \tag{18}$$

where

(i) m^N_k is the execution time of the kth task which has been done at the colony N

(ii) $\overline{m^N_{k-1}}$ is the average execution time of tasks from the 1st task to the kth task

(iii) μ is a predefined parameter which reflects the weight of the execution time of recent moving tasks.

4.3.4. Estimating Communication Time (d_{τ_i}). In order to estimate the communication time we need to model the networked resources in our proposed 3-tier architecture. We denote the system as an undirected graph $G=(V,E)$, where V is the set of vertices representing the physical elements that consist of the cloud (R), fog orchestration node (F) of each fog colony, and fog cells (f), E is the set of physical connections between two vertices in the graph. Formally, the set of connections (links) in the network topology can be represented in

$$E = \bigcup_{k=1}^{K} E^f_k \cup E^N_k \cup E^R_k \tag{19}$$

where

$E^f_k = \{e^f_i =\prec b, d_{prop} \succ | i = 1, \ldots |res(F)^k|\}$ is the set of physical connections between the orchestration node F^k and the fog cells in the colony k. $|res(F)^k|$ is the number of fog cells in the considered colony k,

$E^N_k = \{e^N_i =\prec b, d_{prop} \succ | i = 1, \ldots |N^k|\}$ is the set of physical connections between the orchestration node F^k of the colony k and its neighbors. $|N^k|$ is the number of fog neighbors of the considered colony k,

$E^R_k = \{e^N =\prec b, d_{prop} \succ\}$ is the physical connection between the orchestration node F^k of the colony k and the cloud,

b is the maximum throughput of the considered link when data is transferred, and d_{prop} is the propagation delay of the considered physical link.

As a result, when the task τ_i is distributed from the orchestration node F^k to an appropriate node (i.e., F, f, N, or R), the communication time is calculated in

$$d_{\tau i} = 2\left(d^f_{\tau_i} x^f_{\tau_i} + d^N_{\tau_i} x^N_{\tau_i} + d^R_{\tau_i} x^R_{\tau_i}\right) \tag{20}$$

where

$d^f_{\tau_i}, d^N_{\tau_i}, d^R_{\tau_i}$ are one-way latencies when the task τ_i is deployed on f, N, or R, respectively. It should be noted that the communication time is the double of this one-way latency and the latency when the task is deployed on F (namely, $d^F_{\tau_i}$) is 0; hence it does not appear in equation (20).

For any p representing any type (f, N, or R), $d^p_{\tau_i}$ is calculated in

$$d^p_{\tau_i} = d^p_{\tau_i trans} + d^p_{\tau_i prop} + d^p_{\tau_i queue} = \frac{D_{\tau_i}}{b} + d^p_{\tau_i prop} \tag{21}$$

where

$d^p_{\tau_i trans}, d^p_{\tau_i prop}, d^p_{\tau_i queue}$ are the communication times of task τ_i caused by the transportation, propagation, and queuing times, respectively. This work assumes that the queuing time is 0.

D_{τ_i} is the amount of data (in Byte) of the task τ_i that need to be transferred on the considered link.

It should be noted that reaching this stage all the *deployment time, makespan time,* and *communication time* $(w_{\tau_i}, m_{\tau_i}, d_{\tau_i})$ of the task τ_i have been thoroughly estimated. They can be applied to equation (5) in Section 4.3.1 to estimate the response time of an application, r_{Ak}.

4.3.5. Memory Constraint. In order to deploy a task on an appropriate node p, the memory constraint must be satisfied as shown in

$$\sum_{\tau_j \in J^p_m} S_{\tau_i} \le S^p \tag{22}$$

where

S_{τ_i} is the memory required to executing the task τ_i (Bytes). It is assumed that this value can be estimated by the corresponding orchestration node F.

S^p is the memory capacity of node p (Bytes). It should be noted that the corresponding orchestration node F can examine S^f of any fog cell f belonging to its colony. Meanwhile, the orchestration node of the neighbor colony N must notify to F its available memory S^N. The memory on the cloud S^R always satisfies the required memory of the deployed tasks.

Table 2: Configuration of the general network topology.

Node name	Delay (s)	CPU capacity (MIPS)	Ram capacity (MB)	Power consumption (J, max, and idle)
f_0	2	250	16	200 100
f_1	2	250	16	200 100
f_2	2	250	16	200 100
f_3	2	250	16	200 100
F	0	350	1e3	600 300
N	10	350	1e3	600 300
R	50	1e3	8e3	1e3 300

Table 3: Summary of simulation scenarios.

Number of apps	No. tasks/app	Network topology		
1-6	1-8	4-7 nodes (1 F, 1 N, 1R and 1 - 4 f)		
Deadline (s) <min, mean, max>	Task CPU (MI)	Node f CPU (MIPS)	Delay (s)	Power consumption (J) <idle, max>
<10, 19, 29>	<100, 252, 399>	<205, 389, 581>	<1, 6, 10>	<200, 600>

5. Evaluation

The main purpose of the evaluation is to verify the effectiveness of the proposed approach in terms of reducing of latency, energy consumption, and network load, while increasing the utilization of virtualized resources available on the fog landscape, thus mitigating the operational costs in comparison with the conventional cloud computing model. We have conducted various experiments using simulations including those simulating task distribution problems in real world applications that we are building for a smart city system such as the ITS in Ho Chi Minh City.

5.1. Experiment Environment. In order to evaluate the optimized task distribution model presented in Section 4, we need to describe several experimental variables as follows.

(i) **Network topologies and resource description**: We use iFogsim [32] to generate various network topologies associated with their devices' resource description such as fog colonies, the orchestration node of a specific colony, fog cells in a fog colony associated with their CPU, memory capacities, and the network bandwidth as well as the propagation delays on each link between two physical nodes in the considered topology.

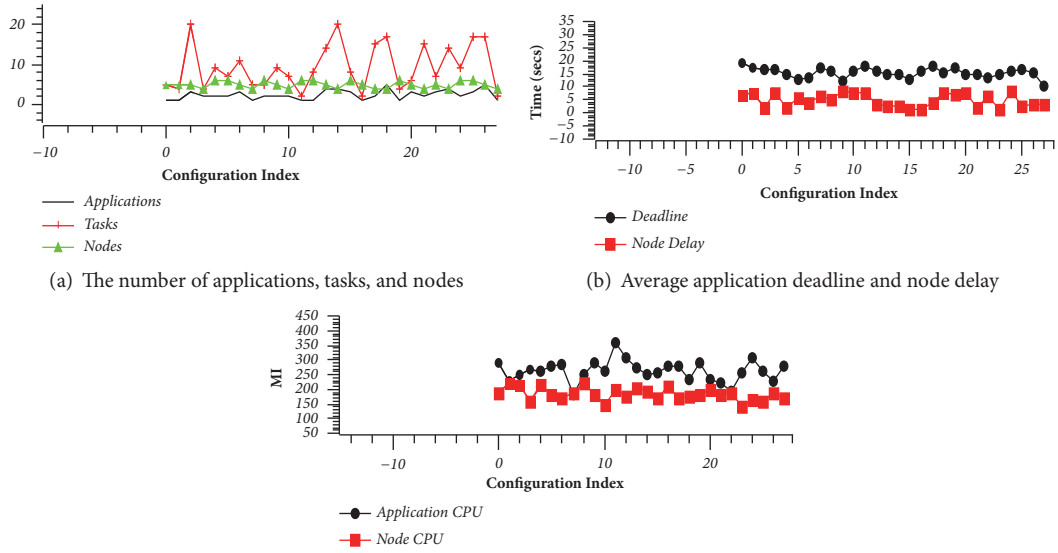
(ii) **Application composition and resource requirement by tasks**: In order to describe applications and tasks we have analyzed several real world applications such as the ITS that we are building for Ho Chi Minh City. We analyzed application composition (which tasks are composed in a specific application) and the resource required by each task (the memory and CPU capacities and the size of each task) as well as the reasonable deadline required by each application in the ITS applications. Concrete examples for these descriptions are presented in the next subsection that describes different application scenarios.

(iii) **Optimize task placement on the fog landscape**: In order to solve the optimization problem of task placement on the fog landscape proposed in Section 4, we implemented a program for optimization problem solving leveraging the IMB CPLEX solver [33]. The result is an optimal placement plan which is used to (i) analyze the network performance and (ii) redeploy on the considering network topology using iFogsim [32] to verify the network performances.

5.2. Experimental Results. As discussed, energy consumption reduction, fog device utilization, and cost saving as functions of application complexities (i.e., number of applications, number of tasks in each application, and resources required by each task) and network topologies (number of fog nodes and the capacity of each node) have been thoroughly evaluated through experiments on general scenarios and real world applications.

5.2.1. Evaluation on General Scenarios. In this section, we evaluate how the proposed approach works to provide task placement plans on different situations in terms of network topologies and applications (including tasks) to be processed on each network topologies. We have simulated 28 different network topologies (configurations), each of which has a structure as follows: one orchestration node F, several fog cells (f_i) controlled by F, one neighbor colony N, and a data center on the cloud. Table 2 depicts an example of a configuration that consists of 1 colony with 4 fog cells managed by an orchestration node F connecting with a neighbor colony N and the cloud R.

In our simulations, the requirements of network topologies are randomly selected with reasonable constraints. In order to ensure the diversity of the simulations, various parameters have been varied as summarized in Table 3. Each configuration is deemed to deploy several applications varying from 1 to 6, each of which consists of 1 to 8 tasks. Each application and task are varied by the required deadline (min, average, and max values are shown in the table) and the required CPU (to which the response time of each



(a) The number of applications, tasks, and nodes

(b) Average application deadline and node delay

(c) Average application CPU and node CPU

Figure 3: Simulation configurations.

Table 4: Applications associated with tasks deployed on the 3-tier architecture.

App name	Task, τ_i	D_{A_k} (s)	task size ($\lvert\tau_i\rvert$, MI)
A_1	0,. . .,5	6	250, 350, 200, 100, 250, 300
A_2	6,. . .,10	7	260,270,280, 290,100
A_3	11,. . .,15	8	200, 250, 200, 250, 300
A_4	16,. . .,21	9	100, 100, 100, 100, 100, 200
A_5	22,. . .,26	9.5	10, 10, 10, 10, 10
A_6	27, 28	8.5	100, 120

task can be inferred in accordance with network topologies and deployment plans), respectively. Network topologies are varied by the number of fog cells (f) in the considering colony which is ranged from 1 to 4. Fog cell is characterized by three parameters, namely, the CPU capacity, delay, and energy consumption as shown in the table.

We have conducted 28 different scenarios and collected information of 65 applications in total with variants mentioned above. The above summary of scenarios is also illustrated in Figure 3. The figures show application deadlines, node's CPU, task's CPU and number of tasks, applications, and nodes with regard to network configurations. Figure 3(a) shows the number of applications, number of tasks need to be deployed, and the number of fog nodes in each network topology in the 28 scenarios. In addition, as a general rule, the available resources of a topology have to be greater than those required by planned applications. While the average node delay is less than the average application deadline in Figure 3(b) to allocate time slots for executing applications, the application CPU is intersected by node CPU because in the worst case tasks could be deployed to the cloud (see Figure 3(c)). An example of application composition is illustrated in Table 4.

As presented, each network configuration consists of 4 to 7 nodes to deploy 1 to 6 applications, each of which is composed of 1 to 8 tasks. As a result, the maximum number of variables in each topology is max(*number of apps* × *number of tasks* × *number of nodes*) = 6 × 8 × 7 = 336; it ensures that the CPLEX resolver is able to produce the results. Other apparent constraints include $C^F > C^f$, $d^R > d^N > d^f$, $P^F_{max} > P^f_{max}$. Obviously, in order to satisfy these constraints accompanied with the objective function presented in equation (3), Section 4.2, each simulation takes into account context about *location, network condition, service type, and expected response time* collected at the starting time of each placement turn.

We have solved the optimization problem of task placement on fog landscape and applied the task distribution plan on the given network topology to evaluate the effectiveness of the proposed approach in terms of satisfying deadline requirement, energy consumption reduction, fog utilization, and cost saving compared to the cloud based approach. The results are presented and analyzed as follows.

Deadline Satisfaction. As we have analyzed, one of the advantages of the proposed 3-tier architecture is its capability of utilizing the locally available resources on the fog landscape for computing tasks; hence it can reduce the communication times (compared to the cloud-based approach) providing more chances to satisfy the required application deadlines. In

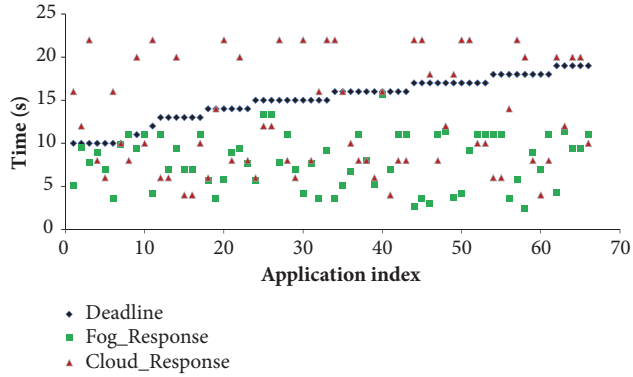


FIGURE 4: Response times in the proposed 3-tier architecture compared with the conventional cloud approach extracted from 65 applications.

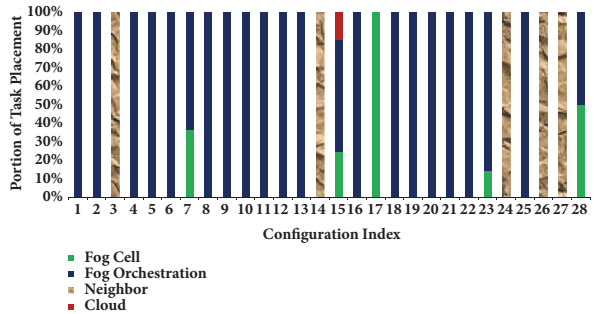


FIGURE 5: Utilization of the fog landscape.

these evaluations, the response time, r_{A_k} of each application is estimated using the model presented in equation (5), while the response time of each task r_{τ_i} is estimated using equation (4), Section 4.3.1. For simplicity, without losing the generality of the response time estimating model, the deployment time at each fog node is ignored ($w_{\tau_i} = 0$), while the execution time (m_{τ_i}) and the communication time (d_{τ_i}) are calculated using equations (10) and (20), respectively, using data provided in Tables 2 and 4. After estimation, r_{A_k} is compared with the corresponding deadline D_{A_k} provided in Table 4. The comparison results are presented in Figure 4.

As shown, around 57% of the applications miss the deadline if they are processed on the cloud. Meanwhile, no application in the proposed 3-tier approach misses the deadline. It should be noted that there are some cases the response times in the proposed approach are larger than the cloud-based ones (both of them satisfy the deadlines). These situations occur when applications consisting of heavy tasks (i.e., tasks require a large number of processing instructions) are deployed on fog nodes with low computational power.

Fog Landscape Utilization. Figure 5 shows an effective utilization of the virtual resources available on the fog landscape in our proposed method. Among 28 configurations (consisting of 65 applications), there is only one configuration that involves the cloud revealing that more than 95% of configurations the applications are fully processed on the fog landscape (without using any cloud-based resource). An interesting point here is that applications are firstly attempted on the fog orchestration node, then on fog cells, then on the neighbor colony, and finally on the cloud. Obviously, the communication time needed for applications to be deployed on the fog orchestration node is 0, and the computation power of this node is commonly much more powerful than those of fog cells; in many cases tasks are referred to be and successfully be deployed on this node. It is also worth to be noticed that, in the cases when cloud resources are involved, only 15% of such resources are required; the remained resources come from the fog landscape.

Energy Consumption Reduction. We also evaluated the energy saving by the proposed architecture and the results are depicted in Figure 6. In most of the cases the proposed architecture saves more than 80% of the energy compared to that of the cloud-based approach. There are only 2 cases among 28 configurations (i.e., 7%); the energy consumption in our proposed method is comparable or a little bit higher than that of the cloud-based method. This is because the main objective of our approach is to maximize the task deployment on the fog landscape when the estimated response time still satisfies the required deadline although the computation time is higher than the case that tasks are processed on the cloud. However, as mentioned, these situations are rare (only 7%) and in fact (in most of the cases) the proposed 3-tier architecture provides benefits in both the response time (computation and communication times) and the energy saving.

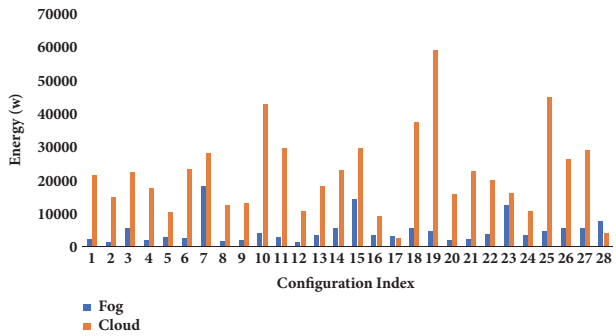


Figure 6: Energy consumption in the proposed scheme compared with the cloud-based approach.

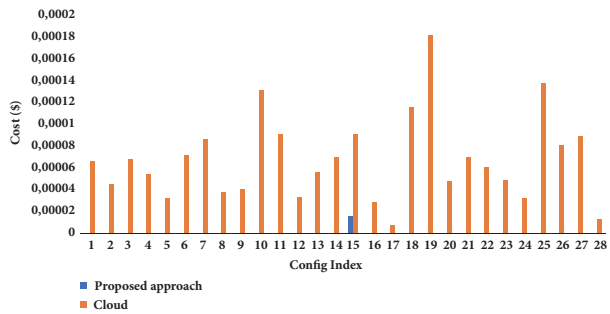


Figure 7: Cost saving in the proposed method.

Cost Saving. Figure 7 reveals the effectiveness of the proposed method in terms of cost saving. As analyzed before, in most of configurations applications are fully deployed on the fog landscape (see Figure 5); no renting cost is required for cloud services in these cases. Cost incurs only in one configuration (configuration 15) when some of cloud resources are required.

5.2.2. Verification under the Real World ITS Application. The communications infrastructure in HCMC is not designed for a specific application such as the ITS. Meanwhile, we could not redesign the network infrastructure physically. Instead, we applied the proposed model to realize the fog computing paradigm, solving the problem of task placement, thus improving the performance of the ITS that we are building for Ho Chi Minh City. In this work we have designed the system with the following principles.

(i) *Decentralization*: The whole system is geographically partitioned into small autonomy regions, each of which is considered a fog colony operating independently with a minimized interference from a "central light-weight controller" (normally a high-end server or a data center). This principle mitigates global communications to lessen the impact of poor connections. Moreover, the autonomy regions help to strengthen the system's availability and fault tolerance.

(ii) *Localization*: Computing resources are distributed across local regions deployed close to data sources; hence the data can be processed locally without invoking many parts of the system. This principle guarantees instant responses even in the context of low and unstable connections. This localization also allows us to provide qualified services at low cost by efficiently utilizing computation resources already available at the network edges. Besides helping to save the high cost of renting computation and storage resources from the cloud, these edge-based devices commonly consume less energy compared to that of data centers on the cloud when processing the same tasks.

We apply principles mentioned above in our ITS system in accordance with the proposed 3-tier architecture with 3 layers, namely, *cloud*, *fog*, and *IoT devices*. We partition the whole city into smaller areas and deploy a fog colony on each area (formed by a group of multiple fog nodes) to handle data and services related to the considered areas such as to estimate the average velocity of a traffic flow on a specific road segment on the considered area. The organization of a fog colony is followed with the designs on the proposed 3-tier architecture presented in Section 4. Each fog colony maintains connection to one (or more) neighbor colony. If the fog colony is overloaded, it moves some tasks to its neighbor and/or to the cloud.

We designed a testbed for emulating our model with the configuration described in Table 5. Note that we use 2 physical machines with Intel Core i7-4790 @ 3.60GHz CPU, 4GB memory and running on Ubuntu 16.04 (64-bit); each machine emulates one fog colony. Concretely, we deploy 1 orchestration node and 3 other fog nodes, each of which is emulated by a single virtual machine, to describe the detailed configuration of a specific colony. The other machine is used to simulate the neighbor colony of the above colony. The fog colonies connect to a data center (cloud) at Ho Chi Minh City University of Technology's high performance computing center (30 km apart from the fog nodes mentioned

Table 5: Configuration of the network topology in the ITS testbed.

Node name	Delay (s)	CPU capacity (MIPS)	Ram capacity (MB)	Power consumption (W, max, and idle)
F	0	100	2e3	50 10
f_0	0.02	3	16	10 2
f_1	0.02	3	16	10 2
f_2	0.02	3	16	10 2
N	0.5	100	2e3	50 10
R	2	684e3	128e3	120 50

Table 6: ITS's applications and associated tasks.

App	Task	Deployment time (p_{τ_i}, ms)	Task size ($\|\tau_i\|$, MI)	Memory consumption (S_{τ_i}, MB)	Deadline (s)
Analyzing traffic flow (a_1)	τ_1	≈ 0	0.02	0.16	1
Traffic light control (a_2)	τ_2	8	6	3	10

Table 7: Effectiveness of the proposed 3-tier scheme compared with the cloud-based approach in the real-world ITS applications.

Real-world case		Deadline missing rate (%)		Energy consumption (J)		Cost ($)	
No. of a1	No. of a2	3-tier	Cloud	3-tier	Cloud	3-tier	Cloud
5	1	0	83.33	452.5	207.4	0	56e-6
10	2	0	83.33	456	414.8	0	113e-6
15	3	0	83.33	459.33	622.2	0	169e-6
50	10	0	50	430.5	2074	0	563e-6
80	16	0	50	448.8	3318.4	0	900e-6

above). The data center is deployed with 2 CPU Intel Xeon E5-2680v3 @ 2.15GHz x 24 CPU and 128GB memory. The connection between fog nodes is emulated by Mininet and the configuration of this topology is described as follows.

In this work, we select two representative applications among many ITS applications for analysis as follows.

(i) **Analyzing traffic flow**: Given a set of GPS signal from vehicles traveling on a road segment, estimate the *average velocity* and the *density* of the considered traffic flow.

(ii) **Traffic light control**: Given a traffic light currently at the beginning of its color phase (that is the sequence of the *green*, *red*, and *yellow* lights), set the duration of each light's color for the next phase appropriately with the traffic condition.

Table 6 shows parameters associated with each application mentioned above.

As presented, each application above is composed of a single task. However, many of similar applications can run concurrently on the same colony (i.e., each application is applied for one road segment in the region covered by the considered colony).

We have solved the optimization problem of task placement on fog landscape to this real world application design. Then, we applied the task distribution plan on the given network topology to evaluate energy consumption reduction, fog utilization, and cost saving of the proposed method compared to the cloud-based approach.

As shown in Table 7, the proposed approach in the 3-tier architecture is significantly more effective compared to the cloud computing counterpart, in all of the three factors (deadline missing rate, energy consumption, and cost). The deadline missing rate in the proposed method is 0% while it is more than 50% in the cloud approach, mainly because of communication latency. As for the energy consumption, when more tasks are deployed on the cloud the energy consumption is large as the cloud-based servers must be up waiting for the arrival of tasks (which are delayed due to upward communication latencies). For example, in the last row, the energy consumptions in the proposed method and the cloud approach are 448.8 (J) and 3318.4 (J), respectively, which are significantly different. The last column shows the cost for renting computing services on the cloud, while the cost on our proposed approach is 0 as the virtual resources available on the network edges are utilized. It should be noted that the cloud service cost is calculated in accordance with the cloud service renting model described in [34, 35].

6. Discussion

This work directly focuses on the feasibility of the optimization on task provision on fog landscape to improve the capacity of fog computing paradigm. In order to reach this target, we have thoroughly analyzed and modeled applications, tasks associated with their required resources, and network configurations with their resource capacity to clarify concrete constraints on the proposed optimization model. The proposed approach is more effective and robust than the existing approaches in terms of practical realization, energy, and cost reduction by means of maximizing the utilization of fog computing paradigm. However, further improvements should be considered.

(1) **Orchestration node selection**: In the current design and prototype, the orchestration node is statically decided in advance based on their computation and storage capacities. As fog devices may die due to energy shortage, a dynamic fog orchestration election model should be thoroughly investigated. This model could be useful for IoT environment with mobile devices.

(2) **Nontree based structure**: The 3-tier architecture is composed based on a tree-based structure which helps to simplify the network management. However, this may danger the network as it is vulnerable to orchestration node failure. As discussed before, when orchestration nodes can be elected dynamically and if this approach is extended thereby every node can serve as an orchestration node if it satisfies some criteria then the architecture in the fog landscape could become "flat." In this sense, the system is more tolerant with node failures but it could be complicated for network management and for task distribution.

(3) **Task distribution**: In this work, task distribution is handled by orchestration nodes. The inherent limitation here is that each orchestration node can obtain only local information about fog nodes in its colony and limited information from its neighbor colonies, rather than examining a global view of the whole network; hence global optimization is not achieved. SDN/Openflow [36] based approaches can help to achieve global optimization by the capability of collecting global network statistics from the Controller. There is a potential direction to apply SDN approach for dynamic task provision in the proposed 3-tier architecture.

(4) **Task dependencies**: In this work, tasks composing an application are assumed to be executed simultaneously to make the proposed 3-tier architecture combining with context-aware service placement viable, before going further. It should be noted that, however, in the IoT applications, the degree of parallelism and dependencies between tasks could be complicated. Therefore, similar to the dynamic of context update the dependence degrees between tasks could also dynamically change. Context-aware service placement in large-scale IoT environments with multiple dependence degrees between tasks is a challenge but potential research direction for our future work which can be developed from this paper.

7. Conclusion

In this article, we have proposed a novel approach to **task placement on fog computing made efficient for IoT application provision** where a systematical fog computing framework consisting of multiple intelligent tiers for the IoT has been introduced and a context-aware task provision mechanism in the fog has been devised. The proposed approach optimally utilizes virtual resources available on the network edges to improve the performance of IoT services in terms of response time, energy, and cost reduction.

The experimental results from both simulated data and data summarized from service deployments in real-world applications, namely, the ITS in Ho Chi Minh City which we are building, show the effectiveness of the proposed approach in terms of maximizing the utilization of fog devices while reducing latency, energy consumption, network load, and operational cost. These results confirm the robustness of the proposed scheme revealing its capability to maximize the IoT potential. In the future, we are planning to implement this method in real applications to support the process of realizing large-scaled IoT applications, smart city ecosystems.

Data Availability

The [simulated evaluation] data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2016.28.

References

[1] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "Cost-effective processing for Delay-sensitive applications in Cloud of Things systems," in *Proceedings of the 15th IEEE International Symposium on Network Computing and Applications, NCA 2016*, pp. 162–169, Cambridge, Mass, USA, November 2016.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the 1st ACM Mobile Cloud Computing Workshop, MCC 2012*, pp. 13–15, Helsinki, Finland, August 2012.

[3] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, *Fog Computing: Principles, Architectures, and Applications*, Internet of Things: Principles and Paradigms, chap. 4. Morgan Kaufmann, 2016.

[4] 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.

[5] D. Miorandi, S. Sicari, F. de Pellegrini, and I. Chlamtac, "Internet of things: vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[7] M. Bertier, F. Desprez, G. Fedak et al., "Beyond the cloud, how should next generation utility computing infrastructures be designed?" in *Cloud Computing. Computer Communications and Networks*, Z. Mahmood, Ed., pp. 325–345, Springer, 2014.

[8] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review Archive*, vol. 44, no. 5, pp. 27–32, 2014.

[9] M. Aazam and E. N. Huh, "Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT," in *Proceedings of the IEEE 29th International Conference on Advanced Information Networking and Applications (AINA '15)*, pp. 687–694, IEEE, Gwangiu, South Korea, March 2015.

[10] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proceedings of the 3rd Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2015*, pp. 73–78, Washington, DC, USA, November 2015.

[11] OpenFog Consortium, 2018, https://www.openfogconsortium.org/.

[12] OpenFog Consortium Architecture Working Group, *OpenFog Reference Architecture for Fog Computing, technical report [PhD thesis]*, 2017.

[13] R. S. Montero, E. Rojas, A. A. Carrillo, and I. M. Llorente, "Extending the Cloud to the Network Edge," *The Computer Journal*, vol. 50, no. 4, pp. 91–95, 2017.

[14] C. C. Byers, "Architectural Imperatives for Fog Computing: Use Cases, Requirements, and Architectural Techniques for Fog-Enabled IoT Networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 14–20, 2017.

[15] P. Hoenisch, I. Weber, S. Schulte, L. Zhu, and A. Fekete, "Four-fold Auto-Scaling on a Contemporary Deployment Platform using Docker Containers," in *Proceedings of the 13th International Conference on Service Oriented Computing (ICSOC 2015)*, vol. 9435 of *Lecture Notes in Computer Science*, pp. 316–323, Springer, 2015.

[16] P. Leitner, W. Hummer, B. Satzger, C. Inzinger, and S. Dustdar, "Cost-efficient and application SLA-aware client side request scheduling in an infrastructure-as-a-service cloud," in *Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, pp. 213–220, USA, June 2012.

[17] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[18] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[19] Y.-J. Yu, T.-C. Chiu, A.-C. Pang, M.-F. Chen, and J. Liu, "Virtual machine placement for backhaul traffic minimization in fog radio access networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1–7, 2017.

[20] 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, 2017.

[21] H. J. Hong, P. H. Tsai, and C. H. Hsu, "Dynamic module deployment in a fog computing platform," in *Proceedings of the 18th Asia-Pacific Network Operations and Management Symposium, APNOMS 2016*, pp. 1–6, Japan, October 2016.

[22] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-Aware Resource Allocation for Edge Computing," in *Proceedings of the 1st IEEE International Conference on Edge Computing, EDGE 2017*, pp. 47–54, USA, June 2017.

[23] V. B. Souza, W. Ramirez, X. Masip-Bruin, E. Marin-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined Fog-cloud scenarios," in *Proceedings of the ICC 2016 - 2016 IEEE International Conference on Communications*, pp. 1–5, Kuala Lumpur, Malaysia, May 2016.

[24] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, "Resource provisioning for IoT services in the fog," in *Proceedings of the 9th IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2016*, pp. 32–39, China, November 2016.

[25] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-Aware Fog Service Placement," in *Proceedings of the 1st IEEE International Conference on Fog and Edge Computing, ICFEC 2017*, pp. 89–96, IEEE, Madrid, Spain, 2017.

[26] 2018, http://parasol.cs.rutgers.edu/.

[27] T. M. Quang, D. T. Nguyen, A. Van Le, H. D. Nguyen, and A. Truong, "Toward service placement on Fog computing landscape," in *Proceedings of the 4th NAFOSTED Conference on Information and Computer Science*, pp. 291–296, Hanoi, Vietnam, 2017.

[28] V. Akman, *Context in Artificial Intelligence: A Fleeting Overview*, McGraw-Hill, Milano, Italy, 2002.

[29] P. Bouquet, C. Ghidini, F. Giunchiglia, and E. Blanzieri, "Theories and uses of context in knowledge representation and reasoning," *Journal of Pragmatics*, vol. 35, no. 3, pp. 455–484, 2003.

[30] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[31] T. T. Do, P. H. Phung, and T. M. Quang, "Toward An IoT-based Expert System for Heart Disease Diagnosis," in *the 28th Modern Artificial Intelligence and Cognitive Science Conference (MAICS)*, pp. 157–164, Fort Wayne, Ind, USA, 2017.

[32] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing En-vironments," Tech. Rep. CLOUDS-TR-2016-2, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, 2016.

[33] IBM CPLEX Optimizer, 2018, https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer.

[34] I. Pietri, G. Juve, E. Deelman, and R. Sakellariou, "A performance model to estimate execution time of scientific workflows on the cloud," in *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science, WORKS 2014*, pp. 11–19, New Orleans, LA, USA.

[35] Elastic Compute Cloud (EC2), 2018, https://aws.amazon.com/ec2/pricing/on-demand/.

[36] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.