Received: 24 July 2024 | Revised: 4 January 2025 | Accepted: 8 January 2025

IET Wireless Sensor Systems

DOI: 10.1049/wss2.70000

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Adaptive power management for multiaccess edge computing-based 6G-inspired massive Internet of Things

Babatunde S. Awoyemi | **Bodhaswar T. Maharaj**

Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, South Africa

Correspondence

Babatunde S. Awoyemi.
Email: awoyemibabatunde@gmail.com

Funding information

SENTECH Chair in Broadband Wireless and Multimedia Communications at the University of Pretoria

Abstract

Multiaccess edge computing (MEC) is a dynamic approach for addressing the capacity and ultra-latency demands caused by the pervasive growth of real-time applications in next-generation (xG) wireless communication networks. Powerful computational resource-enriched virtual machines (VMs) are used in MEC to provide outstanding solutions. However, a major challenge with using VMs in xG networks is the high overhead caused by the excessive energy demands of VMs. To address this challenge, containers, which are generally more energy-efficient and less computationally demanding, are being advocated. This paper proposes a containerised edge computing model for power optimisation in 6G-inspired massive Internet-of-Things applications. The problem is formulated as a central processing unit energy consumption cost function based on quasi-finite system observations. To achieve practicable computational complexity, an approach that uses a search heuristic based on Lyapunov techniques is employed to obtain near-optimal solutions. Important performance metrics are successfully predicted using the online look-ahead technique. The predictive model used achieves an accuracy of 97% prediction compared to actual data. To further improve resource demand, an adaptive controller is used to schedule computational resources on a time slot basis in an adaptive manner while continuing to receive workload levels to plan future resource provisioning. The proposed technique is shown to perform better compared to a competitive baseline algorithm.

KEYWORDS

cloud computing, Internet of Thing, learning (artificial intelligence), massive IoT, optimisation, reliability

1 | INTRODUCTION

The proliferation of next-generation (xG) network services and applications continues to result in the generation of massive amounts of data. In most xG applications, data transmission and usage are usually delay-sensitive [1]. Delay-sensitive data must be processed with minimal delay, which has been a major challenge in network core or central cloud processing network design. In particular, congestion build-up on the network backbone and high transmission delays have been the main challenges faced by network operators [2]. In response, multiaccess edge computing (MEC) solutions are being advocated to see an accelerated shift from cloud processing to edge processing, bringing processing closer to the data sources. This paradigm shift brings many opportunities to improve the network quality of service (QoS), which in turn improves the welfare of end users [3].

An important application of MEC in emerging xG communications is in large-scale or massive Internet of Things (mIoT) networks. Without a doubt, MEC can improve several aspects of the mIoT network. For instance, by adopting edge computing capabilities, communication and computational resources are made more easily available and accessible. This, in turn, can enhance the ability of mIoT devices in the network to return computational results much more quickly, thus helping to improve the network's decision-making and action-taking

processes. Furthermore, edge computing can promote a new round of technological revolution for the mIoT and other similar xG networks with computation-hungry and delay-intensive applications whose QoS requirements cannot be guaranteed by present, prevailing 5G networks. For such applications, high computational capacity is crucial, especially in use cases that employ computationally demanding high-level algorithms [4]. Such algorithmic requirements are indispensable in real-time autonomous missions such as in mobile robots and autonomous driving.

Current 5G networks pose a performance bottleneck to the higher energy demands and transmission rates of emerging xG applications (for instance, the mIoT being considered in this paper). This makes it imperative to develop appropriate power management models and solutions for beyond 5G (B5G) mIoT networks using edge computing capabilities. That is the goal this paper intends to achieve. To achieve the goal, in the paper, an adaptive power management model is investigated in a 6G-inspired mIoT environment using a quasi-finite horizon edge platform. The resulting power management problem is developed as an optimisation problem and solved using appropriate tools to achieve the desired accuracy and latency demands of the mIoT and other similar B5G network applications.

The organisation of the rest of this paper is briefly described. Section 2 provides a review of recent and relevant literature to establish the important body of work. Section 3 explains the proposed system model. Section 4 discusses the mathematical formulation of the problem. Section 5 presents the proposed online adaptive power management algorithm. Section 6 provides the evaluation results and discussions. Lastly, Section 7 gives the concluding remarks. A list of the acronyms and notations used in this paper is provided in Table 1.

TABLE 1 List of acronyms.

Acronym and notations	Description
5 G/6 G	Fifth generation/sixth generation
AI	Artificial intelligence
ANN	Artificial neural network
B5G	Beyond fifth generation
BES	Bald eagle search
BPNN	Belief propagation neural network
CMDP	Constrained Markov decision process
CPU	Central processing unit
ETSI	European Telecommunications Standards Institute
gNB	Generation NodeB or just gNodeB
IoT/mIoT	Internet-of-Things/massive Internet-of-Things
MDP	Markov decision processes
MEC	Multi-access edge computing
NFV	Network function virtualisation
NICs	Network interface controllers
OFDMA	Orthogonal frequency division multiple access
QoE	Quality of experience
QoS	Quality of service
RMSE	Root mean-squared error
SFC	Service function chaining
SINR	Signal-to-interference-plus-noise ratio
VLAN	Virtual local area network
VM	Virtual machine
VNFs	Virtualised network functions
WSN	Wireless sensor networks
xG	Next generation

2 | LITERATURE REVIEW AND RESEARCH MOTIVATION

The MEC paradigm has brought novel computing offloading-intensive capabilities that promise dramatic latency reduction in modern and emerging xG technologies and applications. However, state-of-the-art MEC algorithms tend to have high computational complexity and demand. MEC's high computational demand may be a huge bottleneck for emerging B5G technologies with hopes of achieving extremely impressive speeds, low latency, massive throughput, etc. Recent ongoing research works are directed towards developing appropriate MEC-based models and solutions for B5G networks and applications. From recent works, it is clear that the tools of optimisation and artificial intelligence (AI) are among the most potent tools employed to achieve desirable performance expectations with outstanding learning and prediction capabilities in MEC-based IoT and other emerging xG networks.

In ref. [5], a bald eagle search (BES) algorithm was proposed to reduce the high computational complexity of the deep learning approaches employed shorten the latency and minimise the energy consumed in an MEC-enabled IoT network. Previous BES algorithms employed three stages in the decision-making process, namely (i) select, (ii) search, and (iii) swooping stages. To achieve better performance than previous BES algorithms, the proposed approach introduced an estimation stage to select better resources and edge systems. In this way, devices could offload the most appropriate IoT tasks to the edge servers, minimising the expected time execution. Based on multi-user offloading, the authors then proposed a BES optimisation algorithm which they claimed effectively reduced the end-to-end time, thereby achieving near-optimal solutions for the IoT devices in the network.

To improve the service supply capability of wireless sensor network (WSN)-assisted IoT applications, the authors in ref. [6] proposed a system that integrates MEC technology, simultaneous wireless information, and power transfer

technology. A novel optimisation problem was formulated to minimise the total system energy consumption under the constraints of data transmission rate and transmitting power requirements. The optimisation problem jointly considered power allocation, central processing unit (CPU) frequency, offloading weight factor, and energy harvest weight factor. Since the problem was non-convex, a novel alternate group iteration optimisation algorithm was proposed to decompose the original problem into three subproblems. Each subproblem was alternately optimised using the group interior point iterative algorithm. The numerical results obtained showed that the energy consumption of the proposed approach was lower than that of the benchmark algorithms.

The authors in ref. [7] studied the impact of the behavioural characteristics of users and MEC server pricing policy to determine their optimal offloading strategies. Here, prospect theory concepts were exploited to reflect the subjectivity and satisfaction of users from the data offloading system. The probability of failure of the MEC servers to potential over-exploitation was modelled via the theory of tragedy of the commons. A multi-leader multi-follower Stackelberg game was formulated among the users and the MEC servers to determine optimal pricing policies and offloading strategies. Data offloading decision-making for users was formulated as a non-cooperative game among the users and a Nash equilibrium was determined. The evaluation results demonstrated the superiority of the proposed framework against other benchmark alternatives.

In ref. [8], the authors proposed an efficient solution for multiple data collection tasks in an MEC-enabled WSN environment being used for smart agriculture. The WSN setup in the agricultural space is quite similar to the IoT network setup. For the model [8], edge servers were deployed around the WSN nodes, which then provided computing services for the network. The model developed a WSN framework through which different tasks could be completed at a lower data collection time. This was achieved by selecting the most appropriate WSN node among the available nodes in the network. The node selection was carried out by dynamically configuring the sensor nodes to accomplish the set tasks while keeping within a set time frame. By employing classical optimisation tools, the developed model was shown to process a higher volume of valid data at a much lower data collection time.

The works reviewed (and other similar works in the literature) have shown the importance of optimisation and AI as modern tools for driving MEC capabilities in xG networks. However, the concept and use of containerisation and virtualisation in achieving MEC capabilities in such networks is still very scarcely explored in the literature. Containerisation deals more with software or operating system-level virtualisation, while virtualisation broadly refers more to hardware virtualisation, such as the use of virtual machines (VMs). Particularly for MEC-based 6G-inspired networks, there are a few works in the literature that have explored the concept of virtualisation and/or containerisation. Some works on this subject are briefly reviewed in this Section.

In ref. [9], the authors explored container virtualisation technology as a promising tool for achieving the kind of flexibility required in operations, service demands, resource management etc. of 6G and other B5G networks. The authors explained that for packets to be delivered on time, several network functions have to be deployed close to the mobile device. Virtualisation makes it possible for mobile devices with computational resource capabilities to automatically deploy and run Virtual Network Functions (VNFs) to achieve such high-level packet delivery expectations. The authors then studied and recommended some existing technologies suitable for applying Network Function Virtualisation (NFV) for 5G core network functions in the industrial environments space.

The authors in ref. [10] proposed a virtualised open-source MEC (OS-MEC) scheme for application in B5G networks. The scheme was built on the key principles of network decoupling and reconfiguration. A service-based MEC layer was developed for the OS-MEC scheme to achieve the decoupling or decomposition. The layer was able to decouple the tightly coupled service functions into multiple independent network functions. The authors then employed the concepts of the templates and instances to reassemble the disaggregated NFs and to reallocate the necessary resources in order to provide customised services for users. Some use cases were presented to validate the flexibility and customisation of the test network.

In ref. [11], the authors studied VNF migration and service function chaining reconfiguration problems in dynamic NFV-enabled xG networks, with particular applications in 6G networks. In the work, the authors formulated the VNF migration problem as an optimisation model to minimise the end-to-end delay of all influenced SFCs while guaranteeing network load balance after migration. Then, a deep learning-based two-stage algorithm was designed to solve the VNF migration problem. By combining previous experimental data, realistic VNF traffic patterns were generated and used to evaluate the algorithm. The results obtained showed that a much more load-balanced network was realised.

The works in refs. [12, 13] and a few others have established some of the advantages of using containerisation over VMs in MEC-enabled networks. Some of the most important benefits of the use of containers that have been identified include portability, platform independence, resource savings, scalability, and higher productivity. Although container-based virtualisation technologies significantly increase application availability, in comparison with VMs, however, containers suffer from expensive communication overhead and resource use imbalances. Furthermore, there are still some problematic aspects when containerised edge systems are designed to provide real-time support, especially the aspects of high latency (or migration delays), anomaly detection (or security implications), and automated monitoring and control (or deployment issues), among others. The works mentioned provided some possible solutions to these containerisation challenges. The work in ref. [12] developed some joint task containerisation and container placement methods to reduce communication overhead and balance multi-type computing resource

utilisation, while [13] employed deep reinforcement learning to reduce communication overhead, improve energy consumption, and latency expectations. In other words, although there are some identified challenges, containerised MEC still provides better solutions compared to VM-based MEC, as recent works (such as [12, 13]) have already demonstrated and reported. Interestingly, a lot more work is still being done in this regard.

The works reviewed so far have presented the state-of-the-art in the development of virtualisation and the application of AI-driven solutions in emerging MEC-based xG networks. From the reviewed works, we note that there are still some research gaps as several important aspects and considerations in practical MEC-based B5G networks still need to be addressed. For example, due to the fast data exchange demands in B5G networks, the behaviour of an edge server can change rapidly in a very short time. It is therefore paramount that each edge system is monitored to gain a certain level of stability and reliability, which is still a research area that needs more attention. Furthermore, the aspect of energy or power management in edge computing has yet to be adequately addressed, especially in B5G networks such as the 6G-inspired mIoT applications that are considered in this paper. Then, the increasing deployment of IoT devices and applications continues to result in data explosion and processing energy insufficiency, which are among the most pervasive challenges facing edge computing to date. The main motivation for this research work is the development and use of AI-based prediction and control for energy or power management on 6G edge platforms.

To apply AI-based prediction and control, the paper proposes adaptive power management using predictive control techniques to provide a cost-effective strategy capable of monitoring the energy consumption of the edge platform. Firstly, a containerised edge computing platform for mIoT applications is developed and the power management problem is formulated as a CPU energy consumption cost function. Thereafter, quasi-finite system observations are applied to introduce predictive capabilities for a look-ahead technique that performs forecast operations over a quasi-finite horizon edge platform. An adaptive controller is used to schedule computational resources on a time-slot basis using a search heuristic and Lyapunov technique.

The technique proposed in this work is evaluated in terms of welfare gain, server response rate, and energy consumption and shows better performance against a competitive baseline algorithm. The welfare gain, defined as a function of the network's performance with an increasing number of IoT devices is used as a measure of the quality of experience (QoE) of users in the network. The algorithm used as the baseline for this work is the one presented in ref. [14]. The baseline algorithm is developed using the standard from the European Telecommunications Standards Institute (ETSI) [15] to provide a mobile resource-sharing framework that employs mobile edge servers to achieve cost-effective deployment of 6G edge computing, thereby enabling edge resource sharing for massive IoT devices. Since our developed model also follows the ETSI standard, it is therefore easy to implement using established container orchestration frameworks (such as Docker and Kubernetes) without difficulty. In addition, the developed model is scalable, feasible under fluctuating loads, and can perform relatively well across different networks and/or network conditions or constraints. This is because of the predictive capabilities of the solution model employed. Also, the adaptive power management scheme implemented helps the network achieve improved energy consumption and service overhead. Overall, the proposed algorithm gives better performance over the baseline even with a lower value of CPU frequency in the 6G mIoT network.

3 | PROPOSED SYSTEM MODEL

The system model in consideration is described in Figure 1. The system model is a single-cell scenario of a 6G-inspired mIoT network. In the model, the gNodeB (gNB) is equipped with an edge server and serves a set $\mathcal{K} = \{1, 2, \ldots, K\}$ of mIoT devices. The edge architecture consists of a configured virtual local area network (VLAN), where the gNB is connected to the edge server using a high-speed fibre-based fronthaul connection [16]. As shown in the system model in Figure 1, each mIoT device may process its tasks by the local CPU or offload them to the MEC server. Such a decision is influenced by the computational power required to carry out the tasks. The possible interaction between the MEC server and each local IoT device is described on the right-hand side of Figure 1. In line with the ETSI proposed MEC deployment scenarios, the network is made up of a reconfigurable platform and a related switched VLAN with network interface controllers connected to it. It is assumed that each mIoT device aims to maximise its processing capacity by taking advantage of the CPU and radiofrequency (RF) modules under power and subcarrier constraints. Therefore, the total battery power available in each mIoT device is assigned to a local CPU, which uses RF modules to determine and conduct its operations.

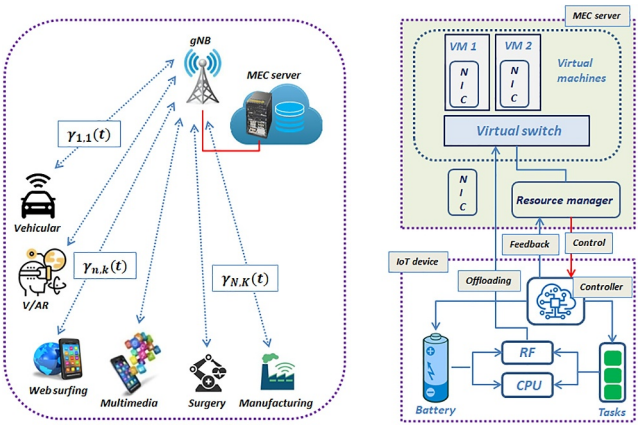

FIGURE 1 Multi-user edge computing applicable in the massive Internet of Things network. Arrows indicate the direction of data flow.

3.1 | The mIoT devices

In terms of local processing, each mIoT device consists of a local processing queue and an intelligent computational agent. The decision-making process of the local CPU is described in Figure 2. In Figure 1, the role of the controller, which is hosted as an application, is to schedule arriving computational tasks, which are then processed locally or by offloading them to the edge server. The local CPU is characterised by a computational frequency $f^{\eta}_{k,\mathrm{cpu}}$, with a frequency upper bound of $f_{k,\max}$. Here, the value $\eta \geq 2$ is a constant determined by the architecture of the CPU core. Also, the effective capacitance of the chip is represented as $\psi_k > 0$ [16].

3.2 | The edge server

If the controller schedules a computational task to be processed at the edge server, the corresponding task is offloaded via a wireless link. It is assumed that the MEC system adopts orthogonal frequency division multiple access (OFDMA) with a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of orthogonal subcarriers. For ease of exposition, it is assumed that both the mIoT devices and the gNB are equipped with a single antenna and communication between them is characterised by a signal-to-interference-plus-noise ratio, $\gamma_{n,k}(t)$. The offloaded tasks reach the virtual switch/router gateway at the rate $r_{n,k}(\mathbf{p}, t)$, where $\mathbf{p}$ represents the vector of possible transmission powers for the K independent offloading links. Arriving tasks are stored in a processing queue of length Q_{in} as shown in Figure 3 [17].

The edge platform shown in Figure 3 consists of an input queue, which is an arrival queue of size Q_{in}; and an output queue, which is a return queue of size Q_{out}; a dynamic VM manager and task scheduler. It is assumed that the edge server hosts a set $\mathcal{C} = \{1, 2, \ldots, C\}$ of worker agents called containers, which are allocated to tasks on demand. The hosted controller is responsible for assigning containers to admitted tasks for processing in terms of VNFs [18]. Control actions $\varsigma(t)$ determine the required processing frequency $f_{n,c}(t)$, which is determined through the virtualised switch by assigning a load-dependent factor $\beta_c(t)$.

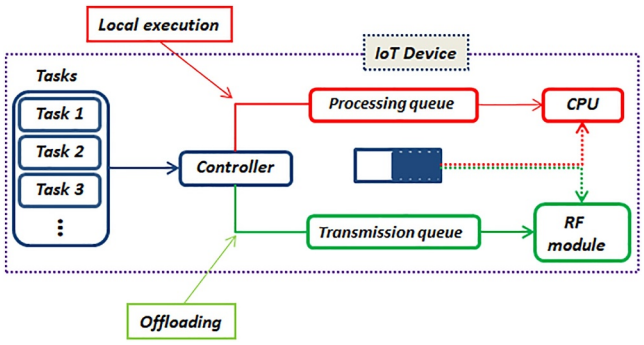

FIGURE 2 Device-level computation and local decision-making.

4 | MATHEMATICAL PROBLEM FORMULATION

In this section, the mathematical formulation of the 6G-inspired mIoT network model is presented and analysed.

4.1 | Local computation model

In the 6G-inspired mIoT network model in consideration, task arrival at the kth mIoT device is considered an event that initiates a scheduling decision on whether the task will be executed locally or remotely. If the decision is for local execution, the local computational power can be modelled as follows:

$$p_k(t) = f^{\eta}_{k,\mathrm{cpu}}, \tag{1}$$

where $f^{\eta}_{k,\mathrm{cpu}}$ In terms of local computation, the local CPU is characterised by a computational frequency $f^{\eta}_{k,\mathrm{cpu}}$, with a frequency upper bound of $f_{k,\max}$. For simplicity and ease of exposition, the value $\eta \geq 2$ is assigned as a constant that is determined by the architecture of the CPU core. Also, the effective capacitance of the chip is represented as $\psi_k > 0$ [16].

4.2 | The communication model

It is assumed that the kth device generates computational tasks at the application layer, and they arrive at the network layer according to a Poisson distribution with arrival rate $\lambda_k(t)$ [19]. This is an event that is assumed to initiate an in-device decision on whether the arrived task will be processed locally or offloaded to the edge server. If the decision is for the computational task to be executed locally, the local computational power can be modelled as follows:

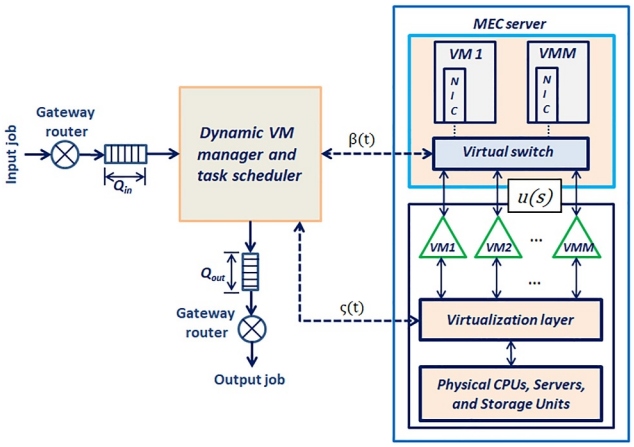

FIGURE 3 Configuration of the MEC server. The solid lines represent bidirectional data flows, while the broken lines indicate bidirectional control actions. MEC, multiaccess edge computing.

$$p_{k,\text{cpu}}^{\text{local}}(t) = f_{k,\text{cpu}}^{\eta}\psi_k. \tag{2}$$

The execution time for the task can be defined as follows:

$$\tau_{k,\text{cpu}}^{\text{local}} = \frac{b_k c_k}{f_{k,\text{cpu}}}, \tag{3}$$

where b_k is the size of the computational task, and c_k is the number of CPU cycles required to process a single byte. From the power consumption defined in ref. [13], the energy consumption for local processing can be expressed as follows:

$$e_{k,\text{cpu}}^{\text{local}} = \kappa \cdot f_{k,\text{cpu}}^{3} \cdot \tau_{k,\text{cpu}}^{\text{local}}, \tag{4}$$

where $\kappa > 0$ is a constant representing the energy consumption due to background data.

4.3 | Computational task offloading

If the kth device offloads the computational task remotely (that is, at the edge server), offloading takes place over the wireless link. It is assumed that the receiving nth gNB adopts an OFDMA scheme to allow multiple simultaneous reception of the K independent offloading links. Therefore, letting B represent the bandwidth of the system, the transmission rate between the kth mIoT device and the nth gNB can be represented as follows:

$$r_{n,k}(t) = B\log_2\left(1 + \frac{p_{n,k}(t)g_{n,k}(t)}{\mathcal{I}_{k'}(t) + N_0}\right), \tag{5}$$

where $p_{n,k}(t)$ is the transmission power, $g_{n,k}(t)$ represents the channel gain, the term $\mathcal{I}_{k'}(t)$ represents the interference caused by simultaneous IoT transmissions excluding the kth device given as follows:

$$\mathcal{I}_{k'}(t) = \sum_{k' \in \mathcal{K}\setminus\{k\}} g_k(t)p_{n,k'}(t). \tag{6}$$

Finally, the term N_0 is the power of the white Gaussian noise. Assuming, for ease of exposition, that all mIoT devices are subjected to similar channel conditions, the transmission rate $r_{n,k}(t)$ in Equation (5) and the task size d_k can be used to define the task offloading latency as follows:

$$\tau_{k,\text{mec}}^{\text{comm}} = \frac{1}{\frac{r_{k,n}(t)}{d_k} - \lambda_k(t)}, \tag{7}$$

where the quotient $\frac{r_{k,n}(t)}{d_k}$ represents the number of computational tasks that can be offloaded from each vehicle through the wireless channel per unit of time. The transmission queue in each vehicle is assumed to be an $M/M/1$ queue and to guarantee queue stability, the condition $\lambda_k(t) < \frac{r_{k,n}(t)}{d_k}$ must be satisfied. Therefore, the offloading energy consumption can be defined as follows:

$$e_{n,k}(t) = p_{n,k}(t) \cdot \tau_{k,\text{mec}}^{\text{comm}}, \tag{8}$$

where $p_{n,k}(t)$ is the transmission power defined in Equation (5). In the edge computing system, the time delay is affected by the load, Equation (7) agrees to flow theory, and its probability distribution follows the Pareto model [20]. So, the evaluation of the computation latency at the edge server can be done by taking into account the number of tasks waiting in the queue and the number of retrials for successful task execution. Therefore, since the average queueing delay can be derived based on Little's Law [21], the task computation delay can be expressed as follows:

$$\tau_{n,\text{mec}}^{\text{comp}} = \frac{1}{\vartheta_n}\left(\frac{1}{\left(f_{n,\text{cpu}}^{\text{mec}} - r_n(\mathbf{n},\mathbf{p},t)\right)}\right) \tag{9}$$

where ϑ_n represents the reliability that a computational task can be executed successfully by the MEC.

According to the Binomial distribution, the expectation of the number of retrying for one successful execution is $\frac{1}{\vartheta}$. The expression $r_n(\mathbf{n},\mathbf{p},t)$ represents the task offloading rate, taking into account the vehicle-gNB association variable and the set of transmission powers $\mathbf{p}$; and $f_{n,\text{cpu}}^{\text{mec}}$ is the computational capability of the CPU processor. The computational power consumption of the MEC can be expressed as follows:

$$p_{n,\text{cpu}}^{\text{mec}}(t) = f_{n,\text{cpu}}^{\text{mec}} \cdot \psi_{n,\text{cpu}}, \tag{10}$$

and the corresponding energy consumption for task completion at the MEC server is expressed as follows:

$$e_{n,\text{cpu}}^{\text{mec}} = \kappa \cdot f_{n,\text{cpu}}^{3} \cdot \tau_{n,\text{cpu}}^{\text{mec}}, \tag{11}$$

where κ is as defined in Equation (4).

4.4 | Formulating the constrained problem

To ensure system stability, the stability of each queue must be monitored using the queue state, which is monitored by the scheduling controller. With the communication rate $r_n(\mathbf{n},\mathbf{p},t)$ being the arrival rate at the computational queue of the MEC server, the evolution of the queue with t can be defined as follows:

$$q_n(t+1) = \max\left\{q_n(t) + r_n(\mathbf{n},\mathbf{p},t) - f_{n,\text{cpu}}^{\text{mec}}(t+1), Q_{\text{in}}\right\}, \tag{12}$$

where $q_n(t+1)$ is the queue length at the upcoming slot, $q_n(t)$ is the current queue length, $f_{n,\text{cpu}}^{\text{mec}} \in \left[f_{\min}^{\text{mec}}, f_{\max}^{\text{mec}}\right]$ is the server's computation capability (in CPU cycles/sec), and Q_{in} is the length of the incoming queue as shown in Figure 3 [22]. Therefore, the processing time is determined by the data transmission time and the execution time in the edge servers, the overall processing delay can be defined as follows:

$$\tau_{\text{delay}}^{\text{total}} = \left\{ \tau_{k,\text{cpu}}^{\text{local}} + \tau_{k,\text{mec}}^{\text{comm}} + \tau_{n,\text{mec}}^{\text{comp}} \right\} \leq \tau^*, \tag{13}$$

where τ^* is the maximum tolerable latency for result return.

4.5 | Formulating a constrained Markov decision process

In use cases where system reliability and time-critical responsiveness are the main concerns, better equipment control is required for fault-free running and decision-making. Since control schemes are formulated better using Markov decision processes [23], a constrained Markov decision process (CMDP) is adopted for the adaptive power management for the quasi-finite horizon edge platform.

4.5.1 | The state space

Contrary to traditional reinforcement learning strategies, where the state space is explicitly defined based on the set of possible states that an agent can be in, in constrained optimisation reinforcement learning, safety is a major concern. Most of the algorithms under CMDP do not consider safety as a concern, which can be a huge challenge for safety-critical IoT use cases supported by the edge. Therefore, enforcing state-wise constraints for challenging environments such as mIoT is essential. The state space of the edge server can be defined as follows:

$$s(t) \triangleq \left\{ r_{n,k}(\mathbf{p}, \mathbf{n}, t), q(t), c(t) \right\}, \quad \forall\, t \in T \tag{14}$$

where T represents the duration over which the system is being observed. It must be noted that all the state components in the state space change with time. More importantly, the number of available computational resources $c(t)$ tends to change rapidly and this needs to be precisely known a priori so that the task offloading rate can be adjusted accordingly. Therefore, an algorithm that will enhance responsiveness to sudden state changes must be applied in such edge computing scenarios.

4.5.2 | The action space and control actions

Edge servers with extremely high computational intensity are required, so the action space of the edge server needs to be designed using its real-time running parameters. In this way, a static policy that cannot adapt to system changes may not be good for real-time decision-making. Therefore, the action space has to be composed of control actions. In control systems theory, control actions are informed by the objective variables, $\upsilon(t)$, which is a sequence of control signals that define the desired behaviour of the processor [24]. Therefore, the action space can be represented as follows:

$$a(t) \triangleq \{\upsilon(t)\}, \quad \forall\, s(t) \in \mathcal{S} \tag{15}$$

which, with reference to Figure 3, includes taking appropriate decisions on the objective variables such as the server processing rate, $f_{n,\text{cpu}}^{\text{mec}}$, to yield a desired computational throughput, $\zeta_{n,\text{cpu}}^{\text{mec}}(t)$. After the action $a(t)$ has been taken, the system migrates to the next state $s(t+1)$, which is the new operational state defined as follows:

$$s(t+1) \triangleq \phi(s(t), \upsilon(t)), \tag{16}$$

where the parameter $\phi(\cdot)$ represents the control input vector that captures the behaviour of the edge system by monitoring the trajectory of the relationship between the current state $s(t)$ and the control action $\upsilon(t)$.

Since the status of the computational queue $q(t)$ and the number of available computational resources $c(t)$ are directly linked to the decision on the processing rate $f_{n,\text{cpu}}^{\text{mec}}$, the power consumption is a direct consequence of the behaviour of the system at that instance. So, if $\hat{x}(t)$ is defined as the power management state of the edge server, the power consumption of the edge computing platform can be represented as follows [25]:

$$p([q(t), c(t)], \upsilon(t)) = \begin{cases} p_{\text{proc}}, & \text{if} \quad \hat{x} = 1, \upsilon = 1 \\ p_{\text{idle}}, & \text{if} \quad \hat{x} = 0, \upsilon = 0 \end{cases}, \tag{17}$$

where as previously defined, υ is the power management action, which is a finite state model consisting of two operational power levels defining server status, that is, on $= 1$ and off $= 0$. The computation of offloaded tasks at the edge server is with respect to the processing rate $f_{n,\text{cpu}}^{\text{mec}}$. Therefore, if the network behaviour is $\hat{x} = 1$, the edge server is in active mode and is actively processing offloaded tasks, and according to Equation (17) $\upsilon = 1$. For the server to positively contribute to the QoS, it must prevent packet losses, and minimise queueing delays, which will subsequently prevent buffer overflows. So, to ensure maximum QoS, the cost function of the computation queue is defined as a function of the network state $s(t)$ and the power management action is as follows:

$$\begin{aligned} \tilde{q}(s(t), \upsilon(t)) = & \sum_{\lambda=0}^{\infty} \sum_{f=0}^{z} p^r(r) p^{\zeta^*}\left(\zeta^* | s, r\right) \\ & \left\{ \hat{\zeta} + \varsigma^* \max\left(\hat{\zeta} + r - Q_{\text{in}}, 0 \right) \right\}. \end{aligned} \tag{18}$$

where the arrival distribution of offloaded tasks is defined using the offloading rate $r_n(\mathbf{n}, \mathbf{p}, t)$ as $p^r(r)$, and the goodput distribution is defined as $p^{\zeta^*}\left(\zeta^* | \cdot, \cdot\right)$ [26], $\hat{\zeta} = \left[q - \zeta^* \right]$ is the holding cost with respect to task arrival rate and processing rate, and Q_{in} is the length of the input queue.

Following Equation (17), the reward of the system is designed based on minimising queueing delays and power consumption. Therefore, the compute-dependent power consumption is defined as follows [27]:

$$P_{\text{comp}}(\upsilon(t), \beta(t)) = \beta(t) \cdot f_{n,\text{cpu}}^3, \tag{19}$$

where the parameter $\beta(t)$ denotes the slope of the trajectory that quantifies the load-dependent power consumption based on arrived tasks. The compute-dependent power consumption Equation (19) translates to the corresponding energy consumption as follows:

$$E_{\text{comp}}(\upsilon(t), \beta(t)) = E_{\text{cpu}}(t) + E_{\text{sw}}(t), \tag{20}$$

where $E_{\text{cpu}}(t)$ is the energy consumption caused by CPU utilisation. Since the containers are instantiated on top of CPU cores, each container processes the currently allocated task by managing its local computing resources.

4.6 | Formulating the optimisation problem

Since the power management objective is to minimise power cost subject to delay constraints, latency minimisation becomes the main objective of the proposed strategy by default, fulfilling the time constraint $\tau_{\max}$. Therefore, the objective of minimising the power costs can be represented as that of minimising the MEC energy consumption due to CPU processing in Equation (23) and can be defined as follows:

$$\mathbf{P} : \text{argmin}_{\Omega} \sum_{t \in T} E_{\text{cpu}}(\upsilon(t), \beta(t)), \tag{21}$$

subject to

$$\begin{aligned}
&\mathbf{C1} : \delta_{\text{k,n}}(t) \in \{0, 1\}, \\
&\mathbf{C2} : r_0 \leq r_{n,k}(\mathbf{p}, \mathbf{n}, t) \leq r_{\max}, \\
&\mathbf{C3} : f_0 \leq f_{n,c}(t) \leq f_{\max}, \\
&\mathbf{C4} : \chi_{n,c}(t) \leq \Delta, \\
&\mathbf{C5} : r(t) \leq r_{\max}, \\
&\mathbf{C6} : \max\{2\zeta_c(t)\} + \Delta \leq \tau^*_{\max},
\end{aligned} \tag{22}$$

where $\Omega \triangleq \{\kappa, \beta(t),\}$ denote the objective decision variables that need to be configured at each time slot t of the finite system observation horizon T. The constraint **C1** is the hard constraint, which is a binary decision variable as defined in Equation (13). The constraint **C2** bounds the maximum task arrivals at the MEC server, noting the size of the incoming queue, Q_{in}, and the fact that in practical applications the maximum per-container computation load is limited by $r_{\max}$. The constraint **C3** bounds the maximum processing rate per CPU considering that container provisioning and workload allocation is governed by $f_{n,c}(t) \triangleq r_{n,k}(\mathbf{p}.\mathbf{n}, t)/\Delta$, where Δ is the server response time. Then, constraint **C4** presents a hard limit on the per-slot and per-container processing time, which guarantees that containers execute the assigned tasks within Δ seconds. The constraint **C5** bounds the aggregate communication rate sustainable by the VLAN to $r_{\max}$, and since the two-way communication and server response time define the upper bound on processing time. Here, it is assumed that the size of the vectors remains constant throughout the time slot. Lastly, constraint **C6** forces the processing rate, $\zeta_c(t)$, and the server response time Δ not to exceed the maximum predefined communication-plus-computation time $\tau_{\max}$.

Considering $E_{\text{cpu}}(t)$ to be related to the number of containers running at time slot t, the CPU frequency allocated to each container can be obtained using a linear relationship between CPU utilisation and the energy consumed as follows:

$$E_{\text{cpu}}(t) = \sum_{c=1}^{C(t)} E_{\text{dyn}}(t) = \sum_{c=1}^{C(t)} \beta(t)(E_{\max}(t) - E_{\text{idle}}(t)), \tag{23}$$

where $E_{\text{dyn}}(t)$ is the dynamic energy component of each container c, with the load-dependent factor $\beta(t) = \left(\frac{f_{n,c}(t)}{f_{\max}(t)}\right)^2$, and $E_{\max}(t)$ is the maximum energy that can be consumed by container c. The term $E_{\text{sw}}(t)$ from Equation (20) is the energy consumed while changing processing rates, which depends on the absolute gap in the processing rates as follows:

$$E_{\text{sw}}(t) = \sum_{n=1}^{N} \kappa_e \left(f_{n,c}(t+1) - f_{n,c}(t)\right)^2, \tag{24}$$

where the parameter κ_e is the per-container reconfiguration cost caused by a unit size frequency switching.

5 | PROPOSED POWER MANAGEMENT ALGORITHM

For the proposed power management algorithm, it is assumed that the stability of the worst-case queue in terms of size and server response times is guaranteed. From constraint **C4**, it must be noted that the workload allocated to container c, that is, $r_{n,k}(\mathbf{p}, \mathbf{n}, t)$, and the processing frequency, $f_{n,c}(t)$ will yield the desired processing time of $\chi_{n,c}(t)$. So, a cost-effective strategy for minimising the cost function in Equation (21) is crucial. However, due to the existence of Equation (18) and constraint **C6**, Equation (21) is a non-convex optimisation problem and the use of convex optimisation to solve the problem is difficult. Even if it were possible to employ convex optimisation approaches, it would be very computationally demanding and impracticable to implement in real-life scenarios. Thus, the use of reinforcement learning solutions with predictive capabilities significantly reduces the computational demand. The solutions provided are near-optimal, hence, optimality is not sacrificed. According to constraint **C6**, the system has to maximise the two-way communication while minimising the round-trip latency, which makes Equation (21) NP-hard. To satisfy the delay constraint $\tau_{\max}$, a reliable link between the scheduler and the server operating at transmission rate $r_{n,k}(t)$ (bits/sec) and processing rate $f_{n,c}(t)$ are respectively assumed. To enhance

the feasibility of this problem with QoS guarantees, constraints **C2** and **C4** must hold—taking into account that the maximum per-slot communication rate is $r_{\max}$. The random queueing delays at the input and output queues together with the server computation time give the overall processing delay in Equation (13), and the cost function in Equation (18) is the QoS guarantee that is reported to the scheduler. Therefore, the energy consumed in sustaining the two-way communication can be defined as follows:

$$E_{\text{comm}}(\upsilon(t),\zeta(t)) = 2\sum_{n=1}^{N} P_{n,k}\big(r_{n,k}(t)\big)\cdot\zeta_{n,c}(t), \tag{25}$$

where the power-rate function $P_{n,k}\big(r_{n,k}(t)\big)$ can be expressed as follows:

$$P_{n,k}\big(r_{n,k}(t)\big) = S_{n,k}\Big(2^{r_{n,k}(t)/B(t)} - 1\Big), \tag{26}$$

which is the Shannon–Hartley exponential discussed in ref. [28], in which case $S_{n,k} = \frac{B(t)\cdot N_0}{g_{n,k}}$, $B(t)$ is the link bandwidth as defined in Equation (5), and the server processing rate $\zeta_{n,c}(t) = \frac{r_{n,k}(\mathbf{p},\mathbf{n},t)}{f_{n,c}(t)}$.

The likely correlation between the workload and energy consumption is required in deriving the adaptive power management scheme. Here, estimates of the server response time and energy consumption will be used in estimating the server processing rate for the next time slot. Let $\Delta t < t$ be the smallest possible step size through which the system is monitored. Based on an initial behaviour estimate $\hat{x}(t)$ Equation (17), the adaptive controller explores possibilities of an appropriate decision υ. It does this by creating a tree of depth T which it traverses in steps of size Δt until it arrives at optimum values for the next time slot $t + 1$. This is described as follows:

$$\hat{x}(t+1) \triangleq \Big(\hat{\omega}(t+\Delta t), \hat{E}(t+\Delta t)\Big), \tag{27}$$

where $\hat{\omega}(t+1)$ is the server response time, and $\hat{E}(t+1)$ is the energy consumption, which is used to predict the next processing rate. The problem can also be solved by using the cost function Equation (21), which can be solved by adopting the solution from ref. [27]. Therefore, if $x(t+\Delta t)$ is the actual estimate that coincide with control actions $a(t+\Delta t) \in \upsilon(t)$, then $J(x(t+\Delta T))$ is the optimal cost that is obtained by solving Equation (21) based on the new current estimate $x(t+\Delta T)$, provided that the current cost function has decreased. Then, the condition to determine the next step is obtained by checking if the current cost function, that is, the optimal cost, has indeed decreased. A Lyapunov candidate [29] is used to check if the current cost function is guaranteed to decrease or not as follows [27]:

$$J(x(t+\Delta T)) - J(x(t)) < 0. \tag{28}$$

It was proven in ref. [28] that the stability of Equation (28) could be achieved using

$$\hat{J}(\hat{x}(t+\Delta T)) - \hat{J}(x(t)) \le -\int_{t}^{t+\Delta T} \phi(\hat{x}(s),\hat{u}(s))ds, \tag{29}$$

where $\hat{J}(\hat{x}(t+\Delta T))$ is the optimal cost obtained when the state estimate at $t+\Delta T$ is $\hat{x}(t+\Delta T)$. This means that the optimal cost is guaranteed to decrease if the actual state follows the optimal state trajectory $x(s) = \hat{x}(s)$ for $s \in [t, t+\Delta T]$. Therefore, from Equation (29), the following result is obtained:

$$\begin{aligned}\hat{J}(\hat{x}(t+\Delta T)) - \hat{J}(x(t)) \le{}& \hat{J}(x(t+\Delta T)) \\ &- \hat{J}(\hat{x}(t+\Delta T)) - \int_{t}^{t+\Delta T} \phi(\hat{x}(s),\hat{a}(s))ds,\end{aligned} \tag{30}$$

where $\phi(\hat{x}(s),\hat{u}(s))$, as defined in Equation (16), is only known at the instant the solution of Equation (21) is obtained. Once Equation (21) is solved, the control action $\varsigma(t) \triangleq (\upsilon(t),\zeta(t))$ is obtained and will be applied in the next time slot $t+1$. Then, a system state vector, $\hat{x}(t) = (c(t),E(t))$, that contains the available number of containers $c(t)$ and the current energy consumption $E(t)$, determines the decision vector $\varsigma(t) \triangleq (\upsilon(t),\zeta(t))$—as shown in Figure 3. This guarantees system stability in terms of the worst-case queue length and server response times. Hence, the estimates of the response time and energy consumption can be used to estimate the server processing rate for the next time slot.

The adaptive controller explores the prediction horizon defined by T which comprises discrete states. The server response time for workload arriving during the interval $[t, t+1]$ can be estimated as follows:

$$\hat{\omega}(t+1) = (1+\hat{q}(t+1))\cdot\hat{\tau}(t+1), \tag{31}$$

where $\hat{q}(t+1)$ is the estimated queue length given as follows:

$$\hat{q}(t+1) = q(t) + \left(\hat{\lambda}(t+1) - \frac{\beta(t+1)}{\hat{\tau}(t+1)}\right), \tag{32}$$

where $\hat{\tau}(t+1)$ is the estimated processing time given as follows:

$$\hat{\tau}(t+1) \triangleq \gamma\cdot\tau(t) + (1-\gamma)\cdot\hat{\tau}(t), \tag{33}$$

where $\tau(t)$ is the actual processing time, and γ is a smoothing constant. Therefore, for an estimated processing rate of $f_{n,c}(t+1)$, the estimated energy consumed by the processor can be given as follows:

$$\hat{E}(t+1) \triangleq \beta^2(t+1), \quad \text{where} \quad \beta(t+1) = \frac{f_{n,c}(t+1)}{f_{\max}}, \tag{34}$$

which is the energy drained based on the pressure of the task completion time at the MEC server.

In describing the algorithm, we recall that the model described in Figure 1 brings together an orchestration of complex cyber-physical components for future mIoT. The combination of these aspects in a distributed edge computing environment gives rise to the need for live migration of multiple computational resources in parallel. This requires an intelligent and dynamic algorithm to perform smart allocation of resources on demand. Migrating containers from the controller to the different edge sites is shown in Figure 4.

In each time slot t, when the controller C receives workload information from the edge sites, it goes through these processes:

- Populate system state information and initialise parameters. The current system state consists of the initial estimate $\hat{x}(t)$ and the container arrival rate $\lambda_c(t)$.
- Receive load-based information from the edge servers.
- Obtain the input vector that will drive the behaviour of the edge server in the auto-scaling and reconfiguration of containers.
- Launch container images, on-demand, as specific functions to execute the specific tasks at the edge sites.
- Recall container images from the respective edge sites to the controller.

The procedure for the proposed algorithm is outlined in Algorithm 1.

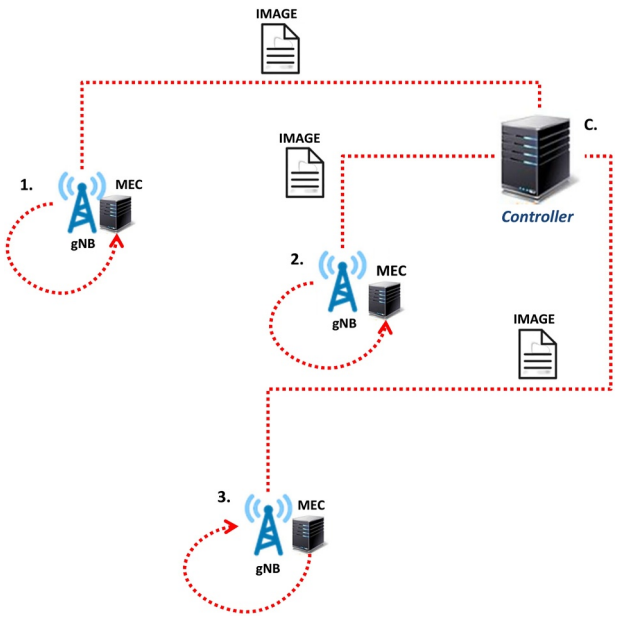

FIGURE 4 Container image migration from controller to edge site.

Algorithm 1 Proposed edge system management algorithm.

Input: Initial estimate, $\hat{x}(t)$; Task arrival rate, $\lambda_c(t)$;
Initial state, $s(t)$.
Output: Control input vector, ϕ^*
For each t in observation horizon of depth T **do**
  Initialise next state as $s(t+1) = \varnothing$
  Use $\lambda_c(t)$ to forecast task arrival rate $\lambda_c(\hat{t+1})$
  For each estimate $\hat{x}(t) \in s(t)$ **do**
    Generate all valid states such that $s(t+1) =$
    $s(t+1) \cup \{\hat{x}(t+1)\}$
  End For

6 | EXPERIMENTAL SETUP AND EVALUATION

The geographical space in the 6G-inspired mIoT network setup has several mIoT devices randomly dispersed to share network resources. A maximum system bandwidth of 20 MHz is considered. The transmission power range was set as $[4, 6]$ for a maximum transmission rate $r_{\max} = 1$ Gpbs. A delay upper-bound of 40 milliseconds was set as the maximum tolerance on latency. Also, a maximum of 10 containers was assumed, with each container assigned to process a maximum of 4000 MB of CPU load per time slot. Each edge site has a maximum of $K = 50$ offloading IoT devices, and the edge sites are set to be 250 m apart and each site has a virtualised MEC server with the specifications for a VMware x86 server H262 3rd Gen CPU, set to operate at maximum CPU operating frequency $f_{\max}$ is 1.6 GHz. The parameters used are selected to be able to favourably compare results obtained in this work with similar results from the baseline algorithm presented in ref. [14], and with other existing models, such as the one in ref. [13].

The achievable offloading rate based on the task admission probability is shown in Figure 5, while the average overhead for the network is shown in Figure 6.

The proposed algorithm, in Figure 5, achieves a better average task offloading rate and fewer overheads with respect to the number of offloading devices than the baseline shown in Figure 6. As the number of offloading devices increases, the competition for admission increases, which also increases the probability of unsuccessful task offloading. However, the learning capability of the proposed algorithm improves the task admission probability, resulting in a better average offloading bit rate and lower overhead than the baseline.

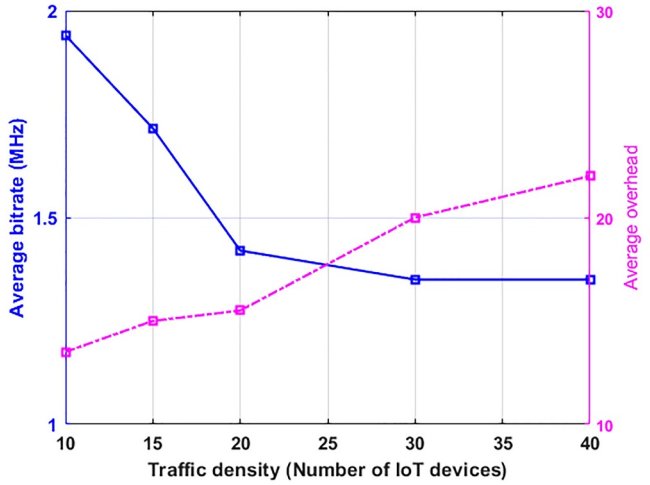

FIGURE 5 The average offloading rate and the average overhead plotted against the number of devices for the proposed algorithm.

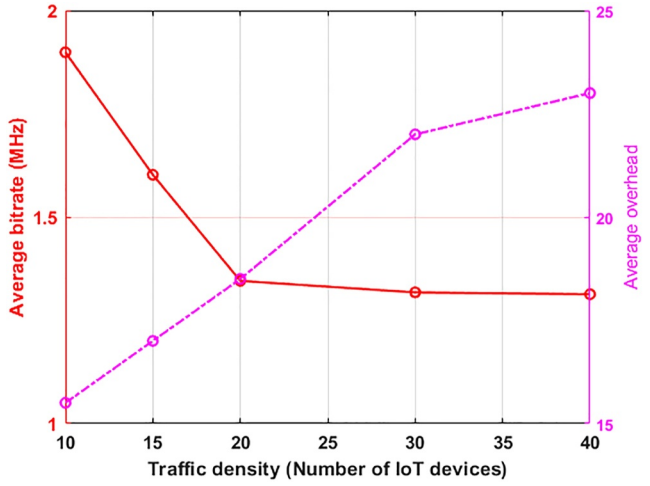

FIGURE 6 The average offloading rate and the average overhead plotted against the number of devices for the baseline algorithm.

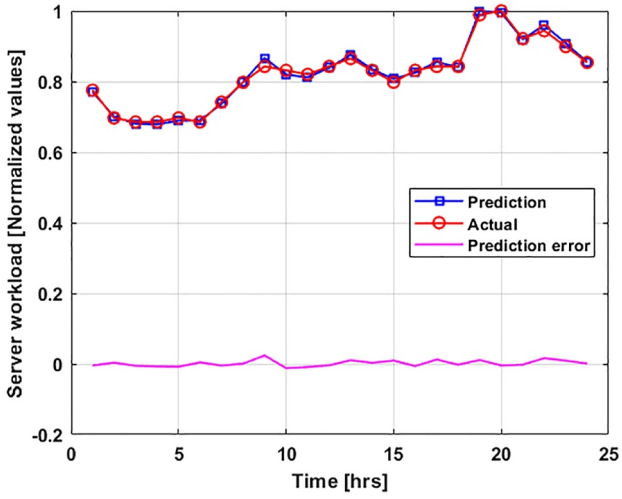

FIGURE 7 Normalised multiaccess edge computing server workload for a period of 24 h.

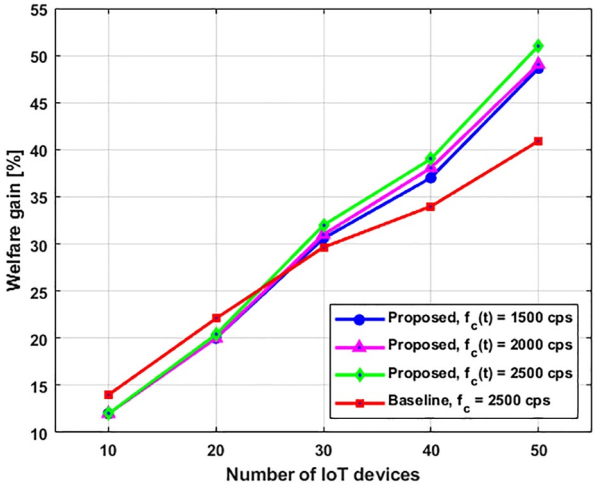

FIGURE 8 Welfare gain with the number of users.

The workload prediction at the MEC is carried out using a belief propagation neural network (BPNN) [30]. The prediction results are shown in Figure 7. The agent used to train the BPNN is the Levenberg-Marquardt algorithm [31]. The root-mean-squared error [32] is used to evaluate the training performance.

The BPNN is employed to estimate traffic patterns to predict server workload, and it has been trained using a time series data set with hourly granularity. From the training process, a 97.08% evaluation accuracy was achieved. From the plot in Figure 7, the prediction error, that is, the difference between the actual and predicted workload, is insignificant and is better than results obtained from using other AI techniques such as the artificial neural network. The prediction accuracy of 97.08% achieved by using the BPNN is deemed good enough for the modelled network in consideration.

The effect of an increasing number of users on the welfare gain is presented in Figure 8. From the plot, an almost consistent improvement in the welfare gain can be observed. The baseline algorithm has better gain compared to the proposed algorithm welfare gain when $0 \geq k \geq 25$, but at $k > 25$ the proposed algorithm outperforms the baseline. This is because when the number of users increases, the combinatorial nature of the algorithm takes significance, and the service is improved, hence the welfare of the users improves.

The effect of an increasing number of users on the server response time and the welfare gain is evaluated in Figure 9. The server response time is defined as the amount of time between when the user makes a request and when the server responds to the request. A better server response rate means a better user experience. In Figure 9, the server response rate is observed to decrease, while the welfare gain increases as the number of users increases. This is because when the number of users increases, the demand for computational resources also increases, which causes the server response time to increase. This can be interpreted as a future proactive point for upgrading the server or performing load balancing.

The effect of an increasing number of users on energy consumption is evaluated in Figure 10. The plot shows an

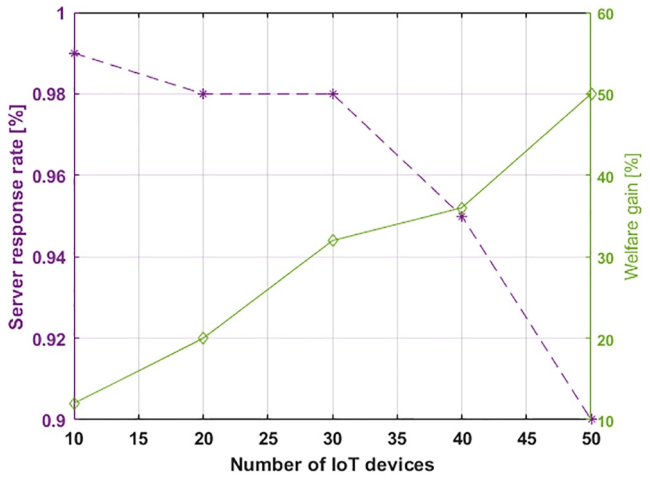

FIGURE 9 Evaluation of server response time and welfare gain with the number of users.

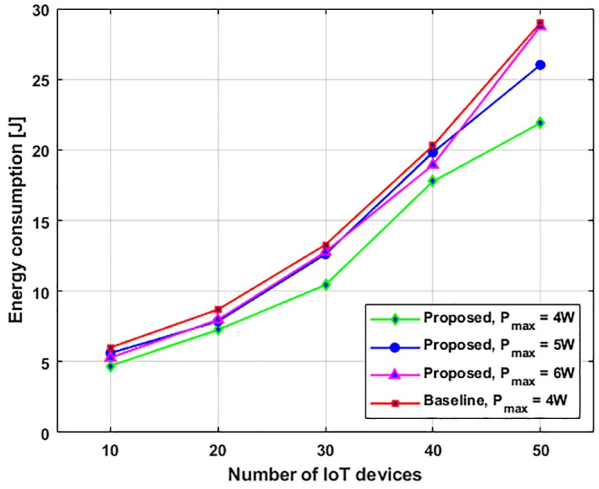

FIGURE 10 Evaluation of energy consumption with the number of users.

almost linear increase in energy consumption as the number of users increases. The energy consumption of the proposed algorithm outperforms the baseline at all the stages of maximum transmission power. This is because the increase in the number of users increases the number of tasks offloaded to the MEC server and this is correlated with high energy usage. As the computational resources are used more heavily, significant amounts of power are required to deliver service to the increased number of users.

7 | CONCLUSION

The paper developed and investigated an adaptive power management scheme that exploits the correlation between workload and energy consumption to monitor the operation of the edge platform in a 6G-inspired mIoT network setting. The containerised edge computing platform is employed for achieving energy efficiency in the 6G-inspired mIoT network. The optimisation problem, formulated as a CPU energy consumption cost function based on quasi-finite system observations, is solved using a search heuristic and Lyapunov technique. From the network analysis and results, it is shown that by incorporating appropriate edge computing techniques, the network can achieve better response times, thereby enhancing user QoE through the computational resource-enriched VMs employed in the network design. Overall, the containerised edge computing platforms utilised in the network significantly minimise the high energy consumption caused by high overheads in the 6G-inspired mIoT network, which can be applied to similar xG networks. The proposed scheme showed better performance than a baseline algorithm in terms of welfare gain, server response rate, and energy consumption. Future works will include developing appropriate digital twin models and solutions for MEC-enabled mIoT and other xG networks.

AUTHOR CONTRIBUTIONS

Babatunde S. Awoyemi: Conceptualization; formal analysis; investigation; methodology; project administration; resources; validation; writing—original draft; writing—review and editing.
Bodhaswar T. Maharaj: Funding acquisition; investigation; project administration; resources; supervision; validation; writing—review and editing.

ACKNOWLEDGEMENTS

The research was financially supported with funds through the SENTECH Chair in Broadband Wireless Multimedia Communications (BWMC) at the University of Pretoria. The authors would like to appreciate our colleague, MC Hlophe for his help with this work.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

ORCID

Babatunde S. Awoyemi https://orcid.org/0000-0003-2136-1318

REFERENCES

1. Awoyemi, B.S., Maharaj, B.T., Alfa, A.S.: Resource allocation in heterogeneous buffered cognitive radio networks. Wireless Commun. Mobile Comput. 2017, 7385627 (2017). https://doi.org/10.1155/2017/7385627
2. Maharaj, B.T., Awoyemi, B.S.: Queuing systems in resource allocation optimisation for cognitive radio networks. In: Developments in Cognitive Radio Networks, pp. 121–139. Springer (2022)
3. Zeng, D., et al.: Convergence of edge computing and next generation networking. Peer-to-Peer Netw. Appl. 14(6), 3891–3894 (2021). https://doi.org/10.1007/s12083-021-01239-7
4. Seisa, A.S., et al.: An edge-based architecture for offloading model predictive control for UAVs. Robotics 11(4), 80 (2022). https://doi.org/10.3390/robotics11040080

5. Hasanin, T., et al.: Efficient multiuser computation for mobile-edge computing in IoT application using optimization algorithm. Appl. Bionics Biomech. 2021, 1–12 (2021). https://doi.org/10.1155/2021/9014559
6. Chen, F., et al.: Energy efficient SWIPT based mobile edge computing framework for WSN-assisted IoT. Sensors 21(14), 4798 (2021). https://doi.org/10.3390/s21144798
7. Mitsis, G., Tsiropoulou, E.E., Papavassiliou, S.: Price and risk awareness for data offloading decision-making in edge computing systems. IEEE Syst. J. 16(4), 6546–6557 (2022). https://doi.org/10.1109/jsyst.2022.3188997
8. Li, X., et al.: Edge computing-enabled wireless sensor networks for multiple data collection tasks in smart agriculture. J. Sens. 2020, 1–9 (2020). https://doi.org/10.1155/2020/4398061
9. Gundall, M., et al.: Towards organic 6G networks: virtualization and live migration of core network functions. In: Mobile Communication – Technologies and Applications; 25th ITG-Symposium, pp. 1–6. VDE, Osnabrück, Germany (2021)
10. Zhao, L., et al.: Open-source multi-access edge computing for 6G: opportunities and challenges. IEEE Access 9, 158426–158439 (2021). https://doi.org/10.1109/access.2021.3130418
11. Yue, Y., et al.: Virtual network function migration considering load balance and SFC delay in 6G mobile edge computing networks. Electronics 12(12), 2753 (2023). https://www.mdpi.com/2079-9292/12/12/2753
12. Liu, A., et al.: Task containerization and container placement optimization for MEC: a joint communication and computing perspective. Processes 11(5), 1560 (2023). https://www.mdpi.com/2227-9717/11/5/1560
13. Hlophe, M., Maharaj, B.: Prospect-theoretic DRL approach for container provisioning in energy-constrained edge platforms. In: 2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring), pp. 1–5. IEEE, Florence, Italy (2023)
14. Cong, R., et al.: EdgeGO: A mobile resource-sharing framework for 6G edge computing in massive IoT systems. IEEE Internet Things J. 9(16), 14521–14529 (2022). https://doi.org/10.1109/jiot.2021.3065357
15. Kekki, S., et al.: MEC in 5G networks. ETSI White Paper 28(28), 1–28 (2018)
16. Hlophe, M.C., Maharaj, B.T., Sande, M.M.: Energy-efficient transmissions in federated learning-assisted cognitive radio networks. In: 2021 IEEE 21st International Conference on Communication Technology, pp. 216–222. ICCT, Tianjin, China (2021)
17. Dlamini, T., Vilakati, S.: LSTM-based traffic load balancing and resource allocation for an edge system. Wireless Commun. Mobile Comput. 2020, 1–15 (2020). https://doi.org/10.1155/2020/8825396
18. Rathore, M.S., et al.: In the direction of service guarantees for virtualized network functions. Wireless Commun. Mobile Comput. 2022, 1–16 (2022). https://doi.org/10.1155/2022/5507845
19. Luo, Y., Li, W., Qiu, S.: Anomaly detection based latency-aware energy consumption optimization for IoT data-flow services. Sensors 20(1), 122 (2019). https://doi.org/10.3390/s20010122
20. Metcalf, L., Casey, W.: Chapter 3 – probability models. In: Metcalf, L., Casey, W. (eds.) Cybersecurity and Applied Mathematics, pp. 23–42. Syngress, Boston (2016). https://www.sciencedirect.com/science/article/pii/B9780128044520000038
21. Gustafson, J.L.: Little's Law. In: Padua, D. (ed.) Encyclopedia of Parallel Computing, pp. 1038–1041. Springer US, Boston, MA (2011). https://doi.org/10.1007/978-0-387-09766-4_79
22. Mir, M.Y., Huang, S.-Z.: Data forwarding with finite buffer capacity in opportunistic networks. In: 2018 27th Wireless and Optical Communication Conference (WOCC), pp. 1–5. IEEE, Hualien, Taiwan (2018)
23. Ding, X., et al.: Optimal control of Markov decision processes with linear temporal logic constraints. IEEE Trans. Automat. Control 59(5), 1244–1257 (2014). https://doi.org/10.1109/tac.2014.2298143
24. Rogers, E., Galkowski, K., Owens, D.H.: Control Systems Theory and Applications for Linear Repetitive Processes, vol. 349. Springer Science & Business Media (2007)
25. Mastronarde, N., van der Schaar, M.: Joint physical-layer and system-level power management for delay-sensitive wireless communications. IEEE Trans. Mobile Comput. 12(4), 694–709 (2012). https://doi.org/10.1109/tmc.2012.36
26. Qu, Y., Ng, B., Homer, M.: A goodput distribution model for planning IEEE 802.11 WBNs in built environments. J. Netw. Comput. Appl. 99, 28–46 (2017). https://www.sciencedirect.com/science/article/pii/S1084804517303181
27. Hlophe, M.C., Maharaj, B.T.: QoS provisioning and energy saving scheme for distributed cognitive radio networks using deep learning. J. Commun. Network. 22(3), 185–204 (2020). https://doi.org/10.1109/jcn.2020.000013
28. Chen, H., Allgöwer, F.: A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. Automatica 34(10), 1205–1217 (1998). https://doi.org/10.1016/s0005-1098(98)00073-9
29. Hamidi, F., Abdelkrim, M.N., Houssem, J.: Searching candidate Lyapunov function with threshold accepting algorithm. In: 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks, pp. 26–31. IEEE, Bali, Indonesia, (2011)
30. Kuck, J., et al.: Belief propagation neural networks. Adv. Neural Inf. Process. Syst. 33, 667–678 (2020)
31. Huang, X., Cao, H., Jia, B.: Optimization of Levenberg Marquardt algorithm applied to nonlinear systems. Processes 11(6), 1794 (2023). https://www.mdpi.com/2227-9717/11/6/1794
32. Awoyemi, B., Walingo, T., Takawira, F.: Predictive relay-selection cooperative diversity in land mobile satellite systems. Int. J. Satellite Commun. Netw. 34(2), 277–294 (2016). https://doi.org/10.1002/sat.1118

How to cite this article: Awoyemi, B.S., Maharaj, B.T.: Adaptive power management for multiaccess edge computing-based 6G-inspired massive Internet of Things. IET Wirel. Sens. Syst. e70000 (2025). https://doi.org/10.1049/wss2.70000