Context-Aware Placement of Industry 4.0 Applications in Fog Computing Environments

Redowan Mahmud, Adel N. Toosi, Kotagiri Ramamohanarao, and Rajkumar Buyya

***Abstract*—The fourth industrial revolution, widely known as Industry 4.0, is realizable through widespread deployment of Internet of Things (IoT) devices across the industrial ambiance. Due to communication latency and geographical distribution, Cloud-centric IoT models often fail to satisfy the Quality of Service requirements of different IoT applications assisting Industry 4.0 in real time. Therefore, Fog computing focuses on harnessing edge resources to place and execute these applications in the proximity of data sources. Since most of the Fog nodes are heterogeneous, distributed, and resource-constrained, it is challenging to place Industry 4.0-oriented applications (I4OAs) over them ensuring time-optimized service delivery. Diversified data sensing frequency of different industrial IoT devices and their data size further intensify the application placement problem. To address this issue, in this article we propose a context-aware application placement policy for Fog environments. Our policy coordinates the IoT device-level contexts with the capacity of Fog nodes and minimizes the service delivery time of various I4OAs such as image processing and robot navigation applications. It also ensures that the streams of input data flowing toward the placed applications neither congest the network nor increase the computing overhead of host Fog nodes significantly. Performance of the proposed policy is evaluated in both real-world and simulated Fog environments and compared with the existing placement policies. The experiment results show that our policy offers overall 16% improvement in service latency, network relaxation, and computing overhead management compared to other placement policies.**

***Index Terms*—Application placement, context-awareness, Fog computing, Industry 4.0, Internet of Things.**

I. Introduction

DEVICE to device connectivity, real-time data access, and advanced automation are rapidly leading the current industrial practice toward its fourth revolution named Industry 4.0. It focuses on building smart industries by enabling robotic assistance, digital twin, and proactive failure management [1]. Internet of Things (IoT) is one of the key elements for Industry 4.0 [2]. Industrial IoT devices generate a huge amount of data per unit time. These data require real-time processing through various industry 4.0-oriented applications (I4OAs) so that different aspects of Industry 4.0 can be achieved [3]. For example, image processing and navigation applications help to launch robotic assistance in the industries. If these applications fail to deliver their services in due time, the performance of industrial robots will degrade significantly. Since Cloud data centers reside at multihop distance from IoT devices, processing of industrial IoT-data using Cloud-based I4OAs is not feasible. It increases data propagation delay, network congestion, and application service delivery time [4]. Therefore, Fog computing aims to overcome such limitations of Cloud-centric IoT-models by harnessing the edge network resources [5].

In Fog-enabled industries, machines with computing processors such as Raspberry PIs, computers, routers, and micro-data centers act as Fog nodes. These nodes offer Infrastructure as a Service (IaaS) like Cloud data centers to assist the execution of different I4OAs [6]. However, Fog nodes are deployed in a distributed manner, and they are heterogeneous in processing speed and networking standards. Additionally, the features of IoT devices such as their data sensing frequency and size of data differ from one to another [2]. These features play vital roles in defining application characteristics. For example, the compute intensity of an I4OA has a proportional relationship with its input data size. Similarly, the network intensity of an I4OA changes based on how frequently the associated IoT devices are sending data to that application [7]. Consequently, these features of IoT devices incite the computational and networking load of host Fog nodes during application execution. If the available capacity of Fog nodes fails to deal with them, network congestion can occur, and the computing overhead of Fog nodes can increase drastically. It also affects the deadline-satisfied service delivery of I4OAs. Hence, it is important to consider these issues while finding the suitable placement option for an I4OA in Fog environments.

Different application placement policies for Fog and other computing paradigms have already been proposed in the literature [8]–[10]. They narrowly exploit data size and sensing frequency of IoT devices while making placement decisions for applications. As a result, they often fail to grasp the characteristics of applications and manage the resources efficiently.

Manuscript received April 30, 2019; revised October 15, 2019; accepted November 4, 2019. Date of publication November 8, 2019; date of current version July 29, 2020. Paper no. TII-19-1646. *(Corresponding author: Redowan Mahmud.)*

R. Mahmud, K. Ramamohanarao, and R. Buyya are with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville VIC 3010, Australia (e-mail: mahmudm@student.unimelb.edu.au; kotagiri@unimelb.edu.au; rbuyya@unimelb.edu.au).

A. N. Toosi is with the Faculty of Information Technology, Monash University, Clayton VIC 3800, Australia (e-mail: adel.n.toosi@monash.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2952412

In some cases, an application is placed on multiple computing nodes, and input data are scheduled to them under the supervision of a centralized entity [8]. When the arrival rate of inputs becomes high, such an application placement policy increases overhead on the centralized entity. It also impels to change the processing destinations of input data very frequently. However, as an alternative, IoT devices themselves can schedule the input data to different replicas of an application. Nevertheless, it increases communication and computation burden for low-energy IoT devices [11]. Thus, in both approaches, application service delivery time degrades. To address these shortcomings, in this article, a context-aware application placement policy for Fog environments is proposed.

Context awareness denotes the ability of a system to deal with the state or contextual information of different entities interacting with the system at any given time and adapt its performance accordingly [12]. In industrial environments, IoT devices, Fog nodes, and I4OAs seamlessly interact with varying data size and sensing frequency, computing and networking capacity, and QoS requirements. Therefore, without context-aware approaches, it is challenging to improve the efficiency of decision-making operations in Industry 4.0. In our proposed placement policy, the sensing frequency and data size of IoT devices are regarded as the device-level contextual information because of their direct impact on Fog node functionalities and application characteristics. Here, based on their implications, the processing and the propagation time of input data for corresponding I4OAs are determined. The proposed policy jointly considers the computation and networking capacity of Fog nodes and the QoS requirements of applications, including their service delivery deadline during application placement. Additionally, it resists the increment of computing overhead on the host Fog nodes and prevents the streams of input data from congesting the network. Thus, it helps to improve service reliability and service time for different I4OAs in Fog environments. The main **contributions** of the article are:

1) proposes a placement policy for I4OAs in Fog environments that optimizes their service time by coordinating IoT device-level contexts with the capacity of Fog nodes;
2) ensures processing of input data streams through placed applications by managing network congestion and computation overhead of host Fog nodes;
3) evaluates the performance of proposed policy in *FogBus*-enabled real [13] and *iFogSim*-based simulated Fog environments [14], and compares it with existing policies in respect of service delivery time, network relaxation and computing overhead management.

The rest of this article is organized as follows. In Section II, related researches are reviewed. Section III provides the system overview and the software architecture of a Fog gateway node. Section IV describes the implication of contextual information in the modeled system, formulates the application placement problem, and discusses our solution. The performance of our policy is evaluated in Section V. Finally, Section VI concludes this article.

TABLE I
SUMMARY OF RELATED WORK AND THEIR COMPARISON

Work	IoT contexts		Meets QoS	Manages overhead	Stable placement
	Sensing rate	Data size			
Haferkamp *et al.* [8]		✓	✓		
Minh *et al.* [9]		✓	✓	✓	
Verba *et al.* [10]	✓	✓	✓		
Lee *et al.* [11]	✓			✓	✓
Lin *et al.* [16]	✓	✓	✓		
Chekired *et al.* [17]		✓	✓	✓	
Wan *et al.* [18]		✓		✓	✓
Pešić *et al.* [21]		✓		✓	✓
Moore *et al.* [22]	✓		✓		✓
Afzal *et al.* [23]	✓	✓	✓	✓	
Gu *et al.* [24]		✓	✓	✓	
This work	✓	✓	✓	✓	✓

II. Related Work

In the literature, there exist several works that highlighted the applicability of Fog computing in Industry 4.0 [6], [15]. Additionally, different placement policies for I4OAs are proposed. For example, Verba *et al.* [10] profiles I4OAs as per their inputs. It helps to place applications with enhanced service time and minimizes the effect of context-variation. Lin *et al.* [16] proposed a Fog node deployment policy in a hierarchical platform that meets the latency and capacity constraints of applications. Similarly, Chekired *et al.* [17] prioritizes the placement of I4OAs based on their latency sensitivity. Wan *et al.* [18] also developed a policy that balances the application execution load on manufacturing components and relates their energy usage with data size.

Not only in Industry 4.0, but the concept of Fog computing has also been extended to other IoT-enabled systems including Healthcare 4.0 [19] and digital agriculture [20]. There exist some application placement policies for such systems. For example, Minh *et al.* [9] proposed a context-aware framework for IoT-Fog-Cloud infrastructure that considers location, application deadline, and resource availability. The application placement policy proposed by Pešić *et al.* [21] deals with the variations of device-level contexts and network topology and places the applications accordingly. Moore *et al.* [22] also provided an application placement policy that engages a centralized entity for context analysis and assists low-latency service delivery of the applications. Afzal *et al.* [23] considered data size and sensing rate of end devices while transferring inputs to applications in energy-efficient and timely manner.

Apart from Fog computing, various application placement policies are also discussed for other computing paradigms. For example, Haferkamp *et al.* [8] developed a policy for cyber-physical systems that exploits payload size and deadline to prioritize the scheduling operations. Lee *et al.* [11] proposed another policy for mobile computing that predicts the launching time of applications and improves the energy usage of smartphones. Gu *et al.* [24] explored the local and remote computation capabilities along with the network condition and latency constraints while placing applications in mobile edge computing environments.

Table I presents a summary comparison of related works with the proposed policy. During application placement, IoT

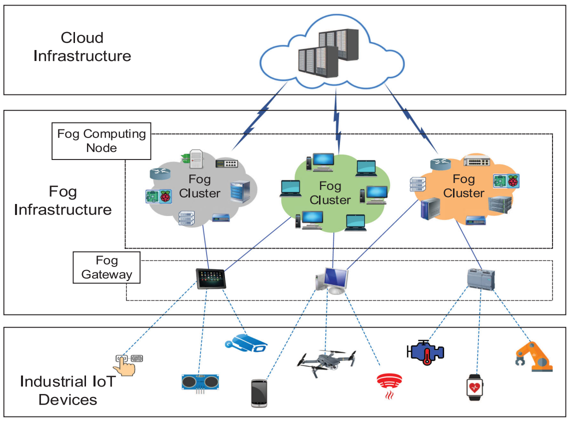

Fig. 1. Computing environment in an industry.

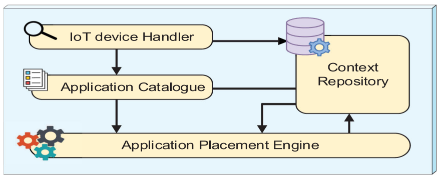

Fig. 2. Software architecture of Fog Gateways.

device-level contexts such as sensing frequency and size of data are not exploited thoroughly in existing works along with the capacity of Fog nodes and application QoS requirements. Furthermore, context parameters are not leveraged to determine the network and computing overhead of Fog nodes. Consequently, they often fail to assist data streams and lead different input of a particular stream to be processed on different Fog nodes. In this article, we address these issues by developing a placement policy for I4OAs. It applies IoT device-level contexts to determine the overhead of Fog nodes and takes the application placement decision accordingly. It ensures application QoS and manages the computational load of Fog nodes. Furthermore, it offers stable placement that resists the changing of processing destination for a particular stream until any context alteration occurs. The proposed policy can also run on different Fog nodes without the supervision of a centralized entity.

III. System Overview

A. Organization of Fog Computing Environments

In industry, IoT devices and Fog Computing Nodes (FCNs) are arranged in the conceptual hierarchical order, as shown in Fig. 1. At the lowest level, IoT devices reside. They sense industrial ambiance and forward data to FCNs for processing through placed I4OAs. Computing capabilities and peer-to-peer communication standards vary from one FCN to another. The applications placed on an FCN can directly access its physical resources through the host operating system. Based on the service outcome of these applications, IoT devices can trigger physical actions through actuators. In either case, service outcomes can be stored for further operations in the future. FCNs can form several clusters among themselves using faster communication protocols such as constrained application protocol and simple network management protocol [4]. In Fog environments, there also exist some Fog Gateways (FGs) that assist the interfacing of IoT devices with the Fog clusters.

IoT devices can subscribe with any of the FGs to launch placement requests for associate applications. They also notify the device-level contexts, such as the sensing frequency and size of data to the FGs. FGs communicate with different FCNs within Fog clusters to grasp their capacity status using RESTful APIs [25]. Later, based on received contextual information of IoT devices, capacity status of FCNs, and QoS requirements of requested applications, FGs identify service delivery time optimized application-FCN placement map. If an FCN is accessible through multiple FGs, their operations on that node are synchronized by the FCN. Whenever the Fog environment becomes overloaded or any latency-tolerant application is requested, the FGs communicate with Cloud data centers to assist them using remote resources. Cloud data centers also help FCNs by offering scalable storage to preserve their accumulated data.

The system model mentioned above facilitates simplified third-party access to I4OAs and Fog clusters. Consequently, it can get affected by security and privacy threats such as impairment of information, disclosure of device identity, replay, and denial of service (DoS) attacks. These threats resist Fog computing to support I4OAs with guaranteed performance. Therefore, we consider the existence of an edge network-based security framework [26] in the modeled Fog environment for securing the application services and infrastructure. We also deem that the system supports preemptive operator migration, request resubmission, and data replication through existing proactive and reactive fault tolerance techniques [27], [28] to resist different anomalies, including request time out, node failure, response error, and application breakdown, and ensures service reliability.

B. Software Architecture for Fog Gateways (FGs)

Fig. 2 presents the software architecture of FGs for context-aware application placement. Its details are given below.

IoT Device Handler: It grasps the context of IoT devices such as the sensing frequency and average size of data, and stores them in a repository. It also narrates the placement requests to a catalogue service and periodically monitors the contextual changes of IoT devices.

Context Repository: It stores the contextual information of IoT devices and the capacity status of accessible FCNs. It also connects Cloud data centers for large-scale storage and maintains a data structure called *PlacementMap* to track which application is placed on which FCN.

Application Catalogue: It contains the details of different I4OAs, including their developer specified execution model, time and space complexity, dependency, and resource requirements. For various inputs, it can also enable the profiling information of an application, such as its processing time and the number of instructions on different FCNs.

Application Placement Engine: It assesses the compatibility of FCNs to host the requested I4OAs based on the contextual

TABLE II
NOTATIONS

Sign	Definition
C	Set of FCNs accessible through an FG.
R	Set of applications requested for placement to an FG.
Z_c	Set of applications placed on FCN $c \in C$.
f^r	Data sensing frequency of IoT devices for application $r \in R$.
$\overline{l^r}$	Average input data size for application $r \in R$.
σ^r	Amount of data dealt by application $r \in R$ in per unit time.
s^r	Number of instructions in application $r \in R$ based on $\overline{l^r}$.
δ^r	Service delivery deadline for application $r \in R$.
μ^c	Instruction execution speed of FCN $c \in C$
λ^c	Network bandwidth of FCN $c \in C$
p_{rc}	Data propagation time to FCN $c \in C$ for application $r \in R$
q_{rc}	Data processing time of application $r \in R$ on FCN $c \in C$
t_{rc}	Service delivery time of application $r \in R$ on FCN $c \in C$
α_{rc}	Networking resource occupancy of application $r \in R$ on FCN $c \in C$ for receiving f^r input data.
β_{rc}	Computing resource occupancy of application $r \in R$ on FCN $c \in C$ for processing f^r input data..
Φ_c	Total networking resource occupancy $\forall r \in Z_c$ on FCN $c \in C$
Ψ_c	Total computing resource occupancy $\forall r \in Z_c$ on FCN $c \in C$
N_r^τ	Set of inputs for application r received in τ amount of time
m_c	CPU usage of FCN c per unit time interval
M_c^τ	Set of m_c values of FCN c monitored for τ amount of time
x_{rc}	Equals to 1 if FCN $c \in C$ is hosting application $r \in R$, 0 otherwise.

information of IoT devices and the capacity status of FCNs. It also initiates the application placement command for the host FCN. Once an application is placed to an FCN, its information is updated on the context repository.

IV. Proposed Application Placement Policy

Based on the implications of contextual information, our proposed context-aware application placement policy identifies application-FCN map and ensures time-optimized service delivery for the requested applications. In a Fog environment, it is executed by the FGs. Basic notations to realize the policy are given in Table II. We have discussed different aspects of our policy in the following subsections.

A. Implications of Contextual Information

The context of IoT devices along with FCN and application-centric information help to determine the data propagation time and processing time for I4OAs. Let R and C be the set of I4OAs and the set of accessible FCNs for an FG, respectively. Data propagation time p_{rc} to an FCN $c \in C$ for an application $r \in R$ is formulated as (1), where average input data size $\overline{l^r}$ is extracted from IoT device-level context and λ^c denotes the network bandwidth of FCN $c \in C$

$$p_{rc} = \frac{\overline{l^r}}{\lambda^c}. \tag{1}$$

Similarly, data processing time q_{rc} for application $r \in R$ on FCN $c \in C$ is calculated using (2). In this case, based on the average size of input data $\overline{l^r}$, the number of instructions s^r in application r is extracted from its profiling information and μ^c refers to the instruction execution speed of FCN c

$$q_{rc} = \frac{s^r}{\mu^c}. \tag{2}$$

Since, the ultimate service delivery of an application ends with either a storage or actuation command, its transferring time within reliable Fog network is considered negligible. Therefore, the service delivery time t_{rc} of application $r \in R$ on FCN $c \in C$ for a single input data is written as (3)

$$t_{rc} = p_{rc} + q_{rc}. \tag{3}$$

These calculations refer that an application requires to occupy networking and computing resources of an FCN for p_{rc} and q_{rc} amount of time to receive and process an input data. *Networking resource occupancy* α_{rc} of application r on FCN c denotes the total amount of time when application r occupies networking resources of that FCN for receiving all its input data sensed in per unit time. It is calculated using (4) where f^r signifies the data sensing frequency of IoT devices for application r. Likewise, using (5), its *Computing resource occupancy* β_{rc} is calculated. It refers the total time that application r requires to process f^r amount of input data occupying computing resources of FCN c

$$\alpha_{rc} = f^r \times p_{rc} \tag{4}$$

$$\beta_{rc} = f^r \times q_{rc}. \tag{5}$$

According to (4) and (5), total networking and computing resource occupancy (Φ_c and Ψ_c, respectively) for all placed applications $r' \in Z_c$ on FCN c is calculated as (6) and (7)

$$\Phi_c = \sum_{r' \in Z_c} \alpha_{r'c} \tag{6}$$

$$\Psi_c = \sum_{r' \in Z_c} \beta_{r'c}. \tag{7}$$

B. Indentification of Placement Map

The applications requested for placement in an industrial scenario can have a diversified level of computing and network intensity. Additionally, their QoS requirements can vary from one to another. Therefore, it is required to focus on a generalized objective for all applications during their collective placement. The proposed context-aware application placement policy resolves this issue by minimizing the service delivery time of applications for each input data. It also helps the policy to deal with the characteristic variations of different applications as the computing and network intensity of applications directly influence the service delivery time. Moreover, the application service delivery time on specific FCN does not vary significantly for each input when the IoT device-level contexts and the load on FCNs remain unchanged. However, in the real world, the placement of multiple applications on a single FCN without considering the effect of different IoT device-level contexts can congest the network and increase the computational overhead of the FCN. As a consequence, service delivery time for all of its occupant applications degrade. Therefore, a balance between their input data admittance and processing on that FCN is required. Furthermore, the application service for each input should be delivered within the deadline to meet QoS. Depending on such facts, we formulate the context-aware application placement as a multiconstrained integer linear programming (ILP) problem as described below.

1) Formulation of Application Placement Problem: Equation (8) signifies the objective function of proposed application placement policy. It minimizes application's service delivery time for each input data and identifies application-FCN mapping through a binary decision variable x_{rc}. Constraints of this ILP problem ensure that an application will not be placed to multiple FCNs (9) and its service delivery time will meet the deadline (10). Furthermore, (11) and (12) refer that networking and computing resource occupancy of all applications placed on an FCN should not surpass the duration of per unit time. Thus, it maintains a balance between input data admittance and processing through the applications within per unit time

$$\min \sum_{r \in R} x_{rc} t_{rc} \tag{8}$$

subject to

$$x_{rc} \leq 1; \quad \forall r \in R \tag{9}$$

$$t_{rc} < \delta^r; \quad \forall r \in R \tag{10}$$

$$\Phi_c + \alpha_{rc} \leq 1; \quad \forall c \in C \tag{11}$$

$$\Psi_c + \beta_{rc} \leq 1; \quad \forall c \in C. \tag{12}$$

Any ILP solver, for example, Solving constraint integer programs (SCIP) [29] can be used to solve this optimization problem and identify the mapping of applications and FCNs. However, in Fog environments, when an FG maintains connections with large number of FCNs and receives placement requests for numerous I4OAs, a longer period of time is required by ILP solvers to solve such optimization problem. It is not acceptable during real-time interactions. Therefore, we propose a heuristic solution to solve the application placement problem.

2) Heuristic Solution for Application Placement: The heuristic solution for application placement is immanent in *PlaceApplication* procedure presented in Algorithm 1. It identifies the FCN for placing an application which ensures least service delivery time for its input. Details of Algorithm 1 is discussed here.

PlaceApplication procedure takes the set of accessible FCNs C and set of applications R requested to an FG g for placement as arguments. It consists of 3 steps:

1) For each application $r \in R$, the amount of data σ^r that an FCN requires to deal with in per unit time for hosting the application is calculated (line 2–3). It refers to the data load of the application which depends on the average size of input data $\overline{l^r}$ and sensing frequency of corresponding IoT devices f^r. An application that deals with huge data load is considered heavyweight and is more likely to promote network congestion and computing overhead compared to lightweight applications having less data load. To reduce the scope of any impediments, it is preferable to place heavyweight applications in earliest convenience than lightweight applications. Therefore, Algorithm 1 sorts all application $r \in R$ on R' in descending order of their σ^r (line 4).

2) For each application $r \in R'$, two variables Υ_r and X_r are initialized (line 5–7). Υ_r stores the minimum service delivery time for application r and X_r tracks the reference of the FCN which delivers the application service in Υ_r amount of time.

Algorithm 1: Application Placement Algorithm.

```
 1: procedure PLACEAPPLICATION $C, R$
 2:     for $r := R$ do
 3:         $\sigma^r \leftarrow f^r \times \overline{l^r}$
 4:     $R' \leftarrow descendingSort(R, \sigma^{\forall r \in R})$
 5:     for $r := R'$ do
 6:         $\Upsilon_r \leftarrow \infty$
 7:         $X_r \leftarrow null$
 8:         for $c := C$ do
 9:             $p_{rc} \leftarrow \frac{\overline{l^r}}{\lambda^c}$
10:             $q_{rc} \leftarrow \frac{s^r}{\mu^c}$
11:             $t_{rc} \leftarrow p_{rc} + q_{rc}$
12:             $\alpha_{rc} = f^r \times p_{rc}$
13:             $\beta_{rc} = f^r \times q_{rc}$
14:             if $t_{rc} < \Upsilon_r$ then
15:                 if $t_{rc} < \delta^r$ then
16:                     if $\Phi_c + \alpha_{rc} \leq 1$ then
17:                         if $\Psi_c + \beta_{rc} \leq 1$ then
18:                             $\Upsilon_r \leftarrow t_{rc}$
19:                             $X_r \leftarrow c$
20:         if $X_r \neq null$ then
21:             $g.PlacementMap(r, X_r)$
22:             $Z_{X_r} \leftarrow Z_{X_r} \cup r$
23:             $\Phi_{X_r} \leftarrow \Phi_{X_r} + \alpha_{rX_r}$
24:             $\Psi_{X_r} \leftarrow \Psi_{X_r} + \beta_{rX_r}$
```

Later, for each FCN $c \in C$, input data propagation time p_{rc}, processing time q_{rc}, service delivery time t_{rc}, networking resource occupancy α_{rc}, and computing resource occupancy β_{rc} are calculated (line 8–13). Based on these calculations, it is checked whether the service delivery time t_{rc} of application r on FCN c is the least or not (line 14). Subsequently other constraints noted in (10), (11), and (12) are also verified (line 15–17). When all constraints are met, Υ_r is updated with t_{rc} and X_r is set to c (line 18–19).

3) For an application $r \in R'$, if X_r refers to an FCN, then r is placed to that FCN. FG g updates its $PlacementMap$ for application r and r is added to the set of applications placed on the host FCN (line 20–22). The total networking and computing resource occupancy for all placed applications on that FCN are also updated (line 23–24).

Whenever an FG receives placement requests for a set of applications, PlaceApplication procedure is executed. If an application is placed to an FCN, it will not be replaced to other FCNs until any device-level contextual parameter for that application is altered. If any alteration happens, the placement request is relaunched. Thus, the procedure helps stabilized placement of applications. However, from line 5 to 24 of Algorithm 1, there are $\mathcal{O}(|R'| \cdot |C|)$ iterations, where $|R'|$ denotes the number of applications requested to FG g for placement and $|C|$ is the number of accessible FCNs through FG g. If any simplified sorting approach such as binary sorting is used to conduct the operation noted in line 4, then the proposed context-aware application placement policy can function with polynomial time complexity.

V. Performance Evaluation

The performance of the proposed policy is evaluated through practical and simulation experiments conducted in FogBus-enabled [13] real-world and iFogSim-based [14] simulated Fog environments, respectively. In the FogBus-enabled setup, a realistic application case scenario is considered that can assist Industry 4.0. However, in simulated setup, synthetic workloads in alignment with this realistic application case scenario are used. The efficacy of our policy is compared with the deadline-prioritized [8], resource-optimized [9], and service time-enhanced [10] application placement policies in both the experimental setup. We have executed the policies separately in a conceptual FG. In deadline-prioritized placement, the applications having stringent deadlines are placed faster compared to others. The resource optimized placement reduces the idle time of FCNs during application execution and the service time-enhanced placement schedules the applications over FCNs by applying the first-come-first-serve principle and improves their service time. In the following subsections, the application scenario, performance metrics, and the details of both experimental setups along with the results are discussed.

A. Application Case Scenario

One of the essential aspects of Industry 4.0 is robotic assistance. In smart industries, for emergency management, different sorts of surveillance equipment such as analog and IP cameras are deployed. Image from these cameras are analyzed by industrial robots to extract the important features of emergency events and take decisions [30]. Since the image quality of analog cameras is ordinary compared to IP cameras, several image processing applications and their services are required to enhance the quality of images before feeding them in robot-embedded image analysis programs. In this article, we consider different image processing applications such as histogram equalization, image noise reduction, and linear contrast adjustment as I4OAs [31]. After processing images using these applications, the enhanced images are forwarded to the industrial robots for further analysis and harnessing robotic assistance in an industry.

B. Performance Metrics

As performance metrics, average service delivery time *Avg. SDT*, average computing resource overhead *Avg. CRO*, and average network relaxation ratio *Avg. NRR* are used in the experiments. The reduced value of Avg. SDT denotes the higher potential of application placement policy. Similarly, the decreased value of Avg. CRO refers to the enhanced performance of the policy in managing computing overhead of FCNs. Conversely, the increased value of Avg. NRR signifies that the policy keeps a stable balance between networking load and networking capacity of FCNs. Avg. SDT and Avg. CRO are determined by following (13) and (14), respectively, where $\tau = 100$ s. Nevertheless, Avg. NRR for an FCN is calculated using (15). Moreover, the percentage of deadline-satisfied inputs *Per. DSI* and required time to identify the placement mapping *TIPM* are

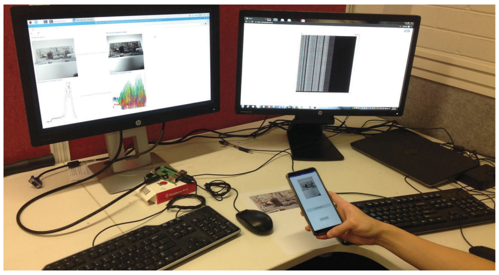

Fig. 3. Real experimental setup.

TABLE III
Specification of Real Experimental Setup

Duration of experiment : 20 minute				
Number of FCNs : 15 Raspberry PIs				
Configuration of FG:				
Processor	RAM	Clock	Uplink	Downlink
Intel Celeron	2 GB	1.60 GHz	2 MBPS	2 MBPS
FCN type ⇒ Configuration ⇓	Raspberry PI 3 A+	Raspberry PI 3 B+	Raspberry PI 3	Raspberry PI 2
System-on-a-chip	Broadcom BCM2837B0	Broadcom BCM2837B0	Broadcom BCM2837	Broadcom BCM2836
RAM	512 MB	1 GB	1 GB	1 GB
Clock	1.4 GHz	1.4 GHz	1.2 GHz	0.9 GHz
Uplink	2 MBPS	2 MBPS	1.5 MBPS	1 MBPS
Downlink	2 MBPS	2 MBPS	1.5 MBPS	1 MBPS
Amount	3	5	4	3
Workload type ⇒ Attributes ⇓	VGA	HD	FHD	QHD
Resolution (Pixel)	640x480	1280x720	1920x1088	2560x1440
Average size (MB)	0.106	0.230	0.358	0.420
Frequency	4	3	2	1
Deadline (second)	0.240	0.320	0.460	0.700

also used here to evaluate the performance of a policy

$$\text{Avg. SDT} = \frac{\sum_{\forall c \in C} \sum_{\forall r \in Z_c} \sum_{\forall i \in N_r^\tau} t_{rc}^i}{\sum_{\forall c \in C} \sum_{\forall r \in Z_c} |N_r^\tau|} \tag{13}$$

$$\text{Avg. CRO} = \frac{1}{|C|} \sum_{\forall c \in C} \frac{\sum_{\forall i \in M_c^\tau} m_c^i}{\tau} \tag{14}$$

$$\text{Avg. NRR} = \frac{1}{|C|} \sum_{\forall c \in C} \frac{\lambda^c - \sum_{\forall r \in Z_c} \sigma^r}{\lambda^c}. \tag{15}$$

C. Experiments on Real Setup

Fig. 3 presents a sample illustration of real experimental setup. Here, different android smart phones are used as the camera-enabled IoT devices. They can capture images in different frequencies and resolution using a customized application. The smart phones are connected to a personal computer which performs the activities of an FG. Furthermore, we deploy several Raspberry PIs as FCNs to form a Fog cluster, and execute the image processing applications. The uplink and downlink speed of FG and FCNs are tuned using the Wondershaper software and they are set to follow a linear relationship with the clock speed of corresponding Fog nodes [25]. Table III presents the details of this setup.

To conduct the experiments in the aforementioned setup, we profile the propagation and processing time of all applications on different FCNs for varying inputs. For instance, Table IV shows the profiling information of histogram equalizing application on

TABLE IV
APPLICATION PROFILING INFORMATION

r = Histogram equalizing application				
Image type ⇒ FCN $c \in C$ ⇓	VGA	HD	FHD	QHD
Raspberry PI 3 B+	p_{rc}=0.052 q_{rc}=0.085	p_{rc}=0.118 q_{rc}=0.154	p_{rc}=0.185 q_{rc}=0.226	p_{rc}=0.218 q_{rc}=0.308

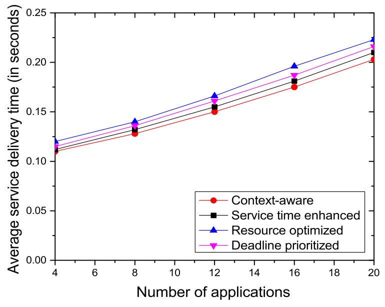

Fig. 4. Avg. SDT versus number of applications.

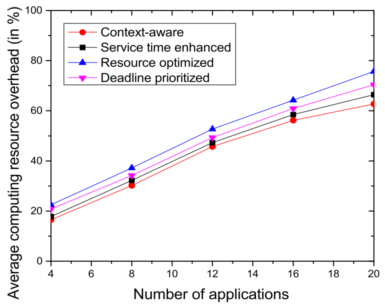

Fig. 5. Avg. CRO versus number of applications.

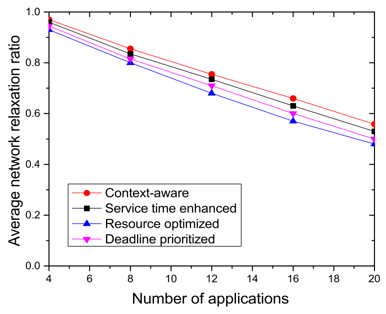

Fig. 6. Avg. NRR versus number of applications.

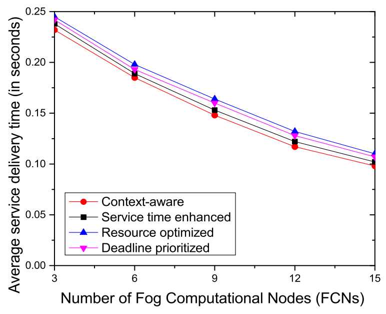

Fig. 7. Avg. SDT versus number of FCNs.

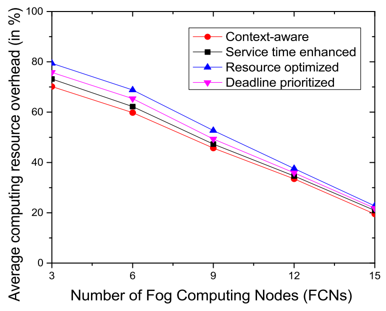

Fig. 8. Avg. CRO versus number of FCNs.

Raspberry PI 3 B+. These information are directly used by the proposed policy while making the placement decisions for any requested applications. The results of experiments are discussed as follows.

1) Impact of Varying the Number of Applications: As the number of applications placed in a certain number of FCNs increases, the Avg. SDT of applications and the Avg. CRO of FCNs elevate (Figs. 4 and 5). It happens due to simultaneous sharing of resources among various applications. However, the proposed policy sorts applications in descending order of their per unit time data load σ^r that implicitly places heavyweight applications on computationally powerful FCNs. As a consequence, the Avg. SDT of applications for this policy remains in the lower values than others. Moreover, our policy makes a balance between the admittance rate of inputs and their processing on an FCN which helps to improve the Avg. CRO of FCNs. Although the service time-enhanced placement shows the similar trend like ours, it often increases the load on computationally powerful FCNs by placing most of the applications over them. Additionally, the deadline-prioritized placement often leads the latency-tolerant applications with higher σ^r to be executed in less computationally powerful FCNs. Both degrade the Avg. SDT of applications and Avg. CRO of FCNs. On the other hand, while dealing with the applications having higher frequency of data, the resource optimized placement significantly increases the computing overhead of FCNs and affects the Avg. SDT and Avg. CRO.

The proposed policy also explicitly measures the effect of data sensing frequency of IoT devices on the networking capacity of FCNs during application placement. Consequently, it helps to offer improved Avg. NRR than other policies where such analysis is narrowly attended. Fig. 6 depicts this aspect of our proposed policy for the increasing number of applications in Fog computing environments.

2) Impact of Varying the Number of FCNs: With the increasing number of FCNs, Avg. SDT of applications and Avg. CRO of FCNs decreases, and Avg. NRR increases (Figs. 7, 8, and 9). For higher number of FCNs, most of the policies exhibit similar trend. However, when there are comparatively lower number of FCNs for application placement, the proposed policy outperforms others. It places applications on limited number of FCNs considering input data size and data sensing frequency of

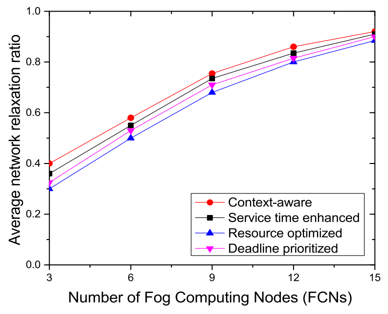

Fig. 9. Avg. NRR versus number of FCNs.

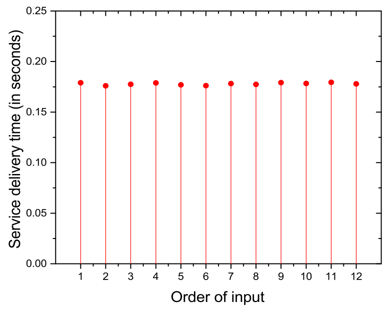

Fig. 10. Service delivery time for different inputs of a stream.

TABLE V
PARAMETERS FOR SIMULATED EXPERIMENTAL SETUP

Parameter	Value
Simulation time	200 Seconds
Sensing duration of IoT devices	1-3 Seconds
Arrival rate of placement requests	10-30 requests/second
Number of FCNs	50
Configuration of FCNs:	
Processing speed	1000-4500 MIPS
RAM	1-2 GB
Downlink bandwidth	1-3 MBPS
Uplink bandwidth	1-3 MBPS
Workload attributes:	
Number of instructions	100 - 1200 MI
Input data size	0.120-0.500 MB
Service deadline	0.300-0.700 seconds
Sensing frequency of IoT devices	1-4 input/second

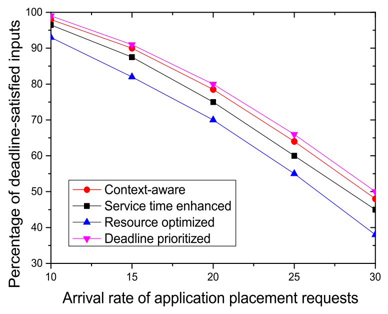

Fig. 11. Per. DSI versus request arrival rate.

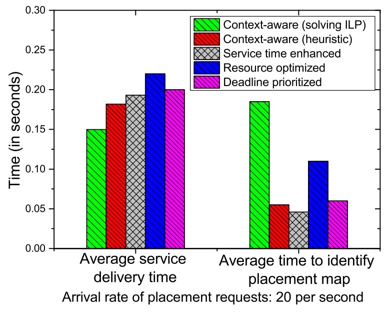

Fig. 12. Avg. SDT and TIPM for different placement policies.

associated IoT devices simultaneously that consequently meets computational and networking commitment of FCNs with their capacity. Thus, despite of having lower number of options for placing applications, computing overhead of FCNs and their networking load do not increase significantly and service delivery time of applications remain in lower values.

3) Observation on Stream Processing: In our evaluation, we monitor application service delivery time for a set of inputs belonging to a particular stream (Fig. 10). In context-aware application placement, service delivery time of inputs do not vary significantly from one to another. Since the proposed context-aware application management reduces the scope of computing overhead and network congestion, the service times for different inputs of a stream remain almost same.

D. Experiments in Simulated Setup

Different parameters used in modeling the simulated setup are listed in Table V. The workload attributes are aligned with the specification of real experimental setup as shown in Table III. Additionally, the configuration of FCNs are set according to the processor benchmarking data provided in [32]. Linear relations are maintained among the parameters of different inputs and FCNs while setting their values from the given range. The simulation experiments are conducted on an *Intel Celeron, 1.60 GHz, 2 GB RAM* configured computer. The results of these experiments are discussed below.

1) Impact of Varying the Arrival Rate of Requests: As the arrival rate of placement request increases, the percentage of deadline-satisfied inputs Per. DSI decreases (Fig. 11). It happens because of the rising number of applications waiting in the queue for execution. However, the deadline-prioritized placement performs better in this case as it immediately executes the deadline-critical applications. Compared to the service time-enhanced placement, our proposed policy offers improved Per. DSI as it does not increase the communication and computation overhead of FCNs unevenly. Conversely, the intention of reducing FCN's idle time often lead the resource optimized-placement to disregard the deadline criticality of applications. As a result, Per. DSI degrades remarkably for this policy.

2) Comparison Between Solution Approaches: The proposed context-aware application placement can be performed either by solving the optimization problem in (8) through any ILP solver or applying the heuristic method of Algorithm 1. Avg. SDT of applications in ILP-based approach is always lower than the heuristic implementation of proposed policy. However, at any arrival rate of placement requests, the heuristic implementation takes less TIPM (time to identify the placement map) compared to the ILP-based approach. Fig. 12 depicts such a scenario when the request arrival is 20 per second. Since the operations of heuristic implementation and the deadline-prioritized placement are almost similar, their TIPM does not very significantly. The TIPM of service time-enhanced policy is lower than others as it does not include any additional sorting operations. Conversely, the resource-optimized placement poses high TIPM because of its extensive search for suitable applications that can reduce the idle time of FCNs.

VI. Conclusion

I4OAs require to offer services in real-time. During the execution of I4OAs, the contexts of IoT devices such as their data size and sensing frequency play influential roles in defining the computing and networking intensity of the applications and help in measuring the processing and communication load of the host Fog nodes. Failure to support them with the capacity of Fog nodes can degrade the application service time. Therefore, in this article, we proposed a policy that implied the IoT device-level contexts to place the applications in Fog environments. It also ensured that the flow of data toward the applications neither increased overhead of Fog nodes nor congested the network deliberately. Thus, it reduced application's service delivery time, relaxed network, managed computing overhead, and increased service reliability. Additionally, the experimental results derived from both real and simulated Fog environments demonstrated the efficiency of our policy.

In the future, we will explore the characteristics of applications such as their execution model, disk and I/O intensity explicitly along with the device-level contexts to place and manage them in Fog environments. Additionally, the hypervisor or docker-based virtualization techniques applied to Fog nodes can affect the service delivery of applications. Therefore, we plan to investigate their effect on the proposed policy as part of its further extension.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Bus. Inf. Syst. Eng.*, vol. 6, no. 4, pp. 239–242, Aug. 2014.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[3] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of industry 4.0: Key technologies, application case, and challenges," *IEEE Access*, vol. 6, pp. 6505–6519, Dec. 2018.

[4] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-aware application module management for Fog computing environments," *ACM Trans. Internet Technol.*, vol. 19, no. 1, Nov. 2018, Art. no. 9.

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

[6] K. Sato and S. Azuma, "Secure real-time control through Fog computation," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 1017–1026, Feb. 2019.

[7] A. Singhvi, S. Banerjee, Y. Harchol, A. Akella, M. Peek, and P. Rydin, "Granular computing and network intensive applications: Friends or foes?," in *Proc. 16th ACM Workshop Hot Topics Netw.*, Palo Alto, CA, USA, 2017, pp. 157–163.

[8] M. Haferkamp, B. Sliwa, C. Ide, and C. Wietfeld, "Payload-size and deadline-aware scheduling for time-critical cyber physical systems," in *Proc. Wireless Days*, Porto, Portugal, 2017, pp. 4–7.

[9] Q. T. Minh, E. Kamioka, and S. Yamada, "CFC-ITS: Context-aware Fog computing for intelligent transportation systems," *IT Professional*, vol. 20, no. 6, pp. 35–45, Nov. 2018.

[10] N. Verba, K.-M. Chao, J. Lewandowski, N. Shah, A. James, and F. Tian, "Modeling Industry 4.0 based Fog computing environments for application analysis and deployment," *Future Gener. Comput. Syst.*, vol. 91, pp. 48–60, Feb. 2019.

[11] J. Lee, K. Lee, E. Jeong, J. Jo, and N. B. Shroff, "CAS: Context-aware background application scheduling in interactive mobile systems," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1013–1029, May 2017.

[12] B. Gu, X. Wang, Y. Qu, J. Jin, Y. Xiang, and L. Gao, "Context-aware privacy preservation in a hierarchical Fog computing system," in *Proc. IEEE Int. Conf. Commun.*, Shanghai, China, 2019, pp. 1–6.

[13] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "FogBus: A blockchain-based lightweight framework for edge and Fog computing," *J. Syst. Softw.*, vol. 154, pp. 22–36, Aug. 2019.

[14] R. Mahmud and R. Buyya, "Modeling and simulation of Fog and edge computing environments using iFogSim toolkit," in *Fog and Edge Computing: Principles and Paradigms*. Hoboken, NJ, USA: Wiley, 2019, ch. 17, pp. 433–465.

[15] X. Yao, H. Kong, H. Liu, T. Qiu, and H. Ning, "An attribute credential based public key scheme for Fog computing in digital manufacturing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2297–2307, Apr. 2019.

[16] C. Lin and J. Yang, "Cost-efficient deployment of Fog computing systems at logistics centers in industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4603–4611, Oct. 2018.

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

[18] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4548–4556, Oct. 2018.

[19] A. Kumari, S. Tanwar, S. Tyagi, and N. Kumar, "Fog computing for Healthcare 4.0 environment: Opportunities and challenges," *Comput. Elect. Eng.*, vol. 72, pp. 1–13, Nov. 2018.

[20] N. Ahmed, D. De, and I. Hussain, "Internet of Things (IoT) for smart precision agriculture and farming in rural areas," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4890–4899, Dec. 2018.

[21] S. Pešić, M. Tošić, O. Iković, M. Ivanović, M. Radovanović, and D. Bošković, "Context aware resource and service provisioning management in Fog computing systems," in *Proc. Int. Symp. Intell. Distrib. Comput.*, Belgrade, Serbia, 2017, pp. 213–223.

[22] P. Moore and H. Van Pham, "FOG computing and low latency context-aware health monitoring in smart interconnected environments," in *Proc. Int. Conf. Emerg. Internetworking, Data Web Technologies*, Tirana, Albania, 2018, pp. 29–40.

[23] B. Afzal, S. A. Alvi, G. A. Shah, and W. Mahmood, "Energy efficient context aware traffic scheduling for IoT applications," *Ad Hoc Netw.*, vol. 62, pp. 101–115, Jul. 2017.

[24] B. Gu, Y. Chen, H. Liao, Z. Zhou, and D. Zhang, "A distributed and context-aware task assignment mechanism for collaborative mobile edge computing," *Sensors*, vol. 18, no. 8, Jul. 2018, Art. no. 2423.

[25] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of experience (QoE)-aware placement of applications in Fog computing environments," *J. Parallel Distrib. Comput.*, vol. 132, pp. 190–203, Oct. 2019.

[26] S. Garg, A. Singh, K. Kaur, G. S. Aujla, S. Batra, N. Kumar, and M. S. Obaidat, "Edge computing-based security framework for big data analytics in VANETs," *IEEE Netw.*, vol. 33, no. 2, pp. 72–81, Mar. 2019.

[27] M. M. Tajiki, M. Shojafar, B. Akbari, S. Salsano, and M. Conti, "Software defined service function chaining with failure consideration for Fog computing," *Concurrency Comput.: Pract. Experience*, vol. 31, no. 8, Apr. 2019, Art. no. e4953.

[28] R. Oma, S. Nakamura, D. Duolikun, T. Enokido, and M. Takizawa, "Energy-efficient recovery algorithm in the fault-tolerant tree-based Fog computing (FTBFC) model," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.*, Matsue, Japan, 2019, pp. 132–143.

[29] T. Achterberg, "SCIP: Solving constraint integer programs," *Math. Program. Comput.*, vol. 1, no. 1, pp. 1–41, Jul. 2009.

[30] M. Afrin, J. Jin, A. Rahman, Y.-C. Tian, and A. Kulkarni, "Multi-objective resource allocation for edge Cloud based robotic workflow in smart factory," *Future Gener. Comput. Syst.*, vol. 97, pp. 119–130, Aug. 2019.

[31] A. K. Vishwakarma and A. Mishra, "Color image enhancement techniques: A critical review," *Indian J. Comput. Sci. Eng.*, vol. 3, no. 1, pp. 39–45, Mar. 2012.

[32] R. Longbottom, "Roy Longbottom's PC benchmark collection," [Online]. Available: http://www.roylongbottom.org.uk

Redowan Mahmud received the B.Sc. degree in computer science and engineering from the Department of Computer Science and Engineering, University of Dhaka, Dhaka, Bangladesh, in 2015. He is currently working toward the Ph.D. degree in engineering at the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC, Australia.

His research interests include Internet of Things, Fog and Edge computing.

Adel N. Toosi received the Ph.D. degree in computer science and software engineering from the University of Melbourne, Melbourne, VIC, Australia, in 2015.

He joined the Faculty of Information Technology, Monash University, Melbourne, Australia, as a Lecturer, in 2018. His research interests include Cloud computing, Fog and edge computing, Internet of Things, software-defined networking, and green computing.

Kotagiri Ramamohanarao received the Ph.D. degree from Monash University, Melbourne, VIC, Australia.

He is currently a Professor with the School of Computing and Information Systems, University of Melbourne, Melbourne, VIC, Australia.

Dr. Ramamohanarao was awarded the Alexander von Humboldt Fellowship. He was the Head of Computer Science and Software Engineering and the Head of the School of Electrical Engineering and Computer Science, University of Melbourne.

Rajkumar Buyya received the Ph.D. degree in computer science and software engineering from Monash University, Melbourne, VIC, Australia.

He is currently a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Melbourne, VIC, Australia.

Dr. Buyya 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.