


# POSITRON: Efficient Allocation of Smart City Multifunctional IoT Devices Aware of Computing Resources

Leandro H. B. da Silva  [ Instituto Federal da Paraíba | [henrique.leandro@academico.ifpb.edu.br](mailto:henrique.leandro@academico.ifpb.edu.br) ]


Jefferson L. F. da Silva  [ Instituto Federal da Paraíba | [ferreira.jefferson@academico.ifpb.edu.br](mailto:ferreira.jefferson@academico.ifpb.edu.br) ]

Ricardo Pereira Lins  [ Instituto Federal da Paraíba | [ricardo.lins@academico.ifpb.edu.br](mailto:ricardo.lins@academico.ifpb.edu.br) ]

Fernando Menezes Matos  [ Universidade Federal da Paraíba | [fernando@ci.ufpb.br](mailto:fernando@ci.ufpb.br) ]

Aldri Luiz dos Santos  [ Universidade Federal de Minas Gerais | [aldri@dcc.ufmg.br](mailto:aldri@dcc.ufmg.br) ]

Paulo Ditarso Maciel Jr.   [ Instituto Federal da Paraíba | [paulo.maciel@ifpb.edu.br](mailto:paulo.maciel@ifpb.edu.br) ]

 Programa de Pós-Graduação em Tecnologia da Informação (PPGTI), Unidade Acadêmica de Informação e Comunicação, Instituto Federal da Paraíba, Av. Primeiro de Maio, 720 - Jaguaribe, João Pessoa/PB, 58015-435, Brazil.

Received: 08 November 2023 • Accepted: 24 April 2024 • Published: 02 July 2024

**Abstract** Many IoT scenarios demand continuous capture of information from multifunctional sensors and smart units, as well as sending those data to cloud centers. However, allocating tasks to these sensors is not straightforward due to the urgency and priority that each type of data collection requires depending on the needs of the urban environment. This paper presents the POSITRON scheme for managing the sensing allocation in a multifunctional IoT network from previously defined policies. The policies take into account the characteristics of the applications running on the network and the different specifications of the available devices. We implemented POSITRON in a network simulator aiming to analyze its efficiency in allocating network resources. The results point out that considering the requirements demanded by applications and the distinct characteristics of multifunctional IoT devices brings benefits to resource allocation.

**Keywords:** Resource Allocation, Policy-based Management, Multifunctional IoT Devices

## 1 Introduction

In recent decades, urban centers have experienced unprecedented population growth, with the estimation that by 2050, about 68% of the world's population will live in cities [Klein and Anderegg, 2021]. A larger population means longer lines, more traffic, competition for jobs, the need for more hospitals and health centers, more housing, and increased generation, capture and distribution of energy and water, among other problems [United Nations, 2019]. The Internet of Things (IoT) has become a basic and essential technology for the operation of various intelligent urban scenarios [Pedroso *et al.*, 2021]. In this context, all environments/objects ("things") can present a better performance through the incorporation of technology, making them have a unique identification and communicate with each other for the transfer of data. This interaction, guided by the actions these objects perform, supports the creation of intelligent and autonomous infrastructures. Functional intelligence is then provided by incorporating heterogeneous objects into environments in a non-perceptible way to users [El Bouanani *et al.*, 2019].

IoT devices from various manufacturers may have different physical characteristics, like battery power capacity and transmission range, which could limit the types of applications that a given set of devices can serve [Perera *et al.*, 2021]. Thus, a single multifunctional IoT network, that could serve different applications and support different types of devices, could then replace several other IoT networks and reduce costs. Therefore, application developers would take advantage of the same existing infrastructure of IoT devices. Re-

search initiatives, however, disregard the potential of such infrastructure and focus on *offloading* the applications in some *Edge/Cloud* environment, as a way of allocating resources and reducing usage metrics. Contrary to this trend, initiatives like the former *Array of Things Project* [Catlett *et al.*, 2022], and now *SAGE: A Software-Defined Sensor Network* [Catlett *et al.*, 2020], exemplify the deployment of an extensible urban monitoring and environmental awareness ecosystem integrated into an IoT infrastructure. Here, multifunctional devices act as robust computing platforms for a wide array of applications. Prioritizing privacy and security above all else, on-device processing of images and sound is emphasized, reducing reliance on cloud-based data storage or transmission.

In that regard, efficiency in allocating resources to applications refers to the ability to effectively and appropriately distribute computational, storage, communication, and energy resources among various IoT applications or services running on a network of connected devices. This is crucial for ensuring a smooth operation, at the same time delivering the desired functionality, and minimizing resource wastage. Efficiency can be defined by several key aspects, such as: **resource utilization**, which measures how effectively the available resources are used, by evaluating the percentage of resources (e.g., CPU, memory, storage) that are actively engaged in processing tasks compared to the total available; **latency**, which refers to the time it takes for a request to be sent from a source to a destination and for a response to be received; **energy efficiency**, which focuses on how efficiently energy is utilized by IoT devices that often have limited battery life; **load balancing**, which involves distributing

workloads evenly across available resources to prevent overloading some devices while leaving others underutilized; and **fault tolerance**, which shows the ability to continue operating in the presence of hardware or software failures, possibly reallocating resources to backup or redundant systems. Efficient resource allocation is often achieved through a combination of hardware utilization in terms of applications' requirements and intelligent software algorithms.

In this paper, we present the POSITRON scheme for managing the "efficient" selection and allocation of multifunctional IoT devices in urban sensing environments. POSITRON takes into account the characteristics of the applications running on the network, as well as the different specifications of available devices. It employs two levels of management through a policy that, at the first level, maps applications to a previously defined workload profile (resource consumption). At a second level, it performs load balancing or saturation (concentration) on possible devices. It should be noted that the feature of "efficiency" in the allocation of resources means an appropriate fit between the application profile and the device where it is executed. Thus, it is intuitive to assume a more efficient use of network resources. The implementation of POSITRON in a network simulator demonstrated its efficiency in resource allocation. The results indicate that considering the requirements demanded by applications and the distinct characteristics of multifunctional IoT devices brings benefits in the allocation of resources. A preliminary version of the proposed scheme was previously published in Rocha *et al.* [2022]. We extended the previous work with a more appropriate representation of the simulated scenarios, expanded the number of applications, increased the number of nodes, and diversified the scenarios to reflect a more accurate real-life situation. These changes allowed for a comprehensive performance evaluation. Mainly, the previous paper was extended with the following contributions:

- A better contextualization of the addressed problem in terms of resource allocation efficiency;
- A wider comparison with the related work literature;
- A broader formalization of the management policies employed by the proposed scheme;
- A detailed description of the solution implemented as a proof of concept;
- A more extensive evaluation of different performance aspects related to allocating and usage of resources, as well as the level of preempted applications.

The rest of the paper is organized as follows. Section 2 presents related works. Section 3 describes the POSITRON scheme. Section 4 details the implemented solution, evaluation methodology, and achieved results. Lastly, Section 5 presents the concluding remarks.

## 2 Related Works

Despite the literature shows up solutions to deal with resource allocation in IoT networks, they have usually focused on reducing specific metrics such as communication costs [Tsai, 2018], latency [Zhao *et al.*, 2019; Sangaiah *et al.*, 2020] and energy consumption [Guim *et al.*, 2022;

Xavier *et al.*, 2022; Nikpour *et al.*, 2023; Andal and Jayapal, 2022]; are aimed at a specific environment/solution such as in Multi-access Edge Computing (MEC) [Xavier *et al.*, 2020; Bolettieri *et al.*, 2021; Wang *et al.*, 2022], Zigbee [Lv *et al.*, 2020], Software-Defined Networking (SDN)/Network Function Virtualization (NFV) [Mukherjee *et al.*, 2020], blockchain [Rafique *et al.*, 2020]; and propose some kind of proof-of-concept (PoC) implementation [Calderoni *et al.*, 2019; Ali and Calis, 2019] or optimization via mathematical models [Li *et al.*, 2021; Bashir *et al.*, 2022]. Further, they mostly focus on applications' *offloading* as a way of allocating resources. Table 1 presents a comparison with related works, taking into account important aspects of the POSITRON design. The proposed allocation scheme aims to leverage resource distribution in multifunctional IoT environments by integrating application requirements, node hardware specifications, and policy-based management. This approach would involve a proof-of-concept implementation to validate technical feasibility before full-scale deployment. By considering application requirements, the system can tailor resource allocation to meet specific smart city user needs, prioritizing critical tasks. Node hardware specifications would ensure fair resource distribution, enhancing performance and reducing bottlenecks. Policy-based management offers flexibility to dynamically adjust resource allocation based on system variability and user priorities. Ultimately, this comprehensive approach promises a robust and efficient resource allocation system suited for complex computing environments like the ones with IoT.

In Tsai [2018], the authors take into account several constraints in a high-performance IoT system on resource allocation, namely, different user requirements, different types of devices, huge need for communications, network bandwidth, and computing power limited. The work proposes an algorithm to solve the issue of resource allocation in such a system, aiming to reduce the total cost of communication between devices. For that, concepts of data clustering and meta-heuristics are employed, in addition to the use of IoT gateways for more efficient communication. However, the solution does not consider the multifunctional characteristics of the network nodes, and nor does it use policies for autonomous management of the IoT network. So, efficiency is measured in terms of data communication, disregarding other aspects of resources such as processing and storage.

In Zhao *et al.* [2019], the authors study the challenge of efficiently allocating resources to edge servers, in a scenario of a growing number of IoT devices and services in smart cities. They optimize hosted applications for satisfactory performance and compare workload *offloading* algorithms assuming response time as the main metric. In Sangaiah *et al.* [2020], the authors propose an optimization algorithm for the allocation and scheduling of resources in IoT systems in order to reduce the total communication cost between devices and a nearby gateway. An unfeasible manual approach due to data heterogeneity and the amount of resources to be allocated. The efficient allocation proposed here, on the other hand, may consider different aspects other than communication to better allocate applications in existing multifunctional IoT devices. Furthermore, in addition to implementing a load balancing policy, we also introduce a saturation policy.

**Table 1.** Comparative summary with related works.

	PoC implementation	Applications' requirements	Specification of nodes	Multifunctional IoT	Policy-based management
Tsai [2018]		✓	✓		
Zhao et al. [2019]		✓	✓		
Calderoni et al. [2019]	✓				
Ali and Calis [2019]	✓				
Sangaiah et al. [2020]					✓
Xavier et al. [2020]		✓	✓		
Lv et al. [2020]	✓				
Mukherjee et al. [2020]	✓				
Rafique et al. [2020]	✓				
Li et al. [2021]				✓	
Bolettieri et al. [2021]		✓			
Clarindo et al. [2021]	✓	✓			
Wang et al. [2022]		✓			
Bashir et al. [2022]				✓	
Guim et al. [2022]			✓		
Xavier et al. [2022]		✓			
Andal and Jayapal [2022]	✓			✓	
Nikpour et al. [2023]	✓			✓	
POSITRON	✓	✓	✓	✓	✓

In Guim et al. [2022], an autonomous lifecycle management is proposed for converged green-computing edge platforms, enabling resource-efficient workload orchestration. Dynamic and intelligent resources that host services from multiple users are configured, ensuring SLOs for each service. However, the efficiency is only considered in terms of energy consumption and multifunctional devices are disregarded. In Xavier et al. [2022], the authors present a fully distributed resource allocation algorithm for an IoT-edge-cloud environment. It manages the usage of resources by promoting collaboration between edge nodes and supports applications' heterogeneity and requirements, reducing latency and increasing energy efficiency at the edge. They also aim at optimizing the energy management, while our proposal deals with the efficient allocation of several computational resources, being able to balance the workload to save battery.

In Nikpour et al. [2023], the authors have developed an IoT framework tailored for energy management within smart cities. This framework not only undertakes data collection and storage but also serves as a platform for application development. Furthermore, the study delves into intelligent energy management solutions and elucidates how the gathered data contributes to the continuous monitoring and enhancement of system efficiency. Despite a shared context that underscores the vital role of continuous data capture through multifunctional sensors and smart devices, it is noteworthy that this study is primarily based on energy consumption reduction, while our work is distinctly focused on the resource management process facilitated by policy-based strategies.

Similarly, the authors in Andal and Jayapal [2022] introduce a real-time energy management system integrated into a suitable platform that leverages the Internet of Things (IoT) for monitoring and processing real-time control data. Multifunctional sensors wirelessly detect data, transmitting it to cloud servers via internet connectivity. The system exhibits the capability to manage energy production across various micro grids and enables remote control of energy consump-

tion and generation. While both articles delve into resource management within IoT networks, our study centers on the equitable allocation of multifunctional devices, whereas this study focuses on a renewable energy system.

In Xavier et al. [2020], the authors propose a resource allocation algorithm for a CoT (*Cloud of Things*) environment, which combines a three-tier architecture with IoT devices, edge nodes and the cloud. The proposal supports the heterogeneity of devices and applications, taking advantage of the distributed nature of edge nodes to promote collaboration during the allocation process. The collaborative algorithm follows a heuristic-based approach inspired by an economic model to solve the resource allocation problem in CoT. The simulated results indicate efficient use of system resources while meeting latency requirements and assuming different priorities of IoT applications, compared to a two-tier cloud-based approach. Unlike our scheme, which works centrally, the collaborative algorithm depends on the use of external resources for greater efficiency in resource allocation.

The study in Clarindo et al. [2021] proposes an architecture that integrates a cloud layer with a fog computing layer for data extraction, transformation, and loading into a spatial data warehouse. It introduces guidelines for smart city managers to implement this architecture effectively. The work addresses the context of using IoT in urban environments. While this article focuses on managing spatial data for analysis in smart cities, ours highlights task allocation in a multifunctional IoT network based on predefined policies.

In Bolettieri et al. [2021], the issue of multiple edge access (MEC) by applications running on IoT devices is investigated. The proposed MEC architecture aims at supporting several IoT service providers on a common platform. From an application perspective, service placement (*placement*) and data management are modeled, assuming data-level dependencies between IoT services and sensing resources. Though, it does not consider the multi-functionality of IoT devices nor different hardware specifications. In

[Wang *et al.*, 2022], the authors propose an *offloading* algorithm based on distributed deep learning in the SD-MEC IoT scenario. Multiple neural networks are invoked in parallel to optimize resource scheduling. Similar to our work, application characteristics are considered in the *offloading* decision. However, the MEC strategy seeks to reduce the IoT utilization, instead of making efficient use of available resources.

The work Lv *et al.* [2020] outlines an intelligent urban environment monitoring system utilizing the ZigBee wireless network. This system comprises a basic monitoring network and a remote receiving terminal. The network nodes are dynamically organized, with each node assigned a unique address. Similar to ours, the work addresses the need for continuous data capture in IoT scenarios. Nevertheless, there are distinctions in terms of the proposed solutions. While POSITRON primarily focuses on the efficient allocation of network resources, the article places emphasis on information gathering and its transmission to a remote terminal.

The research by the authors of Mukherjee *et al.* [2020] introduces the development of a distributed IoT network based on SDN with NFV implementation for smart cities, aiming to enhance network performance, scalability, availability, integrity, and security. Multiple distributed controllers and a clustering scheme are employed to improve load balancing. Despite the similarities, this article proposes a distributed IoT network based on SDN with NFV implementation for smart cities, whereas the POSITRON scheme presents an approach to resource allocation through predefined policies.

The work Rafique *et al.* [2020] suggests the use of Blockchain technology for secure implementation of ITS, providing a framework that securely and efficiently integrates smart vehicles and service infrastructure. While both works propose solutions to optimize network resource utilization and enhance application efficiency in IoT contexts, this article particularly emphasizes the utilization of Intelligent Transportation Systems (ITS) and concerns regarding their security vulnerabilities. In contrast, our work focuses on efficient management through resource allocation via policies.

In Calderoni *et al.* [2019], the authors propose the utilization of the IoT Manager, a comprehensive framework developed at the University of Bologna. IoT Manager presents an open and flexible solution for the management of sensor networks, complete with a detailed implementation strategy and a readily available client for research and educational purposes. While the work addresses the imperative need for IoT network management, IoT Manager is presented as an open-source framework tailored for sensor network management, whereas POSITRON is a specialized scheme designed for resource allocation in multifunctional IoT network. While the former places emphasis on implementation strategy and its accessibility for research and education, the latter focuses in efficiently allocating resources through predefined policies.

In Ali and Calis [2019], the authors outline a centralized intelligent governance framework for monitoring the performance of governmental entities in a smart city. Thousands of IoT devices communicate with each other, generating a vast amount of data. The objective is to control, manage, monitor, operate, and consolidate this network of devices from a central location. The framework employs optimal classification techniques to analyze data and monitor the performance of

governmental entities. Both articles address IoT device management in different contexts. While the article introduces an intelligent governance framework for monitoring the performance of governmental entities, the POSITRON scheme is presented for resource allocation management in a multifunctional IoT network.

In Li *et al.* [2021], the network availability is affected by various types of applications and devices with limited computational capabilities. To deal with this, it is proposed a cooperative *offloading* model of multi-user IoT applications based on a mixed-integer nonlinear programming formulation. Results indicate the efficiency in allocating part of the applications at the edge, by reducing energy consumption, latency and network utilization. Nonetheless, the multifunctional characteristic of devices is not explored and a reference implementation is not provided. In Bashir *et al.* [2022], similar to the previous one, an *offloading* approach is a dynamic resource allocation strategy for *cloud* and *fog* environments, using logistic regression to calculate the required edge load. A framework for ranking available nodes takes into account decisions performed by the TOPSIS algorithm. However, the mathematical formulations in both works do not relate different workload profiles to multifunctional devices. Further, they use an inverse approach to our proposed scheme, which makes a more adequate allocation in the final nodes, based on their specifications.

Overall, POSITRON addresses resource allocation within a multifunctional IoT network through predefined policies while satisfying all the above-mentioned important characteristics. Most works lack consideration for multifunctional characteristics and autonomous management policies, such as Tsai [2018] that emphasize data communication efficiency. Yet, some works deal with resource allocation for edge servers in smart cities, considering workload offloading algorithms, e.g. [Zhao *et al.*, 2019], that take into account communication costs, but lack policy-based management and multifunctional device utilization. The works Guim *et al.* [2022], Xavier *et al.* [2022], Nikpour *et al.* [2023], and Andal and Jayapal [2022] primarily focus on energy management rather than policy-based resource allocation, disregarding multifunctionality. Xavier *et al.* [2020] proposes a platform for collaboration among nodes and Clarindo *et al.* [2021] mainly focuses on spatial data management as a service layer. In contrast, POSITRON emphasizes task allocation in a multifunctional IoT network and can be seen as an infrastructure manager for both works. Bolettieri *et al.* [2021] also disregards multifunctionality or hardware specifications. Wang *et al.* [2022] considers application characteristics in resource scheduling but seeks to reduce IoT utilization rather than making efficient use of available resources like POSITRON. Lv *et al.* [2020] focuses on continuous data capture, similar to POSITRON, but emphasizes information gathering and transmission rather than resource allocation. Mukherjee *et al.* [2020] shares similarities with POSITRON, but it focuses on network enhancement rather than resource allocation through predefined policies. Rafique *et al.* [2020] emphasizes security and efficiency, and Calderoni *et al.* [2019] focuses on sensor network management, but unlike POSITRON, they don't focus on policy-based resource allocation. Ali and Calis [2019] focuses on managing network-

ing aspects of IoT devices, contrasting with POSITRON’s emphasis on resource allocation. Li *et al.* [2021] and Bashir *et al.* [2022] don’t explore multifunctionality, policy-based management, or workload profiles. In particular, Sangaiah *et al.* [2020] focus on minimizing communication costs between resources and gateways by efficiently assigning resources to gateways, while POSITRON aims to optimize resource usage by matching application needs with device capabilities. Both approaches involve load balancing policies, but they differ in implementation: POSITRON has been validated through a PoC, while Sangaiah *et al.* [2020] propose heuristic-based optimization using evolutionary algorithms.

### 3 The POSITRON Scheme

This section details the hierarchical network setting for POSITRON operation and describes its modules that comprise the Controller component, the selection policies, and the allocation algorithm to meet applications’ demands.

#### 3.1 Multifunctional Network Organization

The multifunctional feature comes from different types of IoT devices and configurations. Such environment attends different types of applications requesting services in many contexts (e.g. IA, ITS, blockchain, etc.), as expected in smart cities. Moreover, these applications have the most varied hardware demands. One can assume that monitoring public roads requires a minimum storage capacity available at the network nodes, while performing encryption can take advantage of greater processing power. Certainly, applications sensing some physical phenomenon need to constantly transmit data on the network. However, even though IoT devices have heterogeneous characteristics, it is likely to find groups of nodes with similar configurations. Such similarities may indicate that a given group of nodes is more suitable for running one type of application. Figure 1 depicts the hierarchical network organization where POSITRON operates.

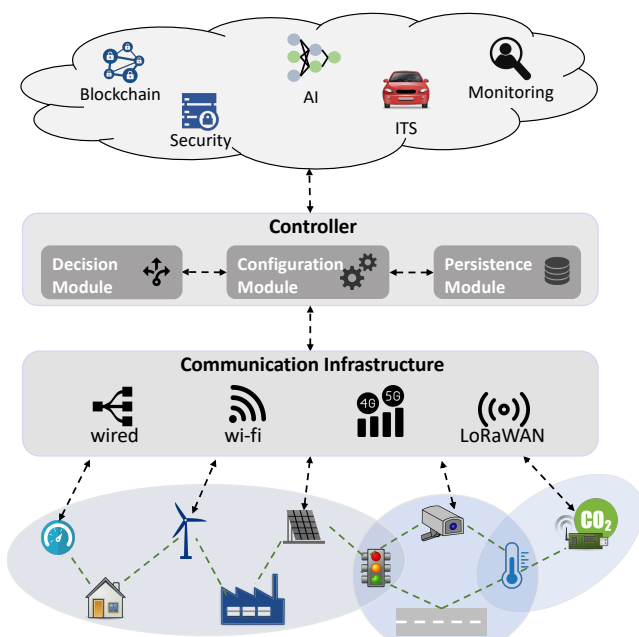


Figure 1. Multifunctional IoT topology (adapted from Rocha *et al.* [2022]).

In this multifunctional IoT environment, a control layer coordinates the efficient allocation scheme employed by POSITRON. The Controller acts as an interface between the applications that demand resources and the adjacent network infrastructure. The Decision Module (DM) evaluates the application demands and uses policies to determine the most suitable set of devices to meet applications’ requirements, according to their current status. This policy-based approach allows the system to make adaptive decisions considering network conditions. The DM result is the input for the Configuration Module (CM), which communicates with the devices through the infrastructure layer, enforcing the allocation suggested by the DM. The Persistence Module (PM) registers and periodically updates all information regarding the IoT nodes in a database, so that POSITRON has an always-updated view of the network state. In addition to nodes’ attributes, the PM records the execution history related to devices and the applications they execute.

#### 3.2 Selection policies

POSITRON employs a hierarchy of policies to dynamically select the devices according to the state of network nodes and applications’ demands, as illustrated in Figure 2. This allows the system auto-configuration based on pre-established rules, with no need to reboot or redeploy [?]. Furthermore, by employing this hierarchy of policies, successive refinements in the selection can be performed to search for the most appropriate subset of nodes. This approach simplifies the addition of new policies to meet different objectives, such as reducing operational costs or fulfilling commercial agreements.

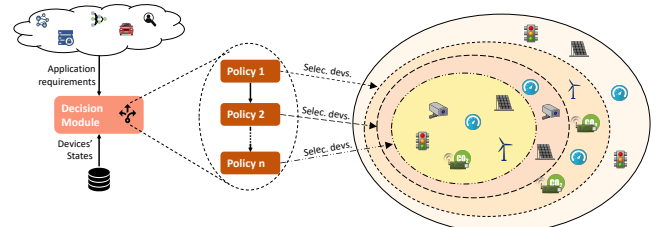


Figure 2. Selection policy hierarchy (adapted from Rocha *et al.* [2022]).

We have applied three simple policies at two hierarchical levels to validate the POSITRON scheme, as can be seen in Figure 3. The function  $satisfy(nodeList, node.resources, app.requirements)$  filtrates all nodes with the minimum amount of free resources to meet the application’s requirements. The  $select(n, nodeList)$  function then selects the first  $n$  nodes from the list. The  $sort(nodeList, criterion, order)$  function sorts  $nodeList$  according to the specific criterion (CPU, ram, storage, or network), in ascending or descending order of node usage (number of running applications). Initially, the system applies a fitness policy by choosing the IoT nodes that meet the application’s requirements. Afterward, POSITRON fine-tunes the selected node list by applying a second policy: load-balancing or saturation. Load balancing aims to balance the use of computational resources in the nodes, sorting the list in ascending order. Thus, the first  $n$  nodes are those that meet the application’s requirements and are least overloaded. On the other hand, the saturation

policy sorts the list in descending order, making the first  $n$  nodes the most overloaded. So, the network has more free nodes to serve future requests.

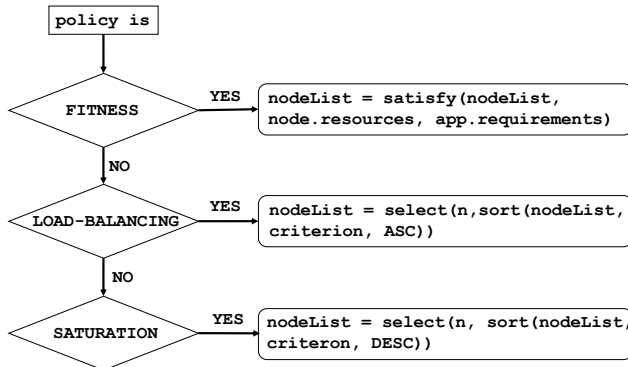


Figure 3. Workflow policies (adapted from Rocha et al. [2022]).

The system assumes that the Controller has access to updated device information stored in a database (PM module), including processing power, storage, memory, transmission rate, and current application usage. It uses this information to search for devices that meet specific application requirements. Messages are exchanged whenever device information is updated or events like application termination occur. The satisfy function finds nodes capable of meeting minimum application requirements by searching the database. It triggers the selection of a set of nodes, which uses a sorting function based on a predefined balancing or saturation policy. ASC parameter sorts devices by ascending usage, while DESC parameter sorts by descending allocation.

Algorithm 1 describes the general node selection process, which uses the application’s demand input. Essentially, the procedure consists of waiting for the arrival of a specific application (line 2) and extracting its description, including type and minimum requirements (e.g. CPU, memory, and storage). Although it is not always easy, we assume that it is possible to estimate such minimum requirements (line 3) and draw a profile (line 4) used for the first hierarchical level of the management policy. Then, POSITRON triggers a query (line 5) to search for a list of nodes that meet these requirements. It is worth noting that this query considers the idle computing resources of the nodes, even if they are running other applications. If the query returns an empty list, the system sends an unavailability message (line 7) and configures a node availability alert (line 8). If a non-empty list returns, the system chooses a policy (load-balancing or saturation) from the second hierarchical level based on network conditions (line 10). As discussed earlier, it sorts the list in ascending or descending order depending on the chosen policy. Finally, POSITRON allocates the application to the node (line 11).

The algorithm’s computational complexity can be analyzed from two perspectives: technological considerations and the algorithm itself. Technological issues like transmission delay and computational power are abstracted, assuming various edge and cloud computing techniques can address them. The algorithmic complexity involves searching for nodes meeting certain criteria and then sorting them based on a chosen policy (balancing or saturation). Searching typically has a complexity of  $O(n)$ , where  $n$  is the number of

### Algorithm 1 Multi-functional node allocation process.

---

**Require:** AppDesc, Policy

```

1: while TRUE do
2:   AppDesc ← waitApp()
3:   requirements ← estimateReq(AppDesc)
4:   profile ← defineProfile(requirements)
5:   nodeList ← queryDB(nodes.resources ≥ requirements)
6:   if (nodeList = empty) then
7:     echo("No available node")
8:     configureAvailabilityAlert()
9:   else
10:    nodeList ← enforces "load balancing" or "saturation"
11:    allocateApp(AppDesc, nodeList[1])
12:  end if
13: end while
    
```

---

elements. Sorting is often  $O(n \log n)$  for known algorithms like *mergesort*. Combining both steps, the overall complexity is the sum. Since sorting usually dominates, especially with large lists, the overall complexity averages to  $O(n \log n)$ . However, improving the computational time by proactively pre-ordering based on node occupancy is ideally possible.

## 4 Evaluation

In this section, we present the POSITRON implementation, simulated scenarios, considered metrics, and results.

### 4.1 Implementation

POSITRON was implemented in C++ at the network simulator NS-3 version 3.38. Regarding IoT specs, we extended the class that implements the network nodes with common attributes to multifunctional devices: **battery level, initial consumption rate, current consumption rate, CPU, memory, transmission rate, and storage capacity**. The control layer considering the three functional modules in Figure 1 was also implemented. This layer runs on a special node called *Controller*, which applies all the functions corresponding to the management policies. To this end, an application conceptual model that echoes UDP packets was used for generating, sending, and handling network packets. IoT nodes accountable for executing the applications are called *Workers*. The POSITRON code is available in a public repository<sup>1</sup>.

Figure 4 shows a sequence diagram with the interaction of modules, representing the arrival of two applications, going through the selection process, efficiently allocating *Workers*, and ending with the applications’ execution. Nodes are selected and sorted based on both the application profile and the previously defined management policy. In this hypothetical example, a battery loss situation of a chosen node (*Worker X*) is also represented. Once the alert message is received, the MC starts again the selection process of a new device (*Worker Y*) in order to run the corresponding application. After each application’s execution, alerts are sent by *Workers Y* and *Z* to the MC so their status is updated in the MP, which in turn maintains current information about *Workers’* usage. For the sake of simplification, status update, node registration and output messages were omitted from the figure.

<sup>1</sup><https://github.com/ifpb/positron>



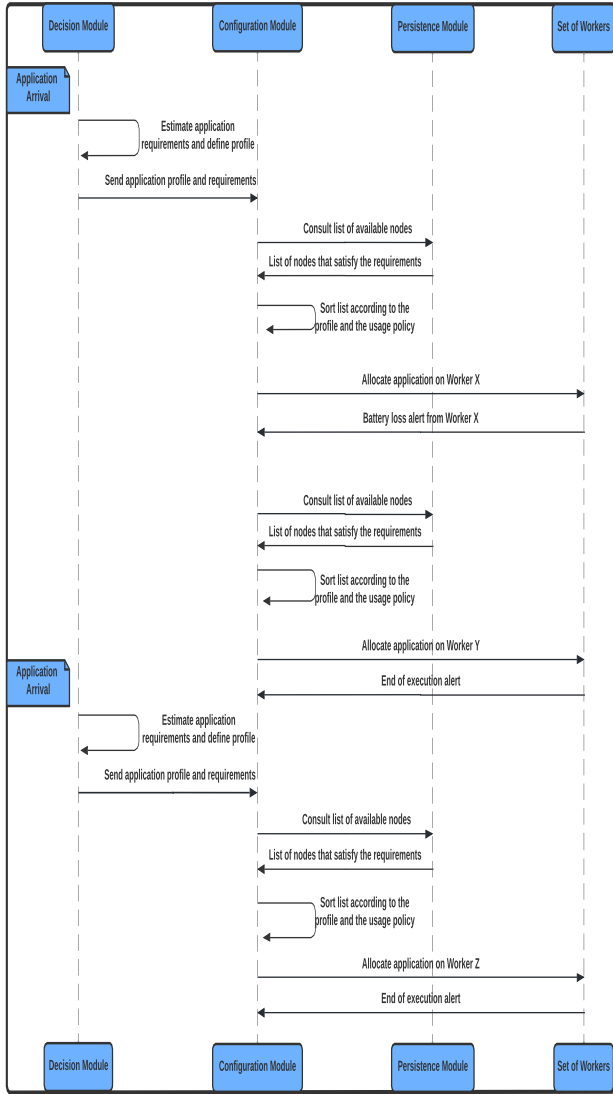


Figure 4. Modules interaction (adapted from Rocha *et al.* [2022]).

## 4.2 Simulated Scenarios

The base configuration scenario comprises one *Controller* and a variable number of *Workers* representing the multifunctional IoT devices. Additionally, network communication takes place through LrWPAN, 6LoWPAN, and IPv6 protocols. In this common configuration, IoT device specs are differentiated through groups with equivalent attributes, like the ones defined in Section 4.1. Similarly, application types are differentiated by their minimum requirements, as detailed below. The arrival of *Workers* occurs at the beginning of the simulation and, consequently, registration in the PM happens as soon as they arrive. Applications' arrivals are randomly set along the simulation period, as further explained.

We assumed three distinct types of applications that require: high processing power, e.g. encryption algorithms; large storage capacities, e.g. monitoring public roads; and high data transmission rate in the network, e.g. environmental sensing. Three different groups of IoT *Workers* were also configured in each scenario, whose specifications vary according to the demands requested by the applications. More specifically, one group more suitable for applications that require processing power, another for applications that need

storage capacity, and a last one for applications requesting high network throughput.

The total number of *Workers* ranges from 30 to 180 nodes in the system, which represents a factor of 10, 20, 30, 40, 50, or 60 times each group. A total batch of 600 applications (200 of each type) arrives in the system, approximately 1/3 of them randomly arrives at each 8h-shift interval along the day; i.e. the first 200 between 0 and 8 a.m., 200 more between 8 a.m. and 4 p.m., and the last 200 ones after 4 p.m., totaling a number of 600 applications through the day. The duration of each application was selected according to a normal distribution, with an average of 6h and a standard deviation of 20%. The idea is to represent a day in a smart city, where different applications may arrive at distinct times and a number of IoT nodes is available for their execution. We evenly match the arrivals with adequate *Workers* to verify the efficiency of the POSITRON scheme. Every scenario was repeated 30 times for the appropriate statistical analysis and the confidence level used was 95%. Regarding battery consumption, we first simulate the scenario with no battery loss, and then consider scenarios where *Workers* lose battery throughout the simulation, i.e. 10%, 20%, and 30% of *Workers* completely losing their battery power. Given that not all nodes can be conveniently located near a power source, our objective is to assess the impact of nodes running out of energy on the proposed efficient allocation scheme and investigated policies. Table 2 presents the simulation parameters.

Table 2. Simulation parameters.

Parameters	Values
Number of applications	600 (200 of each type) <sup>a</sup>
Number of <i>Workers</i>	$n \times 3$ (one of each group) <sup>b</sup> , $n = \{10, 20, 30, 40, 50, 60\}$
Management policies	2 levels (demand <sup>c</sup> and load <sup>d</sup> )
Applications' arrivals	60 (20 of each type <sup>a</sup> ) every 8h, during a total time of 24h
Applications' duration	normally distributed (avg of 6h, sd of 20%)
<i>Workers</i> battery scenarios	No battery loss, 10% of battery loss, 30% of battery loss, 50% of battery loss

<sup>a</sup> Cryptography (cry), monitoring (mon), and sensing (sen).

<sup>b</sup> Processing (pro), storage (str), and transmission rate (tra).

<sup>c</sup> Corresponding application demand and device specification.

<sup>d</sup> Node utilization saturation or load balancing.

## 4.3 Metrics

We developed two metrics to assess the POSITRON efficiency. The first one takes into account the number of nodes actually used, as a way to distinguish the policies of load balancing or saturation in fewer devices. To do that, we assume a finite set of application profiles  $P$ . The number of applications that arrived in each profile is defined as  $I_p$ . Similarly, IoT nodes are classified from a finite set of types  $G$  and the amount of each node, in each type, is represented by the set  $J_g$ . So, the set of arriving applications, the set IoT nodes, and the total number of nodes are defined, respectively, as:

$$A = \{a_{i,p} | i \in I_p; p \in P\}, N = \{n_{j,g} | j \in J_g; g \in G\}, \text{ and}$$

$$|N| = \sum_{g=1}^{|G|} \sum_{j=1}^{|J_g|} (g+j)^0.$$

The first metric, therefore, indicates the percentage of IoT nodes used throughout the simulation and is represented by:

$$\mathcal{P}_{used} = \frac{10^2}{|N|} \sum_{g=1}^{|G|} \sum_{j=1}^{|J_g|} \mathcal{U}(n_{j,g}), \quad (1)$$

where a utilization function is defined by,

$$\mathcal{U}(n_{j,g}) = \begin{cases} 1, & \text{if } n_{j,g} \text{ performed any application;} \\ 0, & \text{otherwise.} \end{cases}$$

This metric is notably related to the allocation policy used and does not necessarily indicate a good or bad value.

The second metric concerns the percentage of ‘‘efficient’’ allocation, which represents the ‘‘hit’’ rate when executing the applications on the most suitable devices. Namely, devices with the more appropriate hardware specs for the considered application profiles. So, the percentage at a given node  $g$  is:

$$\mathcal{P}_{efficient}^g = \frac{10^2}{|N|} \sum_{p=1}^{|P|} \sum_{i=1}^{|I|} \sum_{j=1}^{|J|} \mathcal{U}(n_{j,g}, a_{i,p}) \cdot \mathcal{J}(a_{i,p}, g), \quad (2)$$

where we use a specification of the previous usage function,

$$\mathcal{U}(n_{j,g}, a_{i,p}) = \begin{cases} 1, & \text{if } n_{j,g} \text{ performed any } a_{i,p} \text{ application;} \\ 0, & \text{otherwise.} \end{cases}$$

Also, both application profile and IoT group are related as:

$$\mathcal{J}(a_{i,p}, g) = \begin{cases} 1, & \text{if } g \text{ is the appropriate group to profile } p; \\ 0, & \text{otherwise.} \end{cases}$$

Then, the total percentage of efficient allocation is defined as the sum for each group of IoT nodes:

$$\mathcal{P}_{efficient} = \sum_{g=1}^{|G|} \mathcal{P}_{efficient}^g.$$

Two other metrics that effectively quantify the system’s overload are the percentage of preempted applications, primarily caused by node failures; and the makespan increase, which indicates the percentage growth in applications’ response time. The former can be represented by the ratio of applications that were preempted at least once, given by:

$$\mathcal{P}_{preempted} = \frac{10^2}{|A|} \sum_{p=1}^{|P|} \sum_{i=1}^{|I|} \mathcal{R}(a_{i,p}), \quad (3)$$

where,

$$\mathcal{R}(a_{i,p}) = \begin{cases} 1, & \text{if } a_{i,p} \text{ was re-executed at least once;} \\ 0, & \text{otherwise.} \end{cases}$$

Finally, the percentage increase in makespan takes into account both the application’s duration and the total time in the system; which in turn is the time the application was finished minus the arrival time, both represented respectively by:

$$\mathcal{D}(a_{i,p}), \mathcal{F}(a_{i,p}), \text{ and } \mathcal{A}(a_{i,p}),$$

were

$$\mathcal{M}(a_{i,p}) = \mathcal{F}(a_{i,p}) - \mathcal{A}(a_{i,p})$$

is the makespan (system time) of an application  $a_{i,p}$ , and

$$\mathcal{P}_{makespan} = \frac{10^2}{|A|} \sum_{p=1}^{|P|} \sum_{i=1}^{|I|} \left( 1 - \frac{\mathcal{D}(a_{i,p})}{\mathcal{M}(a_{i,p})} \right), \quad (4)$$

is the percentage of makespan increase. It is noted that

$$\frac{\mathcal{D}(a_{i,p})}{\mathcal{M}(a_{i,p})} \leq 1$$

and

$$\mathcal{D}(a_{i,p}) = \mathcal{M}(a_{i,p})$$

if and only if  $a_{i,p}$  is executed immediately (upon arrival) and uninterruptedly. Basically, the percentage increase in makespan normalizes an application’s system time by duration. A higher percentage increase indicates a system time proportionally higher compared to its overall runtime.

The simulation parameters (Table 2) were instantiated as follows:  $P = \{cry, mon, sen\}$ ,  $I_p = \{1, 2, \dots, 200\} \forall p$ ,  $G = \{pro, str, tra\}$ , and  $J_g = \{1, \dots, n\} \forall g \mid n = \{20, 30, 40, 50, 60\}$ ; the latter varies in each scenario.

#### 4.4 Results

Before detailing the allocation results, Figure 5 demonstrates a balanced execution of the applications over the simulation time, in each one of the 3 distinct groups of *Workers*, for one specific scenario. As also expected, the applications’ arrivals are well distributed among the 3 shifts of the day.

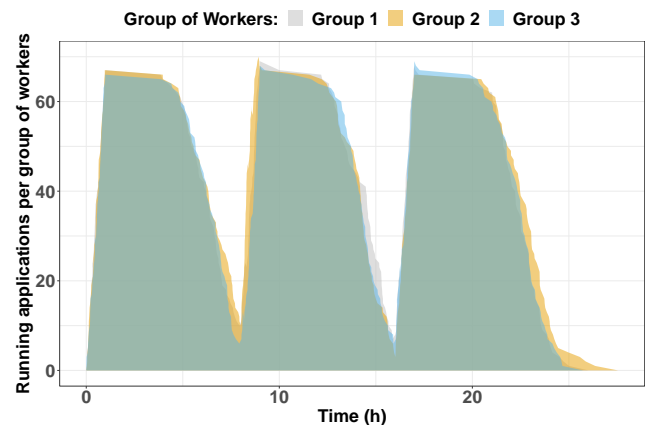
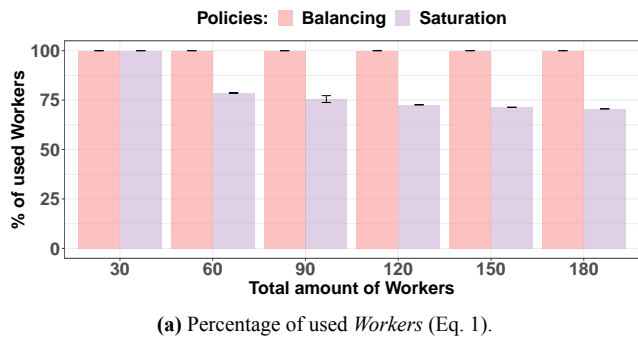


Figure 5. Running applications per group of *Workers*.

Figure 6 illustrates the results for the execution of a simulated scenario with no battery loss among *Workers*. Firstly, as we can see in Figure 6(a), there is a clear difference in the number of used resources according to the applied policy, whether it is load balancing or not. As expected, a resource balancing policy uses a higher number of available



Workers and can be applied in situations where the several network nodes have enough battery to run the applications. On the other hand, in a situation where it is more advantageous to use a smaller number of Workers, given, for example, a power consumption saving scenario, the saturation policy always uses a smaller percentage of nodes. Both plots in Figure 6(b) depict the ability to properly allocate applications, considering the specific demands and specs of existing Workers. In case of a device saturation policy, applications are allocated in the most “efficient” way possible; i.e. to the most suitable Workers for their executions, validating then the POSITRON management scheme. When a load balancing policy is used, there is also a high percentage of “efficient” allocation, with no less than 88% of applications allocated properly. In other words, even using a policy that seeks to balance resource usage and, in this way, save others, it is possible to achieve high allocation efficiency. Thus, the price you “pay” for balancing does not cause great damage to the “efficiency” exercised during the allocation.



wise expected, Figure 7(a) shows a higher percentage of used Workers compared to Figure 6(a), since 10% of nodes run out of battery and leave the simulation. So, the applications running on those Workers are preempted and re-executed after the selection of new ones. There is still a difference between load balancing and saturation policies, which can enforce clear utilization patterns according to network conditions. In Figure 7(b), an almost fully efficient allocation of applications can be seen, no matter the leaving number of Workers. However, an evident impact on the allocation process is noted since an unequal percentage of allocated applications is achieved. In particular, for the saturation policy, the percentage of applications “efficiently” allocated to the Group 1 of Workers increases as the total number of Workers also increases. This happens when a saturated Worker loses its battery and a high number of applications must be preempted. As the applications suitable for the Group 1 type of Workers are more restrictive in terms of requirements (CPU and memory), we observe a higher number of preemption and, consequently, more applications are reallocated on those Workers.

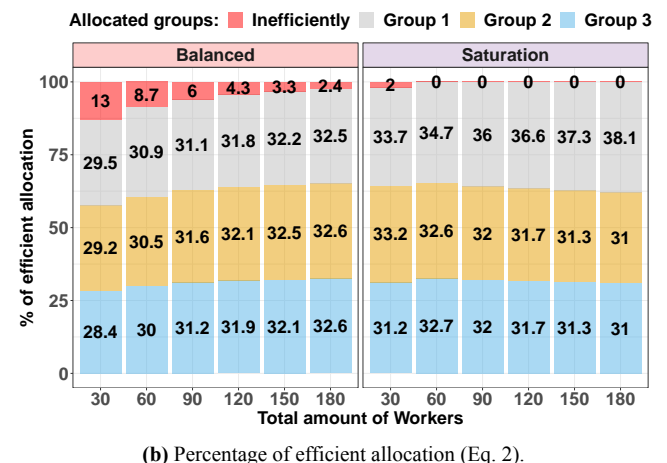
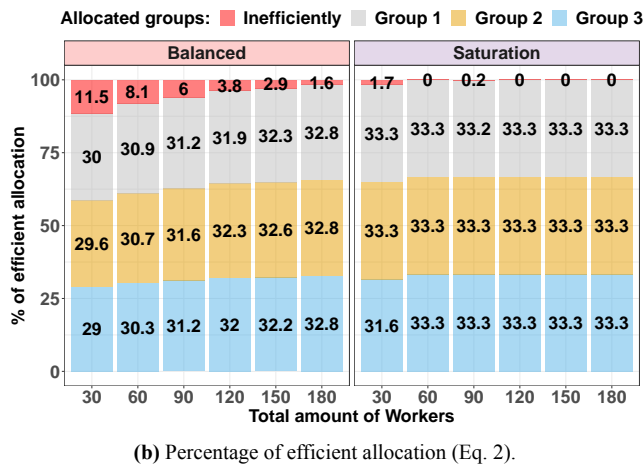
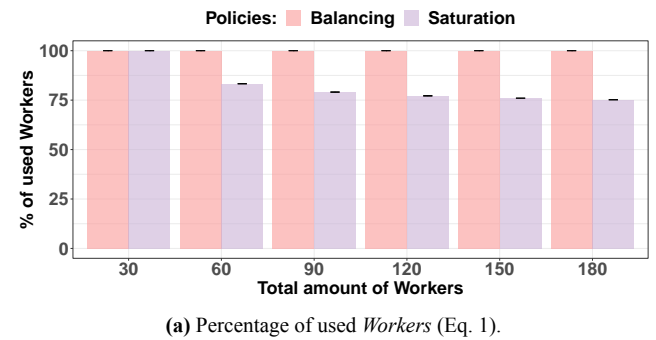


Figure 6. Results for the scenario with no battery loss.

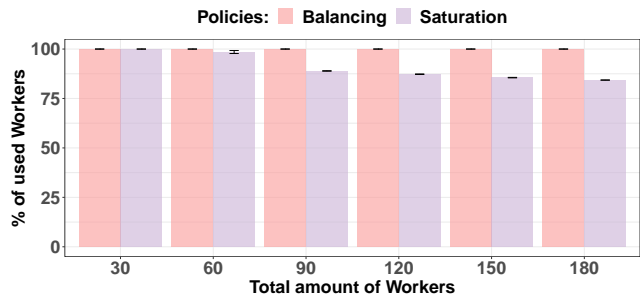
Figure 7. Results for the scenario with 10% of battery loss.

It is noteworthy to highlight that an “inefficient” allocation to a given node does not mean that the application was not executed. This situation represents an allocation where the application would be more appropriate for another type of device, but it wasn’t disregarded. For instance, from the obtained results in Figure 6, the average of applications allocated “inefficiently” was no more than 12%, in the balanced scenario with 30 Workers. Overall, the application execution rate was virtually 100% considering all scenarios.

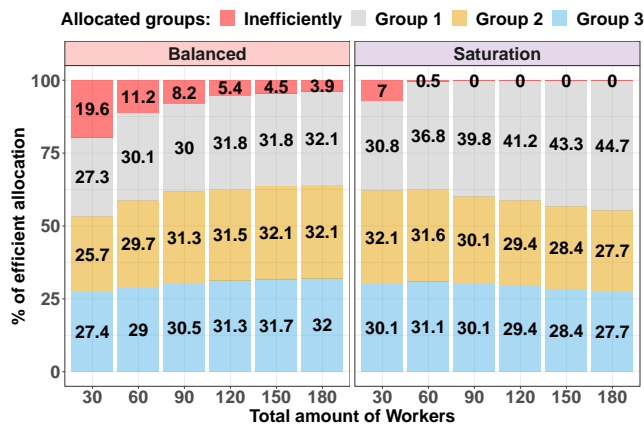
Figure 7 presents the results for the execution of a simulated scenario with 10% of battery loss among Workers. Like-

Figure 8 presents the results for the execution of a simulated scenario with 30% of battery loss among Workers. Just like the previous result, Figure 8(a) shows an increasingly smaller difference between load balancing and saturation policies. Additionally, the percentage of efficient allocation is also strongly impacted at this level of battery loss, as we can see in Figure 8(b). Regarding the load balancing policy, there is still a certain balance considering only applications that were allocated efficiently, especially as the number of Workers increases. However, the number of applications

allocated “inefficiently” increased significantly. The results with the resource saturation policy are even worse, given the more restrictive approach of trying to allocate as few *Workers* as possible. Again, we observe that the saturation policy allocates applications more efficiently, even if this means relocating the same application several times.



(a) Percentage of used *Workers* (Eq. 1).

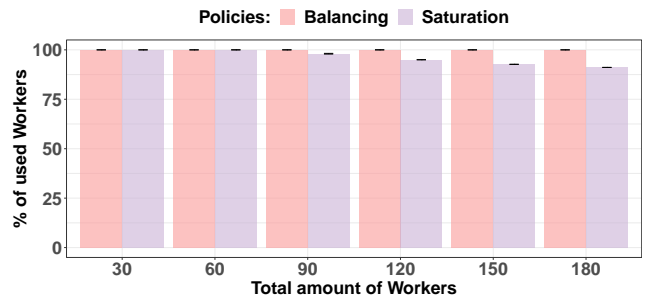


(b) Percentage of efficient allocation (Eq. 2).

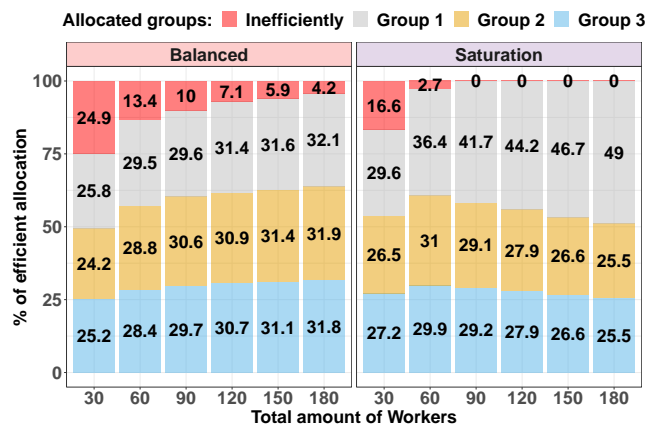
Figure 8. Results for the scenario with 30% of battery loss.

The results with a battery loss level of 50% confirm that POSITRON is strongly impacted in allocating resources, as observed in Figure 9. Figure 9(a) shows that roughly all *Workers* are used to run applications to some extent, as a consequence of nodes leaving the system due to lack of battery. A 50% battery loss is a level at which most *Workers* are required for the simulation. In other words, from this level of decrease on, we almost lose the ability to distinguish between balanced resource utilization and saturation. Likewise, Figure 9(b) illustrates that the efficiency of resource allocation is even more affected, confirming unsatisfactory results with a high *Worker* loss rate. As the number of nodes losing battery increases, it’s clear that remaining nodes in each category are insufficient to execute applications “appropriately”, i.e., preempted applications must be executed by any available nodes and not the most appropriate ones. This is more evident for a smaller number of *Workers*. The bottom line is that, as the level of battery loss increases, efficiently allocating resources becomes increasingly challenging.

Figure 10 corroborates that the saturation policy has a higher percentage of preempted applications, those started in more than one *Worker*. For all battery loss levels, 10% (a), 30% (b), and 50% (c), the resource saturation policy has a higher average number of applications preempted than the



(a) Percentage of used *Workers* (Eq. 1).



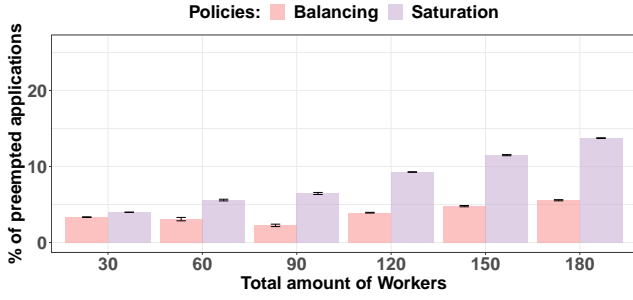
(b) Percentage of efficient allocation (Eq. 2).

Figure 9. Results for the scenario with 50% of battery loss.

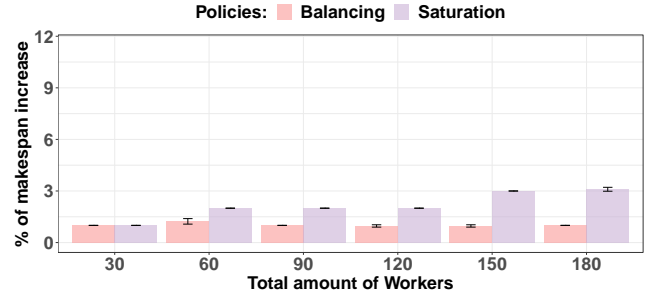
load balancing one, varying in the range of approximately 4 to 25%. Since the former is more restrictive and seeks to overload as few *Workers* as possible, those nodes lose battery more quickly and need to preempt applications again. The latter also presents a significant number of preempted applications, with averages in the range of 2 to 20%.

A significant outcome is the percentage increase in makespan, as depicted in Figure 11. The load balancing policy consistently exhibits lower increases, resulting in makespans closer to those observed in scenarios without application preemption. For this policy, the average makespan increase ranges from approximately 1 to 7%. These values translate to an average makespan increase of 1% with 10% battery loss, 3.2% with 30% loss, and 4.7% with 50% loss. Conversely, the results with the saturation policy range from approximately 1 to 11%. These values translate to an average makespan increase of 2% with 10% battery loss, 5.8% with 30% loss, and 8.1% with 50% loss. We attribute these results to the overload introduced to the system due to battery loss on the part of *Workers*, leading to lost executions and the need for application re-execution.

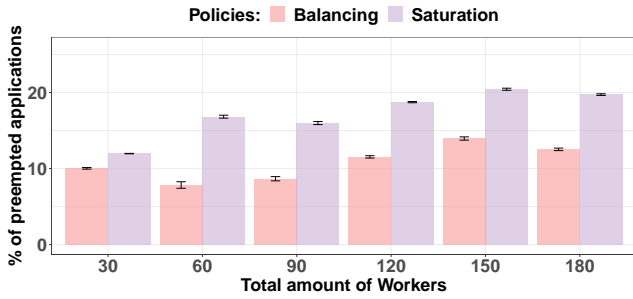
Although the results demonstrate POSITRON’s ability to allocate resources efficiently, it’s also envisioned the inability to react to a very high level of *Worker* loss. This might happen because the battery level is not taken into account when choosing the resources to be allocated. After all, we believe that it’s not common to describe the requirements of an IoT application in terms of the battery level needed, given the heterogeneous and multifunctional environment described in this work. However, the POSITRON scheme can be fully adjustable to also consider the battery level as a *Worker* pa-



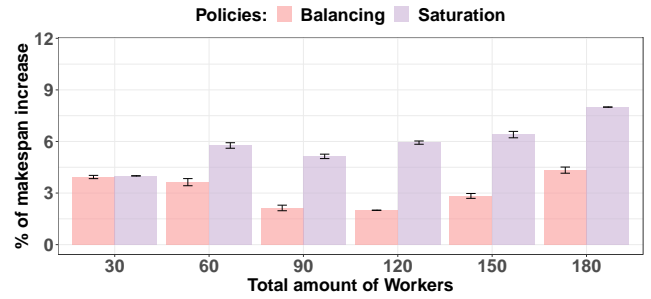
(a) 10% of battery loss.



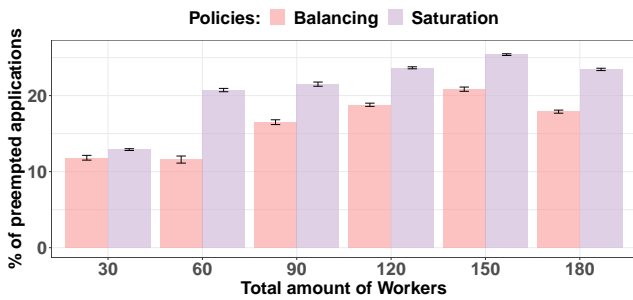
(a) 10% of battery loss.



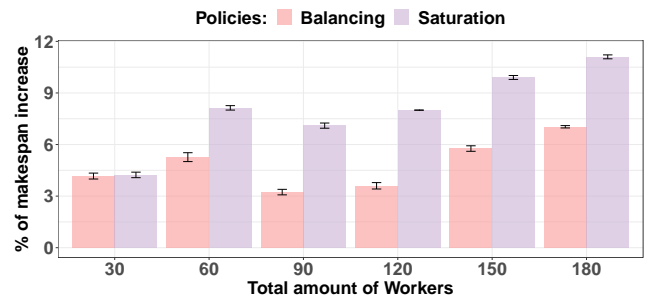
(b) 30% of battery loss.



(b) 30% of battery loss.



(c) 50% of battery loss.



(c) 50% of battery loss.

Figure 10. Results for the percentage of preempted applications (Eq. 3).

Figure 11. Results for the percentage of makespan increase (Eq. 4).

parameter of choice. Trying to infer the minimum battery level during the allocation procedure can be an interesting research problem, in order to mitigate the situation of preemption and battery loss. For example, selecting *Workers* taking into account their battery level and current allocation scenario can represent a multi-variable optimization problem, and this investigation is also a subject of interest for further research.

## 5 Conclusion

The presented work introduces the POSITRON scheme, designed to efficiently manage sensing allocation within a multifunctional IoT network. Unlike previous approaches, this scheme takes into account not only application characteristics but also the unique specifications of available nodes. The implementation in the NS-3 network simulator demonstrates its effectiveness in a controlled IoT network scenario. The findings indicate that POSITRON excels in employing resource usage policies and allocating applications efficiently, particularly in scenarios with minimal node battery loss. However, introducing a node battery-saving policy emerged as a promising enhancement.

The experiments confirm POSITRON’s capabilities in managing multifunctional IoT devices in systems with low

preemption rates under the proposed policies. Future endeavors will focus on devising strategies to further minimize battery loss. Additionally, a more comprehensive characterization of application requirements and IoT node sensing capabilities is planned. Lastly, an investigation into formulating management policies for diverse IoT devices will be pursued.

## Acknowledgements

This research was supported by Edital n° 49/2022 PIBITI/CNPq, Chamada Interconecta n° 07/2023 PIBICT from PRPIPG/IFPB, and partially supported by Paraiba Research Foundation - FAPESQ (call n° 09/2021).

## Funding

This research was partially funded by PRPIPG/IFPB and CNPq.

## Authors’ Contributions

All authors contributed, read and approved the final manuscript.

## Competing interests

The authors declare that they have no competing interests.

## Availability of data and materials

No datasets were generated or analysed during the current study.

## References

- Ali, U. and Calis, C. (2019). Centralized Smart Governance Framework Based on IoT Smart City Using TTG-Classified Technique. In *the 16th IEEE International Conference on Smart Cities*, pages 157–160. DOI: 10.1109/HONET.2019.8908070.
- Andal, C. K. and Jayapal, R. (2022). Design and implementation of IoT based intelligent energy management controller for PV/wind/battery system with cost minimization. *Renewable Energy Focus*, 43:255–262. DOI: 10.1016/j.ref.2022.10.004.
- Bashir, H. et al. (2022). Resource allocation through logistic regression and multicriteria decision making method in IoT fog computing. *Trans. on Emerging Telecom. Technologies*, 33(2). DOI: 10.1002/ett.3824.
- Bolettieri, S. et al. (2021). Application-aware resource allocation and data management for MEC-assisted IoT service providers. *Journal of Network and Computer Applications*, 181:103020. DOI: 10.1016/j.jnca.2021.103020.
- Calderoni, L., Magnani, A., and Maio, D. (2019). Iot manager: An open-source iot framework for smart cities. *Journal of Systems Architecture*, 98:413–423. DOI: 10.1016/j.sysarc.2019.04.003.
- Catlett, C. et al. (2020). Measuring cities with software-defined sensors. *Journal of Social Computing*, 1(1):14–27. DOI: 10.23919/JSC.2020.0003.
- Catlett, C. et al. (2022). Hands-on computer science: The array of things experimental urban instrument. *Computing in Science & Engineering*, 24(1):57–63. DOI: 10.1109/MCSE.2021.3139405.
- Clarindo, J. P., C. Castro, J. P., and D. Aguiar, C. (2021). Combining Fog and Cloud Computing to Support Spatial Analytics in Smart Cities. *Journal of Information and Data Management*, 12(4). DOI: 10.5753/jidm.2021.1798.
- El Bouanani, S., El Kiram, M. A., Achbarou, O., and Outchakoucht, A. (2019). Pervasive-Based Access Control Model for IoT Environments. *IEEE Access*, 7:54575–54585. DOI: 10.1109/ACCESS.2019.2912975.
- Guim, F., Metsch, T., et al. (2022). Autonomous Lifecycle Management for Resource-Efficient Workload Orchestration for Green Edge Computing. *IEEE Transactions on Green Communications and Networking*, 6(1):571–582. DOI: 10.1109/TGCN.2021.3127531.
- Klein, T. and Anderegg, W. R. (2021). A vast increase in heat exposure in the 21st century is driven by global warming and urban population growth. *Sustainable Cities and Society*, 73:103098. DOI: <https://doi.org/10.1016/j.scs.2021.103098>.
- Li, X., Zhao, L., et al. (2021). A cooperative resource allocation model for IoT applications in mobile edge computing. *Computer Communications*, 173:183–191. DOI: 10.1016/j.comcom.2021.04.005.
- Lv, Z., Hu, B., and Lv, H. (2020). Infrastructure monitoring and operation for smart cities based on iot system. *IEEE Transactions on Industrial Informatics*, 16(3):1957–1962. DOI: 10.1109/TII.2019.2913535.
- Mukherjee, B. K. et al. (2020). An SDN Based Distributed IoT Network with NFV Implementation for Smart Cities. In *Cyber Security and Computer Science*, pages 539–552, Cham. Springer International Publishing. DOI: 10.1007/978-3-030-52856-0\_43.
- Nikpour, M. et al. (2023). Intelligent Energy Management with IoT Framework in Smart Cities Using Intelligent Analysis: An Application of Machine Learning Methods for Complex Networks and Systems. <https://arxiv.org/abs/2306.05567>.
- Pedroso, C., de Moraes, Y. U., Nogueira, M., and Santos, A. (2021). Relational Consensus-Based Cooperative Task Allocation Management for IIoT-Health Networks. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 579–585. Available at: [https://ieeexplore.ieee.org/abstract/document/9463938?casa\\_token=eKAs8Fyk2nUAAAAA:DHcwGvi8ND7gsJLVRtHxbkDyk9AHw9dbe7DTEo44\\_GlCK2U1nm545wz513Mz10YgEoogsGIxBw](https://ieeexplore.ieee.org/abstract/document/9463938?casa_token=eKAs8Fyk2nUAAAAA:DHcwGvi8ND7gsJLVRtHxbkDyk9AHw9dbe7DTEo44_GlCK2U1nm545wz513Mz10YgEoogsGIxBw).
- Perera, A. et al. (2021). Light-based Internet of Things: Implementation of an Optically Connected Energy-autonomous Node. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7. DOI: 10.1109/WCNC49053.2021.9417484.
- Rafique, W. et al. (2020). A blockchain-based framework for information security in intelligent transportation systems. In *Intelligent Technologies and Applications*, pages 53–66, Singapore. Springer Singapore. DOI: 10.1007/978-981-15-5232-8\_6.
- Rocha, D., de Gois, A., da Silva, L. H. B., Matos, F., Santos, A., and Maciel Jr., P. D. (2022). Um Esquema para Alocação Justa de Dispositivos IoT Multifuncionais Ciente dos Recursos Computacionais. In *Workshop de Gerência e Operação de Redes e Serviços (WGRS)*. DOI: 10.5753/wgrs.2022.223423.
- Sangaiah, A. K. et al. (2020). IoT Resource Allocation and Optimization Based on Heuristic Algorithm. *Sensors*, 20(2). DOI: 10.3390/s20020539.
- Tsai, C.-W. (2018). SEIRA: An effective algorithm for IoT resource allocation problem. *Computer Communications*, 119:156–166. DOI: 10.1016/j.comcom.2017.10.006.
- United Nations (2019). Department of economic and social affairs, population division (2019). Technical report, World Population Prospects 2019: Highlights (ST/ESA/SER.A/423), United Nations, New York, USA. Available at [https://population.un.org/wpp/Publications/Files/WPP2019\\_Highlights.pdf](https://population.un.org/wpp/Publications/Files/WPP2019_Highlights.pdf).
- Wang, Z. et al. (2022). Computation offloading and resource allocation based on distributed deep learning and software defined mobile edge computing. *Computer Networks*, 205:108732. DOI: 10.1016/j.comnet.2021.108732.
- Xavier, T. C. et al. (2020). Collaborative resource al-

- location for Cloud of Things systems. *Journal of Network and Computer Applications*, 159:102592. DOI: 10.1016/j.jnca.2020.102592.
- Xavier, T. C. *et al.* (2022). Managing Heterogeneous and Time-Sensitive IoT Applications through Collaborative and Energy-Aware Resource Allocation. *ACM Trans. Internet Things*, 3(2). DOI: 10.1145/3488248.
- Zhao, L., Wang, J., *et al.* (2019). Optimal Edge Resource Allocation in IoT-Based Smart Cities. *IEEE Network*, 33(2):30–35. DOI: 10.1109/MNET.2019.1800221.