



SLEEPY-rApp: Delay-aware Sleep Scheduling for Energy Efficiency in MEC-enabled O-RAN

George Levis

Research & Development Department,
Four Dot Infinity (FDI)
Athens, Greece
glevis@fourdotinfinity.com

Sotirios Spantideas

Research & Development Department,
Four Dot Infinity (FDI)
Athens, Greece
sospanti@fourdotinfinity.com

Anastasios Giannopoulos

Department of Ports Management and Shipping,
National and Kapodistrian University of Athens (NKUA)
Athens, Greece
angianno@uoa.gr

Panagiotis Trakadas

Department of Ports Management and Shipping,
National and Kapodistrian University of Athens (NKUA)
Athens, Greece
ptrakadas@uoa.gr

Abstract

This paper presents a novel resource allocation and MEC server deactivation strategy for MEC-enabled O-RAN networks, with the objective of optimizing energy consumption while satisfying strict delay requirements. Our approach leverages an intelligent orchestration application, SLEEPY-rApp, deployed in the SMO layer to dynamically control MEC server activation and request routing. We formulate a joint optimization problem that simultaneously considers computing capacity, end-to-end delay, and energy consumption. To address the NP-hard nature of this problem in real time, we propose a low-complexity heuristic that adapts to varying network conditions and workload patterns. Simulation results indicate that our method significantly reduces energy usage—particularly during peak operating periods—while maintaining the required quality of service. These findings underscore the potential of intelligent orchestration in enhancing the energy efficiency of future MEC-enabled O-RAN systems.

ACM Reference Format:

George Levis, Anastasios Giannopoulos, Sotirios Spantideas, and Panagiotis Trakadas. 2025. SLEEPY-rApp: Delay-aware Sleep Scheduling for Energy Efficiency in MEC-enabled O-RAN. In *2nd International Workshop on MetaOS for the Cloud-Edge-IoT Continuum (MECC '25), March 30–April 3, 2025, Rotterdam, Netherlands*. ACM, New York, NY, USA, Article 112, 6 pages. <https://doi.org/10.1145/3721889.3721927>

1 Introduction

Recent advances in wireless networks have given rise to the Open Radio Access Network (O-RAN) paradigm, which offers enhanced flexibility, interoperability, and scalability through its disaggregated architecture and open interfaces [14]. In this new landscape, traditional network functions are partitioned into distinct modules that

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MECC '25, March 30–April 3, 2025, Rotterdam, Netherlands

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1560-0/2025/03
<https://doi.org/10.1145/3721889.3721927>

can be developed and optimized independently. In particular, managing computing resources at the network edge—implemented via MEC servers—plays a crucial role in supporting latency-sensitive applications and ensuring robust network performance [4].

Integrating MEC servers within an O-RAN framework presents both challenges and opportunities [8]. Whereas conventional architectures rely on tightly coupled systems, the O-RAN approach enables a more distributed and vendor-agnostic deployment, thereby promoting rapid innovation and efficient resource utilization [7]. However, orchestrating these distributed MEC servers demands sophisticated strategies that adapt to dynamic network conditions and diverse service requirements without compromising overall performance [1].

In this paper, we propose a new orchestration framework for MEC-enabled O-RAN networks, featuring SLEEPY-rApp, an intelligent orchestration application deployed in the SMO layer [9, 12]. The primary contributions of our work are as follows:

- (1) We develop a comprehensive system model that captures the key features of an O-RAN environment, with a focus on the distributed management of MEC servers.
- (2) We introduce an innovative orchestration strategy that dynamically configures MEC server deployment to satisfy strict latency and performance requirements in a multi-vendor O-RAN ecosystem.
- (3) We validate the proposed framework through extensive simulations, demonstrating its effectiveness in adapting to real-world network dynamics and service demands.

2 Related Work

Recent research has increasingly focused on improving the energy efficiency of distributed computing infrastructures [11]. Early studies in cloud computing [11, 17] demonstrated that consolidating workloads at the network edge can reduce WAN traffic and lower overall energy consumption. Techniques such as shutting down network devices during periods of low demand have proven effective in achieving energy savings [6]; however, these methods often overlook the stringent latency requirements of modern applications [3].

Subsequent research has proposed advanced resource allocation and routing strategies to enhance energy efficiency in edge computing systems [5, 10]. While some approaches jointly optimize request routing and resource management, many fail to fully address the idle energy consumption of underutilized servers [2]. Other studies have explored clustering and hardware-specific techniques to reduce power usage [13], yet these methods rarely achieve complete server shutdown.

Our work extends these ideas to the MEC-enabled O-RAN context, where the challenges of ultra-low latency and multi-access edge computing necessitate refined resource allocation and server deactivation strategies. In contrast to previous methods, our proposed SLEEPY-rApp framework dynamically manages MEC server activation and request routing to optimize energy consumption without compromising quality of service.

3 System Model

In this section, we present the system model for the MEC-enabled O-RAN framework. The model is structured into four primary layers: the Service Management and Orchestration (SMO) layer, the MEC-enabled O-RAN layer, the Core Network (CN), and the IoT layer. The proposed framework leverages MEC servers co-located with Distributed Units (DUs) and Radio Units (RUs) to perform edge computing, thereby reducing latency and optimizing operational costs.

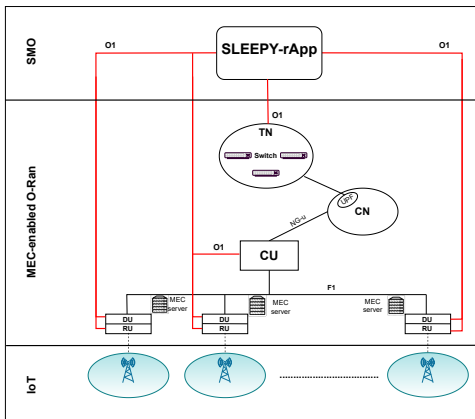


Figure 1: High-level architecture of the MEC-enabled O-RAN system.

3.1 High-Level Architecture Overview

The overall architecture is divided into four primary layers:

- (1) **SMO Layer:** The SMO layer hosts the *SLEEPY-rApp*, an application responsible for network intelligence, monitoring, and optimization. The *SLEEPY-rApp* communicates with the MEC-enabled O-RAN layer through standardized O1 links, which connect both the Centralized Unit (CU) and the Transport Network (TN) to the SMO layer.
- (2) **MEC-enabled O-RAN Layer:** This layer integrates Multi-access Edge Computing (MEC) within an Open RAN (O-RAN) architecture. Its key components include:

- **Centralized Unit (CU):** Performs higher-layer RAN processing, connects to the Core Network via the NG-u interface, and communicates with Distributed Units (DUs) via the F1 interface.
 - **Transport Network (TN):** Comprises the switches and links interconnecting the CU, MEC servers, and other network elements, ensuring efficient data transmission.
 - **MEC Servers:** Co-located with DUs (and, in some cases, RUs), MEC servers execute latency-sensitive computing tasks [16]. Each MEC server h_j is defined by its computing capacity C_j^{\max} (in operations per second) and can be activated or deactivated (with boot time T_j^s) to conserve energy while meeting delay constraints.
 - **Distributed Units (DUs) and Radio Units (RUs):** DUs perform baseband processing and coordinate the activities of multiple RUs, which serve as the primary radio interface for user communications [15].
 - **Interfaces:** The O1 interface connects the SMO layer (and *SLEEPY-rApp*) to the MEC-enabled O-RAN layer, while the F1 interface links the CU with DUs for RAN management.
- (3) **Core Network (CN):** The CN handles user authentication, mobility, and packet routing, including functions such as the User Plane Function (UPF) that manages traffic between the CU and external networks.
 - (4) **IoT Layer:** This layer represents IoT devices (e.g., sensors, smart home devices, industrial applications) that connect to the network via RUs.

3.2 MEC-Enabled O-RAN Components and Their Roles

SMO Layer and SLEEPY-rApp: Within the SMO layer, the *SLEEPY-rApp* continuously monitors network parameters (such as load, latency, and energy consumption) and orchestrates resource allocation decisions. By communicating with the MEC-enabled O-RAN layer via O1 links, the *SLEEPY-rApp* dynamically adjusts the operational states of MEC servers and optimizes the routing of computing requests.

MEC-enabled O-RAN Layer: This layer serves as the core of the computing infrastructure and comprises:

- **CU:** Provides centralized RAN control and interfaces with the CN.
- **TN:** Facilitates connectivity among network components.
- **MEC Servers:** Each MEC server h_j is co-located with a DU (or RU) and offers a computing capacity C_j^{\max} . Similar to traditional edge servers, MEC servers can be activated or deactivated (with a boot time T_j^s) to optimize energy consumption while satisfying delay requirements.
- **DU and RU:** DUs manage baseband processing and coordinate RU operations, with RUs serving as the radio interfaces for user and IoT device connectivity.

3.3 Computing Requests and Resource Allocation

Request Model: Let \mathcal{A} denote the set of latency-sensitive services. Each request for a service $a_k \in \mathcal{A}$ is modeled as a tuple:

$$\langle o_k, V_k^{in}, V_k^{out}, T_k^{max} \rangle,$$

where:

- o_k is the computational workload (in operations),
- V_k^{in} and V_k^{out} are the input and output data sizes (in bytes), respectively,
- T_k^{max} is the maximum allowable delay.

Requests for service a_k arrive at RU i at a rate $\lambda_{i,k}$. MEC servers distribute their processing capacity among services in proportion to the computational load; if a MEC server lacks sufficient capacity, excess requests are rejected in a FIFO manner.

3.4 Orchestrator and Decision Variables

The orchestrator, implemented via the SLEEPY-rApp in the SMO layer, maintains a global view of the network. Its primary functions include:

- Monitoring network performance metrics (e.g., latency and utilization).
- Routing requests from RUs to active MEC servers.
- Dynamically booting or deactivating MEC servers to optimize energy consumption.

The key decision variables are defined as follows:

- χ_j : the operational status of MEC server h_j ($\chi_j = 1$ if active, 0 if inactive).
- $\varphi_{i,k,j}$: the fraction of requests for service a_k routed from RU i to MEC server h_j .
- $\omega_{k,j}$: the fraction of the CPU capacity at MEC server h_j allocated to service a_k .

3.5 Latency and Energy Constraints

Latency Constraints: Each computing request experiences several delay components:

- $\tau_{i,k}^u$: upload delay from the RU to the DU (or source MEC server),
- $\tau_{i,k}^r$: routing delay through the transport network from the source to the destination MEC server,
- $\tau_{i,k}^c$: computation delay at the destination MEC server,
- $\tau_{i,k}^d$: download delay from the destination MEC server (or DU/CU) back to the RU.

The total delay must not exceed the delay budget T_k^{max} for service a_k :

$$\tau_{i,k}^u + \tau_{i,k}^r + \tau_{i,k}^c + \tau_{i,k}^d \leq T_k^{max}, \quad \forall a_k \in \mathcal{A}, \forall \text{RU}_i.$$

Energy Consumption: The energy model comprises:

- **Idle Energy:** Each MEC server h_j consumes idle energy E_j^{idle} when active.
- **Active Energy:** Additional energy E_j is expended per operation executed.
- **Link Energy:** Energy consumption for data transmission is characterized by σ (J/bit) along the transport links.

- **Boot Energy:** When reactivating a MEC server, a boot energy E_j^{boot} is incurred, and the server remains unavailable for T_j^s seconds.

The orchestrator's objective is to minimize overall energy consumption while ensuring that both latency and capacity constraints are satisfied.

3.6 Key Notation Summary

- **SMO Layer:** SLEEPY-rApp – The network intelligence and optimization application.
- **MEC-enabled O-RAN Layer:** $\mathcal{CU}, \mathcal{TN}, \mathcal{M}, \mathcal{DU}, \mathcal{RU}$ – Centralized Unit, Transport Network, MEC servers, Distributed Units, and Radio Units.
- **Core Network:** CN – Core network functions (e.g., UPF).
- **IoT Layer:** \mathcal{I} – IoT devices connected via RUs.
- C_j^{max} : Maximum processing capacity of MEC server h_j .
- $\lambda_{i,k}$: Arrival rate of requests for service a_k at RU i .
- $\varphi_{i,k,j}$: Fraction of requests for a_k routed from RU i to MEC server h_j .
- χ_j : Operational status of MEC server h_j (1 if active, 0 if inactive).
- $\omega_{k,j}$: Fraction of CPU allocated to service a_k at MEC server h_j .
- $\tau_{i,k}^u, \tau_{i,k}^r, \tau_{i,k}^c, \tau_{i,k}^d$: Delay components for upload, routing, computation, and download, respectively.
- $E_j^{idle}, E_j, E_j^{boot}$: Energy consumption parameters for MEC server h_j .

In summary, the proposed system model seeks to optimize the energy consumption of a MEC-enabled O-RAN by dynamically managing MEC server activation and request routing while ensuring that stringent latency and capacity constraints are met. The subsequent sections detail the problem formulation and the optimization algorithms employed by the orchestrator.

4 Problem Formulation

In our MEC-enabled O-RAN framework, the orchestrator's objective is to minimize the overall operational cost—represented here in terms of energy consumption—while ensuring that all computing requests are processed within their respective delay budgets. The decision variables are defined as follows:

- $\chi_j \in \{0, 1\}$: the operational status of MEC server h_j (with $\chi_j = 1$ if active and $\chi_j = 0$ if deactivated),
- $\varphi_{i,k,j} \in [0, 1]$: the fraction of requests for service a_k arriving at RU i that are routed to MEC server h_j ,
- $\omega_{k,j} \in [0, 1]$: the fraction of the computing capacity at MEC server h_j allocated to service a_k .

We denote the set of services by \mathcal{A} , the set of RUs by \mathcal{R} , and the set of MEC servers by \mathcal{M} . For a given service a_k , each request is modeled as a tuple

$$\langle o_k, V_k^{in}, V_k^{out}, T_k^{max} \rangle,$$

where o_k is the computational workload (in operations), V_k^{in} and V_k^{out} denote the input and output data sizes (in bytes), and T_k^{max} is the maximum tolerable delay.

4.1 Capacity and Resource Constraints

For each MEC server h_j , the total computing load assigned to service a_k from all RUs is given by

$$O_{k,j} = \sum_{i \in \mathcal{R}} \varphi_{i,k,j} o_k \lambda_{i,k},$$

where $\lambda_{i,k}$ is the arrival rate of requests for service a_k at RU i . The overall load at MEC server h_j is then

$$O_j = \sum_{a_k \in \mathcal{A}} O_{k,j}.$$

To ensure that the processing capacity of h_j is not exceeded, the following constraint must be satisfied:

$$O_j \leq C_j^{\max}, \quad \forall h_j \in \mathcal{M}. \quad (1)$$

Furthermore, to allocate the computing capacity among services proportionally to their load, we require that

$$\omega_{k,j} \geq \frac{\sum_{i \in \mathcal{R}} \varphi_{i,k,j} o_k \lambda_{i,k}}{O_j}, \quad \forall h_j \in \mathcal{M}, \forall a_k \in \mathcal{A}. \quad (2)$$

4.2 Delay Constraints

Each computing request experiences a combination of delays:

- $\tau_{i,k}^u$: Upload delay from RU i to the associated DU/MEC server,
- $\tau_{i,k}^r$: Routing delay within the transport network,
- $\tau_{i,k}^c$: Computation delay at MEC server h_j ,
- $\tau_{i,k}^d$: Download delay from the MEC server (or DU/CU) back to RU i .

For a request served by MEC server h_j , the computation delay is inversely related to the allocated capacity, and can be modeled as

$$\tau_{i,k}^c = \frac{o_k}{\omega_{k,j} C_j^{\max}}, \quad \text{for } \varphi_{i,k,j} > 0. \quad (3)$$

Thus, the overall delay for a request originating at RU i for service a_k must satisfy:

$$\tau_{i,k}^u + \tau_{i,k}^r + \tau_{i,k}^c + \tau_{i,k}^d \leq T_k^{\max}, \quad \forall a_k \in \mathcal{A}, \forall i \in \mathcal{R}. \quad (4)$$

4.3 Energy Consumption Model

The total energy consumption of the MEC-enabled O-RAN system comprises:

- **MEC Server Energy:** When active, MEC server h_j consumes an idle energy E_j^{idle} and additional energy E_j per operation executed. Thus, the energy consumption of h_j is modeled as:

$$E_j^{\text{total}} = \chi_j (E_j^{\text{idle}} + O_j E_j).$$

- **Transmission Energy:** Energy consumed for data transmission over the transport links is proportional to the data volume and characterized by a per-bit energy cost σ . (A detailed link model is deferred to future work.)
- **Boot Energy:** When a MEC server is reactivated, it incurs a boot energy E_j^{boot} and remains unavailable for T_j^s seconds.

4.4 Optimization Objective

The primary objective is to minimize the overall energy consumption while satisfying the capacity and delay constraints. Mathematically, the optimization problem is formulated as:

$$\begin{aligned} \min_{\varphi, \omega, \chi} \quad & \sum_{h_j \in \mathcal{M}} \chi_j (E_j^{\text{idle}} + O_j E_j) + \sum_{\text{links}} \sigma \cdot (\text{Data Volume}) \\ \text{subject to} \quad & O_j \leq C_j^{\max}, \quad \forall h_j \in \mathcal{M}, \\ & \omega_{k,j} \geq \frac{\sum_{i \in \mathcal{R}} \varphi_{i,k,j} o_k \lambda_{i,k}}{O_j}, \quad \forall h_j \in \mathcal{M}, \forall a_k \in \mathcal{A}, \\ & \tau_{i,k}^u + \tau_{i,k}^r + \frac{o_k}{\omega_{k,j} C_j^{\max}} + \tau_{i,k}^d \leq T_k^{\max}, \\ & \quad \forall a_k \in \mathcal{A}, \forall i \in \mathcal{R} \text{ with } \varphi_{i,k,j} > 0, \\ & \sum_{h_j \in \mathcal{M}} \varphi_{i,k,j} = 1, \quad \forall a_k \in \mathcal{A}, \forall i \in \mathcal{R}, \\ & \varphi_{i,k,j} \leq \chi_j, \quad \forall a_k \in \mathcal{A}, \forall i \in \mathcal{R}, \forall h_j \in \mathcal{M}, \\ & \varphi_{i,k,j} \in [0, 1], \quad \forall a_k \in \mathcal{A}, \forall i \in \mathcal{R}, \forall h_j \in \mathcal{M}, \\ & \omega_{k,j} \in [0, 1], \quad \forall a_k \in \mathcal{A}, \forall h_j \in \mathcal{M}, \\ & \chi_j \in \{0, 1\}, \quad \forall h_j \in \mathcal{M}. \end{aligned} \quad (P1)$$

Constraint (1) (implicitly defined via $O_j \leq C_j^{\max}$) ensures that the total computing load assigned to each MEC server does not exceed its capacity. Constraint (2) guarantees proportional allocation of processing resources, while constraint (4) ensures that the end-to-end delay for each request remains within the service's maximum tolerable delay. The condition $\varphi_{i,k,j} \leq \chi_j$ enforces that requests are assigned only to active MEC servers.

The overall optimization problem in (P1) is a Mixed-Integer Linear Program (MILP) and is NP-hard. In practice, heuristic algorithms (such as the MEC-ON approach described in Section 5) are employed to obtain near-optimal solutions in real time.

In summary, the formulation seeks an optimal routing and resource allocation configuration that minimizes energy consumption while satisfying both capacity and delay constraints in the MEC-enabled O-RAN environment. Furthermore, the problem formulation provides a rigorous mathematical foundation that enables precise evaluation of the trade-offs between energy efficiency and service quality. The model captures the interplay among computing capacity, network delay, and energy costs, which are critical for designing adaptive and sustainable wireless networks. In particular, the constraints ensure that no MEC server is overloaded and that all user requests are processed within the allowable time limits, thereby guaranteeing both operational efficiency and quality of service. This comprehensive formulation facilitates the development of efficient heuristics that can operate in real time, making it highly relevant for future wireless network deployments.

5 SLEEPY-rApp Optimization Algorithms

This section details the algorithms executed by the SLEEPY-rApp (located in the SMO layer) for optimizing energy consumption by dynamically managing MEC server activation and request routing in the MEC-enabled O-RAN system. These algorithms operate in discrete time slots, periodically updating the network configuration.

Algorithm 1 SLEEPY-rApp Initialization Procedure

Require: Current network load λ

- 1: **for** each MEC server j in \mathcal{M} **do**
- 2: Set $\chi_j \leftarrow 1$ \triangleright Activate all MEC servers initially.
- 3: **end for**
- 4: **for** each RU i in \mathcal{R} **and for each** service $a_k \in \mathcal{A}$ **do**
- 5: **if** RU i is associated with a local MEC server **then**
- 6: **if** $\lambda_{i,k} \cdot o_k \leq C_i^{\max} - O_i$ **then**
- 7: Set $\varphi_{i,k,i} \leftarrow 1$
- 8: **else**
- 9: Set $\varphi_{i,k,i} \leftarrow \frac{C_i^{\max} - O_i}{\lambda_{i,k} \cdot o_k}$
- 10: Determine candidate MEC servers reachable within T_k^{\max} and assign the remaining load accordingly
- 11: **end if**
- 12: **else**
- 13: Route the entire load from RU i to the nearest active MEC server that satisfies the delay constraint
- 14: **end if**
- 15: **end for**
- 16: Update ω based on the initial routing φ
- 17: **return** φ , ω , and χ

Algorithm 2 Delay Constraint Handler

Require: Current allocation variables φ , ω , χ , load λ , and maximum iteration limit M

- 1: Initialize counter $c \leftarrow 0$
- 2: **while** a MEC server v is identified with delay violations **and** $c < M$ **do**
- 3: Identify the service a_k causing the delay violation at RU i
- 4: Determine the candidate set $\mathcal{M}_{\text{cand}}$ of MEC servers reachable within the delay limit
- 5: Sort $\mathcal{M}_{\text{cand}}$ in ascending order based on the available capacity gap ($C^{\max} - O$)
- 6: Select the candidate server m with the smallest capacity gap
- 7: **if** $\lambda_{i,k} \cdot o_k \cdot \varphi_{i,k,m} \geq C_m^{\max} - O_m$ **then**
- 8: Select the closest inactive MEC server to RU i and set its status: $\chi_m \leftarrow 1$
- 9: Reassign the load from MEC server v to the newly activated server: update $\varphi_{i,k,m}$ and set $\varphi_{i,k,v} \leftarrow 0$
- 10: **end if**
- 11: Increment counter $c \leftarrow c + 1$
- 12: **end while**
- 13: **return** the updated values of φ , ω , and χ

The SLEEPY-rApp Optimization Algorithms operate in a time-slotted fashion. At the beginning of each time slot $t \in \{1, 2, \dots, T\}$, the SLEEPY-rApp executes the Energy Optimization Heuristic (Algorithm 3) using the maximum expected load λ_t as input. Initially, the allocation is established using the Initialization Procedure (Algorithm 1). Subsequently, MEC servers are evaluated in descending order of their idle energy cost; for each candidate MEC server, the load is redistributed to neighboring servers that satisfy the delay constraints and have sufficient available capacity. The energy savings from deactivating a MEC server are compared with the

Algorithm 3 SLEEPY-rApp Energy Optimization Heuristic

Require: Current network load λ and performance metrics

- 1: Obtain initial values for φ , ω , and χ using Algorithm 1
- 2: Sort the set of MEC servers \mathcal{M} in descending order by their idle energy cost
- 3: Sort the set of services \mathcal{A} in ascending order by their delay budget T_k^{\max}
- 4: **for** each MEC server j in \mathcal{M} **do**
- 5: Compute the current energy cost if active:

$$E_{\text{on}} \leftarrow \chi_j \times (E_j^{\text{idle}} + O_j \cdot E_j)$$
- 6: Initialize the tentative energy cost $E_{\text{off}} \leftarrow 0$
- 7: Tentatively set $\chi_j \leftarrow 0$ \triangleright Attempt to deactivate MEC server j .
- 8: **for** each service $a_k \in \mathcal{A}$ **do**
- 9: **for** each RU i in \mathcal{R} **such that** $\varphi_{i,k,j} > 0$ **do**
- 10: Determine the candidate set $\mathcal{M}_{\text{cand}}$ of MEC servers reachable within T_k^{\max} , sorted in increasing order by estimated transmission cost
- 11: **for** each candidate server m in $\mathcal{M}_{\text{cand}}$ **do**
- 12: **if** the available capacity (both link and server) at m is sufficient to accommodate the load $\varphi_{i,k,j}$ **then**
- 13: Transfer the entire load: set $\varphi_{i,k,m} \leftarrow \varphi_{i,k,m} + \varphi_{i,k,j}$ and set $\varphi_{i,k,j} \leftarrow 0$
- 14: **else**
- 15: Transfer as much load as possible:

$$\varphi_{i,k,m} \leftarrow \varphi_{i,k,m} + \Delta, \quad \varphi_{i,k,j} \leftarrow \varphi_{i,k,j} - \Delta,$$
 where Δ is determined by the available capacity at m and the link capacity
- 16: **end if**
- 17: Update E_{off} based on the new allocation
- 18: **end for**
- 19: **end for**
- 20: **end for**
- 21: **if** $E_{\text{off}} \geq E_{\text{on}}$ **or** any load remains unassigned from MEC server j **then**
- 22: Revert the allocation changes and set $\chi_j \leftarrow 1$ \triangleright Keep MEC server j active.
- 23: **end if**
- 24: **end for**
- 25: Recompute ω for all active MEC servers based on the new load distribution
- 26: **if** any delay constraints are violated **then**
- 27: Invoke the Delay Constraint Handler (Algorithm 2)
- 28: **end if**
- 29: **return** the updated values of φ , ω , and χ

additional energy incurred by rerouting the load. If the savings are insufficient or if some load cannot be reassigned, the MEC server remains active.

After completing the primary optimization loop, the algorithm recalculates the CPU allocation ω for all active MEC servers. If any delay constraints are violated due to the reallocation, the Delay Constraint Handler (Algorithm 2) is invoked to reassign the affected load or to activate additional MEC servers as necessary.

6 Results and Conclusions

Figure 2 illustrates the performance of our proposed SLEEPY-rApp against baseline approaches (Default Operation with continuously active servers and Random Policy with non-optimized task routing) across two operational periods.

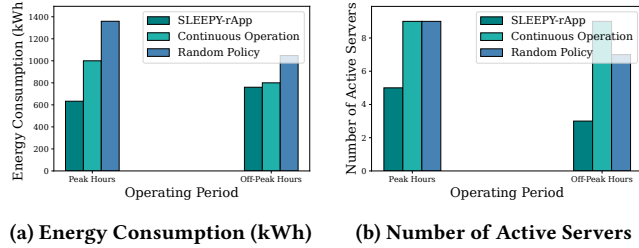


Figure 2: Energy consumption and active server comparison across policies.

During Peak Hours, SLEEPY-rApp achieves significant energy efficiency, consuming 633 kWh—a 36.7% reduction compared to the 1000 kWh under Default Operation. This efficiency persists in Off-Peak Hours, where SLEEPY-rApp (760 kWh) outperforms both Default Operation (800 kWh) and Random Policy (1048 kWh). Concurrently, SLEEPY-rApp demonstrates optimal resource utilization, requiring only 5 and 3 active servers during Peak and Off-Peak Hours, respectively, versus 9 and 8 for Default Operation.

Further analysis reveals that SLEEPY-rApp maintains service quality despite reduced resource usage, with a request satisfaction rate exceeding 99.9%. The heuristic’s decision-making process proves particularly effective during system load transitions, where appropriate server activation/deactivation strategies are crucial. While Random Policy attempts to reduce active servers (achieving 7 during Off-Peak), its non-optimized routing decisions lead to increased energy consumption, demonstrating that server count reduction alone is insufficient without intelligent workload distribution. The simulation also indicates SLEEPY-rApp’s approach differs fundamentally from conventional load-based strategies, as it considers both request characteristics and server energy profiles when making orchestration decisions, resulting in more efficient resource utilization particularly under heterogeneous workloads.

6.1 Conclusions

In this paper, we proposed a novel resource allocation and MEC server deactivation strategy implemented via the SLEEPY-rApp in the SMO layer. Our contributions include:

- The development of an optimization formulation jointly considering capacity, delay, and energy constraints.
- The design of a low-complexity heuristic for dynamic MEC server activation and request routing.
- An empirical demonstration of 36.7% energy reduction during Peak Hours through intelligent orchestration.

These findings confirm that intelligent orchestration through SLEEPY-rApp can substantially enhance energy efficiency and resource utilization in MEC-enabled O-RAN environments. Future extensions of the present work is necessary to further investigate the scalability and complexity of SLEEPY-rApp against baseline

MEC sleep scheduling methods. Additional future work will also focus on refining the heuristic, incorporating adaptive load prediction, and integrating additional performance metrics to improve network efficiency and quality of service.

Acknowledgments

This work has been supported by the EU Horizon programme under grant 101070177 (ICOS project), and co-supported by UNITY-6G project under grant 101192650.

References

- [1] Esmail Amiri and Toktam Mahmoodi. 2024. O-RAN and MEC Integration and Orchestration: An Optimization Approach. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1–6.
- [2] Robert Basnadjian, Florian Niedermeier, and Hermann De Meer. 2012. Modelling and analysing the power consumption of idle servers. In *2012 Sustainable Internet and ICT for Sustainability (SustainIT)*. IEEE, 1–9.
- [3] Xing Chen and Guizhong Liu. 2021. Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks. *IEEE Internet of Things Journal* 8, 13 (2021), 10843–10856.
- [4] I Chih-Lin, Slawomir Kuklinski, and Tao Chen. 2020. A perspective of O-RAN integration with MEC, SON, and network slicing in the 5G era. *IEEE Network* 34, 6 (2020), 3–4.
- [5] Subhash Dhar Dwivedi and Praveen Kaushik. 2012. Energy efficient routing algorithm with sleep scheduling in wireless sensor network. *International Journal of Computer Science and Information Technologies* 3, 3 (2012), 4350–4353.
- [6] Mingjie Feng, Shiwen Mao, and Tao Jiang. 2017. Base station ON-OFF switching in 5G wireless networks: Approaches and challenges. *IEEE Wireless Communications* 24, 4 (2017), 46–54.
- [7] Anastasios Giannopoulos, Ilias Paralikas, Sotirios Spantideas, and Panagiotis Trakadas. 2024. COOLER: Cooperative Computation Offloading in Edge-Cloud Continuum Under Latency Constraints via Multi-Agent Deep Reinforcement Learning. In *2024 International Conference on Intelligent Computing, Communication, Networking and Services (ICCN)*. IEEE, 9–16.
- [8] Anastasios Giannopoulos, Ilias Paralikas, Sotirios Spantideas, and Panagiotis Trakadas. 2024. HOODIE: Hybrid Computation Offloading via Distributed Deep Reinforcement Learning in Delay-Aware Cloud-Edge Continuum. *IEEE Open Journal of the Communications Society* (2024).
- [9] Anastasios Giannopoulos, Sotirios Spantideas, George Levis, Kalafatis Alexandros, and Panagiotis Trakadas. 2025. COMIX: Generalized Conflict Management in O-RAN xApps-Architecture, Workflow, and a Power Control case. *arXiv preprint arXiv:2501.14619* (2025).
- [10] Blas Gómez, Suzan Bayhan, Estefanía Coronado, José Villalón, and Antonio Garrido. 2024. LESS-ON: Load-aware edge server shutdown for energy saving in cellular networks. *Computer Networks* 252 (2024), 110675.
- [11] Yu-Lin Jiang, Ya-Shu Chen, Su-Wei Yang, and Chia-Hsueh Wu. 2018. Energy-efficient task offloading for time-sensitive applications in fog computing. *IEEE Systems Journal* 13, 3 (2018), 2930–2941.
- [12] Simona Marinova and Alberto Leon-Garcia. 2024. Intelligent O-RAN Beyond 5G: Architecture, Use Cases, Challenges, and Opportunities. *IEEE Access* 12 (2024), 27088–27114.
- [13] Vinicius Petrucci, Enrique V Carrera, Orlando Loques, Julius CB Leite, and Daniel Mossé. 2011. Optimized management of power and performance for virtualized heterogeneous server clusters. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 23–32.
- [14] Michele Polese, Leonardo Bonati, Salvatore D’oro, Stefano Basagni, and Tommaso Melodia. 2023. Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1376–1411.
- [15] Sotirios Spantideas, Anastasios Giannopoulos, Marta Amor Cambeiro, Oscar Trullols-Cruces, Eneko Atxutegi, and Panagiotis Trakadas. 2023. Intelligent Mission Critical Services over Beyond 5G Networks: Control Loop and Proactive Overload Detection. In *2023 International Conference on Smart Applications, Communications and Networking (SmartNets)*. IEEE, 1–6.
- [16] Sotirios T Spantideas, Anastasios E Giannopoulos, and Panagiotis Trakadas. 2024. Smart Mission Critical Service Management: Architecture, Deployment Options, and Experimental Results. *IEEE Transactions on Network and Service Management* (2024).
- [17] Lichao Yang, Heli Zhang, Ming Li, Jun Guo, and Hong Ji. 2018. Mobile edge computing empowered energy efficient task offloading in 5G. *IEEE Transactions on Vehicular Technology* 67, 7 (2018), 6398–6409.