EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

# Dynamic resource provisioning in containerized edge systems with reconfigurable edge servers

Babatunde S. Awoyemi[1*], Mduduzi C. Hlophe[1] and Bodhaswar T. Maharaj[1]

*Correspondence:
awoyemibabatunde@gmail.com

[1] Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa

## Abstract

Recent technological advancements have seen powerful computational resource-enriched virtual machines (VMs) being used for processing data in edge servers. However, the high energy demands and excessive overhead associated with launching VMs are major obstacles to achieving energy-efficient operations in multi-access edge computing environments. As a result, there has been a relentless acceleration toward container virtualization to provide containerized services at the edge. The lightweight nature of containers compared to VMs makes them a popular technology for edge computing platforms. However, two significant challenges have been identified. The first is the problem of providing real-time support for containerized edge systems (to combat issues of high latency, anomaly detection, and automated monitoring and control, among others). The other problem is that, although containers help reduce application deployment time, considerable network bandwidth is expended and longer download queues are experienced on each node in the network. We propose a dynamic resource provisioning scheme for containerized edge systems to address these challenges. The proposed scheme employs containerized reconfigurable edge servers, which enable computational task operations to be moved to the data source for easier and quicker completion. Then, a novel adaptive power management technique based on predictive control through finite system observations is used to effectively estimate and regulate the energy consumption in the edge-based network. The adaptive controller schedules computational resources on a time slot basis in an adaptive manner, while continuing to receive updates to plan future resource provisioning. The proposed technique is evaluated using welfare gain, server response rate, and energy consumption metrics and is shown to outperform recent comparative models significantly.

**Keywords:** 6G, Adaptive controller, Containers, Edge computing, Massive IoT, MEC, Power management

## 1 Introduction

Distributed computing, especially edge and mobile computing, has grown tremendously over the past few years and is gaining momentum in achieving efficiency—enough to achieve economic competitiveness in several applications. With 6G networks, the

promise of quasi-infinite networking has arrived, with both arguments on coverage and the perspective of computational effectiveness and efficiency. As infrastructure vendors offer more computational resources with computational capabilities exceeding 2 GHz, and containerized edge servers progressively deployed, the shift to quasi-infinite networking is progressively taking place. However, deploying edge computing infrastructure requires careful placement of edge servers to improve application latencies and reduce data transfer load for opportunistic and mission-critical systems [1].

Control problem specifications from newer applications and application areas have opened new research directions, such as developing newer algorithms for faster online computations in distributed and stochastic situations. For instance, in most upcoming 6G applications, data transmission always has delay-sensitive connotations attached to them [2]. Delay-sensitive data require processing with minimal latency, which is why edge computing is being advocated over central cloud computing in progressive network design. Two main points prompted network designers to consider moving computation away from the core of the network to the edge. The first is the latency problem caused by high transmission delays. The second is the congestion buildup at the backbone of networks [3]. In response, multi-access edge computing (MEC) has become a much-welcome solution for migrating some of the network functions from the core to the edge. Bringing processing closer to the edge of the network, near the data sources, is a paradigm shift that opened the door to a host of opportunities toward improving network quality of service (QoS), thereby improving the welfare of end users [4].

One of the most important and striking applications of MEC in emerging communications is in large-scale or massive Internet-of-Things (mIoT) networks, where, without a doubt, it has been shown that several aspects of mIoT network can be improved by MEC. For instance, through edge computing, the joint distribution of both communication and computational resources can be used to optimize different aspects of 6G networks, such as reliability and efficiency. Also, proper optimization of edge computation processes would help in improving the decision-making processes of the network in terms of taking appropriate actions. Furthermore, delay-intensive applications, such as vehicular communication, whose QoS requirements cannot be guaranteed by the current 5G networks will be accommodated. In vehicular communication, high computational capacity is crucial, especially when highly computationally demanding high-level algorithms are employed [5]. Such algorithmic requirements are indispensable in MEC-enabled real-time autonomous missions and applications, such as autonomous driving and mobile robots.

Due to the requirement of high transmission rates in emerging networks such as the 6G-based mIoT network being considered, the current status of the 5G networks poses a significant performance bottleneck. The main roadblock is the huge energy demand needed to support the network in achieving and maintaining high transmission rates. The energy demand is even more exacerbated in such emerging networks when edge computing capabilities are incorporated. This makes it imperative to develop appropriate power management models to solve the energy consumption problems in beyond 5G (B5G) networks. This is the main motivation of this research work. The solution investigated in this paper views the energy problem as an adaptive power management problem, using the 6G-inspired MEC-based mIoT network as base application. Further,

the optimization problem is approached as a problem observed within a finite horizon of time. In theory, for every finite horizon observation in wireless networking, certain aspects are usually infinite. To handle such aspects, in this paper, a quasi-finite approach is adopted in the optimization process and the problem is solved using appropriate tools to achieve the desired accuracy and latency demands.

### 1.1 Review of related literature

In edge computing, the MEC server, with its high-intensive computing capabilities, is the custodian of bringing novel computing resources for compute-intensive applications. As a result, the dramatic $< 3$ ms end-to-end latency reduction for the emerging 6G applications can be achieved. The high computational capability of the MEC servers in terms of massive throughput and low-latency processing of offloaded computational tasks is a huge advantage for mIoT applications. However, these are extremely impressive computing speeds in terms of ultra-low latency requirements for 6G applications, and high energy consumption is always a direct consequence. Since high energy consumption can become a huge bottleneck for emerging future technologies, efforts need to be made to find energy-efficient solutions tailored for edge computing environments. This section reviews some of the most recent works that have been carried out in this regard.

A careful study of recent contributions in this field indicates that a plethora of candidate solutions are inspired by artificial intelligence (AI). AI strategies, through their exceptional learning and prediction capabilities, are one of the most potent tools employed in achieving the desired performance expectations. With its promise of dramatically reducing latency and energy consumption, even more ultra-low latency use cases have been encouraged. The authors in [6] proposed a bald eagle search (BES) algorithm to reduce the high computational complexity of the currently used deep learning approaches. An extension of their objective was also to shorten the latency as well as minimize the amount of energy being consumed in an MEC-enabled IoT network. The decision-making process of state-of-the-art BES algorithms employed three stages, namely (i) the selection stage, (ii) the searching stage, and (iii) the swooping stage. To improve their performance and achieve better performance than previous BES algorithms, the authors 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, minimizing the expected execution time. Based on multi-user offloading, the authors then proposed a BES optimization algorithm which they claimed effectively reduced the end-to-end latency, thereby achieving near-optimal solutions for the IoT devices in the network.

Service levels in edge computing can be improved by adopting the best strategies with the best responses to offloading requests and task execution. To improve the service supply capability of wireless sensor network-assisted IoT applications, the authors in [7] proposed a system that integrates MEC technology, simultaneous wireless information, and power transfer technology. In this integration, a novel optimization problem was formulated to minimize the total system energy consumption under the constraints of data transmission rate and transmitting power requirements. During the optimization process, the optimization problem was designed to jointly consider 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 optimization algorithm was proposed to decompose the original problem into three sub-problems. Each sub-problem was alternatively optimized using the group interior point iterative algorithm, and the results obtained indicated lower energy consumption when compared to the benchmark algorithms.

It is believed that the effectiveness and efficiency of edge computing are strongly correlated to features of user behavior. To ensure that MEC infrastructure provides users with the required low latency computations, user behavior must be taken into account [8]. In wireless networks, and edge computing in particular, the dynamics and variety of user behaviors have the largest influence on decision-making in terms of the provisioning of infrastructure resources [9]. Therefore, user behavior analysis is most desirable in improving the efficiency and effectiveness of the Edge-IoT continuum. The authors in [10] studied the impact of the behavioral 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. The probability of failure of the MEC servers to potential over-exploitation was modeled via the theory of the 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 them and a Nash equilibrium was determined. The evaluation results demonstrated the superiority of the proposed framework against other benchmark alternatives.

Empowering data analytics at the edge of an IoT network can help create a cloud-ready IoT edge gateway for different IoT applications. In advancing this initiative, the authors in [11] proposed an efficient solution for multiple data collection tasks in an MEC-enabled wireless sensor network (WSN) environment for smart agriculture applications. In the model setup, the edge servers were deployed around the WSN nodes to provide better computing services for the WSN, which was developed to complete different tasks 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 done by dynamically configuring the sensor nodes to accomplish the set tasks while keeping within a set time frame. By employing classical optimization tools, the developed model was shown to process a higher volume of valid data at a much lower data collection time.

Due to the increasing importance of optimization and AI as potent tools for driving edge computing, software-level virtualization is also gaining attention as a modern computational tool to achieve desired results. However, the capabilities of software-level virtualization (containerization, for example) are yet to be extensively explored, particularly for 6G-inspired edge computing solutions [12]. To this effect, the authors in [13] explored container virtualization technology as a promising tool for achieving the kind of flexibility required in operations, service demands, and resource management in emerging networks. The virtualization made it possible for mobile devices with computational resource capabilities and sufficient storage space to automatically deploy and run virtual network functions (VNFs) to achieve high-level packet

delivery expectations. Furthermore, the authors made several recommendations on some existing technologies that can be suitable for the application of NFV in 5G core network functions for different industrial use cases and industry verticals.

The mobility of edge devices, especially in vehicular communications, forces them to migrate between edge servers to maintain quality of experience (QoE) in terms of end-to-end latency [14]. However, the most challenging task in resource migration is deciding when to migrate a certain service, as well as the cost of that migration. Migrating a service requires a cost and QoE trade-off, while the selection of the destination edge server needs to be done considering latency and resource availability constraints to help minimize the number of migrations within the network. To help in migration decisions, the authors in [15] proposed a virtualized open-source MEC (OS-MEC) scheme for application in B5G networks. The scheme was built on the key principles of network decoupling and reconfiguration, whereby in achieving the decoupling, a service-based MEC layer was developed for the OS-MEC scheme. This layer was able to decouple the tightly coupled service functions into multiple independent network functions (NFs). The authors then employed the concepts of the templates and instances to reassemble the disaggregated NFs and to reallocate the necessary resources to provide customized services for users. Some use cases were presented to validate the flexibility and customization of the test network. In another contribution, the authors in [16] studied VNF migration and service function chaining (SFC) reconfiguration problems in dynamic NFV-enabled 6G networks. SFC, also known as network service chaining, uses software-defined networking (SDN) capabilities to create a chain of connected network services [17]. The most prevalent SFCs are L4-7 services, such as firewalls, network address translation, and intrusion protection. In their work in [16], the authors formulated a VNF migration problem as an optimization model to minimize the end-to-end delay of all influenced SFCs while guaranteeing network load balance after migration. Then, a deep learning (DL) two-stage algorithm was designed to solve the migration problem by combining previous experimental data, and realistic VNF traffic patterns were generated and used to evaluate the algorithm. The evaluation results indicate a more balanced network in terms of load distribution.

### 1.2 Research motivation

While performing optimal placement of edge servers, the most important aspects that need to be considered are as follows: (i) the required computing capacity, (ii) the server deployment budget, and (iii) the hardware requirements. Many contributions in this field aspect involve heuristics and meta-heuristics [18–20], where prioritized locations are considered. However, the reliability in terms of computational capacity constraints for load balancing and enabling workload sharing are weak considerations. Generally, in systems and networking, resource management problems often manifest as difficult online decision-making tasks where appropriate solutions depend on understanding the workload and environment. Because of the fast data exchange demands happening in edge computing environments, the behavior of an edge server can change rapidly within 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 needing more attention. Also, the aspect of energy or power management in edge computing is yet to be

adequately addressed, especially in 6G networks such as the 6G-inspired mIoT network being considered in this paper. With the advent of virtualization and the application of AI-driven solutions in edge computing, there exists a huge research gap in the area of resource migration and dynamic power management, which motivates this research work.

### 1.3 Summary of major contributions

Distributed computing is one of the main drivers of 6G networks, especially in relation to edge computing. With distributed computing, such as the containerized edge network in consideration in this work, system throughput can be improved by bringing computational resources close to the data source. Therefore, moving computational task operations to the data source rather than moving the data source to the point of computation form the basic motivation for this research. The contributions of this article are summarized as follows:

- *Distributed computing system and server placement:* This is a collection of multiple physically separated servers and data storage located in different locations, whose components collaborate to achieve the same objective. All the locations of this distributed system behave as independent servers whose computing services can easily be scaled based on computation demand at that particular location through automated processes. This means that the network does not require significant changes to be made to current edge computing-based infrastructure and, therefore, can be implemented with minimal disruption. Furthermore, this provides a unique objective function and compatibility with Markov decision processes (MDP) modeling in a multiuser, multiserver environment. Again, the resource provisioning controller and its interaction with the environment make it possible for optimal resource provisioning decisions to be achieved.
- *Predictive control for adaptive power management:* The proposed method advocates for a controlled resource allocation (RA) strategy at multiple edge sites in metropolitan areas to improve both latency and energy efficiency, making it possible for the system to benefit from real-time learning. So, in addition to being able to process heterogeneous tasks, there is an added capability of estimating the effects on energy consumption, which makes it unique. Here, the power management problem is formulated as a CPU energy consumption cost function, where finite system observations are applied. Then, one-step ahead predictive capabilities are introduced for each edge site, in which an adaptive power management technique is proposed to provide a cost-effective strategy capable of monitoring the critical parameters of each edge server. The developed model is scalable in that it builds on the well-established European Telecommunications Standards Institute (ETSI) standards for edge computing-based networks. In addition, the developed model is feasible even under fluctuating loads and across different networks due to the predictive capabilities of the solution model employed.

The remainder of this article is presented in sections. Section 2 discusses the proposed system model of an edge network in a busy metropolitan area, together with the

components of an intelligent edge system. Section 3 discusses the mathematical formulation of the problem. Section 4 discusses the online adaptive power management algorithm being proposed. Section 6 presents the evaluation results and discussions. Finally, Sect. 7 provides the concluding remarks.

A list of the abbreviations used in this article is provided in Table 1.

## 2 Methods and experimental

This work considers an edge computing scheme by which a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of edge sites is distributed according to a Poisson process in a busy metropolitan area. The channel condition is assumed to be Rayleigh flat fading channel. Each edge architecture consists of a configured virtual local area network (VLAN), where the gNB is connected to an edge server via high-speed fiber-based fronthaul connections using the eCPRI interface [21]. In this model, each edge site is co-located with a gNB which receives offloaded tasks from a set $\mathcal{K} = \{1, 2, \ldots, K\}$ of 6G-capable mIoT devices. A depiction of the model is shown in Fig. 1.

**Table 1** List of Acronyms

| Acronym | Description |
|---|---|
| 5G/6G | 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 |
| DNN | Deep Neural Network |
| ETSI | European Telecommunications Standards Institute |
| FCFS | First Come First Served |
| gNB | generation NodeB or just gNodeB |
| IoT/mIoT | Internet-of-Things/massive Internet-of-Things |
| MDP | Markov Decision Processes |
| MEC | Multi-access Edge Computing |
| MLP | Multi-layer Perceptron |
| NFV | Network Function Virtualization |
| NICs | Network Interface Controllers |
| OFDMA | Orthogonal Frequency Division Multiple Access |
| QoE | Quality of Experience |
| QoS | Quality of Service |
| RL | Reinforcement Learning |
| 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 | Virtualized Network Functions |
| WSN | Wireless Sensor Networks |

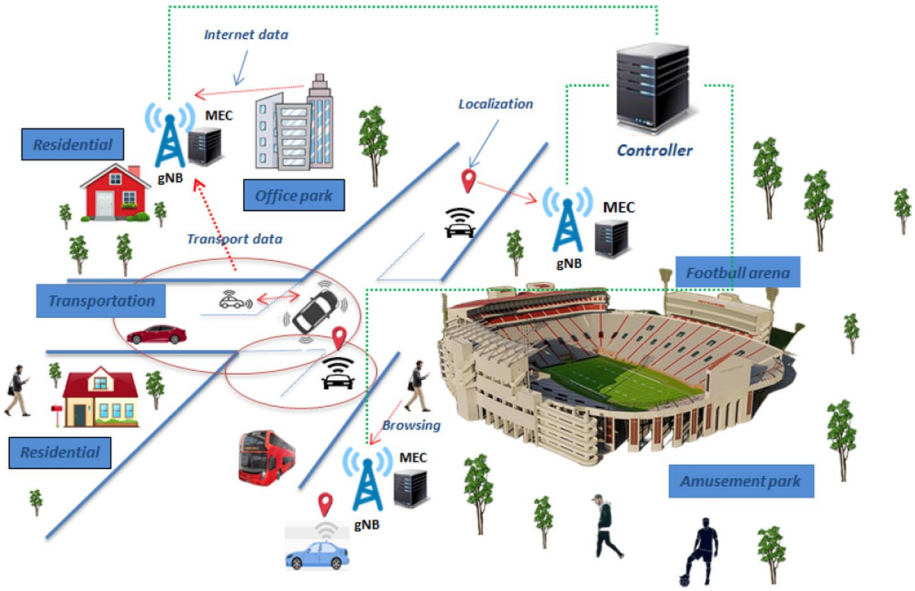


**Fig. 1** An MEC system with devices generating computational tasks with diverse delay sensitivities

As shown in Fig. 1, all the gNBs are connected to an offloading controller that employs an offloading policy to select the appropriate edge site for each task. It is assumed that each computational task generated by the mIoT devices is indivisible, meaning that it can either be processed by the device or completely offloaded to the edge server. A provisioning controller uses the information it collects from the gNBs to manage the computational resources of each edge server. Thus, influenced by the amount of computational power required to process the tasks, the provisioning controller may allocate more computational resources during event days at the football arena than at the office park. Also, since transportation data has high delay sensitivity, more resources might be provided to the gNBs next to the road during peak traffic times than in residential places. The role of the resource provisioning controller, which is hosted as an application, is the on-demand provisioning of computational resources to the different edge computing sites. Since the edge sites are not connected, collaboration between the sites is not assumed. This means that the system does not allow task migration among the edge sites after offloading

### 2.1 The mIoT devices

It is assumed that the mIoT devices that generate latency-critical data are: (i) mobile phones, which generate both generic and video content; (ii) vehicle location systems, such as advanced driver assistance systems (ADAS); and (iii) cooperative vehicular communication, whose details will not be elaborated in this work. Generally, mIoT devices generate different categories of computational tasks with different tolerances for task execution delays. For instance, a short delay in online shopping may not be a major concern for mobile phone users, but that delay can have catastrophic effects on autonomous driving. Thus, as much as each device wishes to maximize its processing capacity through task offloading, latency sensitivity is different for different types of applications. Therefore, the available battery power in each mIoT device is assigned to the radio

frequency (RF) modules for offloading computational tasks under interference and sub-carrier constraints.

### 2.2 The controller and the DNN agent

In general, container virtualization systems are more effective and straightforward to handle than traditional virtualization systems. In this proposed virtualization technique, it is assumed that a DNN agent that is hosted at the controller consists of four modules: (i) decision logic module, which is responsible for communicating with the controller decision logic and calling related components to complete the task according to its requirements; (ii) container migration module, which is responsible for completing the migration of containers; (iii) server power collection module, which is responsible for reading the real-time power consumption of the edge servers; and (iv) container power prediction module, which is responsible for predicting the power consumption of the containers in the server. When the decision logic receives the container power prediction signal from the controller, it uses the multi-layer perceptron (MLP) model to predict the container power in real time.

### 2.3 Reconfigurable edge computing platform

In line with the ETSI proposed MEC deployment scenarios [22], the network is made up of a reconfigurable platform and a related switched VLAN with network interface controllers (NICs) connected to it. It is assumed that each edge server has limited capacity, and it can only store a maximum of $Q_{\text{in}}$ packets in its queue [23], and execute them according to a first-come-first-served (FCFS) scheduling policy before sending the results back to the IoT devices. The dynamic VM manager and task scheduler are used to represent the reconfigurable edge server as shown in Fig. 2.

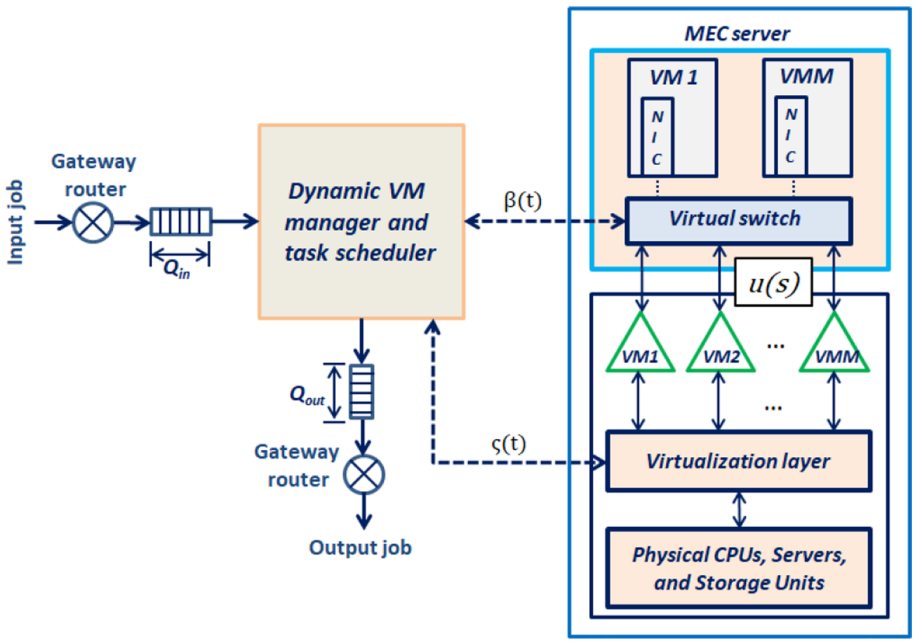


**Fig. 2** The reconfigurable edge computing platform

The edge platform shown in Fig. 2 consists of an input queue, which is an arrival queue of size $Q_{\text{in}}$; an output queue, which is a return queue of size $Q_{\text{out}}$; a dynamic virtual machine (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 virtual network functions (VNFs) [24]. Control actions $\varsigma(t)$ determine the required processing frequency $f^{\text{mec}}_{n,\text{cpu}}$, which in turn is determined through the virtualized switch by assigning a load-dependent factor $\beta_c(t)$ into the process.

## 3 Mathematical problem formulation

In terms of local computation, the local CPU is characterized by a computational frequency $f^{\eta}_{k,\text{cpu}}$, 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 $\psi_k > 0$ [25]. It is assumed that the $k$-th device generates computational tasks at the application layer, and they arrive at the network layer according to a Poisson distribution with arrival rate $\lambda_k(t)$ [26]. 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 is modeled as follows:

$$p^{\text{local}}_{k,\text{cpu}}(t) = f^{\eta}_{k,\text{cpu}} \psi_k. \tag{1}$$

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

$$\tau^{\text{local}}_{k,\text{cpu}} = \frac{b_k c_k}{f_{k,\text{cpu}}}, \tag{2}$$

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 [27], the energy consumption for local processing is expressed as follows:

$$e^{\text{local}}_{k,\text{cpu}} = \kappa \cdot f^{3}_{k,\text{cpu}} \cdot \tau^{\text{local}}_{k,\text{cpu}}, \tag{3}$$

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

### 3.1 Computational task offloading

If the $k$-th 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 $n$-th gNB adopts an orthogonal frequency division multiple access (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 $k$-th mIoT device and the $n$-th gNB can be represented as follows:

$$r_{n,k}(t) = B \log_2 \left(1 + \frac{p_{n,k}(t) g_{n,k}(t)}{\mathcal{I}_{k'}(t) + N_0}\right), \tag{4}$$

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 $k$-th device given as follows:

$$\mathcal{I}_{k'}(t) = \sum_{k' \in \mathcal{K} \setminus \{k\}} g_k(t) p_{n,k'}(t). \tag{5}$$

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 (4) and the task size $d_k$ can be used to define the task offloading latency as follows:

$$\tau_{k,\text{mec}}^{\text{comm}} = \frac{1}{\frac{r_{k,n}(t)}{d_k} - \lambda_k(t)}, \tag{6}$$

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. 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:

$$e_{n,k}(t) = p_{n,k}(t) \cdot \tau_{k,\text{mec}}^{\text{comm}}, \tag{7}$$

where $p_{n,k}(t)$ is the transmission power defined in (4).

In edge computing systems, the time delay is affected by the load. Equation (6) agrees to flow theory and its probability distribution follows the Pareto model [28]. 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 queuing delay can be derived based on Little's Law [29], the task computation delay can be expressed as follows:

$$\tau_{n,\text{mec}}^{\text{comp}} = \frac{1}{\vartheta_n} \left( \frac{1}{(f_{n,\text{cpu}}^{\text{mec}} - r_n(\mathbf{n}, \mathbf{p}, t))} \right) \tag{8}$$

where $\vartheta_n$ 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:

$$p_{n,\text{cpu}}^{\text{mec}}(t) = f_{n,\text{cpu}}^{\text{mec}} \cdot \psi_{n,\text{cpu}}, \tag{9}$$

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

$$e_{n,\text{cpu}}^{\text{mec}} = \kappa \cdot f_{n,\text{cpu}}^{3} \cdot \tau_{n,\text{cpu}}^{\text{mec}}, \tag{10}$$

where $\kappa$ is as defined in (3).

### 3.2 Formulating the constrained problem

To ensure system stability, the stability of each queue must be monitored using the queue state, which the scheduling controller monitors. 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:

$$q_n(t+1) = \max\{q_n(t) + r_n(\mathbf{n}, \mathbf{p}, t) - f^{\text{mec}}_{n,\text{cpu}}(t+1), Q_{\text{in}}\}, \tag{11}$$

where $q_n(t+1)$ is the queue length at the upcoming slot, $q_n(t)$ is the current queue length, $f^{\text{mec}}_{n,\text{cpu}} \in [f^{\text{mec}}_{\min}, f^{\text{mec}}_{\max}]$ is the server's computation capability (in CPU cycles/sec), and $Q_{\text{in}}$ is the length of the incoming queue as shown in Fig. 2 [30]. 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:

$$\tau^{\text{total}}_{\text{delay}} = \left\{ \tau^{\text{local}}_{k,\text{cpu}} + \tau^{\text{comm}}_{k,\text{mec}} + \tau^{\text{comp}}_{n,\text{mec}} \right\} \leq \tau^*, \tag{12}$$

where $\tau^*$ is the maximum tolerable latency for result return.

### 3.3 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 (MDPs) [31], a constrained Markov decision process (CMDP) is adopted for the adaptive power management for the quasi-finite horizon edge platform.

#### *3.3.1 The state space*

The CMDP adopted for use in this work is enhanced by incorporating reinforcement learning (RL) to improve the network's predictive capabilities, help the network adapt to potential changes, and to work toward achieving real-time decision-making. Contrary to traditional RL strategies, where the state space is explicitly defined based on the set of possible states an agent can be in, RL safety is a major concern in constrained optimization. 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:

$$s(t) \triangleq \{r_{n,k}(\mathbf{p}, \mathbf{n}, t), q(t), c(t)\}, \quad \forall t \in T \tag{13}$$

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.

#### *3.3.2 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 behavior of the processor [32]. Therefore, the action space can be represented as follows:

$$a(t) \triangleq \{\upsilon(t)\}, \quad \forall s(t) \in \mathcal{S} \tag{14}$$

which, with reference to Fig. 2, 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:

$$s(t+1) \triangleq \phi(s(t), \upsilon(t)), \tag{15}$$

where the parameter $\phi(\cdot)$ represents the control input vector that captures the behavior 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 behavior 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 [33]:

$$p([q(t), c(t)], \upsilon(t)) = \begin{cases} p_{\text{proc}}, & \text{if } \hat{x} = 1, \upsilon = 1 \\ p_{\text{idle}}, & \text{if } \hat{x} = 0, \upsilon = 0 \end{cases}, \tag{16}$$

where, as previously defined, $\upsilon$ 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 behavior is $\hat{x} = 1$, the edge server is in active mode and is actively processing offloaded tasks, and according to (16) $\upsilon = 1$.

For the server to positively contribute to the QoS, it must prevent packet losses (that is, it must minimize queuing 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:

$$\begin{aligned} \tilde{q}(s(t), \upsilon(t)) = & \sum_{\lambda=0}^{\infty} \sum_{f=0}^{z} p^{r}(r) p^{\zeta^*}(\zeta^* | s, r) \\ & \left\{ \hat{\zeta} + \varsigma^* \max(\hat{\zeta} + r - Q_{\text{in}}, 0) \right\}. \end{aligned} \tag{17}$$

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 $p^{\zeta^*}(\zeta^* | \cdot, \cdot)$ [34], $\hat{\zeta} = [q - \zeta^*]$ is the holding cost with respect to task arrival rate and processing rate, and

$Q_{\text{in}}$ is the length of the input queue. Based on (16), the reward of the system is designed based on minimizing queuing delays and power consumption. Therefore, the compute-dependent power consumption is defined as follows [35]:

$$P_{\text{comp}}(\upsilon(t), \beta(t)) = \beta(t) \cdot f_{n,\text{cpu}}^3, \tag{18}$$

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 (18) translates to the corresponding energy consumption as follows:

$$E_{\text{comp}}(\upsilon(t), \beta(t)) = E_{\text{cpu}}(t) + E_{\text{sw}}(t), \tag{19}$$

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

### 3.4 Formulating the optimization problem

Since the power management objective is to minimize power cost subject to delay constraints, latency minimization becomes the main objective of the proposed strategy by default, fulfilling the time constraint $\tau_{\max}$. Therefore, the objective of minimizing the power costs can be represented as that of minimizing the MEC energy consumption due to CPU processing in (22) and can be defined as follows:

$$\mathbf{P} : \arg\min_{\Omega} \sum_{t \in T} E_{\text{cpu}}(\upsilon(t), \beta(t)), \tag{20}$$

subject to

$$\begin{aligned}
&\mathbf{C1} : \delta_{\text{k,n}}(t) \in \{0, 1\}, \\
&\mathbf{C2} : r_0 \leq r_{n,k}(\mathbf{p}, \mathbf{n}, t) \leq r_{\max}, \\
&\mathbf{C3} : f_0 \leq f_{n,c}(t) \leq f_{\max}, \\
&\mathbf{C4} : \chi_{n,c}(t) \leq \Delta, \\
&\mathbf{C5} : r(t) \leq r_{\max}, \\
&\mathbf{C6} : \max\{2\zeta_c(t)\} + \Delta \leq \tau_{\max}^*,
\end{aligned} \tag{21}$$

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 (12). The constraint **C2** bounds the maximum task arrivals at the MEC server, noting the size of the incoming queue, $Q_{\text{in}}$, 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 $\Delta$ 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 $\Delta$ 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 $\Delta$ not to exceed the maximum predefined communication-plus-computation time $\tau_{\max}$.

Thus, 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 utilization and the energy consumed as follows:

$$E_{\text{cpu}}(t) = \sum_{c=1}^{C(t)} E_{\text{dyn}}(t) = \sum_{c=1}^{C(t)} \beta(t)(E_{\max}(t) - E_{\text{idle}}(t)), \tag{22}$$

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 (19) is the energy consumed while changing processing rates, which depends on the absolute gap in the processing rates as follows:

$$E_{\text{sw}}(t) = \sum_{n=1}^{N} \kappa_e (f_{n,c}(t+1) - f_{n,c}(t))^2, \tag{23}$$

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

## 4 Proposed power management algorithm and computational complexity analysis

In the proposed power management algorithm, it is assumed that the stability in terms of the worst-case queue 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 minimizing the cost function in (20) is crucial. However, due to the existence of (17) and constraint **C6**, (20) is a non-convex optimization problem and the use of convex optimization to solve the problem is difficult. Even if it were possible to employ convex optimization approaches, it would be very computationally demanding and impracticable to implement in real-life scenarios. Thus, the use of RL 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 maximize the two-way communication while minimizing the round-trip latency, which makes (20) 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 queuing delays at the input and output queues together with the server computation time give the overall processing delay in (12), and the cost function in (17) 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:

$$E_{\text{comm}}(\upsilon(t), \zeta(t)) = 2\sum_{n=1}^{N} P_{n,k}(r_{n,k}(t)) \cdot \zeta_{n,c}(t), \tag{24}$$

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

$$P_{n,k}(r_{n,k}(t)) = S_{n,k}(2^{r_{n,k}(t)/B(t)} - 1), \tag{25}$$

which is the Shannon-Hartley exponential discussed in [36], in which case $S_{n,k} = \frac{B(t)\cdot N_0}{g_{n,k}}$, $B(t)$ is the link bandwidth as defined in (4), and the server processing rate $\zeta_{n,c}(t) = \frac{r_{n,k}(\mathbf{p},\mathbf{n},t)}{f_{n,c}(t)}$.

A likely correlation between workload and energy consumption is required to derive the adaptive power management scheme. Here, estimates of the server response time, and energy consumption, as well as the current and predicted channel information are used to estimate the server processing rate for the next time slot. This helps the proposed scheme to be well robust and resilient against channel information uncertainty. Let $\Delta t < t$ be the smallest possible step size through which the system is monitored. Based on an initial behavior estimate $\hat{x}(t)$ (16), the adaptive controller explores possibilities of an appropriate decision $\upsilon$. It does this by creating a tree of depth $T$ which it traverses in steps of size $\Delta t$ until it arrives at optimum values for the next time slot $t + 1$. This is described as follows:

$$\hat{x}(t+1) \triangleq (\hat{\omega}(t + \Delta t), \hat{E}(t + \Delta t)), \tag{26}$$

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 (20), which can be solved by adopting the solution from [35]. 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 (20) 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 [37] is used to check if the current cost function is guaranteed to decrease or not as follows [35]:

$$J(x(t + \Delta T)) - J(x(t)) < 0. \tag{27}$$

It was proven in [36] that the stability of (27) could be achieved using

$$\hat{J}(\hat{x}(t + \Delta T)) - \hat{J}(x(t)) \leq -\int_{t}^{t+\Delta T} \phi(\hat{x}(s), \hat{u}(s))\text{ds}, \tag{28}$$

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 (28), the following result is obtained:

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

where $\phi(\hat{x}(s), \hat{u}(s))$, as defined in 15, is only known at the instant the solution of (20) is obtained. Once (20) 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 Fig. 2. 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:

$$\hat{\omega}(t+1) = (1 + \hat{q}(t+1)) \cdot \hat{\tau}(t+1), \tag{30}$$

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

$$\hat{q}(t+1) = q(t) + \left(\hat{\lambda}(t+1) - \frac{\beta(t+1)}{\hat{\tau}(t+1)}\right), \tag{31}$$

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

$$\hat{\tau}(t+1) \triangleq \gamma \cdot \tau(t) + (1-\gamma) \cdot \hat{\tau}(t), \tag{32}$$

where $\tau(t)$ is the actual processing time, and $\gamma$ 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:

$$\hat{E}(t+1) \triangleq \beta^2(t+1), \quad \text{where} \quad \beta(t+1) = \frac{f_{n,c}(t+1)}{f_{\max}}, \tag{33}$$

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

## 5 Algorithm design and description

The model shown in Fig. 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. The process of migrating containers from the controller to the different edge sites is shown in Fig. 3.

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

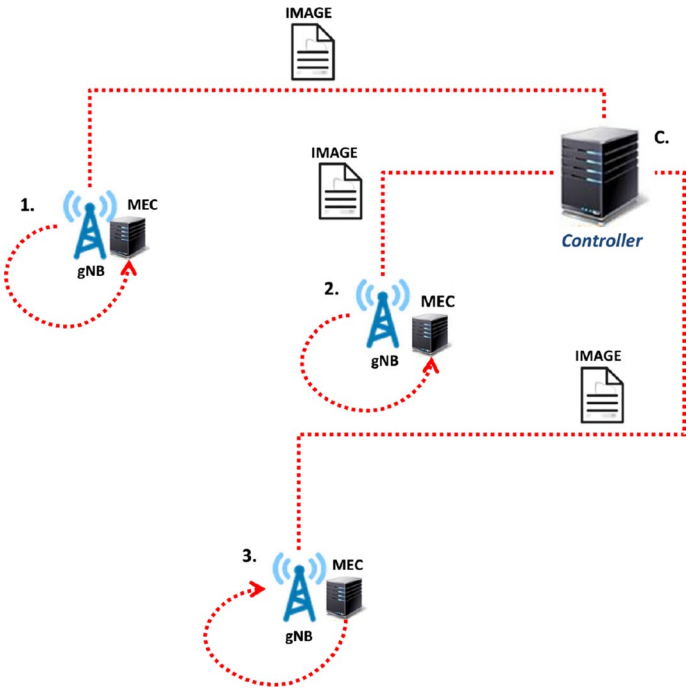


**Fig. 3** Container image migration from controller to edge site

- Populate system state information and initialize 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 behavior 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.

**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, $\phi^*$
**For** each $t$ in observation horizon of depth $T$ **do**
  Initialize next state as $s(t+1) = \emptyset$
  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 Results and discussion

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 ms apart and each site has a virtualized 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 favorably compare results obtained in this paper with the results from the baseline algorithm presented in [38], and with other existing models, such as the one in [12]. The baseline algorithm develops on the ETSI standards to provide a mobile resource-sharing framework that employs mobile edge servers to provide a cost-effective deployment of 6G edge computing, which enables edge resource sharing for massive IoT devices.

One of the most important performance metrics for the model is the achievable data offloading rate for the edge platform. The achievable offloading rate based on the task admission probability is shown in Fig. 4, while the average overhead for the network is shown in Fig. 5. The plot in Fig. 4 shows that the proposed edge system management algorithm (Table 1) achieves a better average task offloading rate and fewer overheads with respect to the number of offloading devices than the baseline shown in Fig. 5. From the two plots, it can also be observed that 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

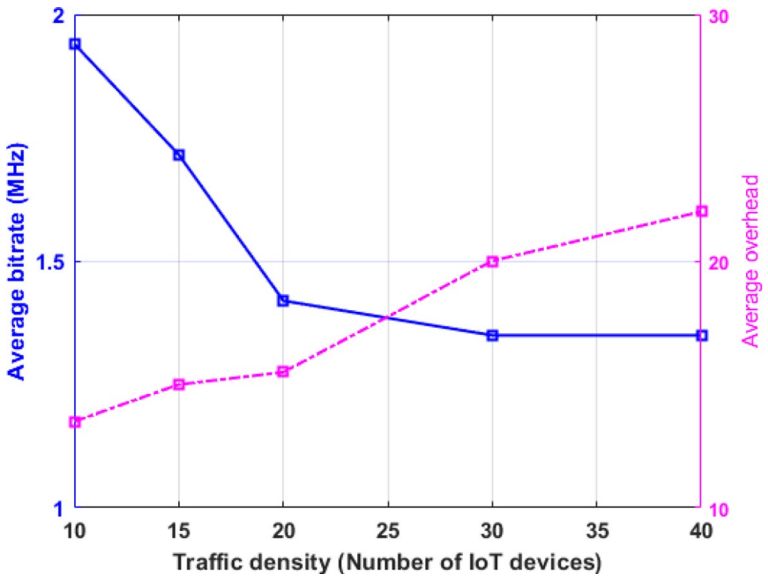


**Fig. 4** The average offloading rate and the average overhead plotted against the number of devices for the proposed algorithm

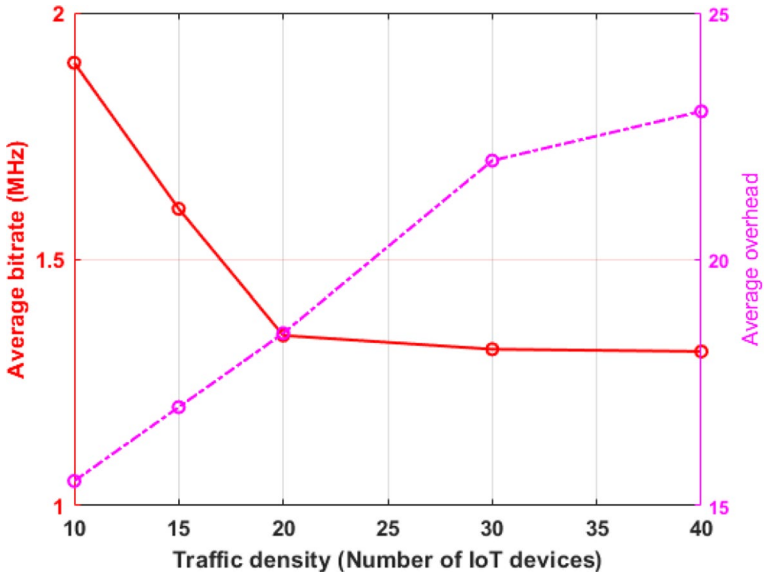


**Fig. 5** The average offloading rate and the average overhead plotted against the number of devices for the baseline algorithm in [38]

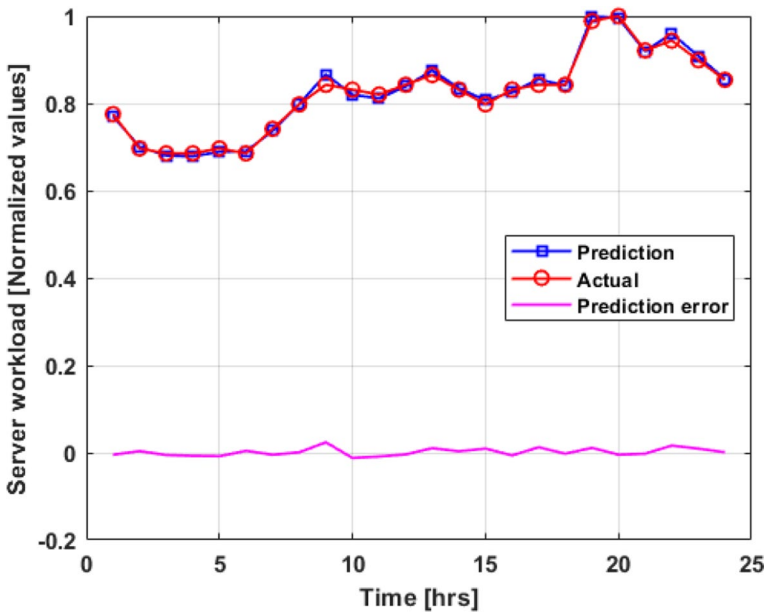


**Fig. 6** Normalized MEC server workload for a period of 24 h

the proposed algorithm improves the task admission probability, resulting in a better average offloading bit rate and lower overhead than the baseline.

Another important performance metric for the network is workload predictability. The workload prediction for the MEC platform investigated is carried out using a belief propagation neural network (BPNN) [39]. The prediction results are shown in Fig. 6. The agent used to train the BPNN is the Levenberg-Marquardt algorithm [40]. The root-mean-squared error (RMSE) [41] 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 Fig. 6, 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 (ANN). The prediction accuracy of 97.08% achieved by using the BPNN is deemed good enough for the modeled network in consideration.

A few other very important performance metrics investigated in this work are the welfare gain, the server response time, and the energy consumed by the network. The effect of an increasing number of users on the welfare gain is presented in Fig. 7. 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 Fig. 8. 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 Fig. 8, 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 Fig. 9. The plot shows an almost linear increase in energy consumption as the number

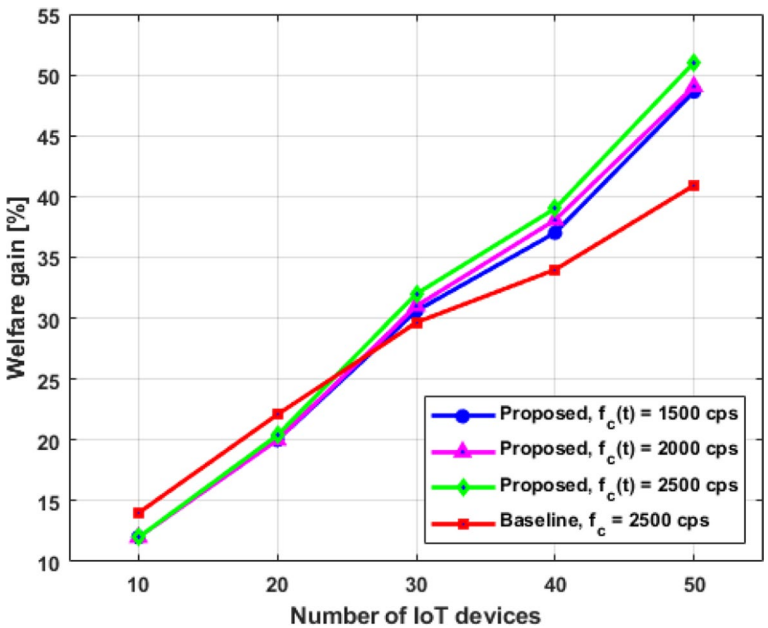


**Fig. 7** Welfare gain with the number of users

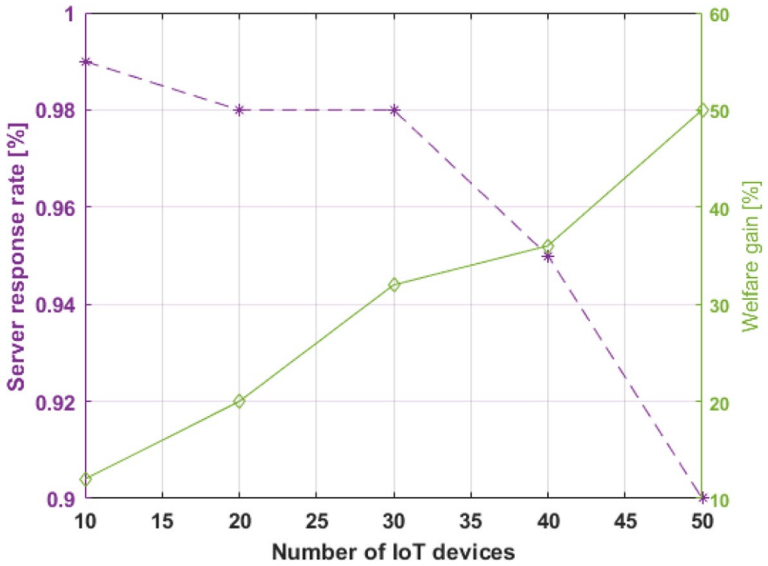


**Fig. 8** Evaluation of server response time and welfare gain with the number of users

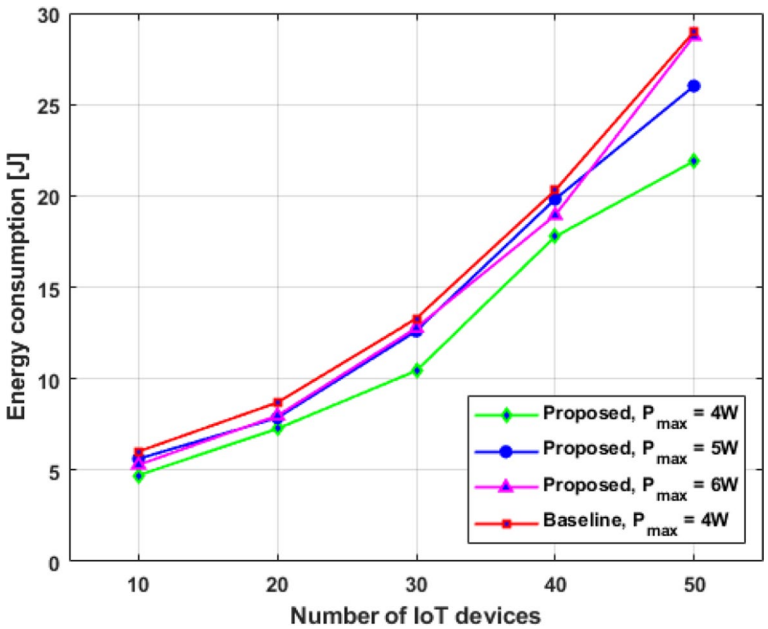


**Fig. 9** Evaluation of energy consumption with the number of users

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, which 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 the use of containerized edge platforms for achieving improved resource management in evolving 6G networks, using the mIoT as a practicable context of an application. The work exploited the correlation between workload and energy consumption to monitor the operation of the edge platform in the 6G-inspired mIoT network setting. The optimization 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 quality of service and experience through the computational resource-enriched VMs employed in the network design. Overall, the containerized edge computing platforms employed in the network are shown to significantly minimize the high energy consumption caused by high overheads in the 6G-inspired mIoT network, which can then be applied to other 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**

BSA developed the concepts, participated in the analysis and drafted the manuscript. MCH studied the problem, developed the models and completed the manuscript. BTJ participated in the concept design and analysis and coordinated and reviewed the draft of the manuscript. All authors read and approved the final manuscript.

**Funding**

The research was financially supported with funds through the SENTECH Chair in Broadband Wireless Multimedia Communications (BWMC) at the University of Pretoria.

**Availability of data and material**

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

## Declarations

**Conflict of interest**

The authors declare they have no Conflict of interest to declare.

Received: 20 July 2024 Accepted: 20 March 2025
Published online: 16 April 2025

**References**

1. M.C. Hlophe, B.S. Awoyemi, B.T. Maharaj, Power-Delay tradeoff in vehicle-to-edge networks using constrained reinforcement learning. IEEE AFRICON **2023**, 1–6 (2023)
2. B.S. Awoyemi, B.T. Maharaj, A.S. Alfa, Resource allocation in heterogeneous buffered cognitive radio networks. Wirel. Commun. Mob. Comput. **2017**(7385627), 1–12 (2017)
3. B.T. Maharaj, B.S. Awoyemi, Queuing systems in resource allocation optimisation for cognitive radio networks, in *Developments in Cognitive Radio Networks*. Springer, pp. 121–139 (2022)
4. D. Zeng, G. Min, Q. He, S. Guo, Convergence of edge computing and next generation networking. Peer-to-Peer Netw. Appl. **14**(6), 3891–3894 (2021)
5. A.S. Seisa, S.G. Satpute, B. Lindqvist, G. Nikolakopoulos, An edge-based architecture for offloading model predictive control for UAVs. Robotics **11**(4), 80 (2022)
6. T. Hasanin, A. Alsobhi, A. Khadidos, A. Qahmash, A. Khadidos, G.A. Ogunmola, Efficient multiuser computation for mobile-edge computing in iot application using optimization algorithm. Appl. Bion. Biomech. **2021**, 1–12 (2021)
7. F. Chen, A. Wang, Y. Zhang, Z. Ni, J. Hua, Energy efficient SWIPT based mobile edge computing framework for WSN-assisted IoT. Sensors **21**(14), 4798 (2021)
8. Y.-S. Jeong, J. Wang, N. Yen, Special issue on user behavior analysis in edge computing based internet-of-things. J. Ambient. Intell. Humaniz. Comput. **13**(3), 1573–1574 (2022)
9. M.C. Hlophe, B.T. Maharaj, From cyber-physical convergence to Digital Twins: a review on edge computing use case designs. Appl. Sci. **13**(24), 13262 (2023)

10. G. Mitsis, E.E. Tsiropoulou, S. Papavassiliou, Price and risk awareness for data offloading decision-making in edge computing systems. IEEE Syst. J. **16**(4), 6546–6557 (2022)
11. X. Li, L. Zhu, X. Chu, H. Fu, Edge computing-enabled wireless sensor networks for multiple data collection tasks in smart agriculture. J. Sens. **2020**, 4398061 (2020)
12. M. Hlophe, B. Maharaj, Prospect-theoretic DRL approach for container provisioning in energy-constrained edge platforms, in *IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*. IEEE **2023**, 1–5 (2023)
13. M. Gundall, J. Stegmann, C. Huber, H.D. Schotten, Towards organic 6G networks: Virtualization and live migration of core network functions, in *Mobile Communication - Technologies and Applications; 25th ITG-Symposium*, pp. 1–6 (2021)
14. L.J. Mwasinga, D.-T. Le, S.M. Raza, R. Challa, M. Kim, H. Choo, Rasm: resource-aware service migration in edge computing based on deep reinforcement learning. J. Parallel Distrib. Comput. **182**, 104745 (2023)
15. L. Zhao, G. Zhou, G. Zheng, X. You, L. Hanzo, Open-source multi-access edge computing for 6g: Opportunities and challenges. IEEE Access **9**, 158426–158439 (2021)
16. Y. Yue, X. Tang, Z. Zhang, X. Zhang, W. Yang, Virtual network function migration considering load balance and sfc delay in 6g mobile edge computing networks. Electronics **12**(12), 2753 (2023)
17. J. Zhang, Y. Liu, Z. Li, Y. Lu, Forecast-assisted service function chain dynamic deployment for sdn/nfv-enabled cloud management systems. IEEE Syst. J. **17**, 4371–82 (2023)
18. S.K. Kasi, M.K. Kasi, K. Ali, M. Raza, H. Afzal, A. Lasebae, B. Naeem, S. UI Islam, J.J. Rodrigues, Heuristic edge server placement in industrial internet of things and cellular networks. IEEE Internet Things J. **8**(13), 10308–10317 (2020)
19. F. Guo, B. Tang, J. Zhang, Mobile edge server placement based on meta-heuristic algorithm. J. Intell. Fuzzy Syst. **40**(5), 8883–8897 (2021)
20. V. Tiwari, C. Pandey, A. Dahal, D.S. Roy, U. Fiore, A knapsack-based metaheuristic for edge server placement in 5g networks with heterogeneous edge capacities. Fut. Gen. Comput. Syst. **153**, 222–233 (2024)
21. C.Y. Son, Digital connectivity: bolstering technical development and shaping the digital economy in South-East Asia (2022)
22. S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin et al., MEC in 5G networks. ETSI White Pap. **28**(28), 1–28 (2018)
23. T. Dlamini, S. Vilakati, LSTM-based traffic load balancing and resource allocation for an edge system. Wirel. Commun. Mob. Comput. **2020**, 8825396 (2020)
24. M.S. Rathore, N. Ahmad, R. Kohli, J. Iqbal, R. Alroobaea, S. Hussain, S.S. Ullah, F. Umar, In the direction of service guarantees for virtualized network functions. Wirel. Commun. Mob. Comput. **2022**, 5507845 (2022)
25. M.C. Hlophe, B.T. Maharaj, M.M. Sande, Energy-efficient transmissions in federated learning-assisted cognitive radio networks, in *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, pp. 216–222 (2021)
26. Y. Luo, W. Li, S. Qiu, Anomaly detection based latency-aware energy consumption optimization for iot data-flow services. Sensors **20**(1), 122 (2019)
27. M.C. Hlophe, B.T. Maharaj, Qoe-driven resource allocation for sus with heterogeneous traffic using deep reinforcement learning, in *2019 IEEE 2nd Wireless Africa Conference (WAC)*, pp. 1–5 (2019)
28. L. Metcalf, W. Casey, Chapter 3 - probability models, in *Cybersecurity and Applied Mathematics*, L. Metcalf and W. Casey, Eds.Boston: Syngress, pp. 23–42. (2016), [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128044520000038
29. J.L. Gustafson, *Little's Law*. Boston, MA: Springer US, (2011), pp. 1038–1041. [Online]. Available: https://doi.org/10.1007/978-0-387-09766-4_79
30. M.Y. Mir, S.-Z. Huang, Data forwarding with finite buffer capacity in opportunistic networks, in, et al., 27th Wireless and Optical Communication Conference (WOCC). IEEE **2018**, 1–5 (2018)
31. X. Ding, S.L. Smith, C. Belta, D. Rus, Optimal control of markov decision processes with linear temporal logic constraints. IEEE Trans. Autom. Control **59**(5), 1244–1257 (2014)
32. E. Rogers, K. Galkowski, D.H. Owens, *Control Systems Theory and Applications for Linear Repetitive Processes*, vol. 349 (Springer, Bern, 2007)
33. N. Mastronarde, M. van der Schaar, Joint physical-layer and system-level power management for delay-sensitive wireless communications. IEEE Trans. Mob. Comput. **12**(4), 694–709 (2012)
34. Y. Qu, B. Ng, M. Homer, A goodput distribution model for planning ieee 802.11 wbns in built environments. J. Netw. Comput. Appl. **99**, 28–46 (2017)
35. M.C. Hlophe, B.T. Maharaj, QoS provisioning and energy saving scheme for distributed cognitive radio networks using deep learning. J. Commun. Netw. **22**(3), 185–204 (2020)
36. H. Chen, F. Allgöwer, A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. Automatica **34**(10), 1205–1217 (1998)
37. F. Hamidi, M.N. Abdelkrim, J. Houssem, Searching candidate lyapunov function with threshold accepting algorithm, in *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 26–31 (2011)
38. R. Cong, Z. Zhao, G. Min, C. Feng, Y. Jiang, Edgego: a mobile resource-sharing framework for 6g edge computing in massive iot systems. IEEE Internet Things J. **9**(16), 14521–14529 (2022)
39. J. Kuck, S. Chakraborty, H. Tang, R. Luo, J. Song, A. Sabharwal, S. Ermon, Belief propagation neural networks (2020)
40. X. Huang, H. Cao, B. Jia, Optimization of levenberg marquardt algorithm applied to nonlinear systems. Processes **11**(6), 1794 (2023)
41. B. Awoyemi, T. Walingo, F. Takawira, 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


## Publisher's Note