IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 August 2022, accepted 29 August 2022, date of publication 1 September 2022, date of current version 12 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3203711

RESEARCH ARTICLE

Resource Allocation and Task Off-Loading for 6G Enabled Smart Edge Environments

SYED USMAN JAMIL[1], (Graduate Student Member, IEEE), M. ARIF KHAN[1], (Member, IEEE), AND SABIH UR REHMAN[2], (Member, IEEE)

[1]School of Computing, Mathematics and Engineering, Charles Sturt University, Wagga Wagga, NSW 2650, Australia
[2]School of Computing, Mathematics and Engineering, Charles Sturt University, Port Macquarie, NSW 2444, Australia

Corresponding author: Syed Usman Jamil (sjamil@csu.edu.au)

This work was supported by the School of Computing, Mathematics and Engineering at Charles Sturt University Australia.

ABSTRACT There has been an enormous increase in information flow and communication data due to the rapid rise in the number of Internet of Everything (IoE) devices and the development of cutting-edge technologies such as the rollout of the Sixth Generation (6G) network. The rising and inevitable off-loading requirements of IoE devices have resulted in an unprecedented increase in the reliance on edge and cloud paradigms. However, such a reliance on far-end technologies to access already scarce resources can often result in increased latency and unstable connection issues due to limited bandwidth. In this paper, we investigate the solution for such a stringent network design by presenting a conceptual cloud architecture based on key components such as resource allocation, scheduling and task off-loading for IoE devices. The IoE devices utilise a scheduler to access resources from nearby higher resourced IoE devices for their task computation, where the scheduler allocates incoming requests according to the availability of resources within a cluster of devices or to other devices in nearby clusters. Motivated by these design characteristics, we propose a design of a novel Main Task Off-loading Scheduling Algorithm (MTOSA) for efficient task allocation and dissemination. We present a theoretical analysis of five different scheduling policies namely Round Robin (RR), Strongest Channel (SC), Max Rate (MR), Proportional Fair (PF) and Priority Base (PB) scheduling to find an optimal technique for task off-loading in futuristic networks. Furthermore, we compare the performance of these five scheduling policies with the two existing scheduling policies from the literature. It is shown through various experiments that the proposed MTOSA algorithm performs better when compared with the existing schemes for different performance parameters.

INDEX TERMS Edge-based cloud, 6G, IoE, smart devices, AI, QoS, task off-loading, machine learning (ML), scheduling, resource allocation.

I. INTRODUCTION

Enormous increase of information flow expected in futuristic networks such as 6G communication will mainly be driven by the constant flow of data generated by smart devices and has now given rise to the Internet of Everything (IoE) paradigm [1], [2]. For such a diverse IoE environment, a smart resource allocation management paradigm is vital to efficiently address resource allocation and processing tasks either locally or off-loading to higher resource-enriched clouds [3]. Scheduling policies [4], [5] play a vital role in situations where tasks are outsourced in order to achieve efficient information dissemination. For the best utilisation of locally available resources, the availability of an efficient task scheduler with the ability to optimise task allocation is necessary. In addition to that, the use of a combination of different scheduling algorithms and policies helps to achieve satisfying outcomes. These scheduling policies vary in nature depending on the type of tasks being generated in the edge network.

It is envisioned that 6G wireless communication will rely on smart edge emerging technologies such as Artificial Intelligence (AI) and Machine Learning (ML) to utilise edge-based cloud architectures to transform task scheduling

The associate editor coordinating the review of this manuscript and approving it for publication was Peng-Yong Kong.

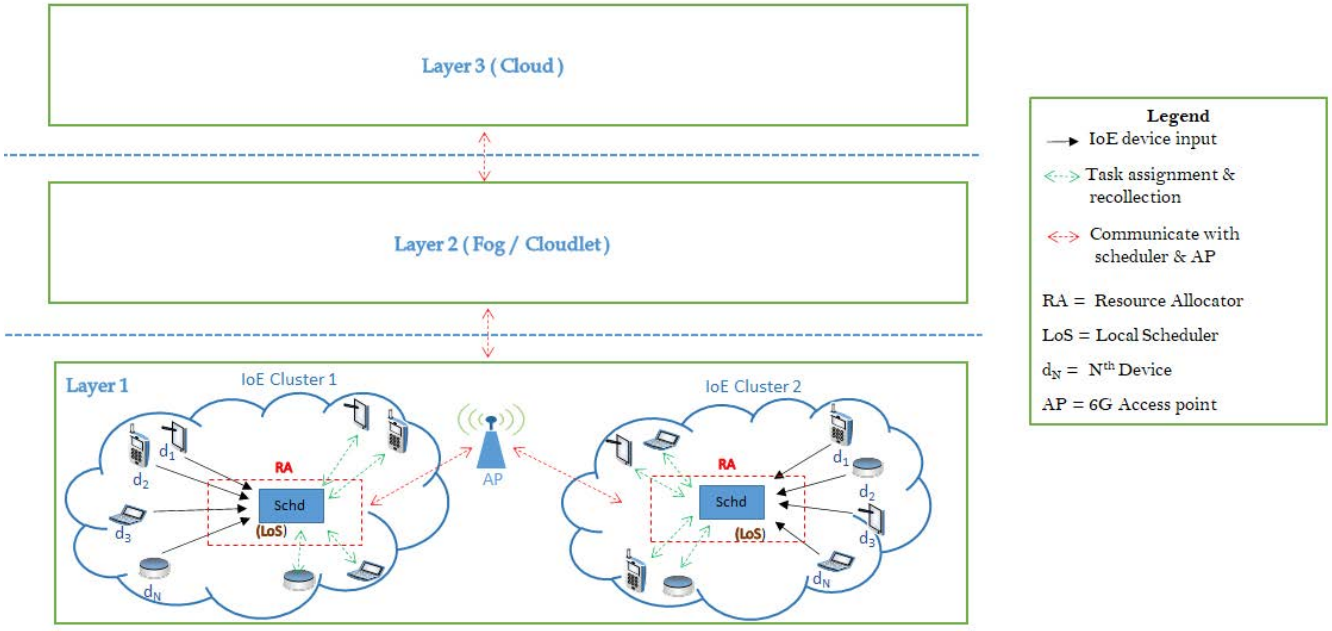

FIGURE 1. Layered architecture model of IoE devices for 6G enabled smart edge environments.

and off-loading for real-time intelligent applications [6], [7]. In [8], authors have proposed an AI-enabled architecture for 6G networks to improve resource management and automatic network adjustment by employing intelligent service provisioning. Authors argue that due to the increasingly complex nature of applications running on 6G networks and their stringent requirements, the 6G networks need to be more revolutionised than their predecessors. In [9] author has presented a task scheduling algorithm based on a Genetic Algorithm (GA) design for a cloud campus platform architecture. The cloud campus platform has been visualised as having users that are connected through virtual machines where resource allocations are dynamically allocated for efficient resource usage. The author has subdivided the problem into two types; first that can be scheduled on the basis of execution time and second that is based on load prediction. In this work, a task scheduling method established upon task scheduling prediction and particle swarm optimisation methods has been utilised by the authors to tackle resource management issues in a cloud architecture.

Recent literature [10], [11], [12] highlights the need of an efficient mechanism to facilitate seamless communication in the resource-intensive futuristic networks. Due to the unprecedented increase in the use of low-power IoE devices and the volume of data it generates, there is a need to develop a state-of-art architecture to support a wide range of applications in order to manage smart environment resources in an efficient and intelligent manner. As highlighted in some of the recently published articles [13], [14], [15], [16], authors have investigated the solutions for some inherent challenges such as minimising energy consumption, the processing efficiency of IoE nodes, task computational and transmission delays due to network congestion, collaborative load balancing approaches for resource allocation. In this article, we investigate a comprehensive solution to address such challenges associated with task off-loading, scheduling and information dissemination in futuristic edge-based networks. We present the detailed elaboration of this architecture in Figure 1 consisting of several IoE devices as key elements. We describe the design components by investigating various scheduling and task off-loading mechanisms within IoE cluster. Using conventional as well as practical scheduling techniques, we evaluate the performance of these algorithms and present a comprehensive theoretical analysis of this architecture. The key contributions of this article are summarised as follows:

1) Presented a comprehensive conceptual design architecture of conventional and practical resource allocation model for IoE devices in a futuristic edge-based 6G network.
2) Formulated the research problem in the proposed system model of the edge-based architecture as a cost/time minimisation problem for the scheduler.
3) Proposed a novel centralised scheduling algorithm, Main Task Off-loading and Scheduling Algorithm (MTOSA), that uses five different scheduling policies namely RR, SC, MR, PF and PB.
4) Evaluated the performance of the proposed MTOSA scheduling algorithm for several practical edge-based scenarios and analysed the results of the algorithm by comparing it with existing schemes.

The rest of the section-wise paper outline is detailed as follows: Section II presents a brief synopsis of recent literature

in this area. Section III describes the system model used in this work and we formulate the central research question by describing the key components of the model in this section. The proposed algorithm working principle is explained in Section IV. Numerical results and evaluation are analysed in Section V. Finally, Section VI concludes the paper.

II. RELATED WORK

A lot of recent research [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43] has been instigated in the domain of task off-loading and resource management to overcome the challenges associated with complex network design. In the following subsections, we present a brief synopsis of recent (i) related articles analytical overview and (ii) a comparative overview in tabular form for meaningful comparison with the proposed algorithm MTOSA.

A. BRIEF OVERVIEW

In this subsection, we review some of the recent significant contributions of different researchers. The analysis helps to signify lessons learnt from existing schemes and highlight unique differences between the proposed work of this paper.

A technological evolution-based survey is presented in relation to the recent challenges in task off-loading in edge network in [17]. Edge computing shifts the functionalities and services to the user's proximity from the cloud. It enhances edge network capabilities regarding data rates, efficient communication, storage, and powerful computing. But, this shift leads to extra challenges such as efficient resource allocation and task scheduling. To address these challenges, the authors in [18] suggested exploring flexible computation models, the attention required on computation migration and considering the nature of tasks through different optimisation variables. Therefore, innovative aspects of this study lead to a flexible resource scheduling and allocation paradigm. Furthermore, it is suggested that the scheduler should employee different scheduling policies to minimise cost and time, based on prioritising tasks according to their nature.

The researchers in [19] modelled efficient and quick off-loading decisions on the basis of optimal computation for the Internet of Vehicles (IoV) architecture. The computation and communication resources required by vehicles at the same instance are argued by researchers as the main reason for such computation off-loading. Therefore, the IoV self-learning scheme based on distributed computation off-loading is proposed with the aid of a fully distributed algorithm to minimise off-loading cost and latency. Task off-loading and scheduling research problem is further analysed in [20] and [21], where the prime focus was to devise solutions to efficiently manage computation and communication resources. Authors have provided the guarantee of the overall system performance while minimising the costs by task allocation in order of arrival through a scheduling algorithm.

In another study [22], a reliable scheduling of the resources in Cloud-Fog environment is implemented through a set of scheduling algorithms and load balancing techniques are utilised to assign specific requests. These requests are classified as time-tolerant, important and real-time. Hence, the scheduling process considered resources failure rate for the provision of high reliability to requested services. So far, different recent studies [23], [24], [25] extensively designed and implemented different sorts of algorithms to optimise communication overheads, focused on minimising time, cost and latency to perform efficient scheduling.

These works [20], [21], [22], [23], [24], [25] considered different scheduling paradigms to minimise time, cost and latency to improve the overall performance efficiency of the system in the task off-loading process. However, there was a lack of emphasis on the local execution of IoE tasks to utilise maximum resources in a cluster by considering the nature of the task. Thus, these algorithms may not fulfil the stringent requirements to optimise task scheduling and allocation in an edge-based futuristic 6G network.

In different real-time scenarios, limited coverage and high cost raise the demand for efficient resource management for transmission in any mode of communication service. The authors explored different mechanisms of device-to-device (D2D) communication and have been presented in [26]. In order to optimise D2D system, the authors considered the willingness of users-end pairing and the performance of the physical link. Similarly, in another study [27] researchers Differentiated Grouping D2D (GD2D) communication model from the traditional D2D communication model. They formulated a resource allocation problem with the aim to guarantee maximum energy efficiency of the system by maintaining the user's Quality of Service (QoS). The convex optimisation problem is transformed into a non-convex optimisation problem to obtain a feasible solution. The researchers have also presented a comprehensive comparative analysis of their technique and showed a better energy efficient solution through their proposed iterative power allocation algorithm.

The authors in [26] and [27] used physical link performance for end-users pairing to improve communication and resource allocation problems in D2D system. Whereas, the proposed algorithms lack scalability and do not perform satisfactorily for tasks of complex nature. Therefore, efficient resource management and task off-loading state-of-the-art schemes are anticipated. Specifically, for end-to-end task computation and ensuring the network's QoS and users Quality of Experience (QoE).

The ultra-dense deployment of 5G specific to IoE will hinder inherent challenges associated with service coverage and limited communication range for futuristic networks such as 6G [28], [29]. To address such limitations, authors in [29] proposed a framework based on mobile resource-sharing through mobile edge-servers to enable edge resource-sharing for 6G cost-effective deployment at the edge. Moreover, task scheduling and path planning as a joint problem is modelled by authors to decouple the resulting and requesting

TABLE 1. Comparison table of existing work in literature.

Papers	Architecture based on								Parameters of interest								
	IoT / IoE	Smart edge	Local execution	Fog	Cloud	6G	Vehicular edge	Internet of Drones	Resource allocation	Scheduling	Off-loading	Multi-objective algorithm	Energy efficiency	Optimisation	Load balancing	Task computation	Intelligence
[17]	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	
[18]	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	
[19]	✓	✓	✓	✓			✓			✓	✓	✓				✓	✓
[20]	✓	✓								✓	✓	✓	✓	✓		✓	
[21]	✓	✓							✓	✓	✓	✓		✓		✓	
[22]	✓			✓	✓				✓	✓		✓			✓		
[23]							✓		✓	✓		✓					
[24]							✓				✓					✓	
[25]		✓							✓		✓		✓	✓			
[26]			✓						✓			✓	✓	✓			
[27]			✓						✓			✓	✓	✓			
[28]						✓											
[29]		✓				✓				✓	✓	✓					
[30]		✓						✓		✓		✓			✓		
[31]							✓	✓		✓						✓	
[32]	✓	✓							✓		✓		✓			✓	
[33]	✓	✓							✓		✓	✓	✓			✓	
[34]	✓	✓				✓				✓	✓	✓		✓			
MTOSA	✓	✓	✓	✓		✓			✓	✓	✓	✓		✓	✓	✓	

edge task off-loading. In order to achieve the overall task efficiency, which is improved through task scheduling and path planning. In comparison, scheduling extended the flexibility to address both task scheduling and movement of the edge servers. Also, the authors proposed a two-layer iterative updating algorithm to provide resource utilisation optimal solutions for IoT systems without needing prior knowledge of task workloads.

In particular, the authors anticipate that 6G technologies [37] remarkable development is mandated for handling massive IoT [38], [39] tasks for complex computations [19], [35], [36] in real-time environments [37], [38], [39], [40], [41], [42], [43]. Thus, an enormous amount of data and the massive number of connections initiated within a cluster of IoE devices are challenging for an edge-based 6G futuristic network.

B. COMPARATIVE OVERVIEW

In this subsection, we provide a comparative overview of the literature in a tabular format (Table 1). Our focus here is to compare various architectures and the relevant performance parameters used in the literature. In particular, we compare what architecture is used in each article and what performance parameters are used to evaluate the performance of proposed algorithms. For clarity and the context in which we considered these architectures and performance parameters, we provide below a brief definition/explanation of each term used in Table 1. We also provide a summary of lessons learned from this study at the end of this subsection.

IoT / IoE: An IoT/IoE architecture is a network of edge devices that communicate with each other for information sharing and various task computations.

Local execution: An architecture based on local execution is defined as a network architecture where tasks from different devices are computed/executed within the cluster without going to the cloud or any other cluster.

Fog: An architecture based on fog is similar to the edge architecture where devices are located far from the centralised access point and nearby the edge of a bigger or more extensive network. The name fog stems from the idea that the actual fog is close to the earth's surface.

Cloud: An architecture based on a cloud is defined as a collection of resources at a centralised location far from the IoT/IoE devices.

6G: 6G are the communication models with the standardised parameters such as frequency, bandwidth, latency, etc., All IoT/IoE devices in a cluster will use 6G communication model.

Vehicular edge: An architecture based on the vehicular edge is defined as an edge network with various vehicles capable of communicating with each other. These vehicles within an edge network can formulate a network and provide task computational capabilities.

Internet of Drones: An architecture based on unmanned aerial vehicles commonly known as drones. These drones can formulate a temporary network at the edge of a bigger network to provide communication services.

Resource allocation: A resource allocation is a parameter of interest within the IoT/IoE network, where a scheduler allocates a device to compute a task.

Scheduling: Scheduling is a mechanism/process to obtain a task from a device, find an appropriate resource and allocate that resource for the computation of the task.

Off-loading: Off-loading is a parameter of interest where a task is assigned to another device for computational or storage purposes.

Multi-objective algorithm: An algorithm that has more than one objective to satisfy under certain constraints.

Energy efficiency: Energy efficiency is a parameter of interest that defines how efficiently and effectively a task is computed by a device. This parameter commonly relates to the battery life of a device in the network.

Optimisation: Optimisation is a process to achieve the best possible outcome of a given problem.

Load balancing: Load balancing is a parameter of interest where a scheduler allocates various tasks among different devices evenly so that a single device may not be getting most of the tasks for computation.

Task computation: Task computation is a parameter of interest where a task allocated to a device is completed successfully from the allocation to execution and returns to the original device.

Intelligence: Intelligence is a parameter of interest where a scheduler may use various characteristics of the devices, such as battery level, task computing history etc., to make the scheduling decisions.

Lessons Learned: It is obvious from the Table 1 that there is a lack of research work that focuses on the local execution of tasks for IoE devices in a fog architecture. Furthermore, there is also a lack of incorporating intelligence as defined above in scheduling and task off-loading processes for an IoE-based edge network. On the other hand, several articles have focused on over-arching multi-objective task off-loading and resource scheduling algorithms. However, they have neglected to design an overall efficient algorithm to find an optimal solution for the resource allocation, scheduling and task off-loading problem. On the architectural side of the networks, there is a significant research interest in the IoT, IoE and smart edge-based networks. It is evident that the futuristic 6G communication network is gaining a lot of attention as an emerging area of research.

Contrary to previous work, we consider a cluster based IoE network and designed a scheduling mechanism that uses several scheduling algorithms for task computation. This work aims to propose an overall solution that minimises cost and time for task scheduling to manage resources efficiently. This proposed algorithm is called MTOSA. The designed algorithm MTOSA leverages the benefits of five different scheduling policies to compute a task efficiently and effectively. These scheduling policies leverage optimal scheduling through optimisation and load balancing for various IoE tasks.

III. SYSTEM MODEL

In this section, we describe the system model used in this paper. Earlier in [44], we presented detailed design of a conventional resource allocation model for IoE devices in 6G environment. The proposed architecture in this paper consists of an IoE based fog network where multiple IoE devices want to compute their various tasks locally. This architecture aims to efficiently compute IoE tasks locally while minimising communication with the servers at the external cloud. It should be noted that the proposed architecture tries to solve the task computation. The long-distance geographical distribution of IoE devices is sub-divided into several IoE clusters. Each IoE cluster has a number of IoE devices within close proximity that can communicate with each other directly. The communication within an IoE cluster and between the clusters is assumed to be using 6G based wireless communication networks.

To handle and allocate tasks for computation to different devices, a computing entity calls the scheduler is available within each IoE cluster. The responsibilities of the scheduler are to collect the tasks from IoE devices for computation, keep up-to-date information about the tasks dispersion and the capability of IoE devices to compute such tasks, allocate the tasks to these available and capable IoE devices, re-collect the computed tasks and re-transmit the results to the original devices. This proposed architecture takes the leverage of having a main task off-loading and scheduling algorithm at the scheduler that has a number of scheduling policies available at its disposal, depending on the objectives of task scheduling. The proposed architecture tries to compute the tasks within the IoE cluster as much as possible. This architecture has the advantages of local task computing, low latency, minimal use of backbone bandwidth and improved quality of service for IoE devices. A complete description of this architecture is given in Figure 1.

Our discussion in this section evolves around the IoE architecture as highlighted in Figure 1 and is focused on the design of the IoE cluster as visualised in layer 1 of this figure. We start our discussion by describing the need of the scheduler for efficient task computation for such a stringent network in the coming subsection. Readers are encouraged to refer to Table 2 that describes the symbols and notations used throughout this article. Furthermore, in the following, we define the entities, communication and assumptions that are frequently used in this article.

Entities: The entities in this paper are considered as IoE devices, scheduler, resource allocator, communication links, 6G Access Point (AP) and types of resources such as storage, information and computation.

Communication: The 'communication' in this paper is defined as the data transmission between IoE devices and the scheduler. This communication happens through multiple wireless links using 6G radio spectrum. These links are considered to work in full-duplex mode and provide ample

TABLE 2. Symbols and notations.

Variable	Description	Variable	Description
$\mathcal{D}$	Set of IoE devices	$\mathcal{T}$	Set of tasks
$m_r^{d_i}$	Binary message	$Schd$	Scheduler
$Sched$	Scheduling algorithms	LoS	Local scheduler
LaS	Layer Scheduler	RA	Resource allocator
$\mathcal{D}_A$	Set of available devices	T	Set of time slot
$\mathcal{R}$	Set of resources	$\mathcal{L}$	Set of communication links
$\mathcal{R}^{typ}$	Resource type	$\mathcal{R}^{stg}$	Resource storage
$\mathcal{R}^{inf}$	Resource information	$\mathcal{R}^{cpt}$	Resource computation
N	Number of devices	$\mathcal{T}_i$	i^{th} task
$\mathcal{T}_s$	Scheduled task	d_i	i^{th} device
d_s	Scheduled device	d_a	Available device
$\lvert\mathcal{T}\rvert$	Total number of tasks	d^*	Selected / scheduled device
$\mathcal{T}_{i\_schd}$	Task scheduling queue	$\mathcal{T}_{i\_schd\_immd}$	Task scheduling immediate queue
$d_{a\_schd\_immd}$	Device scheduling immediate queue	$d_{a\_schd\_wait}$	Device scheduling waiting queue
d_{a_schd}	Device scheduling queue	$\mathcal{T}_{i\_schd\_wait}$	Task Scheduling wait queue
t_i^f	Forward transmission time	$\mathcal{P}_i^{Sched}$	Scheduling research problem
T_{total}	Total task execution time	t_{schd}^f	Forward schedule time in queue
t_s^f	Forward transmission time of the scheduled task	t_c	Computational time required by the d_s
t_s^b	Transmission time from d_s to scheduler, where b presents backward direction communication	t_{schd}^b	Time taken by the scheduler to send the computed task back to original device d_1
t_i^b	Backward transmission time of task	$t_{exp}^{\mathcal{T}_i}$	Expiry time of a task
AP	Access Point	$\mathbf{H}$	Wireless channel matrix
$\mathbb{C}^{m\times n}$	Complex number $m \times n$	BW	Channel bandwidth (Hz)
$\widetilde{r}_i(t)$	Instantaneous data rate	$\overline{r}_i(t)$	Average aggregated data rate
ξ_i	Signal to Interference plus Noise Ratio (SINR)	r_i	Maximum channel rate
σ^2	Additive White Gaussian Noise (AWGN)	ρ	Transmit power
C	Cost function	p	Priority level (high = 3, medium = 2 and low = 1)
$C_p(\mathcal{T}_i)$	Cost associated with a task (time)	$\lvert\lvert.\rvert\rvert$	Norm representation

capacity for stable data transmission between IoE devices and the scheduler.

Assumptions: There are various assumptions made in this paper to formulate a near practical fog based network. It is assumed that the IoE devices in the fog network have different tasks that cannot be computed by the devices themselves. A scheduler is required to schedule these tasks to an appropriate device. The devices predefined these tasks and have a specific size and time to live (TTL), which is considered ten seconds for each task. The arrival of tasks at the scheduler is independent of each other, and the scheduler can hold the task for a certain period of time if required while the tasks are being scheduled. It is further assumed that the network has sufficient wireless communication links, and each device can communicate with each other and the scheduler without having a communication bottleneck. We also assume that all scheduled tasks are computed within the IoE cluster, which means that the task allocation between IoE clusters is not required at this stage and will be investigated in future work.

A. PRELIMINARIES

We start by explaining the need of an appropriate scheduler with the help of two scenarios as elaborated in Figures 2 (a) and (b). In the first scenario, as shown in Figure 2(a), we assume there are four IoE devices d_1, d_2, d_3 and d_4 in the system. These devices want to compute their tasks without the help of any scheduler. Let us assume that IoE device d_1 has a task $\mathcal{T}_1$ that requires computation which cannot be done at the IoE device d_1 itself. Therefore, IoE device d_1 requires support from other devices to compute its task $\mathcal{T}_1$. The IoE device d_1 initiates a request message m_i^1 (where subscript i represents the initiated message and superscript 1 denotes the device number) that is forwarded to all other devices (d_2, d_3 and d_4) to check whether they have the required resources to compute this task $\mathcal{T}_1$ in consideration. This communication from d_1 is shown with solid lines in Figure 2(a). In reply to this request (shown with dotted lines in Figure 2(a)) each IoE device d_2, d_3 and d_4 (except d_1) will send a binary message, $m_r^{d_i}$ $(i = 2, 3, 4)$ to d_1,

$$m_r^{d_i} = \begin{cases} 1, & \text{resources available} \\ 0, & \text{resources not available} \end{cases} \tag{1}$$

The device d_1 then decides to send task $\mathcal{T}_1$ for computation to the available devices. In this scenario, IoE device d_1 is processing all communication to and from other devices directly. This depicts a simple task computation scenario without any scheduler involved.

However, consider a complex scenario, where each IoE device d_i where $i = 1, 2, 3, 4$, has a task $\mathcal{T}_i$ that requires some computation. Each IoE device d_i requires a different IoE device to perform this computation. This scenario is shown in Figure 2(b). All IoE devices will be sending and receiving requests for task computation from each other. For example, as shown in Figure 2(b), the IoE device d_1 is required to handle twelve different communications simultaneously, and

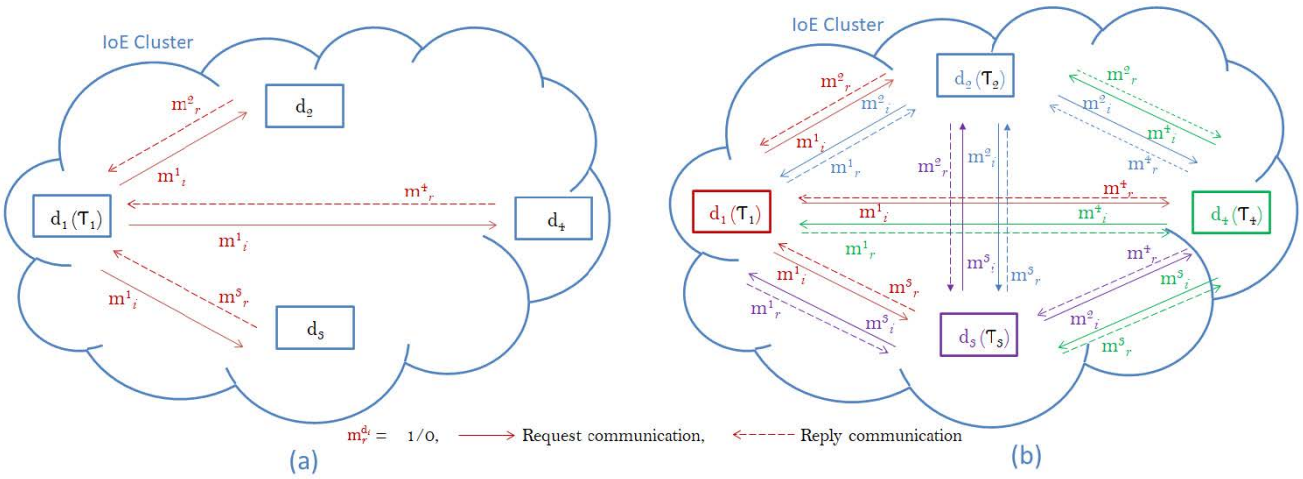

FIGURE 2. Preliminaries: Layered architecture model of IoE devices for 6G enabled smart edge environments.

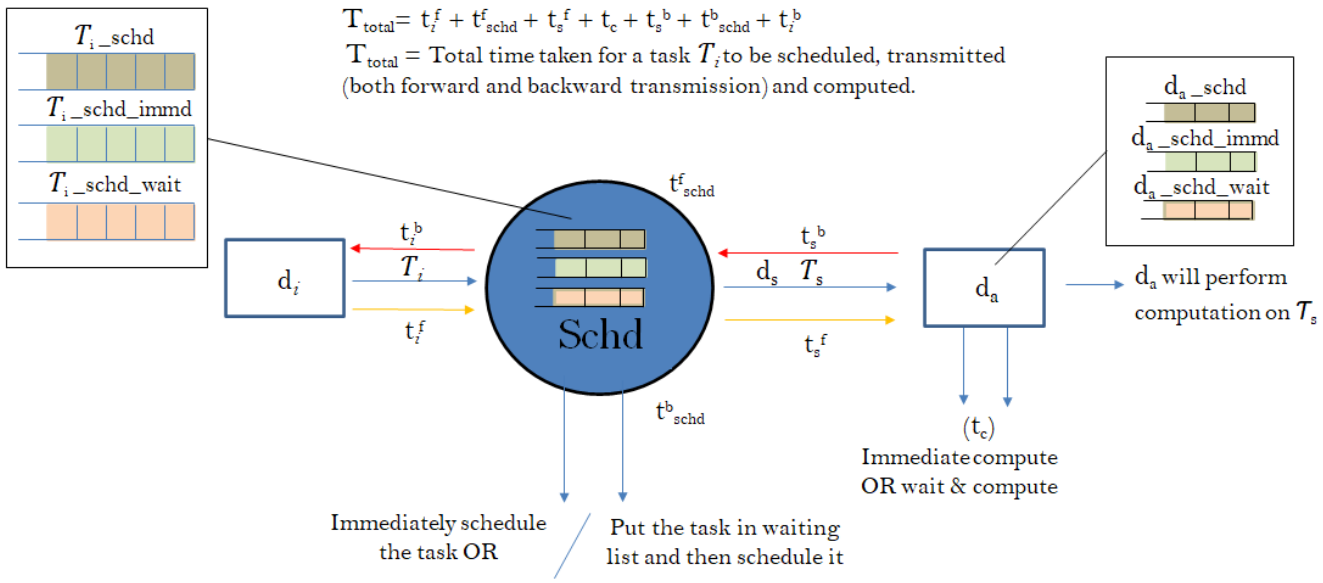

FIGURE 3. An example illustration of a centralised scheduler.

the same is true for the remaining IoE devices as well. In our research scenario, where we have IoE devices with limited communication and computational powers, handling such a complex communication scenario for these devices is a challenging task. Therefore, it is appropriate to have a centralised entity, such as a scheduler, that can handle this communication for all the devices within an IoE cluster. An example illustration of such a scheduler is shown in Figure 3.

This illustrative scheduling example has input tasks $\mathcal{T}_i$, a queuing system to store tasks when required and information on devices that are able to compute the input tasks. A scheduler of such a design is placed at the centre of an IoE cluster. Let us assume there is a set of IoE devices $\mathcal{D}$ as defined below that want their tasks to be scheduled for computation.

$$\mathcal{D} = \left\{d_1, d_2 \ldots\ldots\ldots d_N\right\}. \tag{2}$$

The set of tasks $\mathcal{T}$, to be computed is defined as below:

$$\mathcal{T} = \left\{\mathcal{T}_1, \mathcal{T}_2 \ldots\ldots\ldots \mathcal{T}_N\right\}. \tag{3}$$

Let us assume that the scheduler selects an IoE device, $d_i \in \mathcal{D}$ and its task $\mathcal{T}_i$ to be scheduled at a particular instant of time. We assume that all tasks and their arrival at the scheduler are independent of each other.

The scheduler also selects a device d_a s.t. $d_a \in \mathcal{D}$ and $d_a \neq d_i$ to allocate the selected task $\mathcal{T}_i$. The task $\mathcal{T}_i$ then arrives at the scheduler and it decides whether the task needs

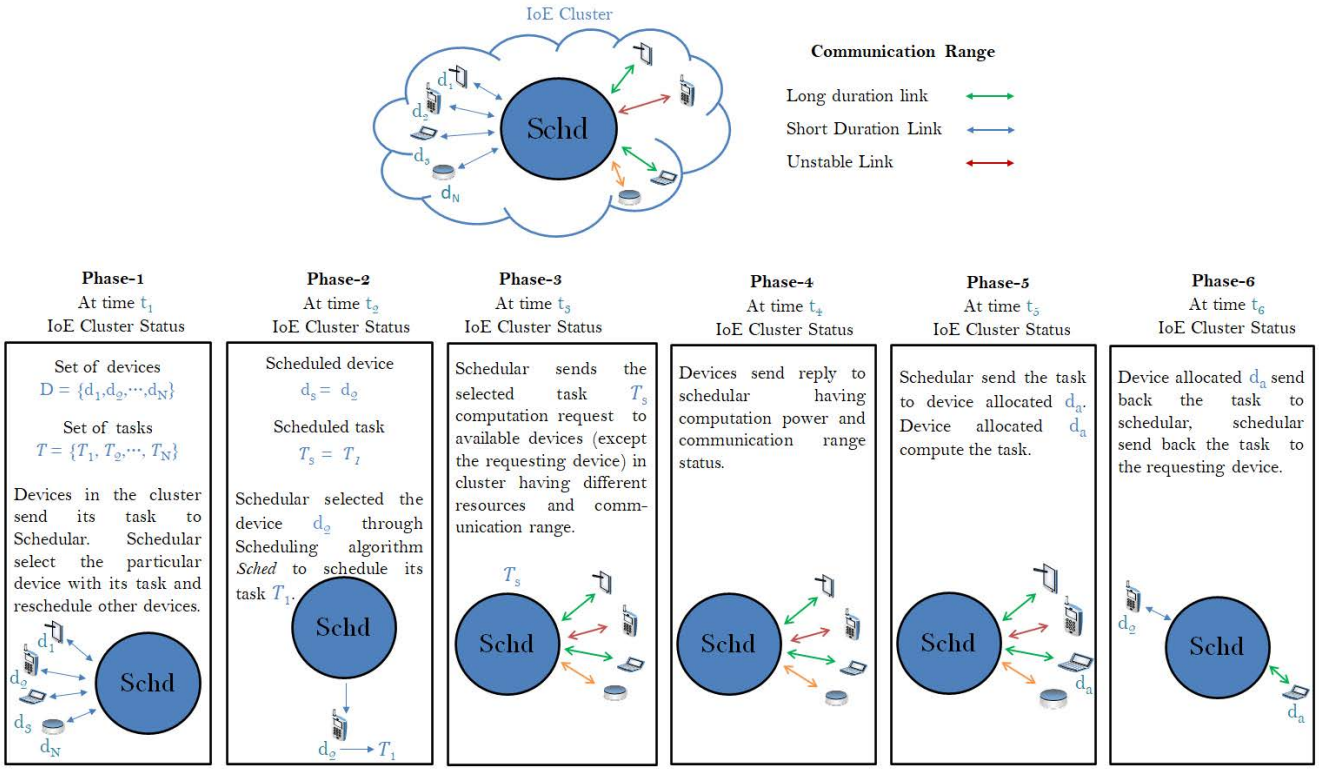

FIGURE 4. Task selection and computational phases.

to wait within the scheduler or it can be scheduled immediately. This depends on the queue status of the device d_a. Once the task is transmitted to d_a, the task may need to wait in the queuing system of d_a for the computational resource to be available. Therefore, the task $\mathcal{T}_i$ is computed at d_a, it is transmitted back to the scheduler where the scheduler again decides to immediately transmit it back to d_i or keep it in the queuing system until d_i is ready to receive. This process explains task scheduling, off-loading and computation of a single task $\mathcal{T}_i$. The scheduler is generally capable of handling multiple tasks simultaneously.

Motivated by the above discussion, in subsequent subsections, we describe various components in an IoE cluster as shown in the proposed architecture (see Figure 1 and 2) to formulate the research problem.

B. IoE CLUSTER COMPONENTS

Components of an IoE cluster, given in our proposed architecture can be divided into two categories. The first category represents the hardware-based components, whereas the second category represents conceptual components. The hardware components are IoE devices and resource allocator. The conceptual components in the IoE clusters are tasks generated by devices, scheduling algorithms, communication links and computational resources.

IoE devices in the cluster can be of different types and can have varying computational powers. Examples of such devices include smart mobile devices, sensors, actuators and any other participating device having little computing and storage capability. They have different hardware architectures, processing abilities, storage capacities, power supply and operating systems. Some high-end devices may be equipped with AI chips, but their computing/storage capacity may still be limited. Also, each IoE device has a different life cycle. We consider a set of devices $\mathcal{D}$ within an IoE cluster where each device does generate tasks $\mathcal{T}_i$ that require some computation. The set of tasks $\mathcal{T}$ corresponding to the set of devices $\mathcal{D}$ is represented as per equation (3).

C. RESOURCE ALLOCATOR

Each IoE cluster in the proposed architecture has a resource allocator, which combines the functions of a scheduler and resource allocation unit. The scheduler has a set of available scheduling algorithms as shown in equation (4) below. The function of the scheduler is to select a specific algorithm from this set that optimises the objective function.

$$Sched = \Big\{RR, SC, MR, PF, PB\Big\}. \tag{4}$$

Depending on the location of the scheduler, it can be divided into two categories as a local scheduler (LoS) and a layer scheduler (LaS).

Local Scheduler (LoS): This scheduler is located within the IoE cluster of layer 1. This scheduler uses multiple scheduling algorithm as per the need. For the sake of brevity,

let $Sched_i$ represent a scheduling algorithm from equation (4) that can be used in LoS. Let $i = 1, 2, \ldots M$ where M is the maximum number of scheduling algorithms as given in equation (4).

Layer Scheduler (LaS): This scheduler is located between the layers i.e., Layer 1, Layer 2 and Layer 3, hence named layer scheduler. The role of LaS is to schedule tasks between layers when the tasks cannot be computed locally within the layer.

Resource: Each device in an IoE cluster has some computational capability called resource. Some common types ($\mathcal{R}^{typ}$) of these resources are storage ($\mathcal{R}^{stg}$), information ($\mathcal{R}^{inf}$) and computation ($\mathcal{R}^{cpt}$). The set of available resources $\mathcal{R}$ is represented as:

$$\mathcal{R} = \Big\{\mathcal{R}_1, \mathcal{R}_2 \ldots\ldots\ldots \mathcal{R}_N\Big\}. \tag{5}$$

The resource allocation part of the system model is responsible for scheduling computational tasks to the potential devices with computational resources available and afterwards getting the computed task back from the devices and returning the information to the original IoE device as shown in Figure 4.

D. COMMUNICATION MODEL

The communication model used in the design is assumed to be based on 6G technology. As illustrated in Figure 1, multiple wireless links between IoE devices and the scheduler are considered, as well as between the scheduler and AP. We represent such a wireless link as $\mathcal{L}_{(i,j)}$ where (i, j) represents a source and a destination device pair and this link can support a full duplex mode of communication. A set of such communication links in an IoE cluster is represented as:

$$\mathcal{L} = \Big\{\mathcal{L}_1, \mathcal{L}_2 \ldots\ldots\ldots \mathcal{L}_N\Big\}, \tag{6}$$

where N represents the total number of devices in an IoE cluster as shown in Figure 2.

E. PROBLEM FORMULATION

The main objective here is to use one of the scheduling algorithms presented in section IV for selecting an appropriate device to compute the scheduled task $\mathcal{T}$. An overarching goal of the scheduler is to minimise the total time T_{total} for a task to be scheduled, computed and returned back to the original IoE device. We formulate each scheduling problem as $\mathcal{P}_i^{Sched}$, where $\mathcal{P}$ denotes the research problem (time minimisation problem), depending on certain selection criteria as described in Problem 1. The subscript i in $\mathcal{P}_i^{Sched}$, where $i = 1, 2, \ldots, 5$, represents each minimisation problem. Mathematically, we can write these research problems, $\mathcal{P}_i^{Sched}$, as follows:

Problem 1:

$$\begin{aligned} \min_{\{\mathcal{D}\}} \quad & \Big\{T_{total}\Big\}, \\ s.t. \quad & i) \text{ Select } d_s \text{ using}(Sched) \\ & ii) \text{ Use } \mathcal{P}_i^{Sched} \end{aligned}$$

Each minimisation problem, formulated from Problem 1 has two main parts. The first part is to select a scheduling IoE device using a particular scheduling algorithm *Sched* such that the schedule device d_s minimised T_{total} and it can take any value as per equation (4). Note that we use d_s and d^* in the same context throughout this paper such that both symbols represent the scheduled/selected device after applying the respective scheduling algorithm. Whereas in the second part, we use the subsequent research problem $\mathcal{P}_i^{Sched}$ formulated in the set of following Problems 2 - 6 to implement the scheduling algorithm.

Problem 2:

$$\begin{aligned} \Big\{\mathcal{P}_1^{RR}\Big\} &\Longrightarrow \text{Select } d_s \\ d_s &= \text{rand}\{\mathcal{D}\} \end{aligned}$$

Problem 3:

$$\begin{aligned} &\Big\{\mathcal{P}_2^{SC}\Big\} \Longrightarrow \text{Select } d_s \\ &s.t. \; ||\mathbf{h}_s||_F^2 > \Big\{||\mathbf{h}_i||_F^2\Big\}_{i=1}^{N} \end{aligned}$$

Problem 4:

$$\begin{aligned} &\Big\{\mathcal{P}_3^{MR}\Big\} \Longrightarrow \text{Select } d_s \\ &s.t. \; r_{d_s} > \Big\{r_i\Big\}_{i=1}^{N} \end{aligned}$$

Problem 5:

$$\begin{aligned} &\Big\{\mathcal{P}_4^{PF}\Big\} \Longrightarrow \text{Select } d_s \\ &s.t. \; \Big\{\frac{\widetilde{r}_s(t)}{\overline{r}_s(t)}\Big\} > \Big\{\frac{\widetilde{r}_i(t)}{\overline{r}_i(t)}\Big\}_{i=1}^{N} \end{aligned}$$

Problem 6:

$$\begin{aligned} &\Big\{\mathcal{P}_5^{PB}\Big\} \Longrightarrow \text{Select } d_s \\ &s.t. \text{ using equations (15 and 19)} \end{aligned}$$

In Problems 2 - 6, to resolve the scheduling problem $\mathcal{P}_i^{Sched}$, five different scheduling algorithms (RR, SC, MR, PF and PB) are defined in detail in subsections (IV-A1 - IV-A5). Problem 2 describes RR scheduling, Problem 3 describes SC scheduling with the condition that the scheduled device has the largest channel strength, Problem 4 describes the MR scheduling algorithm where the selected device has the maximum data rate between the selected and the original device, Problem 5 describes the PF scheduling algorithm where selected device is the fairest scheduled device. Finally, Problem 6 describes the PB scheduling algorithm where a device is selected in such a way that it is capable of computing the prioritised tasks.

IV. PROPOSED ALGORITHM

In this section, we present the main task off-loading scheduling algorithm referred as MTOSA. This algorithm takes a task when it becomes available from a requesting IoE device, selects an appropriate scheduling algorithm, handles all communication related to the task from a source device to a scheduled device, and then returns the task-related information to

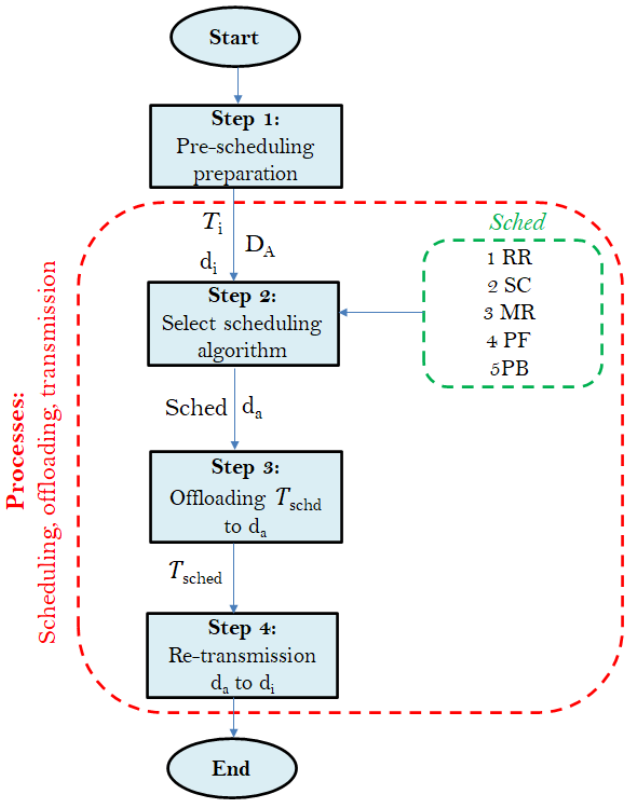

FIGURE 5. Flow diagram of MTOSA.

the source device. In the following subsection, we explain the detailed working of the proposed MTOSA algorithm.

A. MTOSA ALGORITHM

The scheduling algorithm MTOSA in Algorithm 1 serves as the main algorithm to select a task waiting to be computed, schedule the task by selecting the appropriate algorithm, off-loading to compute the task and then re-transmitting the computed task back to the actual device as shown in Figure 5. The MTOSA algorithm has five main steps. Initial variables are set as input to the main algorithm.

In step-1, the pre-scheduling preparation is done where task $\mathcal{T}_i$ and device d_i are identified.

In step-2, MTOSA algorithm uses the function *Sched*, equation (4), to select an appropriate scheduling algorithm from the pool of five available algorithms such as RR, SC, MR, PF and PB. When *Sched* equals to 1, 2, 3, 4 and 5, the scheduler selects RR, SC, MR, PF and PB algorithms respectively. It should be noted that the performance of MTOSA algorithm depends on one of these selected scheduling policies.

In step-3, MTOSA algorithm performs task off-loading from source device d_i which is requesting to compute its task, to device d_a (available device having resources to compute the scheduled task).

In step-4, MTOSA algorithm performs the back transmission of the computed task from device d_s (note that any available device d_a can become the scheduled device d_s once the task is assigned to it) to device d_i.

Algorithm 1: MTOSA Scheduling Algorithm

Input: Set of tasks $\{\mathcal{T}\}$ to be computed, Set of available devices $\{\mathcal{D}_A\}$ s.t. $\mathcal{D}_A = \{\mathcal{D}\} \setminus d_i$
Processes: Scheduling, off-loading, transmission
Output: Computed task
Step1: Pre-scheduling preparation; Pick a task $\mathcal{T}_i$ of device $d_i \in \{\mathcal{D}\}$
Step2: Select a scheduling algorithm using equation (4) Select d^* accordingly
Subroutine (*Sched*)
for *iter = 1 : No. of Scheduling Algorithms* **do**
  Sched = {RR, SC, MR, PF, PB}
  if *Sched* == 1 **then** *Sched* $\Longleftarrow$ Algorithm 2 (RR)
  if *Sched* == 2 **then** *Sched* $\Longleftarrow$ Algorithm 3 (SC)
  if *Sched* == 3 **then** *Sched* $\Longleftarrow$ Algorithm 4 (MR)
  if *Sched* == 4 **then** *Sched* $\Longleftarrow$ Algorithm 5 (PF)
  if *Sched* == 5 **then** *Sched* $\Longleftarrow$ Algorithm 6 (PB)
end
Step3: Off-load $\mathcal{T}_{schd}$ to d_a
Step4: Collect and transmit back the computed task to d_i
Step5: End of the algorithm

In step-5, after allocating all available tasks the algorithm terminates.

The relationship between MTOSA algorithm and the other scheduling schemes such as RR, SC, MR, PF and PB is evidenced through the performance of MTOSA algorithm. For example, when the MTOSA algorithm selects SC scheduling policy, prioritising the available tasks will not be the scheduling objective of MTOSA algorithm. In this case, the device with the strongest wireless channel will be allocated the task for computational purposes. Similarly, the performance of MTOSA algorithm can be explained when other scheduling policies such as RR, MR, PF and PB are selected.

1) ROUND ROBIN SCHEDULING

Round Robin (RR) scheduling is the simplest scheduling algorithm (Algorithm 2) used in many applications. We use RR as a base scheduling algorithm for the proposed system. Let us explain the working principle of RR. In RR algorithm, all devices are scheduled equally when the algorithm is run for a long period of time. The function *rand* used in equation (7) given below, randomly selects a device d^* from the set of given devices $\mathcal{D}$.

$$d^* = \underset{d^* \in \{\mathcal{D}\}}{\text{rand}} \Big\{ d_1, d_2, d_3, \ldots, d_D \Big\}. \tag{7}$$

It should be noted that each device in equation (7) has equal selection probability and the set of selected devices follow a uniform distribution. The RR scheduling (Algorithm 2) takes a set of tasks $\mathcal{T}$ and a set of devices $\mathcal{D}$ as input, the output of RR algorithm is d^* which is fed back to MTOSA algorithm (Algorithm 1).

Algorithm 2: Round Robin Scheduling Algorithm

Input: $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2 \ldots, \mathcal{T}_N\}$, $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$
where N = Total number of devices
Output: $d^* \Longleftarrow Sched\{\mathcal{D}\}$
for *iter = 1 : TimeSlot* **do**
  $d_a \leftarrow \{\mathcal{D} \setminus d_s\}$
  Select scheduled device d^* using equation (7)
end

2) STRONGEST CHANNEL SCHEDULING

The strongest channel (SC) scheduling algorithm (Algorithm 3) selects a device d^* with the strongest channel. The purpose of SC algorithm is to provide the best QoS in guaranteeing the task completion to the scheduled device. In this algorithm, it is assumed that wireless channels of all the devices are feasible for communication. This means that when the scheduler selects any particular device or set of devices for communication, their wireless channels are capable of supporting the data transmission. The selection process of SC algorithm can be described as below:

$$d^* = \underset{d^* \in \mathcal{D}}{\text{argmax}} \Big\{ ||\mathbf{h}_1||, ||\mathbf{h}_2||, \ldots, ||\mathbf{h}_N|| \Big\}, \tag{8}$$

where $||.||$ represents norm, $\mathcal{D}$ is a set of devices with the total number of N elements and $d^* \in \mathcal{D}$ is the selected device with the strongest channel. The norm of a channel matrix $\mathbf{H} \in \mathbb{C}^{m \times n}$ can be calculated by using the Frobenius norm expression as:

$$||\mathbf{H}||_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} \Big(|\mathbf{h}_{ij}| \Big). \tag{9}$$

where h_{ij} represents the ij^{th} column of the channel matrix $\mathbf{H}$.

Algorithm 3: Strong Channel Scheduling Algorithm

Input: $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_N\}$,
where N = Total number of devices.
Output: $d^* \Longleftarrow Sched\{\mathcal{D}\}$
for *iter = 1 : TimeSlot* **do**
  Calculate channel norm using equation (9)
  Select d^* **with the strong channel** using equation (8)
end

Input to the SC algorithm is set of device $\mathcal{D}$, time slots T and wireless channel matrix $\mathbf{H}$. During each iteration, the algorithm calculates the channel norm of each device using equation (9), orders all the norm values in descending order and then selects the device with the strongest norm value.

3) MAX RATE SCHEDULING

The Max Rate (MR) scheduling algorithm (Algorithm 4) selects a device d^* based on the maximum channel rate r_i of the devices [45]. This ensures to meet the data rate requirements of the requesting device as per the nature of the task. In a sense, MR is similar to SC but ensures that the wireless channel has the required data rate. Let us define the data rate of i^{th} device as follows:

$$r_i(t) = BW \times \log_2 \Big(1 + \xi_i\Big), \quad i = 1, \ldots, d_N, \tag{10}$$

where BW is the channel bandwidth (Hz) and ξ_i is the Signal to Interference plus Noise Ratio (SINR) of the i^{th} device and is defined as:

$$\xi_i = \frac{\rho_i(\mathbf{h}_i \mathbf{h}_i^*)}{\sigma^2 + \sum_{j=1, j \neq i}^{d_N} \Big(\rho_j \mathbf{h}_j \mathbf{h}_j^*\Big)}, \tag{11}$$

where ρ and σ^2 represent the transmit power and additive noise respectively. Now we can write the MR device selection as:

$$d^* = \underset{d^* \in \mathcal{D}}{\text{argmax}} \Big\{ r_1(t), r_2(t), r_3(t), \ldots, r_{d_N}(t) \Big\}. \tag{12}$$

Algorithm 4: Max Rate Scheduling Algorithm

Input: $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_N\}$,
where N = Total number of devices.
Output: $d^* \Longleftarrow Sched\{\mathcal{D}\}$
for *iter = 1 : TimeSlot* **do**
  Select scheduled device d^* using equation (12)
end

4) PROPORTIONAL FAIR SCHEDULING

The Proportional Fair (PF) scheduling algorithm (Algorithm 5) selects a device d^* based on the ratio of instantaneous data rate $\widetilde{r}_i(t)$ and the average aggregated data rate $\overline{r}_i(t)$. The purpose of using this ratio is to introduce fairness in device selection. A device with the highest data rate may get the first few time slots allocated, however, since its average aggregated data rate will keep increasing, which will decrease the possibility of selection for this device in the next time slots. This way, selecting those devices with low instantaneous data rates can be possible through this algorithm. Mathematically, PF algorithm can be written as:

$$d^* = \underset{d^* \in \mathcal{D}}{\text{argmax}} \left\{ \left(\frac{\widetilde{r}_1(t)}{\overline{r}_1(t)}\right), \frac{\widetilde{r}_2(t)}{\overline{r}_2(t)}, \frac{\widetilde{r}_3(t)}{\overline{r}_3(t)}, \ldots, \frac{\widetilde{r}_D(t)}{\overline{r}_D(t)} \right\}, \tag{13}$$

where $\widetilde{r}_i(t)$ and $\overline{r}_i(t)$ represent the instantaneous and average aggregate data rates of the i^{th} device.

Algorithm 5: Proportional Fair Scheduling Algorithm

Input: $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, Set of TimeSlot = {T}
Output: $d^* \Longleftarrow Sched\{\mathcal{D}\}$
for *iter = 1 : TimeSlot* **do**
  Select scheduled device d^* using equation (13)
end

5) PRIORITY BASED SCHEDULING

Although RR is the simplest and the fairest scheduling algorithm, in practical situations, one needs to prioritise certain tasks as per their importance. Hence, we present a priority based scheduling algorithm (Algorithm 6) for the devices in the network. This algorithm identifies the priorities of each task and schedules it accordingly. A complete description of PB scheduling algorithm and related components is given below.

Classification of Tasks: In PB scheduling algorithm, we define that each device can assign a priority level p to its task $\mathcal{T}$. There can be three priority levels for each task defined as:

$$\text{Priority} = \begin{cases} p = 3, & \text{High} \\ p = 2, & \text{Medium} \\ p = 1, & \text{Low} \end{cases} \tag{14}$$

Let us represent task $\mathcal{T}_i$ with its priority level denoted by p as $\mathcal{T}_i^p$ where $p = \{3, 2, 1\}$. For example $\mathcal{T}_i^3$ means a task with the highest priority over all other tasks. A high priority task can be considered an emergency task, whereas a low priority task can be considered a simple calculation or information request.

This should be noted that the role of the PB scheduler is to allocate the tasks to available devices that are capable of computing the tasks and have computational resources available. The scheduler does not analyse the nature of the tasks, which means the scheduler itself is unaware of the priority levels of the tasks. Instead, each device in an IoE cluster assigns a priority level to its own task.

There can be situations when multiple tasks will have the same priority levels, whereas in other situations tasks can have different priorities. Based on these scenarios, PB scheduling algorithm can have the following two cases.

Case - 1: Tasks with Different Priority Levels: In this case, let us consider that the scheduler has all tasks with different priority levels. We can represent this scenario as follows:

$$\mathcal{T}^{p \neq p} = \left\{ \mathcal{T}_1^p, \mathcal{T}_2^p, \ldots, \mathcal{T}_D^p \right\}. \tag{15}$$

Left hand side of the equation (15) shows that each task in the set of tasks available to the PB scheduler has a different priority level, i.e. $p_i \neq p_j$. In this case, PB scheduler selects a set of tasks with the highest priority levels. For example, if the scheduler can schedule a maximum of $|\mathcal{T}|$ tasks, it will select these tasks from the set of tasks available as in equation (15). If the tasks with the highest priority level in equation (15) are greater than the maximum capacity of the scheduler, it will randomly pick the tasks equal to its capacity with the highest priority level. This scenario of PB algorithm is shown in Algorithm 6.

Case - 2: Tasks with Same Priority Level: In this case, the scheduler has all tasks $\mathcal{T}$ from various devices with same priority level, i.e. $p_i = p_j$ where i, j are device indices. This is a complicated scheduling scenario where the scheduler needs to implement some strategy S to resolve the conflict of which tasks need to be scheduled. In order to resolve this conflict, we propose the following conflict resolution strategy.

Algorithm 6: Priority Based Scheduling Algorithm

Input: $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, Set of TimeSlots = {T}
Output: $d^* \Longleftarrow Sched\{\mathcal{D}\}$
where *Sched* = PB
for *iter = 1 : TimeSlot* **do**
  if *Case == 1* **then**
    $\mathcal{T}_s \Longleftarrow \underset{\text{p}}{\text{argmax}} \left\{ \mathcal{T}^{p \neq p} \right\}$
  else
    if *Case == 2* **then**
      $\mathcal{T}_s \Longleftarrow$ Go to equation (19)
    end
  end
end

Conflict Resolution Strategy: Let us assume that the scheduler has the following set of tasks:

$$\mathcal{T}^{p=p} = \{\mathcal{T}_1^3, \mathcal{T}_2^3, \ldots, \mathcal{T}_D^3\}, \tag{16}$$

where the left hand side of the equation (16) shows that tasks have the same priority level and the right hand side of this equation represents that all tasks are with the highest priority level, i.e. $p = 3$. This priority level can also be either 1 or 2 but all tasks will have the same priority level. The scheduling algorithm for this case is presented in Algorithm 6.

The Strategy S: In order to resolve the conflict mentioned above, the scheduler uses a strategy S to schedule the tasks based on their cost function represented by C. This cost function is associated with each task in $\mathcal{T}^{p=p}$. More formally, our cost function $C_p(\mathcal{T}_i)$ represents the total computational cost in terms of time of a task $\mathcal{T}_i$ with priority p, when scheduled to device d_j for computation and returned back to the scheduler.

Let us define (i, j) as the source-destination (SD) device pair where $i, j \in \mathcal{D}$ denote the source and destination devices respectively. It is important to note that the set of available devices for the scheduler to allocate the selected task is $\mathcal{D} \setminus i$.

Let us represent device d_a as the available device willing and having the resources to compute $\mathcal{T}_i^p \in \mathcal{T}^{p=p}$. We assume that the scheduler has complete information of all the available devices $d_a \in \mathcal{D}$, for example through a dynamic intelligent resource availability table. Once a device is selected for task scheduling its status becomes a scheduled device, represented as d_s. It should be noted that it is not necessary for all d_a to become d_s, i.e. the device with the scheduled task.

The scheduler sends a query message m_j ($j \in \{\mathcal{D} \setminus i\}$) to all potential $d_a \in (\mathcal{D} \setminus i)$ about the tasks available for scheduling. The content of m_j consists of calculating the total computational cost C in terms of time for each task and returning this information to the scheduler.

Each available device d_a calculates the cost corresponding to each task as follows:

$$C^{\mathcal{T}}_{d_a \in (\mathcal{D} \setminus i)} = \left\{C_1, C_2, \ldots, C_{|\mathcal{T}|}\right\} = \left\{C_i\right\}_{i=1}^{|\mathcal{T}|}, \tag{17}$$

where $|\mathcal{T}|$ represents the total number of tasks in equation (16). Each device $d_a \in (\mathcal{D} \setminus i))$ then calculates the cost function C_i as follows:

$$C_i = t^f_i + t^f_{schd} + t^f_s + t_c + t^b_s + t^b_{schd} + t^b_i, \tag{18}$$

where variables in above equation (18) are defined in Table 2. The scheduler now has the information of C from all available devices. Then the strategy S for each device d_a can be defined as:

$$\begin{aligned} j &= \text{argmin}\left\{C^{\mathcal{T}}_{d_a \in (\mathcal{D} \setminus i)}\right\}, \\ \mathcal{T}_s &\Longleftarrow \mathcal{T}_j. \end{aligned} \tag{19}$$

This leads us to an optimisation problem at the scheduler where the task of the scheduler becomes to select a task with the minimum computational time such that the expiry time of the task does not reach. Let us define the expiry time of a task $\mathcal{T}_i$ as $t^{\mathcal{T}_i}_{exp}$. Then the minimisation problem at the scheduler becomes as:

***Problem** 7*:

$$\begin{aligned} &min \left\{C_i\right\}_{i=1}^{|\mathcal{T}|}, \\ &s.t. \quad C_i < t^{\mathcal{T}_i}_{exp}, \end{aligned}$$

where the objective function of Problem 7 is similar to the objective function of Problem 1, but defined in terms of the total cost for computing each task.

B. COMPLEXITY ANALYSES

The main computational complexity of the proposed MTOSA algorithm (Algorithm 1) comes from one of the scheduling algorithms selected by MTOSA algorithm. There are three stages in terms of the overall complexity analysis of the proposed algorithm. Let us represent the complexity of MTOSA algorithm as:

$$Comp^{MTOSA} = Comp^{Stage1} + Comp^{Stage2} + Comp^{Stage3}, \tag{20}$$

where term *Comp* represents the computational complexity.

Stage 1: This is the input and preparation stage of MTOSA algorithm. During this stage, the main algorithm MTOSA reads the set of devices and the number of tasks to be scheduled. Both of these tasks are linear in nature, and the algorithm complexity follows linear additive operations. So the input complexity of the algorithm can be written as $(c_1 + c_2)n$ where c_i where $i = 1, 2$ is a constant and n is a variable representing number of operations. Given this is a linear expression, we can write the input complexity of the proposed algorithm as $\mathcal{O}((c_1 + c_2)n)$. For the best-case scenario, when only a single cycle is needed to read each input, the complexity will follow $\mathcal{O}(2)$. Hence, we can write $Comp^{Stage1} = \mathcal{O}((c_1 + c_2)n)$.

Stage 2: In this stage, the main MTOSA algorithm invokes one of the scheduling algorithms from RR, SC, MR, PF and PB.

From this list of the algorithms, RR is the simplest algorithm with the computational complexity of $\mathcal{O}(1)$ as it requires a single operation to select an available device for task allocation.

The second scheduling algorithm that we consider is SC which uses equation (9) to calculate channel norms and then selects the device with the best channel. To compute channel norms, there are three operations required including a mod computation followed by two summations. The overall complexity of this operation can be written as $\mathcal{O}(cn)$ where $c = 3$ in this case. Also, after norm calculation, the algorithm sorts out all norm values to pick the best norm. This sorting operation has the complexity of $\mathcal{O}(n^2)$. The squared value of n shows the two steps of selecting the best value and replacing it with the head element in the array. This gives us the overall computation complexity of SC algorithm as $\mathcal{O}(cn) + \mathcal{O}(n^2)$.

For MR scheduling algorithm, two steps are required by the algorithm to compute channel rates and then sort them in decreasing order. The complexity of the wireless channel capacity formula is given as $\mathcal{O}(\log n)$ whereas, as mentioned before, the complexity of the sorting operation is $\mathcal{O}(n^2)$. Hence the computational complexity of MR algorithm can be written as $\mathcal{O}\log(n) + \mathcal{O}(n^2)$.

The complexity of PF algorithm is similar to the complexity of MR algorithm with an extra step where PF algorithm calculates the ratio between the instantaneous and average rates of each device. Since this is a linear calculation operation, the complexity of this step can be written as $\mathcal{O}(n)$. Hence, giving us the overall complexity of PF algorithm as $\mathcal{O}\log(n) + \mathcal{O}(n) + \mathcal{O}(n^2)$ which can be further simplified as $\mathcal{O}\log(n) + \mathcal{O}(n^3)$.

The last scheduling algorithm in the list is PB scheduling algorithm, where tasks are sorted into three priority levels which are low, medium and high. Then based on the priority level assigned to each task, it is allocated to the available devices for computation. Given three sorting operations are happening in PB scheduling algorithm, the computational complexity of this algorithm can be written as $\mathcal{O}(n^3)$.

Hence, we can write the complexity of stage 2 depends on the selected scheduling algorithm as explained above. Mathematically we can write as $Comp^{Stage2} = Comp$ (selected scheduling algorithm).

Stage 3: This stage includes the complexity of collecting and re-transmitting the computed tasks back to the original devices. These are linear operations with a constant and a variable depending on the number of tasks being handled. For example, for n number of tasks with $c = 2$ representing two operations (collection and re-transmission), the complexity of stage 3 can be written as $\mathcal{O}(cn)$.

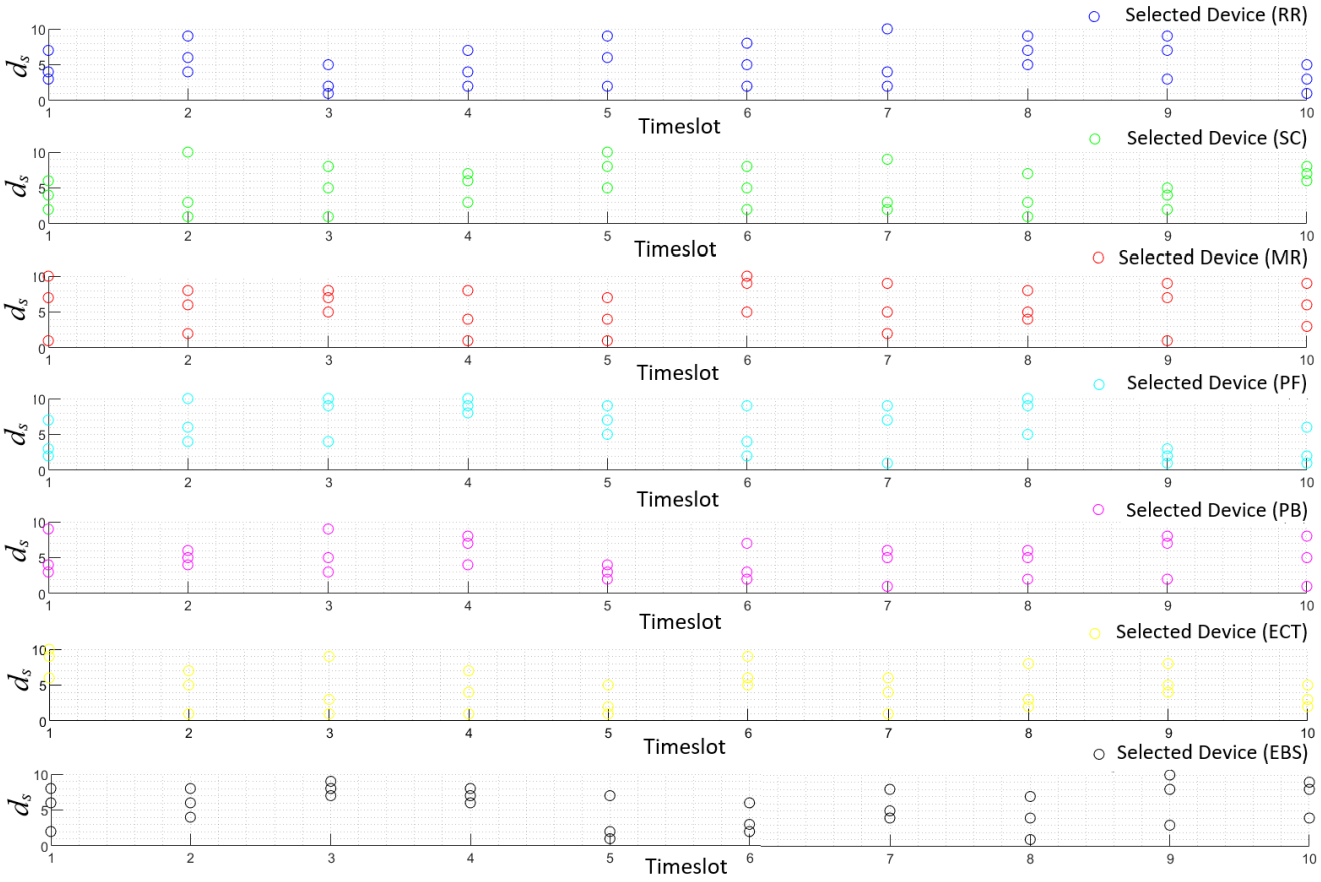

FIGURE 6. Scenario I: Pictorial representation of various selected devices using RR, SC, MR, PF, PB, ECT [46] and EBS [47] scheduling algorithms. ECT and EBS scheduling algorithms are used for comparison purposes.

Hence, using equation (20) and the complexity expressions presented above in stage 1, 2 and 3, we can re-write equation (20) as equation (21):

$$\begin{aligned} &Comp^{MTOSA} \\ &\quad = \mathcal{O}(c_1 + c_2)n \\ &\qquad + Comp\ (\text{selectedscheduling algorithm}) + \mathcal{O}(cn). \end{aligned} \tag{21}$$

V. RESULTS AND DISCUSSION

In this section, we analyse the performance of the proposed MTOSA scheduling algorithm (Algorithm 1) by presenting a number of simulation scenarios. We compare the results of various scheduling policies presented in this work, and also compare them with the existing scheduling policies such as from [46], [47]. Similar to many other works in literature, such as [46], [47], [48], we also used MATLAB to create and simulate different scenarios using system parameters that are close to real-life setups.

In the following, we describe the experimental setup used in this work, and elaborate on the two existing scheduling policies namely, ECT (Expected Computation Time) [46] and EBS (Energy Based Scheduling) [47], which are used for comparison purpose. The reasons for selecting ECT and EBS policies for the comparison purposes are (i) the relevance to our work in this paper, (ii) the recent relevant literature (iii) the practical nature of these policies for implementation purposes.

The experimental setup used in this work mimics an IoE based cluster where a number of IoE devices co-locate within the cluster in close proximity to each other. Within the IoE cluster, there is a scheduler and a number of IoE devices. The scheduler is provided with the fixed set of devices $\mathcal{D}$ as defined in equation (2) within the IoE network. Simulation parameters used for the experiments in this work are similar to [46], [47]. We used a HP-laptop with the following configuration: CPU AMD Ryzen 5 4500U, operating system 64-bit Windows-10 and memory of 8GB. The scheduler and IoE devices are configured using the computing resources of this laptop. Furthermore, we used a single scheduler within IoE cluster, and the number of IoE devices varies according to each simulation scenario. The obtained results are manifested through independent simulation repetition of each experiment by 30 times. The results obtained and presented are the average over the total number of the simulation run.

TABLE 3. Representation of preliminaries for simulation results.

Device (d_i)	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
Device (d_i) **ID**	2	8	10	4	9	1	6	5	3	7
Task ($\mathcal{T}_i$) **ID**	6	1	3	1	6	4	2	10	8	3
Resource type ($\mathcal{R}^{typ}$)	$\mathcal{R}^{inf}$	$\mathcal{R}^{inf}$	$\mathcal{R}^{cpt}$	$\mathcal{R}^{stg}$	$\mathcal{R}^{inf}$	$\mathcal{R}^{stg}$	$\mathcal{R}^{cpt}$	$\mathcal{R}^{cpt}$	$\mathcal{R}^{inf}$	$\mathcal{R}^{stg}$
Cost (C)	12	18	30	40	9	1	16	15	13	17
Priority (p)	2	1	2	1	3	2	3	3	2	2

TABLE 4. Scenario I: Tabular representation of various selected devices using RR, SC, MR, PF, PB, ECT [46] and EBS [47] scheduling algorithms.

Sched **Policy**	t_1			t_2			t_3		
	d^*			d^*			d^*		
RR	3	4	7	4	6	9	1	2	5
SC	2	4	6	1	3	10	1	5	8
MR	1	7	10	2	6	8	5	7	8
PF	2	3	7	4	6	10	4	9	10
PB	3	4	9	4	5	6	3	5	9
ECT [46]	9	10	6	5	7	1	1	9	6
EBS [47]	2	8	6	4	6	8	8	7	9

In the following paragraph, we describe two existing scheduling schemes from the current literature, namely ECT [46] and EBS [47]. In ECT based scheduling scheme, a task from a particular IoE device is scheduled with the minimum execution time by the allocated device. This scheme uses the parameters such as task length and allocated device computing power to minimise the execution time. Whereas, in EBS scheme, the authors defined the QoE parameter for each fog node and computed the task based on QoE. The scheduler computing power is divided equally in the beginning for all the competing nodes, and the task with minimum execution time is given priority during the scheduling. The authors used Game Theory to maximise each user's own QoE compared to the other user's strategies. Furthermore, they proposed a weighted potential game to achieve the Nash equilibrium in the scheduling process.

Preliminaries: We start our discussion by describing the key simulation preliminaries such as IoE cluster, devices, tasks, resource type, cost and priorities that are an integral components of the entire design on which simulations are built. The key parameters of these components related to this simulation are tabulated in Table 3. For example, a device $d_1 = 2$ is available for scheduling in IoE cluster at time slot t_1. The relevant parameters of this device $d_1 = 2$ are shown in the first column of Table 3 and explained below.

A predefined task with the task ID as $\mathcal{T}_1 = 6$ is assigned to the task of this device ($d_1 = 2$). The type of the requested resource by this device is 'information', denoted by $\mathcal{R}^{inf}$. The total cost (C) required to compute this task is 12 units. The task has a medium level priority represented by $p = 2$.

A set of ten devices is considered in our simulations as shown in the first row of Table 3. During each simulation scenario, a device d_i is selected as per the scheduling policy used by MTOSA algorithm. The selected device ID is shown in second row of Table 3. Each device has its own task denoted by $\mathcal{T}_i$ having a specific task ID as shown in third row of Table 3. The requested resources from devices in the IoE cluster can have various types such as *information*, *computation* and *storage* which are shown in the fourth row of Table 3. The fifth row of the Table 3 shows the *total cost* in terms of computational time for each scheduled task. The sixth and the final row of Table 3 shows the priority level of each task (3- high), (2- medium) and (1-low) to be used when PB scheduling policy is used.

Relevance between Performance Parameters and the Application: The parameter to evaluate the performance of the proposed MTOSA algorithm in this article, *is the knowledge of selected devices (device selection) by the scheduler*. This information (device selection) is directly relevant to the application performance of the overall system considered in this paper, which is an IoE based edge network. In all three evaluation scenarios given below, device selection and its derived parameter, such as the percentage of device selection, are the focus parameters. The proposed MTOSA algorithm selects different devices for the same task when different scheduling algorithms (such as RR, SC, MR, PF, PB) are used. The knowledge of selected devices determines the overall performance of the devices in the edge network. If a particular device or a group of devices are being selected repeatedly, their long-term performance will degrade dramatically due to the overloading of the scheduled tasks. These devices being selected for task computation again and again will lead to fast failure because of the extra workload assigned to them. With this information, the scheduler can balance the workload for the devices in the network that are repeatedly selected and distribute the tasks to those devices that are assigned fewer tasks.

Scenario I: In this scenario, we show the device selection using all five scheduling algorithms (RR, SC, MR, PF, PB) as described in section IV. We also show the device selection using existing schemes ECT and EBS for comparison purposes. Simulation results of this scenario are shown in Table 4 and Figure 6.

The results in Table 4 are shown for only three time slots (t_1, t_2, t_3) whereas in Figure 6 results are shown for ten time slots. First column of Table 4 shows scheduling policies which are the five scheduling algorithms investigated in this paper and two scheduling policies used for result comparison from literature. Second column-first row of Table 4 shows the first time slot t_1, whereas second column-second row shows

TABLE 5. Scenario II: Tabular representation of the percentages of the selected devices.

Time slot (t_i)	$(t_1, \cdots, t_{10})$									
Device (d_i)	d_1 (%)	d_2 (%)	d_3 (%)	d_4 (%)	d_5 (%)	d_6 (%)	d_7 (%)	d_8 (%)	d_9 (%)	d_{10} **(%)**
RR	20	50	30	40	40	20	40	20	40	10
SC	30	40	40	20	40	30	30	40	10	20
MR	40	20	10	30	40	20	40	40	40	20
PF	30	40	20	30	20	20	30	10	60	40
PB	20	40	40	40	50	30	30	30	20	0
ECT [46]	50	30	30	30	50	30	20	20	30	10
EBS [47]	20	30	20	40	10	40	40	70	20	10

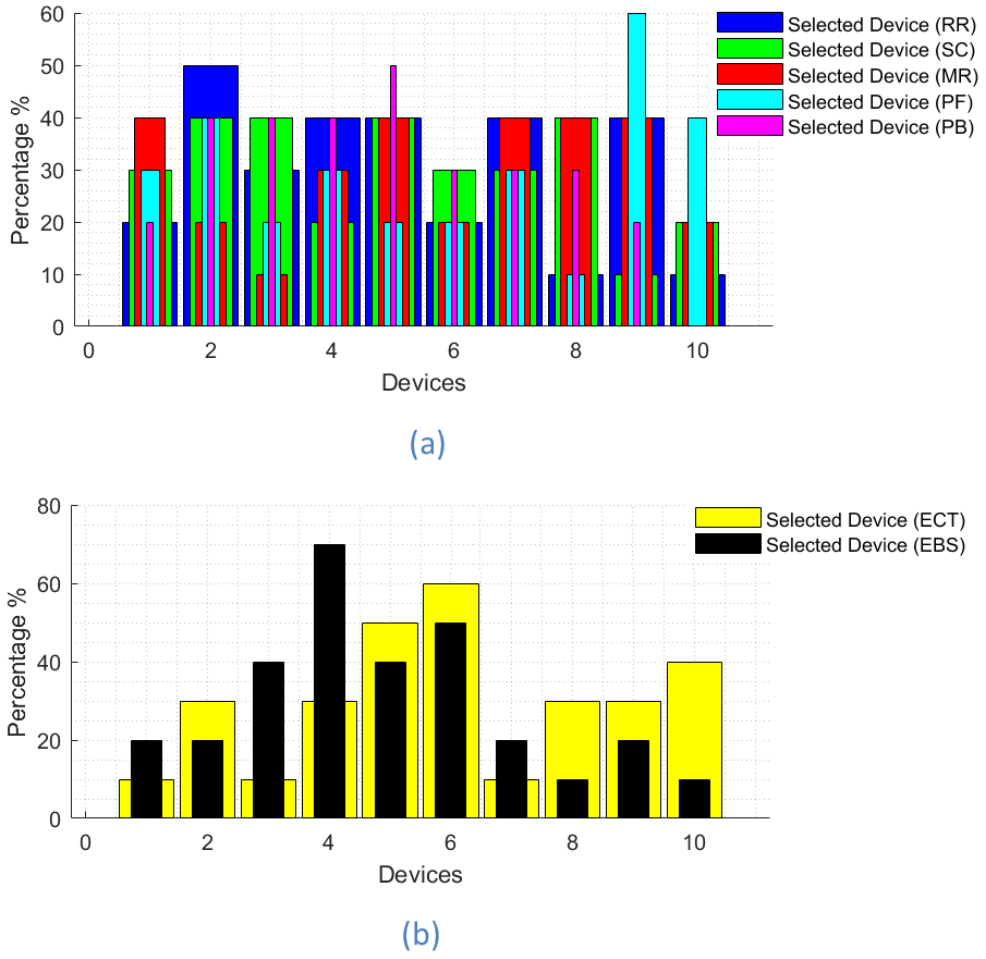

FIGURE 7. Scenario II: Pictorial representation of the percentages of the selected devices.

scheduled / selected devices d^*. There are three out of a total of ten devices selected in each time slot. Similarly second and third columns in Table 4 show selected devices during time slots t_2 and t_3 respectively.

Figure 6 shows the pictorial representation of the selected devices for ten time slots $t_1, \ldots, t_{10}$ using all five scheduling algorithms and two existing scheduling policies from literature. It is clear from this scenario that various devices are being selected during each time slot when using different scheduling algorithms. This scenario sets the foundation of other simulation scenarios and confirms the working accuracy of each scheduling algorithm through different selected devices.

Scenario II: In this scenario, we show the selection percentage of each device for all five scheduling algorithm, as well as for ECT [46] and EBS [47], in the simulation. The selection percentage is defined as that how many times a device is selected by a specific scheduling algorithm over total time slots. For example, if a device is selected twice during ten time slots, then the selection percentage of that device

TABLE 6. Scenario III: Devices selection in IoE cluster with PB scheduling algorithm.

Device (d_i)	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
Case - 1										
Device (d_i)	5	1	3	4	10	6	2	8	9	7
Priority (p)	3	1	1	1	3	1	1	1	2	1
Case - 2										
Device (d_i)	1	7	3	9	2	4	5	8	6	10
Priority (p)	3	3	3	3	3	3	3	3	3	3
Cost (C)	12	33	21	13	27	4	26	21	30	1
Case PB - Cost										
Device (d_i)	2	10	1	6	8	7	3	5	4	9
Cost (C)	9	2	17	15	39	3	35	24	26	17
Case PB - Type										
Device (d_i)	2	8	5	9	1	4	7	10	6	3
Resource Type ($\mathcal{R}^{typ}$)	$\mathcal{R}^{inf}$	$\mathcal{R}^{inf}$	$\mathcal{R}^{cpt}$	$\mathcal{R}^{stg}$	$\mathcal{R}^{stg}$	$\mathcal{R}^{inf}$	$\mathcal{R}^{cpt}$	$\mathcal{R}^{cpt}$	$\mathcal{R}^{cpt}$	$\mathcal{R}^{stg}$

TABLE 7. Scenario III: A comparison of device selection RR, ECT [46] and EBS [47] under PB various cases.

PB Cases / **Policy**	t_1			t_2			t_3		
	d^*			d^*			d^*		
Case - 1									
RR	1	2	8	3	8	10	3	4	5
PB - C1	5	9	10	2	3	10	4	5	7
ECT [46]	10	6	3	2	5	1	6	8	4
EBS [47]	10	9	1	7	1	9	9	6	1
Case - 2									
RR	2	5	10	4	5	7	1	4	6
PB - C2	1	4	10	1	6	9	2	3	8
ECT [46]	2	7	3	9	6	5	6	8	2
EBS [47]	10	5	6	7	6	5	4	6	3
Case PB - Cost									
RR	2	3	9	2	3	8	4	5	10
PB - Cost	2	7	10	4	5	6	1	5	9
ECT [46]	6	3	8	5	7	2	10	9	2
EBS [47]	5	7	2	1	9	4	1	3	8
Case PB - Type									
RR	3	4	7	2	8	10	1	8	10
PB - Type	2	4	8	1	4	8	1	5	7
ECT [46]	4	2	3	5	1	4	4	7	2
EBS [47]	3	9	1	9	8	6	5	4	9

is twenty percent. This study is important to understand the computation workload of each device within the cluster. If a particular device is being scheduled for task computation most of the time, it may damage the life span of that particular device. Therefore, the scheduler examines this information and can manage the scheduling of each device in the cluster. The results of this scenario are shown in Table 5 and Figure 7.

The results in Table 5 are shown for ten time slots $(t_1, \ldots, t_{10})$ and all ten devices $N = 1, \ldots, 10$. First column of Table 5 shows all five scheduling algorithms and two scheduling policies, ECT [46] and EBS [47] which are used for comparison. The top row of Table 5 shows time slots and the second row of the table shows the percentage of selection of each device in the cluster. For example, third row of Table 5 shows the selection percentages of all ten devices when RR scheduling algorithm is used. The remaining four rows in the table show similar results for the remaining four scheduling algorithms as well. Whereas, the last two rows in the table show the results of ECT [46] and EBS [47].

Figure 7 shows the pictorial representation of the selection percentages for each device. Vertical axis of the figure shows the selection percentage and the horizontal axis shows the number of devices in the cluster. Different colours of bars in Figure 7(a) represent five different scheduling algorithms and the height of each bar represents the selection percentage of each device. Similarly, Figure 7(b) represents the same results for ECT [46] and EBS [47] policies for the comparison purposes. For example, it is clear that under RR scheduling algorithm, device d_2 is selected most of the time and has a selection percentage value of 50%, whereas amongst all scheduling algorithms and all devices, d_9 has been selected 60% under PF scheduling algorithm. Based on this data, the scheduler can manage the selection workload for d_9 to optimise its life span. In comparison, device d_4 is selected by ECT [46] 70% and device d_6 has been selected by EBS [47] 60%. The results of these policies highlight the imbalance in device selection and require application of load balancing through the scheduler to improve the quality of service.

Scenario III: In this scenario, we investigate the performance of PB scheduling algorithm while considering the tasks with different priorities (see equation (14)) and different computation requirements. Description of task priorities, computational requirements and strategies to schedule such tasks is given in subsection IV-A5. Simulation results of various studies on PB scheduling algorithm are shown in Tables 6, 7, 8 and also in Figures 8 and 9.

Table 6 describes the simulation preliminaries for PB scheduling algorithm. We assume there are ten total devices $N = 10$ and $(d_1, \ldots, d_{10})$ with each device has a task with a pre-defined priority level. Each device and its task have the same IDs, i.e. for example device 4 has its own task with the task ID number 4. Note from equation (14) that priority levels $p = 1, 2, 3$ represent low, medium and high priorities respectively. Each task in this scenario is assigned a random priority

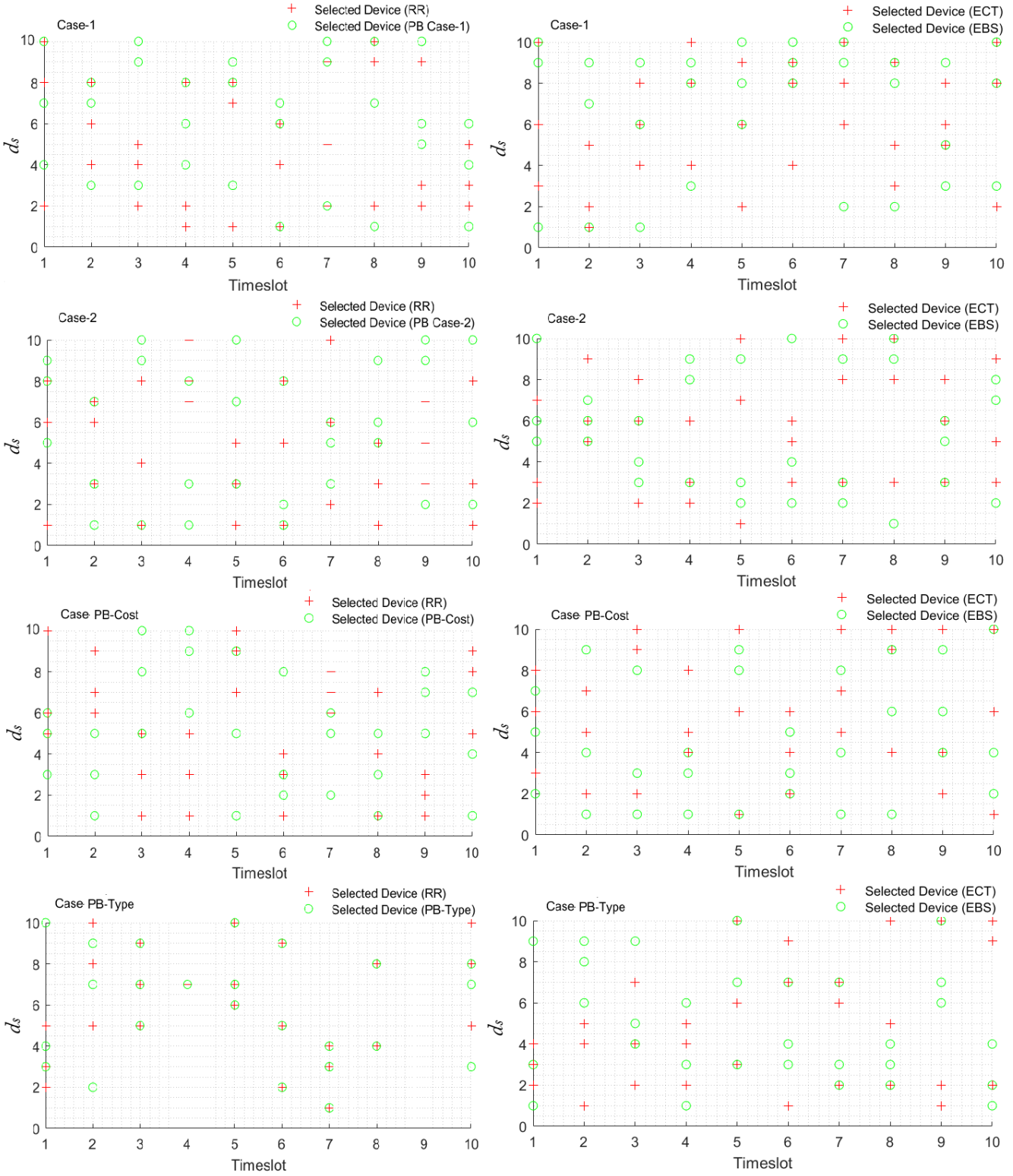

FIGURE 8. Scenario III: Devices selection in IoE cluster with PB scheduling under different cases. Also, comparison of PB various cases with RR, ECT [46] and EBS [47] device selection.

level. For example, from Table 6, under case-1, task 5 has high priority ($p = 3$), device 1 has its task with low priority ($p = 1$) and so on. The table also shows two cases where tasks have different and have the same priority levels respectively,

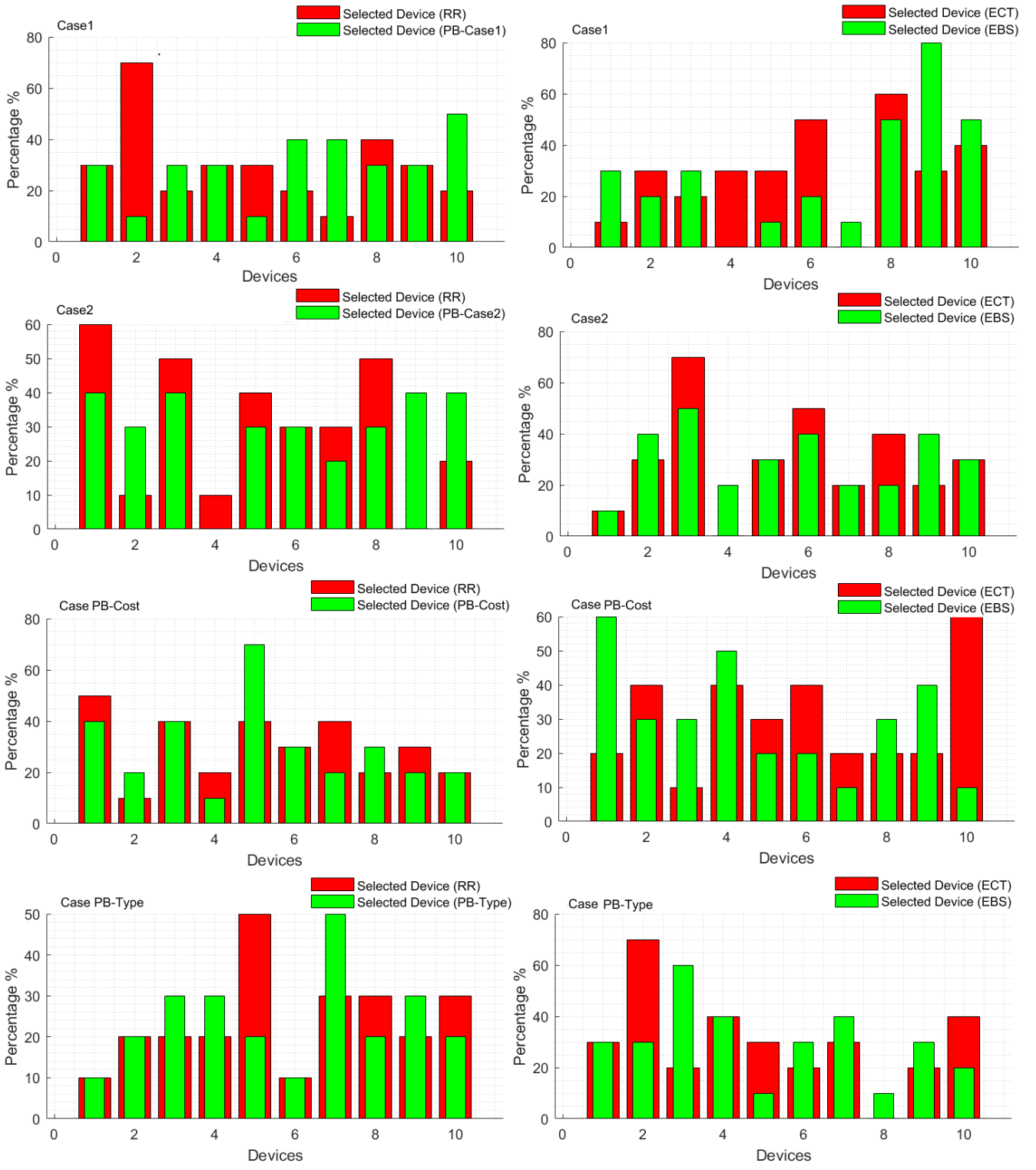

FIGURE 9. Scenario III: A comparison of selection percentage RR, ECT [46] and EBS [47] with PB different cases.

case PB-Cost where the device with the minimum cost is selected and case PB-Type, where a device is selected that can best compute the task based on its type, e.g. information, computation and storage types.

Table 7 shows the simulation results of PB algorithm for the scenario explained above and compare these results with RR, ECT [46] and EBS [47] algorithms. These results are shown for only three time slots while during each time slot

TABLE 8. Scenario III: Device selection percentage comparison RR, ECT [46], EBS [47] with PB various cases.

Time slot (t_i)	($t_1 \cdots t_{10}$)									
Device (d_i)	d_1 (%)	d_2 (%)	d_3 (%)	d_4 (%)	d_5 (%)	d_6 (%)	d_7 (%)	d_8 (%)	d_9 (%)	d_{10} (%)
Case - 1										
RR	30	70	20	30	30	20	10	40	30	20
PB - C1	30	10	30	30	10	40	40	30	30	50
ECT [46]	10	30	20	30	30	50	0	60	30	40
EBS [47]	30	20	30	0	10	20	10	50	80	50
Case - 2										
RR	60	10	50	10	40	30	30	50	0	20
PB - C2	40	30	40	0	30	30	20	30	40	40
ECT [46]	10	30	70	0	30	50	20	40	20	30
EBS [47]	10	40	50	20	30	40	20	20	40	30
Case PB-Cost										
RR	50	10	40	20	40	30	40	20	30	20
PB - Cost	40	20	40	10	70	30	20	30	20	20
ECT [46]	0	50	20	40	40	20	30	40	60	0
EBS [47]	40	30	20	30	40	50	30	20	40	0
Case PB-Type										
RR	10	20	20	20	50	10	30	30	20	30
PB - Type	10	20	30	30	20	10	50	20	30	20
ECT [46]	20	40	10	40	30	40	20	20	20	60
EBS [47]	60	30	30	50	20	20	10	30	40	10

three devices are selected by the PB, ECT, EBS and RR algorithms, whereas, Figure 8 represents these simulation results for the complete run of the simulation, i.e. for time slots $t_1, \ldots, t_{10}$.

Finally, Table 8 and Figure 9 present a comparison among ECT [46], EBS [47], RR and PB scheduling algorithms in terms of selection percentage of the selected device. Similar to the earlier argument, this study is important to manage the workload of each selected device. It is clear from Table 8 that during case-1, EBS selects 90% of times the device d_9, ECT selects 60% of times the device d_8, RR selects 70% of times the device d_2 whereas the device d_{10} is selected 50% by the PB algorithm. Also, from Figure 9, it is clear that during case-1, EBS selects the ninth device, ECT selects the device number eighth, RR selects the second device whereas PB selects the tenth device most of the time. Similarly, during PB-Type scenario, EBS, ECT, RR and PB policies selects the third, second, fifth and seventh device most of the time respectively. This information helps the scheduler balance the workload of all IoE device much more effectively in the cluster.

VI. CONCLUSION

This paper presents a conceptual design architecture of the conventional resource allocation model for IoE devices in 6G enabled smart edge based communication environment. Based on the proposed conceptual model, a novel task off-loading, scheduling and resource allocation algorithm MTOSA is presented and its performance is investigated for several edge scenarios. The proposed algorithm, MTOSA, schedules the devices by choosing the appropriate scheduling policy from the list of five scheduling policies, i.e., RR, SC, MR, PB and PF. The analysis of MTOSA results shows that each scheduling algorithm performs optimally under the specific and predefined scenario with the IoE cluster. This scheduling performance analysis helps to optimise the IoE device life span and working within the cluster. The paper showed that for tasks of trivial nature, RR algorithm provides acceptable performance. RR algorithm was also found to be suitable for most tasks since IoE devices have low power and low computational profiles. For tasks requiring more stringent conditions to be met, such as minimum data rates, time constraints and priorities, one of the more sophisticated scheduling policies such as SC, MR, PF and PB should be used for optimal solutions. These results are compared with the two existing scheduling policies from the literature, such as ECT [46] and EBS [47]. It is shown through various experiments that the proposed algorithm MTOSA performs better with respect to every performance parameter studied in this paper compared to the existing policies.

As a part of future work, we would investigate the possible integration of intelligence within the scheduler to achieve more efficiency within the design by analysing the performance history of IoE devices as a parameter to make informed decisions. A future extension can help to refine the overall design for a better edge-based smart environment for futuristic 6G networks.

REFERENCES

[1] C. De Lima, D. Belot, R. Berkvens, A. Bourdoux, D. Dardari, M. Guillaud, M. Isomursu, E. S. Lohan, Y. Miao, A. N. Barreto, and M. R. K. Aziz, "Convergent communication, sensing and localization in 6G systems: An overview of technologies, opportunities and challenges," *IEEE Access*, vol. 9, pp. 26902–26925, 2021.

[2] L. U. Khan, I. Yaqoob, M. Imran, Z. Han, and C. S. Hong, "6G wireless systems: A vision, architectural elements, and future directions," *IEEE Access*, vol. 8, pp. 147029–147044, 2020.

[3] S. U. Jamil, M. A. Khan, and S. Rehman, "Edge computing enabled technologies for secure 6G smart environment-an overview," in *Proc. 12th Int. Conf. Soft Comput. Pattern Recognit. (SoCPaR)*. Cham, Switzerland: Springer, 2021, pp. 934–945.
[4] S. Wenqi, S. Yuxuan, H. Xiufeng, Z. Sheng, and N. Zhisheng, "Scheduling policies for federated learning in wireless networks: An overview," *ZTE Commun.*, vol. 18, no. 2, pp. 11–19, 2020.
[5] U. Bhoi and P. N. Ramanuj, "Enhanced max-min task scheduling algorithm in cloud computing," *Int. J. Appl. Innov. Eng. Manage. (IJAIEM)*, vol. 2, no. 4, pp. 259–264, 2013.
[6] J. Kaur, M. A. Khan, M. Iftikhar, M. Imran, and Q. E. Ul Haq, "Machine learning techniques for 5G and beyond," *IEEE Access*, vol. 9, pp. 23472–23488, 2021.
[7] M. M. Razaq, B. Tak, L. Peng, and M. Guizani, "Privacy-aware collaborative task offloading in fog computing," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 1, pp. 88–96, Feb. 2021.
[8] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-enabled intelligent 6G networks," *IEEE Netw.*, vol. 34, no. 6, pp. 272–280, Nov./Dec. 2020.
[9] Z.-C. Chen, "Task scheduling algorithm based on campus cloud platform," in *Proc. Int. Conf. Artif. Intell. Secur.* Cham, Switzerland: Springer, 2019, pp. 299–308.
[10] A. Lakhan, M. A. Mohammed, M. Elhoseny, M. D. Alshehri, and K. H. Abdulkareem, "Blockchain multi-objective optimization approach-enabled secure and cost-efficient scheduling for the Internet of Medical Things (IoMT) in fog-cloud system," *Soft Comput.*, vol. 26, pp. 1–14, May 2022.
[11] Y. Wang, X. Qi, X. Lin, and X. Wang, "Computing offloading-based task scheduling for space-based cloud-fog networks," in *Proc. 2nd Int. Seminar Artif. Intell., Netw. Inf. Technol. (AINIT)*, Oct. 2021, pp. 266–270.
[12] M. K. Hussein and M. H. Mousa, "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2020.
[13] N. Magaia, P. Ferreira, P. R. Pereira, K. Muhammad, J. D. Ser, and V. H. C. De Albuquerque, "Group'n route: An edge learning-based clustering and efficient routing scheme leveraging social strength for the internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, early access, May 16, 2022, doi: 10.1109/TITS.2022.3171978.
[14] R. Lin, T. Xie, S. Luo, X. Zhang, Y. Xiao, B. Moran, and M. Zukerman, "Energy-efficient computation offloading in collaborative edge computing," *IEEE Internet Things J.*, early access, May 30, 2022, doi: 10.1109/JIOT.2022.3179000.
[15] Q. Zhang, L. Gui, S. Zhu, and X. Lang, "Task offloading and resource scheduling in hybrid edge-cloud networks," *IEEE Access*, vol. 9, pp. 85350–85366, 2021.
[16] A. Hazra, P. K. Donta, T. Amgoth, and S. Dustdar, "Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial IoT applications," *IEEE Internet Things J.*, early access, Feb. 9, 2022, doi: 10.1109/JIOT.2022.3150070.
[17] S. K. Uz Zaman, A. I. Jehangiri, T. Maqsood, Z. Ahmad, A. I. Umar, J. Shuja, E. Alanazi, and W. Alasmary, "Mobility-aware computational offloading in mobile edge networks: A survey," *Cluster Comput.*, vol. 24, pp. 1–22, Dec. 2021.
[18] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2131–2165, 4th Quart., 2021.
[19] Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for internet of vehicles: Model and algorithm," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5913–5925, Sep. 2021.
[20] F. Zhao, Y. Chen, Y. Zhang, Z. Liu, and X. Chen, "Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 2154–2165, Jun. 2021.
[21] I. Kovacevic, E. Harjula, S. Glisic, B. Lorenzo, and M. Ylianttila, "Cloud and edge computation offloading for latency limited services," *IEEE Access*, vol. 9, pp. 55764–55776, 2021.
[22] F. Alqahtani, M. Amoon, and A. A. Nasr, "Reliable scheduling and load balancing for requests in cloud-fog computing," *Peer-to-Peer Netw. Appl.*, vol. 14, pp. 1–12, Jul. 2021.
[23] X. Huang, R. Yu, D. Ye, L. Shu, and S. Xie, "Efficient workload allocation and user-centric utility maximization for task scheduling in collaborative vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3773–3787, Apr. 2021.
[24] N. Cha, C. Wu, T. Yoshinaga, Y. Ji, and K.-L.-A. Yau, "Virtual edge: Exploring computation offloading in collaborative vehicular edge computing," *IEEE Access*, vol. 9, pp. 37739–37751, 2021.
[25] S. Zhou and W. Jadoon, "Jointly optimizing offloading decision and bandwidth allocation with energy constraint in mobile edge computing environment," *Computing*, vol. 103, pp. 1–27, Dec. 2021.
[26] X. Huang, K. Wu, M. Jiang, L. Huang, and J. Xu, "Distributed resource allocation for general energy efficiency maximization in offshore maritime device-to-device communication," *IEEE Wireless Commun. Lett.*, vol. 10, no. 6, pp. 1344–1348, Jun. 2021.
[27] J. Cao, X. Song, S. Xu, Z. Xie, and Y. Xue, "Energy-efficient resource allocation for heterogeneous network with grouping D2D," *China Commun.*, vol. 18, no. 3, pp. 132–141, Mar. 2021.
[28] M. Wang, T. Zhu, T. Zhang, J. Zhang, S. Yu, and W. Zhou, "Security and privacy in 6G networks: New areas and new challenges," *Digit. Commun. Netw.*, vol. 6, no. 3, pp. 281–291, Aug. 2020.
[29] R. Cong, Z. Zhao, G. Min, C. Feng, and Y. Jiang, "EdgeGO: A mobile resource-sharing framework for 6G edge computing in massive IoT systems," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14521–14529, Aug. 2021.
[30] C. Pu and L. Carpenter, "Psched: A priority-based service scheduling scheme for the Internet of Drones," *IEEE Syst. J.*, vol. 15, no. 3, pp. 4230–4239, Sep. 2020.
[31] X. Fan, B. Liu, C. Huang, S. Wen, and B. Fu, "Utility maximization data scheduling in drone-assisted vehicular networks," *Comput. Commun.*, vol. 175, pp. 68–81, Jul. 2021.
[32] X. Li, R. Fan, H. Hu, N. Zhang, X. Chen, and A. Meng, "Energy-efficient resource allocation for mobile edge computing with multiple relays," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10732–10750, Jul. 2021.
[33] X. An, R. Fan, H. Hu, N. Zhang, S. Atapattu, and T. A. Tsiftsis, "Joint task offloading and resource allocation for IoT edge computing with sequential task dependency," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16546–16561, 2022.
[34] M. Afhamisis and M. R. Palattella, "SALSA: A scheduling algorithm for LoRa to LEO satellites," *IEEE Access*, vol. 10, pp. 11608–11615, 2022.
[35] E. M. Shiriaev, N. N. Kycherov, and V. A. Kuchukov, "Analytical review of the methods of dynamic load balancing under conditions of uncertainty in the execution time of tasks," in *Proc. IEEE Conf. Russian Young Res. Electr. Electron. Eng. (ElConRus)*, Jan. 2021, pp. 674–677.
[36] C. Wu, E. Haihong, and M. Song, "A distributed task scheduling system suitable for massive environments," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Feb. 2020, pp. 750–755.
[37] T. K. Rodrigues, J. Liu, and N. Kato, "Offloading decision for mobile multi-access edge computing in a multi-tiered 6G network," *IEEE Trans. Emerg. Topics Comput.*, early access, Jun. 21, 2021, doi: 10.1109/TETC.2021.3090061.
[38] Z. Liao, J. Peng, J. Huang, J. Wang, J. Wang, P. K. Sharma, and U. Ghosh, "Distributed probabilistic offloading in edge computing for 6G-enabled massive Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5298–5308, Apr. 2021.
[39] Z. Wang, B. Lin, L. Sun, and Y. Wang, "Intelligent task offloading for 6G-enabled maritime IoT based on reinforcement learning," in *Proc. Int. Conf. Secur., Pattern Anal., Cybern. (SPAC)*, Jun. 2021, pp. 566–570.
[40] H. Hu, W. Song, Q. Wang, F. Zhou, and R. Q. Hu, "Mobility-aware offloading and resource allocation in MEC-enabled IoT networks," in *Proc. 16th Int. Conf. Mobility, Sens. Netw. (MSN)*, Dec. 2020, pp. 554–560.
[41] G. Feng, X. Li, Z. Gao, C. Wang, H. Lv, and Q. Zhao, "Multi-path and multi-hop task offloading in mobile ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5347–5361, Jun. 2021.
[42] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.
[43] D. Wu, L. Deng, Z. Liu, Y. Zhang, and Y. S. Han, "Reinforcement learning random access for delay-constrained heterogeneous wireless networks: A two-user case," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2021, pp. 1–7.
[44] S. U. Jamil, M. A. Khan, and S. U. Rehman, "Intelligent task off-loading and resource allocation for 6G smart city environment," in *Proc. IEEE 45th Conf. Local Comput. Netw. (LCN)*, Nov. 2020, pp. 441–444.
[45] S. Ur Rehman, M. A. Khan, M. Imran, T. A. Zia, and M. Iftikhar, "Enhancing Quality-of-Service conditions using a cross-layer paradigm for ad-hoc vehicular communication," *IEEE Access*, vol. 5, pp. 12404–12416, 2017.

[46] M. Abd Elaziz, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," *Future Gener. Comput. Syst.*, vol. 124, pp. 142–154, Nov. 2021.
[47] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.
[48] B. Wang, H. Zhu, H. Xu, Y. Bao, and H. Di, "Distribution network reconfiguration based on NoisyNet deep Q-learning network," *IEEE Access*, vol. 9, pp. 90358–90365, 2021.

SYED USMAN JAMIL (Graduate Student Member, IEEE) received the M.Sc. degree in computer science from GC University Lahore, Pakistan, in 2004, the Master of Business Administration degree from Allama Iqbal Open University, Pakistan, in 2009, the Master of Information Technology degree in network computing from the University of Canberra, Australia, in 2014, and the Bachelor of Computing degree (Hons.) from Charles Sturt University, in 2019. He is currently a Ph.D. Scholar at Charles Sturt University. He was included on the Executive Dean's List for outstanding academic achievement with the School of Computing, Mathematics and Engineering, Charles Sturt University. He has previously worked in industry as a System Administrator for a number of years. His research interests include the areas of cloud computing technologies, the Internet of Everything (IoE), resource allocation, scheduling, intelligence, and information security for wireless networks.

M. ARIF KHAN (Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Engineering and Technology Lahore, Pakistan, the M.S. degree in electronic engineering from the GIK Institute of Engineering Sciences and Technology, Pakistan, and the Ph.D. degree in electronic engineering from Macquarie University Sydney, Australia. He is currently a Senior Lecturer with the School of Computing, Mathematics and Engineering, Charles Sturt University, Australia. His research interests include future wireless communication technologies, smart cities, massive MIMO systems, and cyber security. He was a recipient of the Prestigious International Macquarie University Research Scholarship (iMURS), and ICT CSIRO scholarships for his Ph.D. degree. He also has the competitive GIK Scholarship for his master's degree.

SABIH UR REHMAN (Member, IEEE) received the bachelor's degree (Hons.) in electronics and telecommunication engineering from the University of South Australia, Adelaide, and the Ph.D. degree in the area of wireless sensor networks from Charles Sturt University, Australia. He is currently a Senior Lecturer and the Course Director of the School of Computing, Mathematics and Engineering, Charles Sturt University. He has extensive industry experience in delivering large scale networking solutions along with providing system administration, security, and data integration services. His current research interests include wireless communication, network planning, routing and switching, information security, the Internet of Things (IoT), robotics and big data analytic, especially in the domains of intelligent transport systems, environmental sustainability, e-health and precision agriculture, with the aim to positively influence social, economic, and environmental sustainability of communities in rural Australia via digitally enabled solutions. He regularly publishes his research and serves as a reviewer for a number of respected journals and conferences. He is a member of Australian Computer Society (ACS).