Journal of Parallel and Distributed Computing 135 (2020) 177–190

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Profit-aware application placement for integrated Fog–Cloud computing environments

Redowan Mahmud [a,*], Satish Narayana Srirama [b], Kotagiri Ramamohanarao [a], Rajkumar Buyya [a]

[a] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia
[b] Mobile and Cloud Lab, Institute of Computer Science, University of Tartu, Estonia

ARTICLE INFO

Article history:
Received 15 January 2019
Received in revised form 13 September 2019
Accepted 1 October 2019
Available online 11 October 2019

Keywords:
Fog–Cloud integration
Internet of Things
Application placement
Profit-awareness
Pricing model

ABSTRACT

The marketplace for Internet of Things (IoT)-enabled smart systems is rapidly expanding. The integration of Fog and Cloud paradigm aims at harnessing both edge device and remote datacentre-based computing resources to meet Quality of Service (QoS) requirements of these smart systems. Due to lack of instance pricing and revenue maximizing techniques, it becomes difficult for service providers to make comprehensive profit from such integration. This problem further intensifies when associated expenses and allowances are charged from the revenue. Conversely, the rigid revenue maximizing intention of providers affects user's budget and system's service quality. To address these issues, we propose a profit-aware application placement policy for integrated Fog–Cloud environments. It is formulated using constraint *Integer Linear Programming* model that simultaneously enhances profit and ensures QoS during application placement on computing instances. Furthermore, it provides compensation to users for any violation of Service Level Agreement (SLA) and sets the price of instances according to their ability of reducing service delivery time. The performance of proposed policy is evaluated in a simulated Fog–Cloud environment using *iFogSim* and the results demonstrate that it outperforms other placement policies in concurrently increasing provider's profit and user's QoS satisfaction rate.

1. Introduction

The Internet of Things (IoT) paradigm interconnects numerous devices through Internet to collect and share data from physical environments. By using existing Cloud-centric IoT models, the computational demand of different IoT-enabled systems such as smart city and healthcare can be met [15]. However, execution of their latency-sensitive applications at remote Cloud datacentres can decrease the service quality and excessive dataflow towards the datacentres can congest the network [3]. To overcome such limitations and deal with large number of IoT devices at the edge network, Fog computing is introduced. The computing components within Fog such as Raspberry Pi devices, personal computers, network routers, switches and micro datacentres, commonly known as Fog nodes, are heterogeneous and distributed. They offer infrastructure services to host and develop IoT-applications, and process data closer to sources [10]. Thus, Fog computing facilitates reduced application service time and network congestion for different IoT-enabled systems compared to that scenario when IoT-data is solely processed by remote Cloud datacentres [9,27].

Fog nodes have less computational capabilities than Cloud datacentres that resist accommodation of every IoT application at the edge level [29]. Therefore, different Cloud providers such as Amazon, Microsoft and Google initiate integrating Fog and Cloud infrastructure to offer extensive placement options for IoT applications [6]. The inclusion of Fog computing to current Cloud-centric IoT model is expected to add US$ 203.48 million more in their combined marketplace by 2022 [20]. It will also increase the operational cost in computing environments for consuming additional energy, deploying Fog infrastructure and utilizing more network bandwidth [14]. In this case, without revenue maximization, it will be difficult for providers to make profit from integrated environments. Contrariwise, firm intention of maximizing revenue often instigates providers to compromise application Quality of Service (QoS) that increases Service Level Agreement (SLA) violations. The imprecise price of Fog instances that is set for revenue maximization can also add overhead to user's budget [4]. Hence, it becomes challenging to enhance provider's profit in integrated environments as it urges to make a balance among expectations of users, expenses of

* Corresponding author.
E-mail address: mahmudm@student.unimelb.edu.au (R. Mahmud).

https://doi.org/10.1016/j.jpdc.2019.10.001
0743-7315/© 2019 Elsevier Inc. All rights reserved.

providers and performance of Fog–Cloud infrastructure. Failure to ensure such balance inhibits providers and users to realize the potential of integrated computation [23].

In integrated environments, placement of applications on suitable instances is very crucial to enhance profit of providers and meet application QoS for users. Although different application placement policies for Fog computing are proposed prioritizing deadline, completion time and revenue [13,32,37], it is critical for these policies to attain the aforementioned objectives individually for integrated environment. Diversified affordability level of users, uneven expenses of operating heterogeneous instances and commitment of providing compensation to users for service failure further intensify the complexity of such application placement problem [28]. Therefore, it is demanding to develop an application placement policy for integrated Fog–Cloud computing environments that can comply with their economic and performance-based attributes simultaneously.

In Internet economics, providers are encouraged to charge users more for improved services [17]. Since Fog instances upgrade application service delivery time, it creates a scope for providers to charge users an extra amount for these instances on top of their actual Cloud-based price. To users, providers can advertise this additional charge as the price for extending the instance from Cloud to Fog infrastructure. However, it should be justified with the scale of performance improvement and user budget constraint. It is also required for clarifying the impreciseness of instance pricing and assisting users to identify how much they need to pay for executing applications in Fog. Additionally, to attain loyalty, providers can offer compensation to users on SLA violations. With such instance pricing model and compensation method, an application placement policy in integrated environments can boost the revenue and arouse the necessity of meeting application QoS that will consequently enhance provider's profit. However, in existing works such policy has not been explored yet. Therefore, we propose a profit-aware application placement policy for integrated Fog–Cloud environments that increases revenue and reduces their number of failures in meeting application's service delivery deadline. It also sets price of Fog instances in accordance with their capabilities of improving service quality and provides compensation to users based on SLA violation rate of computing environments.

The major **contributions** of this paper are:

- Proposes an application placement policy for integrated Fog–Cloud environments based on an *Integer Linear Programming (ILP)* model that enhances provider's profit and meets application's QoS simultaneously.
- Presents a pricing model for Fog instances which increases provider's revenue by incorporating their Cloud-based pricing with the service delivery time improvement ratio of applications placed on those instances.
- Develops a user compensation method that supports both user's and provider's interest through inverse relationship between compensation amount and performance of the computing environments in observing SLA requirements.
- Demonstrates the performance of proposed policy in enhancing profit, satisfying QoS and managing waiting time via simulation on *iFogSim* [16] and compares them with the outcomes of existing policies.

The rest of the paper is organized as follows. Section 2 highlights several relevant works form literature. Section 3 provides the architecture of integrated environments along with revenue estimation, pricing model and compensation method. The proposed application placement policy and its illustrative example are presented in Sections 4 and 5 respectively. Section 6 presents the simulation environment and performance evaluation of the proposed policy. Finally, Section 7 concludes the paper.

2. Related work

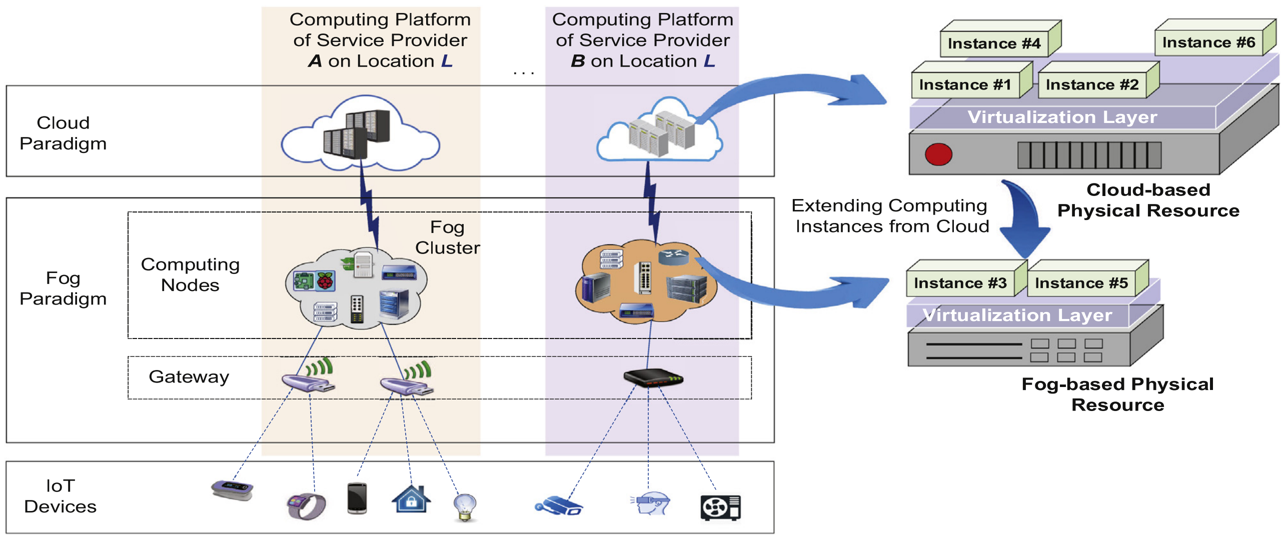
Provider's profit and cost maintenance have already been studied extensively in Cloud computing [24,25]. However, Fog computing is different from Cloud as it is more distributed and composed of numerous resource-constrained and heterogeneous Fog nodes. Service expectations of users from Fog-based applications, their anticipated run-time and budget for execution are also diversified compared to that of Cloud-based applications. Therefore, it is very complicated to develop interoperable resource and application management policies for both Fog and Cloud computing and tedious to customize any existing Cloud policy for Fog computing [28]. Nevertheless, there exists several works that discuss about financial aspects of integrated Fog–Cloud environments. Nan et al. [32] provided an online solution that minimizes task completion time and provider's cost in integrated environments. It also reduces overall response time by discarding infeasible applications directly from the queue. In another work, Deng et al. [12] made trade-off between power consumption and transmission delay. Their policy solves the placement problem distributively and allocate resources at the Fog to complement Cloud for improving performance. Moreover, Pham et al. [36] did trade-off between execution time and cost of Cloud-based processing while placing applications in integrated environment. Their Cost-Makespan-aware placement policy meets application deadline constraints. Yu at al. [43] also focused on reducing processing cost in Cloud by placing applications in Fog. Their policy saves bandwidth cost by serving users with Fog resources and compensates Fog providers for processing data on behalf of Cloud.

The efficacy of Fog has also been extended to other computing paradigms. Lin et al. [14] minimized the expenses in Fog assisted Cyber Physical System (CPS) considering instance deployment, data uploading and inter-nodal communication cost. To overcome high complexity, their policy linearizes the cost-minimization problem then solves it through a two-phase linear program-based heuristic algorithm. Likewise, Yang et al. [41] explored cost-efficient service placement and load distribution in Fog enabled Mobile Cloud Computing (MCC) environments. Their algorithms make trade-off between the average response delay and the expenses of providers by considering mobility and service access pattern of users. Yao et al. [42] considered instance deployment cost and diverse mobility pattern of the users while placing applications on heterogeneous Fog nodes (Cloudlets). Their greedy solution, at first, generates candidate set of Cloudlets that meets user's requirements, then selects a Cloudlet from the candidate set to place the applications with minimum deployment cost. Kiani et al. [21] proposed an auction-based profit maximization policy for Fog enabled Mobile Edge Computing (MEC) environment. Their policy is developed on a binary linear programming model and incorporates a two-time scale technique while allocating both the computing and communications resources to the mobile users.

In literature, profit and budget-aware resource estimation for Fog computing are also studied. Fan et al. [13] discussed deadline-aware application placement that enhances provider's profit and user's QoS satisfaction. It exploits provider's owned Fog resources and rented Cloud instances combinedly. Neetu et al. [37] explored the competition between Fog providers in setting service price and minimizing their cost through Equilibrium Constraints model. It aims at enhancing the profit and balancing the service requirements between providers and users by facilitating incentivization. A dynamic resource estimation and pricing policy for Fog computing was developed by Aazam et al. [1]. While allocating resources and charging services, their policy considers user's behaviour, provider's profit and category of IoT devices

Table 1
Summary of relevant works.

Work	Decentralized Decision	Provides Compensation	Considers Budget	Maintains QoS	Enhances Profit	Models Price	Targets Cost	
							Fog	Cloud
Nan et al. [32]	✓			✓			✓	✓
Deng et al. [12]	✓			✓			✓	✓
Pham et al. [36]				✓				✓
Yu et al. [43]	✓	✓						✓
Lin et al. [14]	✓			✓			✓	
Yang et al. [41]				✓			✓	
Yao et al. [42]				✓			✓	
Kiani et al. [21]			✓		✓		✓	
Fan et al. [13]	✓			✓	✓		✓	
Neetu et al. [37]	✓	✓	✓		✓	✓	✓	
Aazam et al. [1]	✓	✓			✓	✓	✓	
Ni et al. [33]	✓		✓	✓			✓	
Profit-aware (this work)	✓	✓	✓	✓	✓	✓	✓	✓



Computing Platform of Service Provider A on Location L
. . .
Computing Platform of Service Provider B on Location L
Cloud Paradigm
Fog Paradigm
Computing Nodes
Fog Cluster
Gateway
IoT Devices
Instance #4
Instance #6
Instance #1
Instance #2
Virtualization Layer
Cloud-based Physical Resource
Extending Computing Instances from Cloud
Instance #3
Instance #5
Virtualization Layer
Fog-based Physical Resource

Fig. 1. Integrated Fog–Cloud environments.

based on their mobility pattern. Ni et al. [33] proposed a resource provisioning policy for Fog that enables users to meet demand from a set of reserved resources. It considers cost and deadline along with user's affordability and features of Fog nodes during resource allocation.

Table 1 provides a summary of different application placement policies that investigate monetary issues and service objectives of integrated environments. In these works, enhancement of profit and maintenance of QoS are not simultaneously ensured during application placement. They barely apply performance-driven instance pricing model and compensation method to promote provider's revenue and subsidize user's losses. In comparison, the proposed policy contains two important features; (*a*). sets price of Fog instances based on their competency of improving application's service time and (*b*). offers compensation according to the SLA-violation rate of computing environments. Both jointly offer a systematic way to support user's and provider's interests. The profit-aware application placement problem is also formulated as a function of application's service delivery deadline that resists their QoS degradation. These aspects form the core innovation part of the policy that helps to overcome the limitation of existing placement policies. Additionally, the proposed policy deals with various Fog and Cloud-based costs and works in decentralized manner so that it can be synthesized with distributed Fog nodes.

3. System design

3.1. Features of integrated Fog–Cloud environments

As a supplement to IoT, Fog computing executes latency-sensitive applications in proximate of data sources to offer services in real time. Conversely, as an extension of Cloud computing, Fog conducts IoT-data pre-processing so that communication and computation overhead from Cloud datacentres can be reduced. Thus, Fog computing maintains an intermediate layer between IoT and Cloud computing [32,43]. Based on this concept, the *Computing Platforms* for IoT applications are considered to be expanded across the Fog and Cloud infrastructure of integrated environments. Different providers deploy such platforms on various locations with their owned physical resources (Fog nodes and Cloud datacentres), for example, Fig. 1 shows the Computing Platforms deployed by provider *A* and *B* on location *L*. At the Fog part of each platform, provider-specific Gateways are deployed, and their number can be scalable as per the load of external connections with the platform. When multiple Gateways are associated to a Computing Platform, their operations are synchronized, and monetary calculations are performed in collective manner. Within a Fog cluster, the communication is maintained by faster Constrained Application Protocol and Simple Network Management Protocol. Since Gateways and Fog clusters are localized, their data exchange delay is considered negligible [30]. In Fog clusters, cybersecurity frameworks are used to identify and monitor malicious Fog nodes that defend the Fog part of platforms from uncertain attacks in future [39]. Fog clusters can extend different types of virtualized instances (virtual machines and containers) from Cloud datacentres for application execution [34,43]. While making application placement decisions in Fog infrastructure, the Cloud-based attributes of extended instances such as their configurations, price and cost model are used extensively along with other performance parameters [5,38].

Moreover, it is regarded that most of the IoT devices are cramped of performing large-scale data processing and maintaining direct communication with Cloud because of their resource scarcity and bandwidth limitations [30]. In such circumstance, Bottom-up interaction among IoT, Fog and Cloud computing plays an important role where IoT devices at first communicate with Fog infrastructure and notify their service requirements. Fog infrastructure tries to meet these requirements with its available resources. If it is infeasible, Fog infrastructure asks Cloud to deal with the issue [1,36]. To enable such interaction in the devised integrated environments, IoT devices are configured with the Gateways of any accessible Computing Platforms and the reliable links between Fog and Cloud part of Computing Platforms help the Gateways to reach out Cloud-based services via Fog clusters on behalf of the IoT devices. However, due to mobility of IoT devices, their associated Gateway and Computing Platform can change with the course of time. Therefore, to maintain connectivity and deal with data traffic, related network Service Function Chains (SFCs) are transferred from one place to another. In integrated environments, efficient SFC migration approaches are applied to support this operation with reduced reconfiguration cost [44].

While configuring an IoT device with a Gateway, the placement request for corresponding application is narrated. A placement request comprises specifications of the application including its number of instructions, input packet size, data receiving frequency, expected service delivery time limit and user's budget for its execution. However, IoT device and user-driven contexts can vary from time to time. Therefore, integrated environments endorse time series analytic frameworks for dependable data extraction so that varying contexts of these entities can be tracked and placement requests for the applications can be updated [40]. After assimilation of an application placement request, the associated Gateway grasps status such as processing speed, bandwidth, per unit time costs and price of available instances [1]. Based on these parameters, the Gateway finds suitable placement option for the application. If any instance satisfies minimum resource requirements of an application, its deployment time on that instance becomes trivial [31].

Unremitting application placement requests received by a Gateway can intensify its management overhead. Therefore, Gateways conduct placement of applications after a certain interval, for example 0.100 s. It helps to manage the overhead of Gateways, simplifies their synchronization with a Computing Platform and resists unnecessary reporting. However, it can increase resource wastage and redundancy to some extent. Their effect can be mitigated by setting the interval between two placement rounds to a minimum value or dynamically tuning it according to the average run-time of applications. In devised integrated environment, providers can follow any of these approaches so that placement round intervals neither burden the Gateways nor decrease resource utilization [26]. Within this interval, Gateways receive new placement requests and instances execute the applications placed at previous round. Before initiating a placement round k, available instances C_k and requested applications R_k for that round are identified. Later, the Gateway estimates profit of the platform provider for executing each application. During placement, a single instance can host at most one application. A placement request is successful if the application is mapped to a computing instance, and its service is ensured to be delivered within the deadline. For kth placement round, the set of successful applications is noted as R_k^{χ}. If an application is not scheduled in a placement round, it is considered for scheduling in the next round along with newly received placement requests. This process continues unless the application is placed, or its estimated service delivery time surpasses the deadline. During a billing period, a Gateway can run numerous placement rounds targeting the Computing Platform. However, after a billing period, compensation for users based on the SLA violation rate of corresponding platform is determined and total profit of its provider is calculated. Relevant notations for these calculations are shown in Table 2.

3.2. Gross profit estimation for providers

Before placing an application $r \in R_k$ on instance $c \in C_k$, *Gross profit* e_c^r of provider for executing the application is estimated. Usually, Gross profit is calculated by deducting the cost of production from the revenue [22]. Here, the revenue refers to the service charge of instance that is collected by provider from user to execute the application. Conversely, the cost of production is the operating cost of instance that is paid by provider to others for application execution. For Gross profit estimation, input packet size l^r and number of instructions in the application s^r are extracted from the placement request. Processing speed μ^c and network bandwidth λ^c of the instance are also considered. Input processing time t_{rc}^p and input propagation time t_{rc}^n are calculated for the application using Eqs. (1) and (2);

$$t_{rc}^p = \frac{s^r}{\mu^c}, \tag{1}$$

$$t_{rc}^n = \frac{l^r}{\lambda^c}. \tag{2}$$

If the instance is deployed in Cloud part, service charge of the instance for executing the application depends on its per unit time price ω^c and the summation of input processing time t_{rc}^p and input propagation time t_{rc}^n. Its operating cost also relies on t_{rc}^p and t_{rc}^n along with its per unit time processing cost α^c and networking cost β^c. In α^c and β^c, providers encapsulate certain portion of various expenses such as deployment, migration, energy and security management costs separately. The Gross profit for executing the application in Cloud-based instance is estimated using Eq. (3);

$$e_{c\in Cloud}^r = \omega^c(t_{rc}^p + t_{rc}^n) - (t_{rc}^p\alpha^c + t_{rc}^n\beta^c). \tag{3}$$

However, if the instance resides in Fog part, input propagation time t_{rc}^n becomes negligible. Therefore, its impact on service charge and operational cost are omitted while estimating the Gross profit. To align with the characteristics of Internet economy [17], providers can also charge users ϵ^c price per unit time on top of ω^c for ensuring improved service through Fog-based placement of applications. It is usually advertised to users as the price for extending the instance from Cloud to Fog. Hence, the Gross profit for application execution on Fog-based instance is estimated by Eq. (4);

$$e_{c\in Fog}^r = t_{rc}^p(\omega^c + \epsilon^c) - t_{rc}^p\alpha^c. \tag{4}$$

Combining Eqs. (3) and (4), a general narration of Gross profit for executing an application is shown in Eq. (5);

$$e_c^r = (t_{rc}^p + \eta_c t_{rc}^n)\{\omega^c + (1-\eta_c)\epsilon^c\} - (t_{rc}^p\alpha^c + \eta_c t_{rc}^n\beta^c). \tag{5}$$

Here, the binary variable η_c tracks whether the instance is deployed in Cloud or extended to Fog part of the Computing Platform. Based on Eq. (5), Gross profit of provider per placement

Table 2
Notations.

Symbol	Definition
K	Total number of placement rounds per billing period on a Computing Platform.
Υ	Total Gross profit of providers from a Computing Platform per billing period.
I_k	Gross profit of providers from a Computing Platform during kth placement round.
C_k	Set of all computational instances during kth placement round on a Computing Platform.
R_k	Set of all requested applications during kth placement round on a Computing Platform.
R_k^χ	Set of successful applications during kth placement round on a Computing Platform. $R_k^\chi \subseteq R_k$.
e_c^r	Estimated Gross profit for executing the application $r \in R_k$ on instance $c \in C_k$.
m_c^r	Profit Merit (PM) of an application $r \in R_k$ on any instance $c \in C_k$.
l^r	Input packet size for the application $r \in R_k$.
s^r	Number of instructions in the application $r \in R_k$.
z^r	Minimum resources required for hosting the application $r \in R_k$, $z \in$ {processing cores, memory}.
ψ^r	Users budget for executing the application $r \in R_k$.
δ^r	User expected service delivery time limit for the application $r \in R_k$.
ξ^r	Latency sensitivity index for the application $r \in R_k$.
μ^c	Processing speed of a computing instance $c \in C_k$.
λ^c	Network bandwidth of a computing instance $c \in C_k$.
Z^c	Available resources on a computing instance $c \in C_k$, $Z \in$ {processing cores, memory}.
ω^c	Cloud-based price of a computing instance $c \in C_k$ for per unit time.
α^c	Cost of computing instance $c \in C_k$ for processing resource consumption per unit time.
β^c	Cost of computing instance $c \in C_k$ for network resource consumption per unit time.
ϵ^c	Additional price of a computing instance $c \in C_k$ for per unit time.
τ_a^r	Arrival time stamp of placement request for application $r \in R_k$.
τ_ϑ^r	Placement time stamp of application $r \in R_k$.
τ	Current time stamp.
P	Net profit for the provider per billing period.
ρ	Compensation given for SLA violation per billing period on a Computing Platform.
Φ	Set of all requested applications per billing period on a Computing Platform. $R_k \subset \Phi$
ϕ	Set of all QoS-satisfied applications per billing period on a Computing Platform. $R_k^\chi \subset \phi$
φ	Set of all SLA-violated applications per billing period on a Computing Platform.
t_{rc}^p	Input processing time on computing instance $c \in C_k$ for application $r \in R_k$.
t_{rc}^n	Input propagation time to computing instance $c \in C_k$ for application $r \in R_k$.
t_{rc}	Total time required to complete the execution of application $r \in R_k$ on instance $c \in C_k$.
υ_{rc}	Performance improvement grade of application $r \in R_k$ for extending instance $c \in C_k$ form Cloud to Fog.
Ω_{rc}	Service charge to users for executing the application $r \in R_k$ on instance $c \in C_k$.
Γ_{rc}	Operational cost of providers for executing the application $r \in R_k$ on instance $c \in C_k$.
$\eta_c \in \{0, 1\}$	Equals to 1 if computing instance $c \in C_k$ is running in remote Cloud, 0 otherwise.
$x_{rc} \in \{0, 1\}$	Equals to 1 if the application $r \in R_k$ is mapped to $c \in C_k$, 0 otherwise.

round and per billing period from a Computing Platform in respect of a Gateway is given by Eqs. (6) and (7);

$$I_k = \sum_{r \in R_k^\chi} e_c^r, \tag{6}$$

$$\Upsilon = \sum_{k=1}^{K} I_k. \tag{7}$$

3.3. Pricing model for Fog instances

To increase provider's Gross profit from Fog-based placement of applications in integrated environment, the following condition needs to be satisfied;

$$e_{c \in Fog}^r > e_{c \in Cloud}^r.$$

One of the possible ways to satisfy this condition is to raise provider's revenue from the Fog instance. It can be achieved by setting a higher ϵ^c value while charging users for the instance. Providers can set this value as per their interest with no guarantee of user acceptance. To attain user's acknowledgement, ϵ^c should reflect the value of improving performance for placing applications on Fog instances. Therefore, in defining ϵ^c, the performance improvement grade υ_{rc} of application $r \in R_k$ is used that denotes per unit time improvement in service delivery of the application for extending its assigned instance $c \in C_k$ from Cloud datacentre to Fog cluster.

When instance c remains in Cloud, service delivery time of application r is the summation of input processing time t_{rc}^p and input propagation time t_{rc}^n. However, if the instance is extended to Fog, service delivery time of application r becomes dependent to input processing time t_{rc}^p and its rough improvement is equivalent to t_{rc}^n compared to Cloud-based placement. Hence, the performance improvement grade υ_{rc} of application r can be narrated as Eq. (8);

$$\upsilon_{rc} = \frac{t_{rc}^n}{t_{rc}^p}. \tag{8}$$

Moreover, providers save a larger portion of networking cost when the application is executed in Fog [10]. It is also observed in assessing the value of ϵ^c. Considering performance improvement and cost saving attributes, to boost the revenue from Fog-based application placement, providers should set the value of ϵ^c satisfying the following condition;

$$\epsilon^c > \upsilon_{rc}(\omega^c - \beta^c).$$

This condition can also be certified with the help of Eqs. (3) and (4). In proposed profit-aware application placement policy, it is applied by adding of a very small charge ∂ per unit time, for example, 0.005 \$/s, with ϵ^c as shown in Eq. (9);

$$\epsilon^c = \upsilon_{rc}(\omega^c - \beta^c) + \partial. \tag{9}$$

3.4. Compensation method and net profit Calculation

SLA of an application $r \in R_k$ violates when the Computing Platform fails to assist it in meeting the service delivery deadline [31]. This deadline is determined by adding the user's expected service delivery time limit δ^r with the request's arrival time stamp τ_a^r. If service of the application is delivered within deadline, its QoS to users is satisfied. In a Computing Platform, per billing period users are only charged for the set of QoS satisfied applications ϕ and compensated for the set of SLA-violated applications φ. The compensation is given as a percentage of average

Gross profit of providers that is accumulated from QoS-satisfied applications [11]. It is calculated using the ratio of SLA-violated and total number of requested applications $|\Phi|$ per billing period; where $|\Phi| = |\phi| + |\varphi|$. Hence, the total amount of compensation ρ given by the provider is shown in Eq. (10);

$$\rho = |\varphi| \times \frac{\Upsilon}{|\phi|} \times \frac{|\varphi|}{|\Phi|}. \tag{10}$$

This compensation method works as per the performance of Computing Platform. If the Computing Platform assists to increase the number of QoS satisfied applications, the total compensation reduces. Conversely, if its performance degrades, increased amount of compensation helps to retain the user's loyalty. This inverse relationship balances the financial support of Computing Platform for both users and providers [8]

After determining the compensation, *Net profit P* of provider from the Computing Platform for a billing period is assessed. Net profit is calculated by deducting the non-operational cost from the total Gross profit [22]. Here, the non-operational cost of providers refers to the amount of compensation that is repaid to the users. Repaying the compensation ρ by applying Eq. (11), the residual portion of Gross profit Υ per billing period is regarded as the provider's Net profit P;

$$P = \Upsilon - \rho. \tag{11}$$

4. Profit-aware application placement

4.1. Problem formulation

According to Eq. (11), provider's Net profit P from a Computing Platform enhances if the Gross profit Υ per billing period increases and the amount of compensation ρ decreases. To support these conditions during placement rounds, the proposed Profit-aware Application Placement policy prioritizes each application $r \in R_k$ in terms of estimated Gross profit e_c^r for execution on any instance $c \in C_k$ and latency sensitivity index ξ^r. On current time stamp τ, ξ^r refers to the remaining time from application's service delivery deadline as shown in Eq. (12);

$$\xi^r = \tau_a^r + \delta^r - \tau. \tag{12}$$

In the proposed policy, based on e_c^r and ξ^r, *Profit Merit (PM)* m_c^r of application $r \in R_k$ is calculated using Eq. (13);

$$m_c^r = \frac{e_c^r}{\xi^r}. \tag{13}$$

On an instance $c \in C_k$, if the estimated Gross profit e_c^r remains same $\forall r \in R_k$, the application having stringent deadline will have the higher PM value. Conversely, if two applications have identical latency sensitivity index, the application estimating elevated Gross profit for execution will have the higher PM value. According to these two cases, in other scenarios, the PM value of an application r on any instance c signifies the relative weight of estimated Gross profit for execution and latency sensitivity index. Additionally, latency sensitivity index ξ^r of an application decreases with the course of time. As a result, if an application is placed at kth round, its PM value on particular instance increases by $k + 1^{th}$ round. Based on these features of PM, the objective function of profit-aware application placement for any placement round k is formulated as Eq. (14), where a binary decision x_{rc} helps to identify optimal mapping of an application $r \in R_k$ to an instance $c \in C_k$;

$$\max \sum_{r \in R_k} x_{rc} m_c^r. \tag{14}$$

subject to,

$$\sum x_{rc} \le 1; \forall r \in R_k. \tag{15}$$

$$x_{rc} z^r \le Z^c; \forall r \in R_k, \forall Z, \forall z. \tag{16}$$

$$(t_{rc}^p + \eta_c t_{rc}^n) \le \xi^r; \forall r \in R_k. \tag{17}$$

$$\Omega_{rc} \le \psi^r; \forall r \in R_k, \tag{18}$$

where,

$$\Omega_{rc} = (t_{rc}^p + \eta_c t_{rc}^n)\{\omega^c + (1 - \eta_c)\epsilon^c\}. \tag{19}$$

Eq. (14) is a constrained ILP model that maximizes the total PM of applications during kth placement round and Eqs. (15)–(18) specify its constraints. This objective function is required to solve at the beginning of each placement round. It can be solved with any ILP solver such as SCIP [2]. The constraints of Eq. (14) are discussed in the following subsections.

4.1.1. Placement constraint

To deliver uninterrupted services, an application $r \in R_k$ requires exclusive access to the assigned instance $c \in C_k$. The constraint presented in Eq. (15) supports this condition by compelling a computing instance to host at most one application per placement round.

4.1.2. Resource constraint

A computing instance $c \in C_k$ can host the application $r \in R_k$, if its available resources Z^c such as processing cores and memory meet minimum resource requirements z^c of the application. Eq. (16) enforces this constraint signifying that minimum resources to host an application is always available on its assigned computing instance.

4.1.3. QoS constraint

Placement of the application $r \in R_k$ on an instance $c \in C_k$ will not be successful unless its QoS satisfaction is ensured. QoS of the application is satisfied when the propagation and processing of input data are completed within the remaining time from service delivery deadline. Eq. (17) imposes this constraint to the proposed application placement policy.

4.1.4. Budget constraint

Total service charge Ω_{rc} of executing the application in $r \in R_k$ on an instance $c \in C_k$ should be within the affordability of the user. If user's budget is not sufficient compared to the total service charge, execution of that application will trigger negative gearing for the provider. Eq. (18) defines this constraint during application placement.

4.2. Enhancement of profit

Complexity of solving the optimization problem noted in Eq. (14) using an ILP solver is very high. Through this method, it is not feasible to identify the application-instance mapping within stringent time frame for profit-aware application placement [7]. Therefore, a heuristic-method to solve the placement problem is proposed. The heuristic is immanent in the *EnhanceProfit* procedure presented in Algorithm 1. It identifies the best-fit solution in terms of Profit Merit (PM) for placing applications to instances. Details of the EnhanceProfit procedure is described as follows.

To determine the application-instance mapping for the kth placement round, EnhanceProfit procedure takes the current time stamp τ, set of available instances C_k and set of requested applications R_k as arguments. While identifying the mapping, at first Fog-based instances and later, the Cloud-based instances are considered. As shown in Algorithm 1, EnhanceProfit procedure consists of 4 steps:

(1) For each instance $c \in C_k$, firstly, it is checked whether the instance has already been allocated to any application or not (line 2–3). If the instance is not allocated, two variables M_c and

Algorithm 1 Profit enhancement algorithm

```
1: procedure EnhanceProfit($\tau$, $C_k$, $R_k$)
2:   for $c := C_k$ do
3:     if $!c.allocated$ then
4:       $M_c \leftarrow -\infty$
5:       $X_c \leftarrow null$
6:       for $r := R_k$ do
7:         if $!r.placed$ then
8:           $t^p_{rc} \leftarrow \frac{s^r}{\mu_c}$
9:           $t^n_{rc} \leftarrow \frac{I^r}{\lambda_c}$
10:          $\upsilon_{rc} \leftarrow \frac{t^n_{rc}}{t^p_{rc}}$
11:          $\xi^r \leftarrow \tau^r_a + \delta^r - \tau$
12:          $t_{rc} \leftarrow t^p_{rc} + \eta_c t^n_{rc}$
13:          $\Omega_{rc} \leftarrow t_{rc}[\omega^c + (1-\eta_c)\{\upsilon_{rc}(\omega^c - \beta^c) + \partial\}]$
14:          $\Gamma_{rc} \leftarrow t^p_{rc}\alpha^c + \eta_c t^n_{rc}\beta^c$
15:          $e^r_c \leftarrow \Omega_{rc} - \Gamma_{rc}$
16:          $m^r_c \leftarrow \frac{e^r_c}{\xi^r}$
17:          if $z^r \le Z^c$ then
18:            if $t_{rc} \le \xi^r$ then
19:              if $\Omega_{rc} \le \psi^r$ then
20:                if $M_c < m^r_c$ then
21:                  $M_c \leftarrow m^r_c$
22:                  $X_c \leftarrow r$
23:      if $X_c \neq null$ then
24:        $c.assignedApplication \leftarrow X_c$
25:        $X_c.placed \leftarrow true$
26:        $c.allocated \leftarrow true$
27:        $\tau^{X_c}_{\vartheta} \leftarrow \tau$
28:        $\phi.add(X_c)$
```

X_c are initialized (line 4–5). M_c tracks the maximum PM value associate with the instance c and X_c stores the application which is responsible for the value in M_c.

(2) For the instance c, the placement request of each application $r \in R_k$ is exploited (line 6). If the application has not been placed yet (line 7), the following parameters are determined for its placement in respect of instance c.
i. input data processing time t^p_{rc} through Eq. (1) (line 8).
ii. input data propagation time t^n_{rc} applying Eq. (2) (line 9).
iii. performance improvement grade υ_{rc} by Eq. (8) (line 10).
iv. latency sensitivity index ξ^r according to Eq. (12) (line 11).
v. required time t_{rc} to complete the application execution based on t^p_{rc} and t^n_{rc}(line 12) .
vi. service charge Ω_{rc} for users to execute the application using Eq. (19) (line 13). The value of ϵ^c is derived from Eq. (9).
vii. operational cost Γ_{rc} for executing the application considering its input data propagation time t^n_{rc} and processing time t^p_{rc} along with per unit time networking cost β^c and processing cost α^c (line 14).
viii. Gross profit e^r_c for application execution by deducting operational cost Γ_{rc} from service charge Ω_{rc} (line 15).
ix. PM m^r_c by applying Eq. (13) (line 16).

(3) Based on the calculation of step 2, resource, QoS and budget constraint for the placement are explored (line 17–19). The estimated m^r_c is also compared with M_c provided that the constraints are satisfied (line 20). If m^r_c is higher than M_c, then M_c is updated with the value of m^r_c and X_c is set to r (line 21–22). The intuition for performing these operations is to select an application r for placing on instance c which meets all the imposed constraints and has the maximum relative weight of estimated Gross Profit and latency sensitivity index on c. It not only increases the proportion of Gross Profit for executing application r on instance c but also reduces the scope of SLA violation for r. Consequently, it enhances the Net profit of providers.

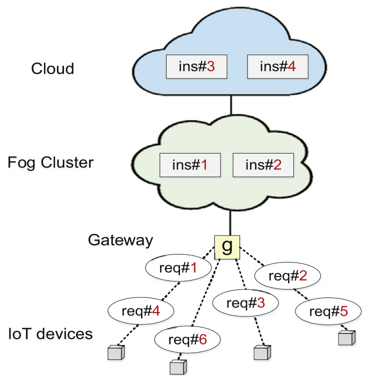


Fig. 2. Illustrative integrated Fog–Cloud environments.

Table 3
Parameters of computing instances.

Instances	λ^c(KB/s)	μ^c(TI/s)	ω^c($/s)	α^c($/s)	β^c($/s)
ins#1	840.00	190.00	0.0380	0.0085	0.0065
ins#2	824.00	167.00	0.0364	0.0064	0.0050
ins#3	820.00	162.00	0.0360	0.0060	0.0047
ins#4	845.00	193.00	0.0386	0.0088	0.0068

(4) If X_c refers to any application (line 23), that application is assigned to instance c (line 24). To ensure the placement constraint, $X_c.placed$ and $c.allocated$ are set true (line 25–26). The placement time $\tau^{X_c}_{\vartheta}$ of X_c is also set to current time stamp τ and X_c is added to the set of QoS satisfied application placement requests ϕ (line 27–28).

For each placement round of a billing period, EnhanceProfit procedure is required to be executed. However, from lines 2 to 28 in Algorithm 1, there are $\mathcal{O}(|C_k| \cdot |R_k|)$ iterations, where $|C_k|$ is the number of available computing instance and $|R_k|$ denotes the number of received application placement requests during kth placement round. Therefore, while identifying application-instance mapping per placement round, Algorithm 1 functions with polynomial time complexity. Theoretically, it also takes less amount of time to operate than ILP solvers. In addition, the proposed heuristic-method simultaneously enhances the Net profit of providers, ensures the QoS satisfaction of applications and meets the budget constraint of users which makes the method more effective for profit-aware application placement.

5. Illustrative example

To numerically illustrate the basic steps of proposed profit-aware application placement policy, we have considered an integrated Fog–Cloud environment as depicted in Fig. 2.

Here, the Computing Platform offers 4 instances: two instances (ins#1, ins#2) are extended from Cloud to Fog part and two instances (ins#3, ins#4) remain at Cloud part. Configuration of the instances are summarized in Table 3. Here, Kilo byte per second (KB/s) and Thousand instruction per second (TI/s) refers to the unit of network bandwidth and processing speed for the instances respectively.

At time $\tau = 0.0$ second, the Gateway g starts receiving placement requests and the placement round interval is set to 0.100 s. Thus, the first placement round occurs at $\tau = 0.100$ second. Details of requested applications before the first placement round are given in Table 4.

For the first placement round, entries of Tables 3 and 4 are denoted as C_1 and R_1, and as part of EnhanceProfit procedure, the input data processing time t^p_{rc} and propagation time t^n_{rc}, $\forall r \in R_1$ on each $c \in C_1$ are determined. The value of t^p_{rc} and t^n_{rc} for this

Table 4
Parameters of applications.

Requests	l^r(KB)	s^r (TI)	ψ^r ($)	δ^r (sec)	τ_a^r
app#1	110.00	27.00	0.9054	0.5838	0.0170
app#2	80.00	23.00	0.8062	0.5115	0.0190
app#3	140.00	30.00	0.8915	0.6115	0.0330
app#4	90.00	26.00	0.9797	0.6075	0.0340
app#5	100.00	28.00	0.8384	0.6719	0.0360
app#6	130.00	21.00	0.9210	0.5448	0.0410

Table 5
Input data processing and propagation time.

Input data processing time t^p_{rc}

Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.1421	0.1617	0.1667	0.1399
app#2	0.1211	0.1377	0.1420	0.1192
app#3	0.1579	0.1796	0.1852	0.1554
app#4	0.1368	0.1557	0.1605	0.1347
app#5	0.1474	0.1677	0.1728	0.1451
app#6	0.1105	0.1257	0.1296	0.1088

Input data propagation time t^n_{rc}

Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.1310	0.1335	0.1341	0.1302
app#2	0.0952	0.0971	0.0976	0.0947
app#3	0.1667	0.1699	0.1707	0.1657
app#4	0.1071	0.1092	0.1098	0.1065
app#5	0.1190	0.1214	0.1220	0.1183
app#6	0.1548	0.1578	0.1585	0.1538

Table 6
Performance improvement grade υ_{rc} for Fog instances.

Instances → Applications ↓	ins#1	ins#2
app#1	0.9215	0.8257
app#2	0.7867	0.7049
app#3	1.0556	0.9458
app#4	0.7830	0.7015
app#5	0.8078	0.7238
app#6	1.4002	1.2546

Table 7
Total service charge and operational cost.

Total service charge Ω_{rc}

Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.0102	0.0109	0.0108	0.0104
app#2	0.0082	0.0088	0.0086	0.0083
app#3	0.0120	0.0128	0.0128	0.0124
app#4	0.0093	0.0099	0.0097	0.0093
app#5	0.0101	0.0108	0.0106	0.0102
app#6	0.0096	0.0102	0.0104	0.0101

Total operational cost Γ_{rc}

Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.0012	0.0010	0.0016	0.0021
app#2	0.0010	0.0009	0.0013	0.0017
app#3	0.0013	0.0011	0.0019	0.0025
app#4	0.0012	0.0010	0.0015	0.0019
app#5	0.0013	0.0011	0.0016	0.0021
app#6	0.0009	0.0008	0.0015	0.0020

Table 8
PM of applications for first placement round.

Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.0180	0.0197	~~0.0184~~	~~0.0166~~
app#2	0.0167	0.0183	0.0170	0.0152
app#3	0.0196	~~0.0213~~	~~0.0200~~	~~0.0182~~
app#4	0.0150	0.0164	0.0152	0.0137
app#5	0.0145	0.0159	0.0148	0.0133
app#6	0.0179	0.0193	0.0182	~~0.0167~~

Table 9
Placement map for first round.

Instances	Applications
ins#1	app#3
ins#2	app#1
ins#3	app#6
ins#4	app#2

Table 10
PM of applications for second placement round.

Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#4	0.0237	~~0.0260~~	~~0.0242~~	~~0.0217~~
app#5	0.0217	0.0237	~~0.0221~~	~~0.0198~~

Table 11
Placement map for second round.

Instances	Applications
ins#1	app#4
ins#2	app#5

round are shown in Table 5. Later, $\forall r \in R_1$, the performance improvement grade υ_{rc} are determined. They are figured out in respect of ins#1 and ins#2 those are extended from Cloud to Fog part of the platform. Since ins#3 and ins#4, remain in Cloud part, according to the proposed policy, performance improvement grade of applications regarding them are irrelevant. Moreover, execution service charge Ω_{rc} and operational cost Γ_{rc} of applications on each $c \in C_1$ are estimated. Here, ∂ is set to 0.005 $/s. These estimations are presented in Tables 6 and 7. Additionally, the PM values of each $r \in R_1$ on different $c \in C_1$ are calculated and they are listed in Table 8.

The proposed policy selects that instance for placing an application which ensures maximum PM value (in red colour on Table 8) for the application satisfying all constraints. The placement map for the first round is shown in Table 9.

In this example, the second placement round is supposed to occur at $\tau = 0.200$ second. Since all instances were busy on that time in executing previously placed applications, the second round occurs at $\tau = 0.300$ second. By this time all instances become available for C_2 and due to not receiving new requests, R_2 encapsulates only app#4 and app#5. The PM values of each $r \in R_2$ on $\forall c \in C_2$ are shown in Table 10. According to them, the placement map for the second round is shown in Table 11.

The illustrative example shows that the PM value of applications waiting for longer period of time gradually increases. However, the aforementioned operations for each placement round are conducted on Gateway g of the Computing Platform. We implement Gateway g with *Intel(R) Core(TM)2 Duo CPU E6550 @ 2.33 GHz 2 GB DDR2 RAM*. On this configuration, Gateway g takes 0.008 s on average to identify placement map for each round.

6. Performance evaluation

In this section, performance of the proposed profit-aware application placement policy is compared with the basic concept of *Completion time* [32], *Deadline* [13] and *Revenue-prioritized placement* policies [37]. In Deadline-prioritized placement policy, deadline-critical applications are placed on computationally powerful instances in higher precedence whereas in Revenue-prioritized placement policy, it is done for economically beneficial

Table 12
Simulation attributes.

Parameter	Numerical specification
Simulation duration	500 s
Interval between placement rounds	0.020–0.180 s
Total number of instances	50
Frequency of request arrival	10–50 applications/s
Instance:	
Network bandwidth	700–1000 KBPS
Processing speed	120–220 TIPS
Price	0.025–0.06 \$/s
Cost of processing	0.005–0.01 \$/s
Cost of networking	0.002–0.007 \$/s
Application:	
Input packet size	80–140 KB
Number of instructions	20–30 TI
User's budget	0.80–1.00 \$ per application
Service delivery time limit	0.500–0.700 s

applications. Alternatively, through Completion time-prioritized placement policy, applications are placed on those instances which collectively ensure minimized application execution time. The profit-aware application placement problem for the proposed policy is also solved by following two approaches; in *Optimized Profit-aware placement*, the gateway runs SCIP [2] to find solution for Eq. (14), and in *Heuristic Profit-aware placement* the gateway executes EnhanceProfit procedure from Algorithm 1 to identify application-instance map during each placement round.

6.1. Simulation environment

The experiments for performance evaluation of proposed policy are conducted in a simulated Fog–Cloud environment using iFogSim [16]. Instances of this environment are containerized and their specifications such as network bandwidth, processing speed and expenses are extracted from real-world references [35]. Since the instances offer short-term services, their per unit time price is comparatively higher than the instances provisioned for long-term services [19]. Additionally, to model the placement requests, synthetic workload is used since real-world workload for large number of different IoT applications are not currently available. The arrival pattern of these requests is Poisson distributed and their parametric standards are congruent with the configuration of instances. Numerical values of simulation attributes are determined from their given range in Table 12 through discrete uniform distribution.

6.2. Performance metrics

In experiments, the following metrics are used to evaluate the performance of proposed application placement policy.

1. Percentage of QoS-satisfied applications: The percentage of QoS satisfied applications $\phi^{\wp}$ is calculated as the ratio between number of QoS-satisfied applications $|\phi|$ and total number of requested application $|\Phi|$ per billing period using Eq. (20);

$$\phi^{\wp} = \frac{|\phi|}{|\Phi|} \times 100\%. \tag{20}$$

The higher percentage denotes the improved performance of integrated environment.

2. Average waiting time of applications: Waiting time of an application is defined as the interval between its requesting and placement time. The average waiting time $\bar{\tau^w}$ of QoS-satisfied applications per billing period is calculated using Eq. (21);

$$\bar{\tau^w} = \frac{1}{|\phi|} \sum_{\forall r \in \phi} \tau^r_{\vartheta} - \tau^r_a. \tag{21}$$

The lower the waiting time signifies the higher the performance of integrated environment.

3. Total Gross profit of providers from a Computing Platform: Per billing period, this metric is calculated using Eq. (7). The increased total Gross profit refers to higher balance between service charge and operational cost. It also reflects the efficiency of providers in setting price of the instances.

4. Net profit of providers from a Computing Platform: This metric is calculated by Eq. (11) after each billing period. Enhanced Net profit signifies improved performance of the platform in reducing SLA violation and compensation.

Besides, *Percentage of compensation to Gross profit* is calculated in the experiments by Eq. (22);

$$\rho^{\wp} = \frac{\rho}{\Upsilon} \times 100\%. \tag{22}$$

Proportional relation between $\rho^{\wp}$ and the rate of SLA violation helps to support financial aspects of both users and providers. Moreover, *Average application completion time* and *Average service charge* for Fog and Cloud instances are analysed during experiments to demonstrate the improvement in application's service delivery and justify the applicability of proposed pricing model. *Average time to identify placement map* is also used as an performance metric. Reduced time to identify the placement map denotes the higher feasibility of applying corresponding approach for solving the placement problem.

The aforementioned performance metrics can have a significant impact on any real-world IoT-enabled system. For example, a remote health management system can request an integrated computing environment to place various IoT applications for measuring heartbeat, determining oxygen saturation level, monitoring electrocardiogram pattern and analysing sleep apnea data of different patients [28]. In such circumstance, high percentage of QoS satisfied applications is required for ensuring that the computing environment can offer services for most of the requested applications within their deadline. Similarly, the computing environment should guarantee lower waiting time for all applications so that their execution can initiate in quicker time. Both will help the IoT-enabled system to deal with emergency situations of critical patients. However, the computing environment would need to manage the applications for remote health management system in such a way that can maximize Gross and Net profit of providers. Otherwise, the computing environment will be beneficial only for the IoT-system operators and patients, and there will be no financial support for the providers. Apart from these issues, the high percentage of compensation for increased number of SLA violations will assist the computing environment to attain loyalty of other IoT-enabled systems that can eventually work in favour of the providers.

6.3. Experimental results

The experiments for performance evaluation are conducted by varying the number of requested applications, the number of instances, the percentage of Fog instances and the interval between placement rounds separately. For each variation of these parameters, simulation is run for 500 s and the performance metrics are calculated only when the simulation is over. For simplicity, results of all homogeneous variations for a specific metric are combined in a single two-dimensional graph. These graphs are described in the following subsections.

6.3.1. Impact of varying number of applications

Due to receiving placement requests at higher frequency compared to the available rate of instances, the percentage of QoS satisfied applications decreases (Fig. 3.a). In Profit-aware application

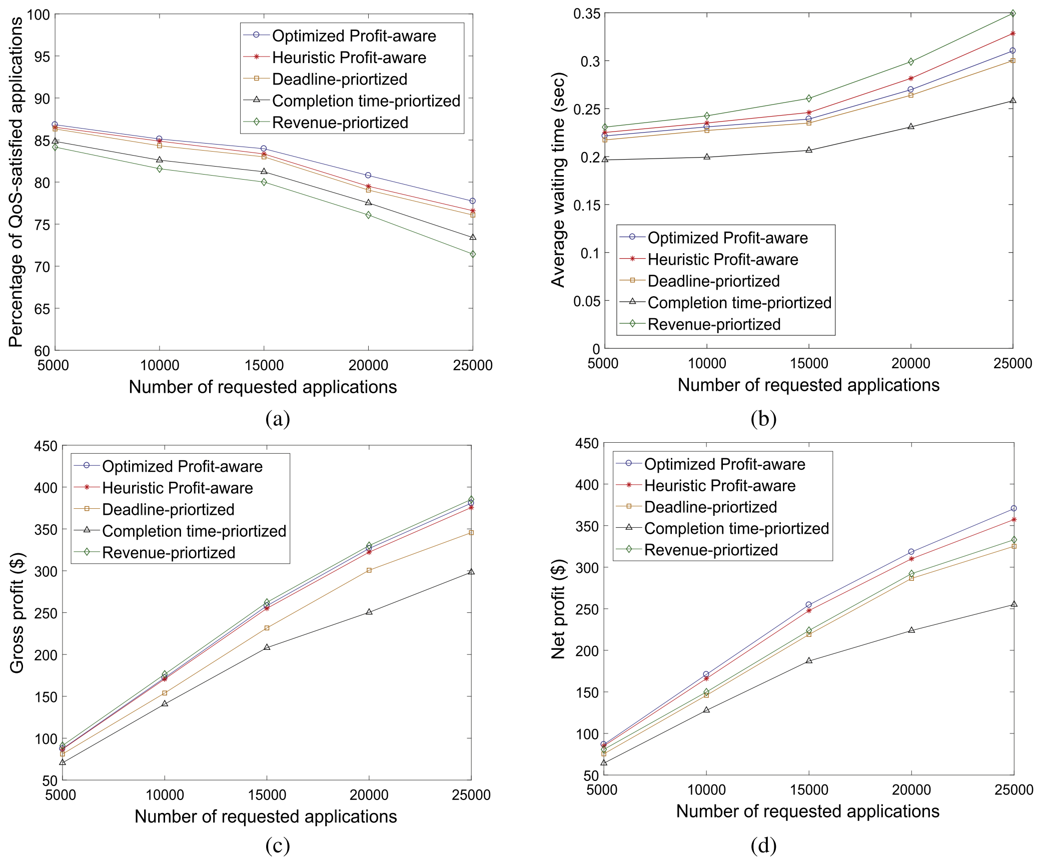


Fig. 3. Impact of varying number of applications on (a) Percentage of QoS satisfaction (b) Waiting time (c) Gross profit (d) Net Profit.

placement, this down trend is slower and closer to the Deadline-prioritized placement policy. The proposed policy schedules applications in precedence of their service delivery deadline. It raises the QoS satisfaction percentage compared to the Completion time and the Revenue-prioritized application placement where priority of applications largely depends on their program size and prospect of earning revenue. Moreover, average waiting time of placed applications prolongs as the number of requested application increases (Fig. 3.b). In this experiment, the proposed policy performs better than the Deadline-prioritized and the Revenue-prioritized policy since it increases the PM value of applications per placement round. However, it awaits latency-tolerant and less economical applications for a longer period of time. Conversely, the Completion time-prioritized placement policy releases instances quickly that consequently helps to place more applications in next rounds and reduces their waiting time.

Since Net profit is accumulated from Gross profit, it always remains greater than Net profit. However, the instance pricing model of the proposed policy helps providers to increase revenue from Fog-based placement of applications that consequently boosts their Gross profit. Hence, the amount of Gross profit with the increasing of applications becomes higher for the proposed policy compared to Completion time and Deadline-prioritized placement policy (Fig. 3.c). Moreover, the proposed policy ensures QoS satisfaction for significant percentage of applications that resists SLA violations and assists providers to pay less compensation. For this reason, provider's Net profit elevates at higher rate in favour of the proposed policy compared to others as the number of applications increases (Fig. 3.d). Lower mount of Gross profit also results in reduced Net profit for Completion time and Deadline-prioritized placement policy. Nevertheless, the Revenue-prioritized placement policy performs almost similar to the proposed policy in increasing provider's Gross profit. Since the percentage of SLA violation rises for the Revenue-prioritized placement policy with the increasing of applications, it leads provider's to pay high compensation. Therefore, despite of elevating Gross profit, it offers less Net profit than the proposed policy.

The impact of varying number of requested applications on different performance metrics signify that placement request receiving frequency of a Computing Platform should be congruent with the availability rate of instances. It helps to maintain acceptable level of provider's profit and waiting time of applications with higher QoS satisfaction of users.

6.3.2. *Impact of varying percentage of fog instances*

In a Computing Platform, if the percentage of Fog instances rises, the percentage of QoS satisfied application increases (Fig. 4.a). The higher number of Fog instances assist more applications to reduce their input propagation delay and to meet QoS. Besides, for explicitly dealing with application's service delivery deadline, the proposed and the Deadline-prioritized placement policy performs better in this case compared to the others.

Similarly, as the number of Fog instances increases, average waiting time of applications increases (Fig. 4.b). Due to charging additional price for using Fog instances, Cloud instances remain as the only option to place low-budget applications. When the number of Cloud instances becomes less, these applications wait for a longer period of time. In this case, compared to other policies, the Completion time-prioritized placement policy performs better as it releases all kinds of instances quickly. Moreover, increased number of Fog instances elevates both Gross and Net profit for providers (Fig. 4.c and 4.d). In these experiments, for concurrently maximizing revenue and reducing SLA violations, the proposed policy performs better than others.

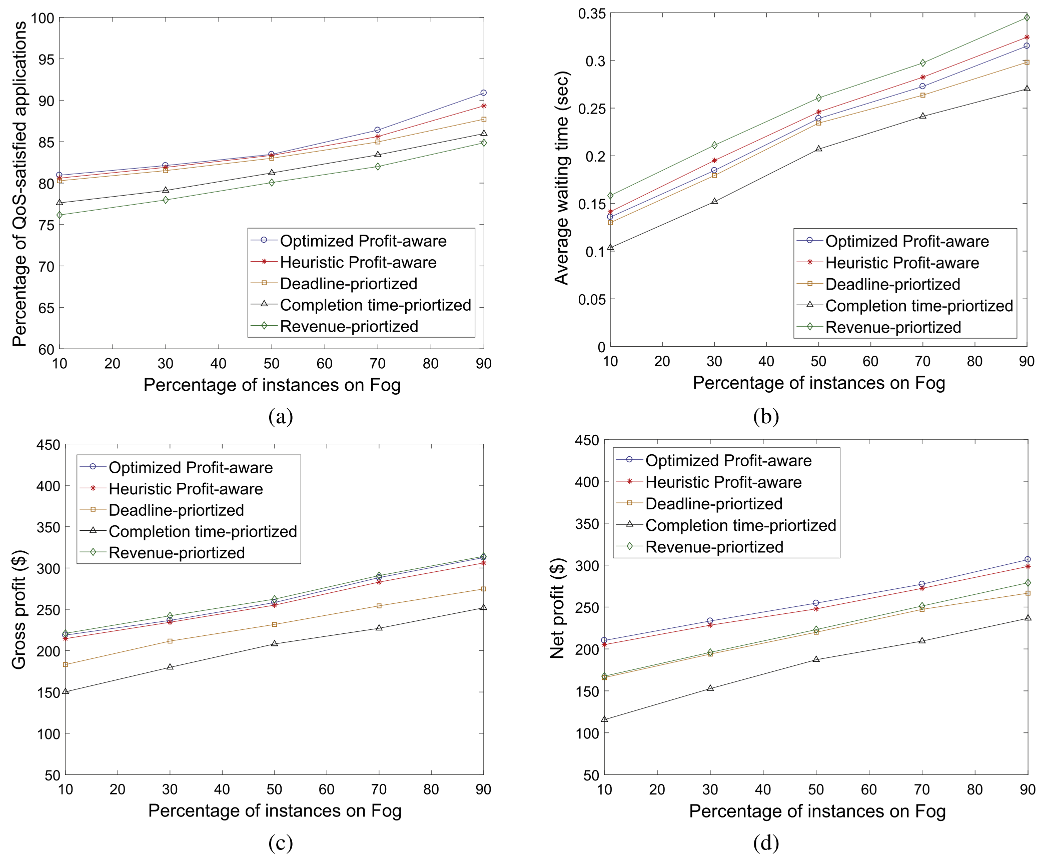


Fig. 4. Impact of varying percentage of Fog instances on (a) Percentage of QoS satisfaction (b) Waiting time (c) Gross profit (d) Net profit.

However, results of the aforementioned experiments signify that a balanced ratio of Fog and Cloud instances assists both profit enhancement and waiting time management.

6.3.3. *Impact of varying placement round interval*

As the interval between two placement rounds increases, the percentage of QoS satisfaction decreases and waiting time of applications increases (Fig. 5.a and 5.b). This interval halts placement of applications even when the instances are already available and resists applications to meet service delivery deadline. However, for considering deadline during application placement, the proposed policy performs better in this experiment than others specially in terms of QoS satisfaction. Moreover, lower QoS satisfaction occurred form increased placement round interval decrease both Gross and Net profit of providers from a Computing Platform (Fig. 5.c and 5.d). Since the proposed policy offers compensation as a variable percentage of provider's total Gross profit, it ensures moderate Net profit despite of higher SLA violation than others.

Outcomes of these experiments recommend tuning the placement round interval in such way that neither creates overhead on gateway by invoking placement process frequently nor degrades the percentage of QoS satisfied requests.

6.3.4. *Feasibility of placement problem solving approaches*

In aforementioned results, the Optimized Profit-aware placement performs slightly better than the Heuristic Profit-aware placement. However, as the number of requested applications increases or the number of instances increases, or both happens due to increasing the placement round interval, the Optimized Profit-aware placement takes longer period of time to identify the application-instance map than the heuristic approach (Fig. 6.a, 6.b and 6.c). Thus, the Optimized Profit-aware placement increases overhead on gateways and degrades their performance. Since the outcomes of both Optimized and Heuristic Profit-aware placement do not vary significantly, for in-time performance, it is feasible to apply the heuristic approach instead of the optimized one to implement the proposed policy.

6.3.5. *Justification for compensation and instance pricing*

The proposed policy facilitates compensation according to the performance of Computing Platform. It ensures high compensation for higher percentage of SLA violation and vice versa (Fig. 7.a). Conversely, in fixed rate compensation method (10% of total revenue), the percentage of compensation to Gross profit remain static despite of performance improvement or degradation [18]. As a result, the Computing Platform fails to show its financial support to both providers and users. Besides, in such method, the distribution of compensation amount is uniform among the affected users on different SLA violation rate. Therefore, the variable rate compensation method of the proposed policy is more conducive to build a financially supportive computing environments for both users and providers than the fixed rate compensation method.

Moreover, it is already proven that Fog-based placement of an application deliver services in reduced time compared to its Cloud-based placement. It happens due to less or negligible input propagation delay while executing an application in Fog instance. The experiment result shown in Fig. 7.b also certifies this fact with almost 55% improved completion time of applications for their Fog-based placement in context of the devised computing environment and proposed placement policy. For this experiment, a candidate set of applications is placed through the proposed policy on fixed number of Fog and Cloud instances separately. Identical configuration is maintained for each instances and workload of the applications are kept unchanged during

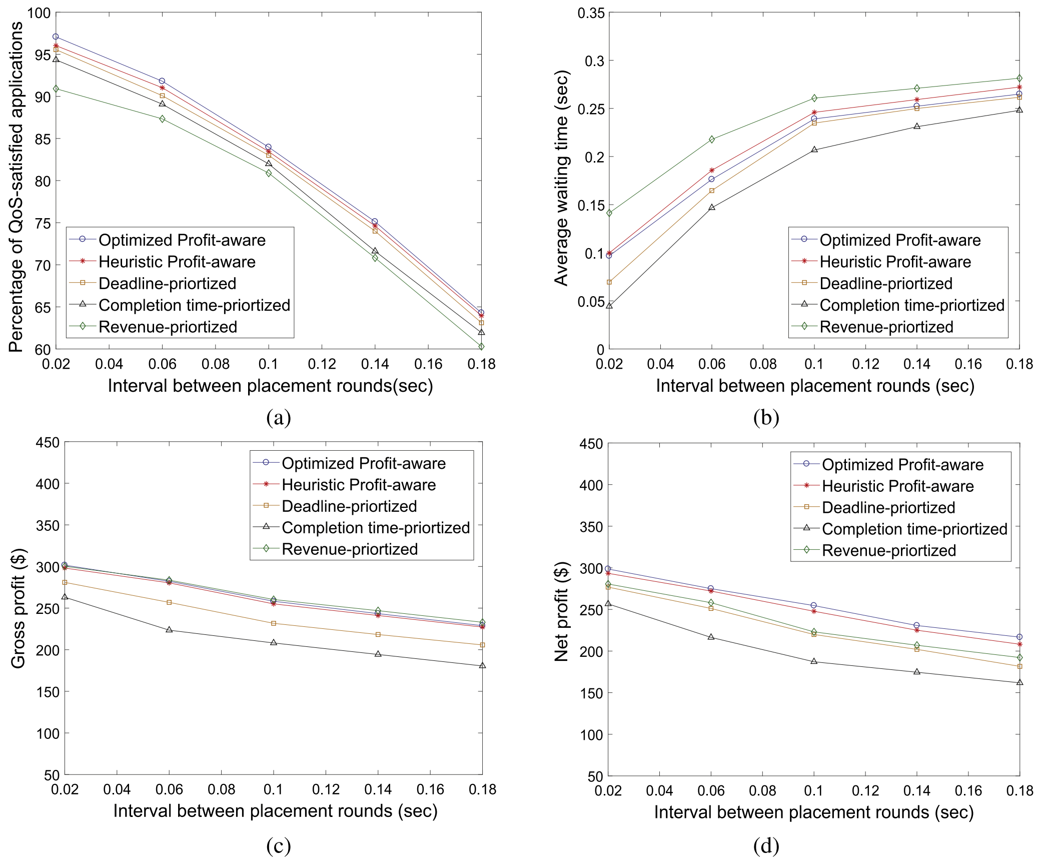


Fig. 5. Impact of varying placement round interval on (a) Percentage of QoS satisfaction (b) Waiting time (c) Gross profit (d) Net Profit.

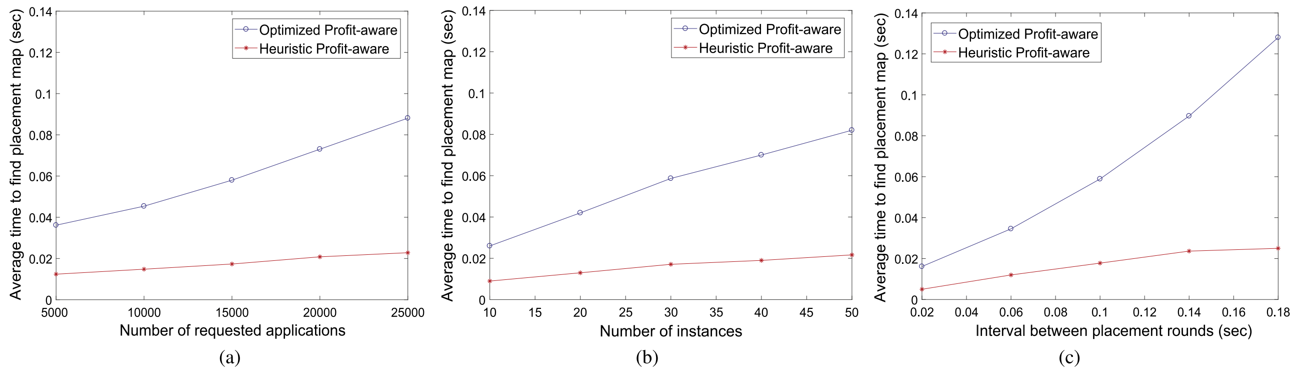


Fig. 6. Required time to find placement map varying (a) Number of requests (b) Number of instances (c) Placement round interval.

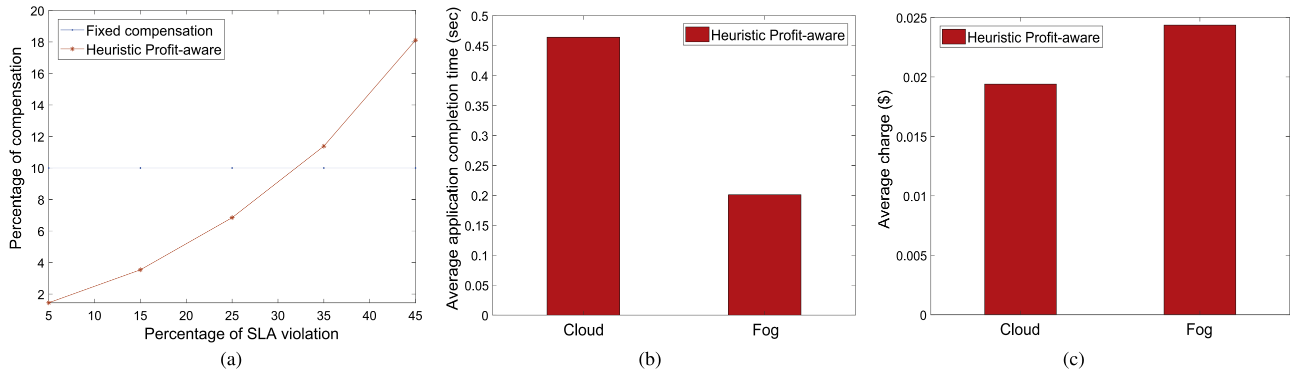


Fig. 7. Comparison between (a) Fixed and variable compensation rate (b) Average application completion time (c) Average charge for Fog and Cloud instances.

the experiment. Thus, its fairness and validity are ensured. On same experimental setup, it is also observed that Fog instances charge additional 20% on average in contrast to Cloud instances (Fig. 7.c), which is quite less compared to the scale of performance improvement. Hence, it is realized that instance pricing model applied in the policy is justified and reflects an acceptable value for improving performance.

7. Conclusions and future work

A profit-aware application placement policy for integrated Fog–Cloud environments is proposed in this work. The policy simultaneously increases provider's Gross and Net profit by placing applications on suitable instances without violating their deadline constraint. It incorporates a pricing model that tunes the service charge of Fog instances according to their capability of reducing application service delivery time. The policy follows a compensation method to mitigate the effect of SLA violation on users. The compensation method depends on the performance of computing environments and supportive for both providers and users. The proposed policy can identify application-instance map by using any ILP solver or best-fit heuristic approach. To demonstrate the efficacy of proposed policy, we applied the heuristic approach in an iFogSim-simulated environment and compared its performance with several existing application placement policies. The experimental results show improvements in Gross and Net profit, waiting time and QoS satisfaction rate for the proposed policy. They also manifested that heuristic implementation of the policy finds closer to optimal solution within minimal time, and pricing of the Fog instances are justified to their performance.

Since Fog computing is a very recent inclusion in computing paradigms, there exists many research challenges including Fog resource management and their resiliency and security maintenance. In this work, we mainly focused on profit-aware application placement in Fog computing that enhances provider's profit and meets application QoS simultaneously through efficient management of Fog instances. However, vulnerable security measures within Fog computing can degrade the efficiency of any Fog resource management policies significantly. Therefore, we plan to extend this work in future incorporating multi-dimensional security aspects so that provider's profit can remain in acceptable level even after any anomaly happens. We will also explicitly explore the expenses for ensuring security in Fog while calculating provider's profit and dynamically determine the placement round interval for IoT applications.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.jpdc.2019.10.001.

References

[1] M. Aazam, E.-N. Huh, Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT, in: Advanced Information Networking and Applications, AINA, 2015 IEEE 29th International Conference on, IEEE, 2015, pp. 687–694.
[2] T. Achterberg, SCIP: solving constraint integer programs, Math. Program. Comput. 1 (1) (2009) 1–41.
[3] M. Afrin, M.R. Mahmud, M.A. Razzaque, Real time detection of speed breakers and warning system for on-road drivers, in: Proc. of the IEEE International WIE Conference on Electrical and Computer Engineering, WIECON-ECE, 2015, pp. 495–498.
[4] M. Afrin, M.A. Razzaque, I. Anjum, M.M. Hassan, A. Alamri, Tradeoff between user quality-of-experience and service provider profit in 5G cloud radio access network, Sustainability 9 (11) (2017) 2127.
[5] M. Al-khafajiy, T. Baker, H. Al-Libawy, A. Waraich, C. Chalmers, O. Alfandi, Fog computing framework for Internet of Things applications, in: 2018 11th International Conference on Developments in ESystems Engineering, DeSE, IEEE, 2018, pp. 71–77.
[6] I.F. All, The big three make a play for the fog, 2018, https://www.iotforall.com/big-three-make-play-fog/ [Online]. (Accessed 06-October-2018).
[7] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties, Springer Science & Business Media, 2012.
[8] K. Barbosa, A. Bucione, A.P. Souza, Performance-based compensation vs. guaranteed compensation: contractual incentives and performance in the Brazilian banking industry, Econ. Apl. 18 (1) (2014) 5–33.
[9] L.F. Bittencourt, J. Diaz-Montes, R. Buyya, O.F. Rana, M. Parashar, Mobility-aware application scheduling in fog computing, IEEE Cloud Comput. 4 (2) (2017) 26–35.
[10] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of Things, in: Proc. of the First Edition of the MCC Workshop on Mobile Cloud Computing, in: MCC '12, ACM, 2012, pp. 13–16.
[11] A.T. Coughlan, K. Joseph, 26 sales force compensation: research insights and research potential, Edward Elgar Publishing, 2012.
[12] 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 Things J. 3 (6) (2016) 1171–1181.
[13] J. Fan, X. Wei, T. Wang, T. Lan, S. Subramaniam, Deadline-aware task scheduling in a tiered IoT infrastructure, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.
[14] L. Gu, D. Zeng, S. Guo, A. Barnawi, Y. Xiang, Cost efficient resource management in fog computing supported medical cyber-physical system, IEEE Trans. Emerg. Top. Comput. 5 (1) (2017) 108–119.
[15] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions, Future Gener. Comput. Syst. 29 (7) (2013) 1645–1660.
[16] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, Ifogsim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments, Softw. - Pract. Exp. 47 (9) (2017) 1275–1296.
[17] L. He, J. Walrand, Pricing and revenue sharing strategies for internet service providers, in: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 1, IEEE, 2005, pp. 205–216.
[18] N. Hogue, Service Level Agreements with Penalty Clause, South Carolina State Documents Depository, South Carolina State Library, 2009.
[19] Z. Hoque, Handbook of Cost and Management Accounting, Spiramus Press Ltd, 2005.
[20] F.M. Insights, Fog computing market: Global industry analysis and opportunity assessment 2017-2027, 2017, https://www.futuremarketinsights.com/reports/fog-computingmarket [Online]. (Accessed 23-June-2018).
[21] A. Kiani, N. Ansari, Toward hierarchical mobile edge computing: An auction-based profit maximization approach, IEEE Internet Things J. 4 (6) (2017) 2082–2091.
[22] B.K. Kwok, Accounting Irregularities in Financial Statements: A Definitive Guide for Litigators, Auditors and Fraud Investigators, Routledge, 2017.
[23] Y. Lin, H. Shen, CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service, IEEE Trans. Parallel Distrib. Syst. 28 (2) (2017) 431–445.
[24] C. Liu, K. Li, K. Li, Minimal cost server configuration for meeting time-varying resource demands in cloud centers, IEEE Trans. Parallel Distrib. Syst. 29 (11) (2018) 2503–2513.
[25] C. Liu, K. Li, C. Xu, K. Li, Strategy configurations of multiple users competition for cloud service reservation, IEEE Trans. Parallel Distrib. Syst. 27 (2) (2015) 508–520.
[26] R. Mahmud, M. Afrin, M.A. Razzaque, M.M. Hassan, A. Alelaiwi, M. Alrubaian, Maximizing quality of experience through context-aware mobile application scheduling in cloudlet infrastructure, Softw. - Pract. Exp. 46 (11) (2016) 1525–1545, spe.2392.
[27] R. Mahmud, R. Buyya, Modelling and simulation of fog and edge computing environments using iFogSim toolkit, in: Fog and Edge Computing: Principles and Paradigms, John Wiley & Sons, Inc. Hoboken, NJ, USA, 2019, pp. 1–35.
[28] R. Mahmud, F.L. Koch, R. Buyya, Cloud-fog interoperability in IoT-enabled healthcare solutions, in: Proceedings of the 19th International Conference on Distributed Computing and Networking, ICDCN '18, ACM, New York, NY, USA, 2018, pp. 32:1–32:10.
[29] R. Mahmud, R. Kotagiri, R. Buyya, Fog computing: A taxonomy, survey and future directions, in: Internet of Everything, Springer, 2018, pp. 103–130.
[30] R. Mahmud, K. Ramamohanarao, R. Buyya, Latency-aware application module management for fog computing environments, ACM Trans. Internet Technol. 19 (1) (2018) 9.
[31] R. Mahmud, S.N. Srirama, K. Ramamohanarao, R. Buyya, Quality of experience (QoE)-aware placement of applications in Fog computing environments, J. Parallel Distrib. Comput. 132 (2019) 190–203.

[32] Y. Nan, W. Li, W. Bao, F.C. Delicato, P.F. Pires, A.Y. Zomaya, A dynamic tradeoff data processing framework for delay-sensitive applications in cloud of things systems, J. Parallel Distrib. Comput. 112 (2018) 53–66.
[33] L. Ni, J. Zhang, C. Jiang, C. Yan, K. Yu, Resource allocation strategy in fog computing based on priced timed petri nets, IEEE Internet Things J. 4 (5) (2017) 1216–1228.
[34] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K.-K.R. Choo, M. Dlodlo, From cloud to fog computing: A review and a conceptual live VM migration framework, IEEE Access 5 (2017) 8284–8300.
[35] C. Pahl, S. Helmer, L. Miori, J. Sanin, B. Lee, A container-based edge cloud paas architecture based on raspberry pi clusters, in: Future Internet of Things and Cloud Workshops, FiCloudW, IEEE International Conference on, IEEE, 2016, pp. 117–124.
[36] X.-Q. Pham, N.D. Man, N.D.T. Tri, N.Q. Thai, E.-N. Huh, A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing, Int. J. Distrib. Sens. Netw. 13 (11) (2017).
[37] N. Raveendran, H. Zhang, Z. Zheng, L. Song, Z. Han, Large-scale fog computing optimization using equilibrium problem with equilibrium constraints, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–6.
[38] D. Rosário, M. Schimuneck, J. Camargo, J. Nobre, C. Both, J. Rochol, M. Gerla, Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support, Sensors 18 (2) (2018) 329.
[39] A.S. Sohal, R. Sandhu, S.K. Sood, V. Chang, A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments, Comput. Secur. 74 (2018) 340–354.
[40] C. Wang, Y. Zhu, W. Shi, V. Chang, P. Vijayakumar, B. Liu, Y. Mao, J. Wang, Y. Fan, A dependable time series analytic framework for cyber-physical systems of IoT-based smart grid, ACM Trans. Cyber-Phys. Syst. 3 (1) (2018) 7.
[41] L. Yang, J. Cao, G. Liang, X. Han, Cost aware service placement and load dispatching in mobile cloud systems, IEEE Trans. Comput. 65 (5) (2016) 1440–1452.
[42] H. Yao, C. Bai, M. Xiong, D. Zeng, Z. Fu, Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing, Concurr. Comput.: Pract. Exper. 29 (16) (2017).
[43] L. Yu, T. Jiang, Y. Zou, Fog-assisted operational cost reduction for cloud data centers, IEEE Access 5 (2017) 13578–13586.
[44] D. Zhao, G. Sun, D. Liao, S. Xu, V. Chang, Mobile-aware service function chain migration in cloud–fog computing, Future Gener. Comput. Syst. 96 (2019) 591–604.

Redowan Mahmud is a Ph.D. student at the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, the University of Melbourne, Australia and awarded Melbourne International Research Scholarship (MIRS) for supporting his studies. He received B.Sc. degree in 2015 from Department of Computer Science and Engineering, University of Dhaka, Bangladesh. His research interests include Internet of Things, Fog and Mobile Cloud Computing.

Satish Narayana Srirama is a Research Professor and the head of the Mobile and Cloud Lab at the Institute of Computer Science, University of Tartu, Estonia. He received PhD in computer science from RWTH Aachen University, Germany, in 2008. His research focuses on cloud computing, mobile cloud, Internet of Things, fog computing, migrating scientific and enterprise applications to the cloud and large scale data analytics on the cloud. He is an Editor of Wiley Software: Practice and Experience.

Kotagiri Ramamohanarao received Ph.D. from Monash University. He was awarded the Alexander von Humboldt Fellowship. He has been at the University of Melbourne since 1980 and was appointed as a professor in computer science in 1989. He was the Head of Computer Science and Software Engineering and Head of the School of Electrical Engineering and Computer Science, University of Melbourne. He is on the editorial boards for Universal Computer Science and Data Mining, IEETKDE and VLDB Journal.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. He is one of the highly cited authors in computer science and software engineering. Microsoft Academic Search Index ranked him as the world’s top author in distributed and parallel computing during 2007–2012. He was founding Editor of the IEEE Transaction on Cloud Computing and is an Editor of Wiley Software: Practice and Experience.