ODESA: Load-Dependent Edge Server Activation for Lower Energy Footprint

Blas Gómez*, Suzan Bayhan†, Estefanía Coronado*‡, José Villalón* and Antonio Garrido*
*High-Performance Networks and Architectures, Universidad de Castilla-La Mancha, Albacete, Spain
Email: {Blas.Gomez, Estefania.Coronado, JoseMiguel.Villalon, Antonio.Garrido}@uclm.es
†University of Twente, Enschede, The Netherlands. Email: s.bayhan@utwente.nl
‡I2CAT Foundation, Barcelona, Spain. Email: estefania.coronado@i2cat.net

***Abstract*—5G networks promise to deliver an unprecedented performance that can accommodate novel services with stringent Quality of Service (QoS) requirements that were not possible with previous generations of networks. Edge Computing plays a fundamental role by providing computing resources closer to the user, reducing round trip times. However, the deployment of edge computing poses new challenges, including the energy footprint of a potentially large number of servers. Even in idle state, these servers consume a significant amount of energy, which is worth considering for reducing their energy footprint. In cloud computing environments, server shutdown during low-demand periods is a typical energy-saving strategy. However, this approach has received less attention in edge computing due to the strict latency requirements of its use cases. This work presents ODESA, an edge server shutdown strategy with polynomial time complexity that provides a trade-off between the idle energy consumption of the edge servers and energy consumed by the backhaul to route requests to active servers. Our numerical investigation shows that thanks to the reduction in idle energy consumption, ODESA reduces the total consumption by 42% over the common always-on approach during low-demand periods and 11% over 24 hours, all while meeting the latency requirements of the applications.**

***Index Terms*—Edge Computing, Energy Consumption, Energy Efficiency, 5G, Wireless Networks, Green Communications.**

I. Introduction

Wireless networks have recently made great strides, offering unprecedented features such as sub-millisecond latencies, and multi-Gbps data rates that will take Quality of Service (QoS) and user experience to previously unknown levels. Consequently, Mobile Network Operators (MNOs) and third parties can offer novel applications and services whose strict QoS requirements cannot be satisfied by previous network generations, such as augmented reality and medical robotics. In this context, edge computing plays a key role by providing computing resources closer to the user, reducing the delays experienced in numerous use cases. However, global warming and the ongoing energy crisis raise concerns about the energy footprint of communications and computing infrastructures, with different studies such as [1], [2] calling for network infrastructure to be designed and operated with energy footprint and sustainability aspects in mind. The European Union expects edge systems to account for 12% of the computing infrastructure's Carbon Dioxide Equivalent (CO_2e) emissions by 2025 [3], highlighting the increasing need for more sustainable computing infrastructures with lower energy footprint.

A large body of work on cloud computing such as [4]–[8] report significant energy savings enabled by shutting down servers during low-demand periods, e.g., when servers are not performing any user operation [9], as energy consumption during these idle periods is significant. Similarly, edge computing involves the potential deployment of a multitude of servers whose capacity is scaled to meet the demand at peak hours, leaving a significant fraction of resources idle during low-demand periods. However, server shutdown strategies have received less attention in edge computing due to the strict latency requirements of most of its use cases. Since edge computing's low latencies are possible thanks to the proximity of the edge servers to the users, the impact of shutdown strategies in the QoS of edge computing applications is unclear. Although studies such as [10], [11] leverage this approach for edge infrastructures, an assessment of the effect they can have on deadline satisfaction is missing. Moreover, these strategies neglect energy consumption due to routing requests from offline servers to active ones through backhaul links.

This work presents ODESA, an edge server shutdown algorithm with a polynomial time complexity that aims at reducing the energy footprint of a cellular network's co-located edge computing infrastructure by selectively shutting down edge servers when latency requirements of the served applications can be fulfilled with the existing active edge servers. Introduced together with the network orchestrator, ODESA provides a trade-off between the idle energy consumption of the servers and the energy consumed by the backhaul derived from routing the requests to the active servers while maintaining acceptable QoS. In particular, this paper addresses the following questions: (i) How much energy can be saved by the proposed scheme over two baseline schemes, namely *Always-on* (i.e., where all edge servers are active) and *Threshold-based* server shutdown (i.e., in which servers are switched-off when their load is below a certain threshold)? (ii) How does the proposed scheme perform in terms of deadline satisfaction in a delay-constrained scenario?

The rest of the paper is organized as follows: Sec. II provides an overview of the related work while Sec. III presents the system model. Sec. IV introduces our energy-saving approach, and Sec. V provides and discusses the performance evaluation results. Finally, Sec. VI concludes the paper with a list of future research directions.

II. Related Work

This section reviews the most relevant work on energy-efficient resource management in edge computing systems, namely the studies [10]–[14] that propose several approaches to decrease the energy consumption of edge servers.

In [12], the authors propose server clustering to save energy using Dynamic Voltage Frequency Scaling (DVFS) CPUs. However, they never turn off the servers. Similarly, the authors [13] assume that each edge server has multiple CPUs that can be independently turned off based on two thresholds. This makes these two approaches constrained to specific hardware, which limits their applicability. Contrary to these studies, ODESA is hardware-agnostic. Both [12], [13] assign tasks to servers to leave as many servers idle as possible. Then, they rely on specific hardware to decrease idle energy consumption. ODESA shuts down the servers, reducing idle energy to zero without harming QoS and working with any kind of hardware. The authors of [14] introduce a location-based load prediction algorithm that uses historical data from an edge server and the neighboring edge servers. While this study acknowledges the potential benefits of using this algorithm to implement server shutdown strategies, their focus is on load prediction, and they do not quantify the mentioned benefits.

The approach proposed in [10] enables edge servers to sleep when the load falls below a threshold, but it does not consider the energy consumption of the backhaul needed to route incoming requests to active servers. As per [1], the transmission of information also significantly affects energy consumption to the point that the energy consumed by the backhaul might outweigh the energy savings from server shutdown. Their study also lacks an evaluation of the impact in latency-constrained services. In contrast, ODESA finds a compromise between the energy saved by shutting down servers and the energy consumed by routing the requests through the backhaul to the remaining active servers. Moreover, we study the impact of ODESA on latency-constrained scenarios. Another edge server shutdown study is presented in [11]. The authors propose a game-theory approach to optimize costs (instead of energy consumption) by shutting down routers and servers. Their study, which assumes fixed running costs, shows latencies that are unsuitable for latency-constrained use cases, such as vehicular safety. In contrast, our approach focuses on applications with stringent delay requirements and our evaluation models energy consumption dynamically. Moreover, ODESA considers an architecture that places edge servers closer to the user.

In summary, our contributions are three-fold: (i) proposing a hardware-agnostic edge server shutdown strategy that (ii) considers the backhaul infrastructure and (iii) evaluating its impact on latency-constrained services in a realistic scenario.

III. System Model

This section presents the considered system model with its components as illustrated in Fig. 1.

Communication network: We consider a 5G Radio Access Network (RAN) with the set of Base Stations (BSs) denoted by $\mathcal{B} = \{BS_1, \ldots, BS_N\}$. The BSs are interconnected through a given set of links $\mathcal{L}$, which establishes connectivity to the Internet and the management plane through the backhaul network. We denote this connectivity graph by $\mathcal{G} = \langle \mathcal{B}, \mathcal{L} \rangle$. Each link ℓ is identified by its source and destination BSs and associated link capacity denoted by $\alpha_{i,j}$. We assume that the link capacity from BS_i to BS_j is equal to the capacity in the reverse direction. Consequently, we represent each link as a three tuple: $\ell_{i,j} = \langle BS_i, BS_j, \alpha_{i,j} \rangle$. For the sake of simplicity, we assume a fixed routing algorithm between two BSs, and the associated routing matrix is denoted by Γ, which consists of paths between two BSs.

Edge computing infrastructure: The cellular network has a co-located set of edge servers denoted by $\mathcal{H} = \{h_m\}$. Without loss of generality, we assume that each edge server is associated with a BS. Since MNOs might not prefer deploying edge servers at every BS, we indicate by a binary variable e_i whether BS_i hosts an edge server. We assume that a particular edge server $h_m \in \mathcal{H}$ can accommodate a finite set $\mathcal{A}$ of computation services, such as accident detection or traffic control in vehicular scenarios. We denote the computing capacity of h_m by C_m^{max} in operations per second. To summarize, an edge server is defined by the following tuple $\langle BS_i, C_i^{max} \rangle$. When we refer to the edge server at BS_i, we denote it by h_i. An edge server located at BS_i might serve computing requests originating from other BSs, e.g., a BS without an edge server.

Computing requests: Each computing request is characterized by a four tuple $\langle o_k, V_k^{in}, V_k^{out}, T_k^{max} \rangle$ where o_k is the workload, V_k^{in} is the size of the input data, V_k^{out} is the size of the output, and T_k^{max} is the delay budget for this task, which indicates the maximum time before a request's result has to be delivered to the user[1]. The total delay of a request is given by $T_{i,k}^u + T_{i,k}^r + T_{i,k}^c + T_{i,k}^o + T_{i,k}^d \leq T_k^{max}$, where $T_{i,k}^u$ is the delay to upload a request to a_k through BS_i, $T_{i,k}^r$ is the delay of routing the request through the backhaul, $T_{i,k}^c$ is the computing delay, $T_{i,k}^o$ is the delay to route the output back to the corresponding BS and $T_{i,k}^d$ is the delay of downloading the output from the BS. Each request made to $a_k \in \mathcal{A}$ results in a computing workload of o_k in terms of number of operations required to accomplish the task. Requests to BS_i for a particular service a_k arrive at a service request rate $\lambda_{i,k}$, which depends on the number of users and characteristics of the service [15]. Consequently, the workload produced by the requests received at BS_i due to a_k is given by $o_k \lambda_{i,k}$. We assume that the computing capacity of h_i is proportionally split between the set $\mathcal{A}$ that it hosts according to the number of operations to be executed by each service in $\mathcal{A}$ [16].

Operation of the edge orchestrator: The edge computing infrastructure is managed by an orchestrator located at the Service Management and Orchestration (SMO) layer, which also encloses the government of the RAN and the transport network.[2] It manages the capacity of the servers and selects

[1]For the sake of simplicity, we implicitly assume that the result of the computation is delivered to the same user submitting the request. More sophisticated patterns, e.g., one-to-many, can also be considered.

[2]Federation across MNOs is out of the scope of this work.

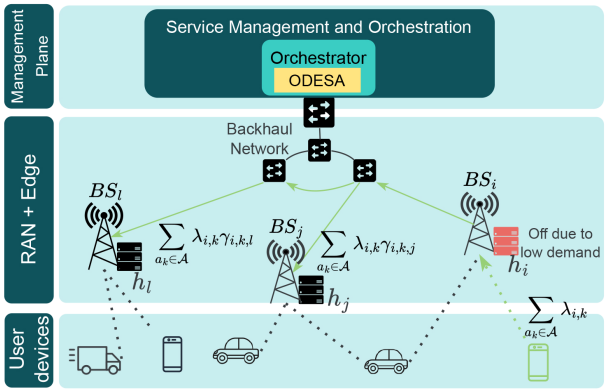


Fig. 1. A cellular network with edge servers. The edge orchestrator uses ODESA to determine an efficient resource allocation and determines where to execute the requests, how to route the requests to the assigned edge servers, and which servers to switch off to minimize the network's energy consumption.

TABLE I
SUMMARY OF KEY NOTATION.

Symbol	Definition
$\mathcal{B}$, N	The set of BSs and the number of BSs
$\mathcal{L}$, $\mathcal{G}$, Γ	The set of links, the connectivity graph and the routing matrix
$\ell_{i,j}$, $\alpha_{i,j}$	The link connecting BS_i and BS_j and its capacity
$\mathcal{H}$, h_m	The set of edge servers and a particular server
C_m^{max}	The maximum capacity of h_m in operations per second
e_i	Variable indicating the presence of an edge server at BS_i
$\mathcal{A}$, a_k	The set of services and a service instance
V_k^{in}, V_k^{out}, o_k	Size of the input and output and the workload of a req. to a_k
$\lambda_{i,k}$	Arrival rate of requests from BS_i to service a_k
$\gamma_{i,k,j}$	Fraction of $\lambda_{i,k}$ to be routed to h_j
η_i	Status (on/off) of edge server $h_i \in \mathcal{H}$
$p_{i,j}^{o,p}$	Indicator of $\ell_{o,p}$'s presence in the path from BS_i to BS_j
T_k^{max}	Delay budget of service $a_k \in \mathcal{A}$
$T_{j,k}^c$	Time to compute a response
$T_{i,k}^u$, $T_{i,k}^d$	Time to send a req. and to download it in the RAN
$T_{i,k}^r$, $T_{i,k}^o$	Request and output routing times for $h_m \in \mathcal{H}$.
E_m^{idle}, E_m	Energy of h_m when idle and energy per operation
O_m	Total workload in operations per second of $h_m \in \mathcal{H}$

the appropriate one to serve the requests. For energy-saving purposes, the orchestrator can shut down edge servers. When the edge server $h_i \in \mathcal{H}$ attached to $BS_i \in \mathcal{B}$ is off, computing requests received by BS_i are redirected to another active edge server $h_j \in \mathcal{H}$. At a given time, the orchestrator can decide on the following parameters to optimize the QoS or save energy:

- Fraction of requests of a_k received by BS_i to be computed by the edge server of BS_j denoted by $\gamma_{i,k,j} \in [0, 1]$.
- Status of each edge server denoted by $\eta_i \in \{0, 1\}$ where $\eta_i = 0$ means that h_i will be switched off.

In the next section, we introduce ODESA, our approach to finding a combination of these values that reduces the energy footprint while meeting the deadlines of the requests.

Energy consumption: An edge server $h_m \in \mathcal{H}$ has a specific baseline energy consumption (E_m^{idle}) when it is not performing user operations. Each operation carried out by h_m adds a certain energy consumption on top of E_m^{idle} denoted as E_m [17]. We denote the total number of operations of a server h_m by O_m, which is given by the fraction of requests routed to h_m for each service ($\gamma_{i,k,m}\lambda_{i,k}$) and the workload of each request (o_k). Then, the energy consumed by the set $\mathcal{H}$ is given by:

$$\sum_{h_m \in \mathcal{H}} \eta_m \left(E_m^{idle} + O_m E_m\right). \tag{1}$$

Routing requests to another BS's edge server also requires energy. Let $\sigma_{o,p}$ denote the energy consumed per bit transmitted over a link $\ell_{o,p}$. To reach the designated edge server, a request might need to be routed through multiple hops, being the total $\sigma_{i,j}$ of the path given by $\sum_{\ell_{o,p} \in \mathcal{L}} \sigma_{o,p} p_{o,p}^{i,j}$ where $p_{o,p}^{i,j}$ indicates whether a link is in the shortest path between BS_i and BS_j. The volume of data traversing a link $\ell_{o,p}$, denoted as $V_{o,p}$, determines total energy consumption. $V_{o,p}$ is given by $\left(V_k^{in}+V_k^{out}\right) \lambda_{i,k}\gamma_{i,k,j} p_{i,j}^{o,p}$ where $\lambda_{i,k}\gamma_{i,k,j} p_{i,j}^{o,p}$ is the number of requests routed from any BS_i to any BS_j whose shortest path goes through $\ell_{o,p}$ and $V_k^{in}+V_k^{out}$ is the input and output data generated by each request. Thus, we can denote the total energy consumed by all links $\mathcal{L}$ to route the requests as follows:

$$\sum_{\ell_{o,p} \in \mathcal{L}} \sigma_{o,p} V_{o,p}. \tag{2}$$

IV. ODESA: LOAD-DEPENDENT EDGE SERVER ACTIVATION FOR LOWER ENERGY FOOTPRINT

The problem of resource allocation at the edge is known to be typically NP-hard [18]. In our system model, assuming that there is only one service ($|\mathcal{A}| = 1$) and that the link capacities are sufficiently large (i.e., $\alpha_{i,j} = \infty \; \forall \ell_{i,j}$), this problem could be formulated as a Capacitated Facility Location Problem (CFLP). CFLP has two stages: first, determining which subset of edge servers to turn on (facilities to open), and second, deciding what fraction of requests is routed from each BS to each edge server (assigning customer demand to facilities). Because CFLP is NP-hard [19], we propose a lower-complexity heuristic for determining $\gamma_{i,k,j}$ and η_i for all i, j, k.

Our heuristic, whose steps are listed in Alg.1, is based on the DROP procedure [20], a greedy algorithm used to solve CFLP. Our adaptation of DROP begins by initializing the vector η with all the edge servers active. Then, the matrix γ is initialized to route requests to the closest server (the local one for BSs with a co-hosted server, i.e., $i = j$), where they are attended in a First In First Out (FIFO) manner. If the closest server cannot handle all the requests from a BS, the remaining ones are routed to the closest server. After checking all reachable servers, the remaining requests are rejected.

After the initialization, the algorithm loops through all the servers. At each iteration, the edge server h_j with the highest E^{idle} is shut down. Consequently, requests to all to all services a_k received at all BS_i that were previously served by h_j (i.e., $\gamma_{i,k,j} > 0$) must be routed to other servers. To give priority to the services with tighter latency constraints in the closest servers, the services are sorted according to their delay budget, T^{max}. Additionally, candidate servers in the set $\mathcal{H}^{can}$

Algorithm 1 ODESA energy saving heuristic

1: Initialize η and γ
2: Sort $\mathcal{H}$ according to E^{idle} in descending order
3: Sort $\mathcal{A}$ according to T^{max} in ascending order
4: **for each** h_j **in** $\mathcal{H}$ **do**
5: $E^{on} \leftarrow$ current energy, $\eta_j \leftarrow 0$, $E^{off} \leftarrow 0$
6: **for each** a_k **in** $\mathcal{A}$ **do**
7: **for each** BS_i **in** $\mathcal{B}$ **do**
8: $\mathcal{H}^{can} \leftarrow \{h_m : \exists\, \ell_{i,m} \wedge T^r_{i,k,m} + T^u_{i,k,m} < T^{max}_k\}$
9: Sort $\mathcal{H}^{can}$ according to $\sigma_{i,j}$ in descending order
10: **for each** h_m **in** $\mathcal{H}^{can}$ **do**
11: **if** $\lambda_{i,k}\gamma_{i,k,j}V^{in}_k \leq \alpha$ **then**
12: **if** $o_k\lambda_{i,k} < C^{max}_m - O_m$ **then**
13: $\gamma_{i,k,m} \leftarrow \gamma_{i,k,j}$, $\gamma_{i,k,j} \leftarrow 0$
14: **else**
15: $\gamma_{i,k,m} \leftarrow (C^{max}_m - O_m)/\lambda_{i,k}$
16: $\gamma_{i,k,j} \leftarrow \gamma_{i,k,j} - \gamma_{i,k,m}$
17: $\alpha_{i,m} \leftarrow \alpha_{i,m} - V^{in}_k\lambda_{i,k}\gamma_{i,k,m}$
18: **else**
19: **if** $o_k\lambda_{i,k} < C^{max}_m - O_m$ **then**
20: $\gamma_{i,k,m} \leftarrow \lambda_{i,k}\alpha_{i,m}/V^{in}_k$
21: $\gamma_{i,k,j} \leftarrow \gamma_{i,k,j} - \gamma_{i,k,m}$, $\alpha_{i,m} \leftarrow 0$
22: **else**
23: $\gamma_{i,k,m} \leftarrow \gamma_{i,k,j}$, $\gamma_{i,k,j} \leftarrow 0$
24: $\alpha_{i,m} \leftarrow \alpha_{i,m} - V^{in}_k\lambda_{i,k}\gamma_{i,k,m}$
25: $E^{off} \leftarrow E^{off} + E^{off}_{k,m}$
26: **if** $E^{off} \geq E^{on}$ **or** $\sum_{BS_i\in\mathcal{B}}\sum_{a_k\in\mathcal{A}}\gamma_{i,k,j} \neq 0$ **then**
27: $\eta_j \leftarrow 1$, Undo changes
28: **if any** $T^u + T^r + T^c + T^o + T^d \leq T^{max}$ **then**
29: **while** Server with deadline violations found **do**
30: $h_j \leftarrow$ Last server turned off, $\eta_j \leftarrow 1$
31: Move load from server with deadline violations to h_j

(made of all h_m that have a path connecting them with BS_i and can be reached within the delay budget) are ordered according to the $\sigma_{i,m}$ of the path to prioritize shorter and more efficient paths. Then, for each candidate server h_m, the algorithm checks its free capacity (given by the difference between its maximum capacity C^{max}_m and the capacity in use O_m) and the remaining capacity of the link, denoted as $\alpha_{i,m}$. If both capacities are sufficient, the orchestrator reroutes all requests previously routed to h_j to h_m. If not, it routes as many requests as possible to h_m and tries to route the remaining requests to the rest of the candidates.

Considering the energy consumption of routing requests through the backhaul is essential, as it may negate the energy saved by shutting down the edge server. ODESA addresses this by keeping track of the energy difference between shutting down the edge server and routing the requests to other servers (E^{off}) and keeping it on (E^{on}). If $E^{off} \geq E^{on}$, the server h_j is kept on, the original routing is maintained, and the changes calculated by ODESA are rejected. ODESA also checks if all requests previously attended at h_j are routed to other edge servers ($\sum_{BS_i\in\mathcal{B}}\sum_{a_k\in\mathcal{A}}\gamma_{i,k,j} = 0$). If that is not the case, the routing changes are also rejected, and the server is kept on. Similarly, when the main loop has ended, ODESA checks if all T^{max} are satisfied. If deadline violations are found, ODESA turns on h_j and routes requests from the overloaded servers, which are typically those causing such violations.

The time complexity of Alg. 1 is a function of the number of edge servers in $\mathcal{H}$, which must be traversed twice (Lines 4 and 10 of Alg. 1), the number of services in $\mathcal{A}$, which is iterated once (Line 6), and the number of BSs in $\mathcal{B}$, which is also traversed once (Line 7). Thus, the time complexity of the energy optimization heuristic is $\mathcal{O}(|\mathcal{H}|^2\times|\mathcal{A}|\times|\mathcal{B}|)$. However, in practical scenarios, the number of edge servers and BSs greatly exceeds the number of services ($|\mathcal{B}|\gg|\mathcal{A}|$, $|\mathcal{H}|\gg|\mathcal{A}|$) and the number of edge servers is, at most, equal to the number of BSs ($|\mathcal{B}|\geqslant|\mathcal{H}|$). Therefore, the complexity can be represented as $\mathcal{O}(N^3)$ where $N = |\mathcal{B}|$.

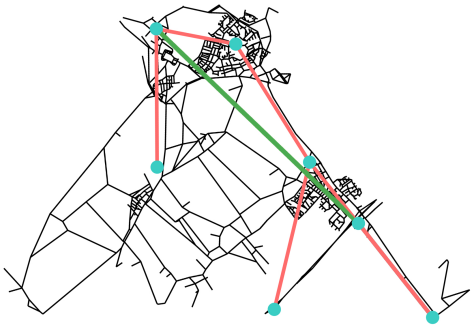


Fig. 2. Simulation scenario in the map of Elburg, Netherlands, with the location of the BSs' of a real MNO and the added backhaul links.

TABLE II
SIMULATION PARAMETERS

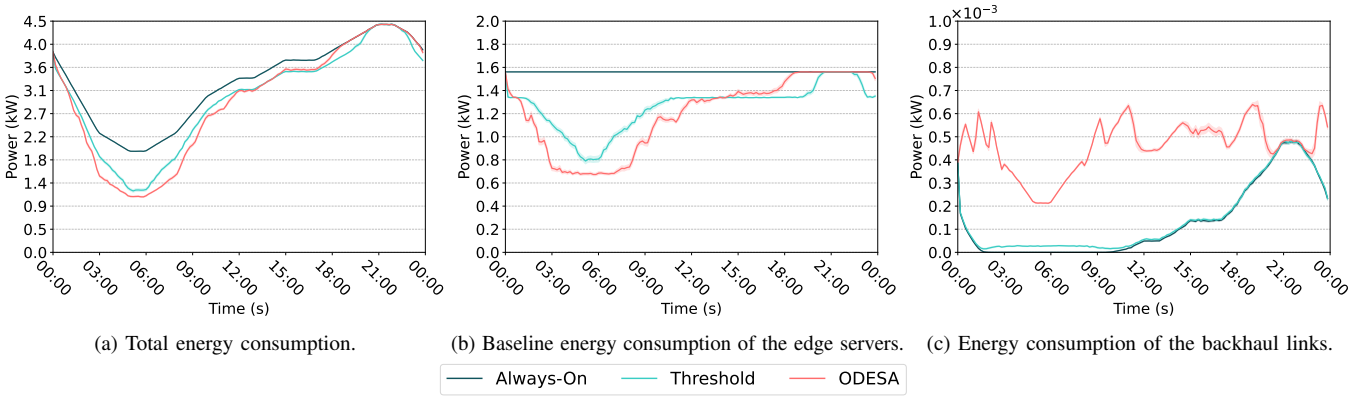
Parameter	Value
Max. Users	3800
Max. Server Capacity (C^{max})	11500941 ops/s
Idle server consumption E^{idle}	223 J/s
Max. server consumption (E^{max})	719 J/s
Link Capacity (α)	1 Gbps
Link energy consumption (σ)	5.9 nJ/bit
Req. workload (o_k)	7000 ops/s
Delay budget (T^{max})	5 ms
Request arrival rate (λ)	15 req/s
Data volumes (V^{in} and V^{out})	1600 and 100 bytes

V. Performance Evaluation

In this section, we evaluate the performance of ODESA using the following baselines for benchmarking: (i) always-on, (ii) servers sleep when their load is below a 10% threshold [10].

A. Scenario and Parameters

This scenario aims to evaluate ODESA in a realistic small-scale setting. Hence, we consider the infrastructure based on real data [21] of a single MNO in the municipality of Elburg, the Netherlands, which consists of 7 BSs located in various positions across the municipality, as shown in Fig. 2. For the sake of simplicity, we assume all BSs have an edge server with identical hardware. Specifically, we assume the servers are HP Enterprise ProLiant DL360 Gen11 equipped with Intel Xeon 8480+ CPUs. Table II shows the energy consumption and capacity of the servers driven by the benchmarks in [22]. Note that the energy per operation can be obtained as $E = (E^{max} - E^{idle})/C^{max}$ where E^{max} represents the energy consumption when the server operates at 100% of its capacity. We assume



(a) Total energy consumption. (b) Baseline energy consumption of the edge servers. (c) Energy consumption of the backhaul links.

Fig. 3. Energy consumption of the infrastructure for the three compared strategies over a 24 hour period.

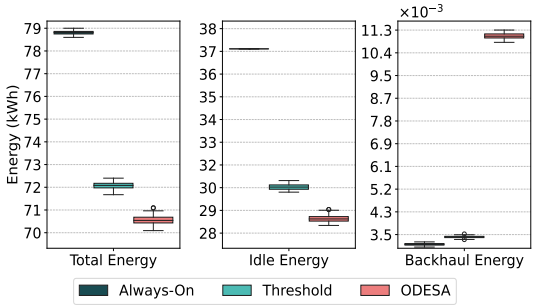
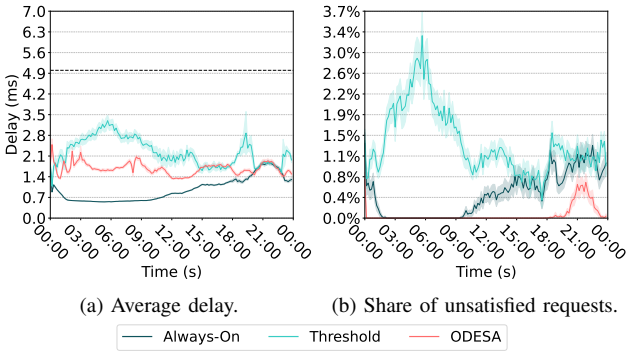


Fig. 4. Accumulated total energy consumption, idle energy consumption of the servers, and energy consumption of the backhaul in kWh after the 24-h period.



(a) Average delay. (b) Share of unsatisfied requests.

Fig. 5. Average delay and share of requests with an unsatisfied deadline (5 ms) over the 24 hour simulated period.

that the BSs are connected through the X2 interface with 1 Gbps fiber optics links following the topology shown in Fig. 2 using Edge Routers Cisco ASR9010, which consume 5.9 nJ per bit transmitted. The layout of the links is partially obtained from [21], where one radio link is provided, shown in green in Fig. 2. However, as this information is insufficient to connect all the BSs and determine their capacity, we assumed fiber optic links in some of the BSs depicted in red in Fig. 2.

Since our main objective is to assess the deadline satisfaction of latency-constrained services, we consider a single service and simulate a vehicular safety service with a stringent delay budget of 5 ms. We model users as vehicles moving randomly through the roads according to the speed limit. The arrival rate of the service requests for each user follows a Poisson distribution, with an expected number of arrivals of 15 requests/s [23]. The input data of each request is 1600 Bytes, in line with evaluation use cases in [24], while we assume the output is 100 Bytes. Additionally, each request requires 7000 ops/s in the servers [17]. We simulate a 24-hour period in which the number of active users varies according to the time-traffic distribution presented in [24]. According to this, 16% of all users are active during the peak hour, and only 2% are active during the valley hour. We set the total number of users to 3800 to ensure that the computing infrastructure can handle the peak load with a safety margin of 20%, ensuring that the system is not overloaded. This is a common approach to dimensioning the capacity of this type of infrastructure.

B. Results Discussion

Energy consumption: Fig. 3 shows the breakdown of the power consumption over a 24-hour period. ODESA reduces the total power consumption shown in Fig. 3a, which is mainly attributed to ODESA's ability to shut more edge servers down during periods of low and moderate demand, as well as keeping them off for more extended periods, thereby reducing the idle power, as depicted in Fig 3b. The threshold-based approach cannot shut down as many servers and turns them back on sooner, which results in lower energy savings. Although increasing the load threshold may allow it to shut down more servers, this approach's impact on deadline satisfaction renders it unfeasible, as discussed below. Shutting down edge servers increases ODESA's backhaul power consumption, as shown in Fig. 3c, but the savings achieved by doing this outweigh this increase as the backhaul energy represents just 0.01% of the total energy consumption, whereas the idle and processing energies represent 43% and 57%, respectively. It is noteworthy that in scenarios with larger input and output data or more energy-consuming links, the portion of backhaul energy would be more significant. In that case, ODESA would strike a balance between shutting down servers and using the backhaul links. ODESA achieves an average reduction of 11% in total energy consumption compared to the always-on approach and 2.25% compared to the threshold-based approach, as shown in Fig. 4. Notably, ODESA reduces the energy consumption by 42%

compared to the always-on approach and 16% compared to the threshold-based approach during the lowest demand period (between 03:00 and 08:00). However, during peak demand at 21:00, when all servers must be active to provide enough capacity for all requests, ODESA is unable to reduce energy consumption.

Deadline satisfaction: The energy footprint reduction has minor effects on the QoS, as shown in Fig. 5. As more edge servers are shut down, the requests must be routed to more distant servers, and users have to share less resources. Therefore, a degradation in the average delay is expected for the shutdown strategies. However, this is acceptable as long as it is below the 5 ms delay budget. Fig. 5a depicts how this degradation is more pronounced during low-demand periods for ODESA as it leaves fewer servers active. As computing time tends to be the biggest contributor to the total delay, ODESA redistributes load across the servers during low-demand periods when it detects potential deadline violations. This gives it an advantage over the threshold-based approach, as shown in Fig. 5a. Despite the increment in the average delay, ODESA can satisfy the requests, as shown in Fig. 5b, where it is the approach with the lowest share of deadline violations, only 0.06%. The always-on approach presents 0.4% of deadline violations, and the threshold-based approach delivers an average of 1.4% of requests after their deadline. Since this scheme bases its decision only on the servers' load, requests are redirected to overloaded servers. ODESA prevents this by rerouting requests from overloaded servers to the most recently reactivated server.

VI. Conclusions

In this work, we presented ODESA, an energy-saving strategy for edge computing environments that shuts down edge servers to save energy achieving a compromise between the energy consumed by idle servers and by the backhaul to route requests to active servers. Our analysis shows that ODESA achieves a considerable reduction of 11% in the energy footprint on the operation of the edge servers compared to the commonly implemented always-on approach while ensuring deadline satisfaction. By dynamically shutting down idle servers during low-demand periods, ODESA offers a sustainable and cost-effective solution for deploying edge computing services. Future work will focus on investigating the impact of varying server densities and load on the performance of ODESA, as well as reducing the computational complexity of the heuristic to enable faster decision-making. Additionally, we plan to explore the use of prediction schemes to enable proactive changes to server states and further improve energy efficiency.

Acknowledgements

This work has been supported by MCIN/AEI/10.13039/501100011033 (FEDER "a way of making Europe") under grants PID2022-142332OA-I00 and PID2021-123627OB-C52. This work is also funded by UCLM and the European Social Fund (Grant 2019-PREDUCLM-10921), the Government of Castilla-La Mancha (project SBPLY/21/180501/000195) and UCLM (project 2023-GRIN-34056). This work is also supported by the EU "NextGenerationEU/PRTR", MCIN, and AEI (Spain) under project IJC2020-043058-I and the EU's H2020 XGain project (GA No 101060294). The authors from UT acknowledge the support of the Faculty of EEMCS under the research grant EERI: Energy-Efficient and Resilient Internet. Blas Gómez thanks UCLM's Vice-rectorate of Science Policy for the mobility grant.

References

[1] B. Ramprasad, A. da Silva Veith *et al.*, "Sustainable computing on the edge: A system dynamics perspective," in *Proc. of ACM HotMobile*, 2021, pp. 64–70.

[2] R. Jacob and L. Vanbever, "The internet of tomorrow must sleep more and grow old," in *Proc. of HotCarbon*, 2022.

[3] "Energy-efficient Cloud Computing Technologies and Policies for an Eco-friendly Cloud Market | Shaping Europe's digital future," https://digital-strategy.ec.europa.eu/en/library/energy-efficient-cloud-computing-technologies-and-policies-eco-friendly-cloud-market, 2020.

[4] U. Wajid, C. Cappiello *et al.*, "On Achieving Energy Efficiency and Reducing CO2 Footprint in Cloud Computing," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 138–151, 2016.

[5] T. Mastelic, A. Oleksiak *et al.*, "Cloud Computing: Survey on Energy Efficiency," *ACM Comput. Surveys*, vol. 47, no. 2, pp. 1–36, Dec. 2014.

[6] I. Raïs, A.-C. Orgerie *et al.*, "Impact of Shutdown Techniques for Energy-Efficient Cloud Data Centers," in *Proc. Springer ICA3PP*, J. Carretero, J. Garcia-Blas *et al.*, Eds., vol. 10048, 2016, pp. 203–210.

[7] R. Buyya, A. Beloglazov *et al.*, "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges," in *Proc. of PDPTA*, 2010.

[8] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomput.*, vol. 60, no. 2, pp. 268–280, May 2012.

[9] D. Meisner, B. T. Gold *et al.*, "PowerNap: Eliminating Server Idle Power," in *Proc. of ACM ASPLOS*, 2009.

[10] S. Wang, X. Zhang *et al.*, "Cooperative Edge Computing With Sleep Control Under Nonuniform Traffic in Mobile Edge Networks," *IEEE Internet of Things J.*, no. 3, pp. 4295–4306, Jun. 2019.

[11] B. Wu, J. Zeng *et al.*, "New Game-Theoretic Approach to Decentralized Path Selection and Sleep Scheduling for Mobile Edge Computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 8, pp. 6125–6140, Aug. 2022.

[12] J. Ahn, J. Lee *et al.*, "Power Efficient Clustering Scheme for 5G Mobile Edge Computing Environment," *Mobile Netw. and Appl.*, vol. 24, no. 2, pp. 643–652, Apr. 2019.

[13] A. A. Amer, I. E. Talkhan *et al.*, "Optimal Power Consumption on Distributed Edge Services Under Non-Uniform Traffic with Dual Threshold Sleep/Active Control," in *Proc. of IEEE NILES*, 2021.

[14] C. N. L. Tan, C. Klein *et al.*, "Location-aware load prediction in Edge Data Centers," in *Proc. of IEEE FMEC*, 2017.

[15] J. Navarro-Ortiz, P. Romero-Diaz *et al.*, "A Survey on 5G Usage Scenarios and Traffic Models," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 905–929, 2020.

[16] P. Wang, Z. Zheng *et al.*, "HetMEC: latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4942–4956, 2019.

[17] P. Wiesner and L. Thamsen, "LEAF: Simulating large energy-aware fog computing environments," in *Proc. of IEEE ICFEC*, 2021, pp. 29–36.

[18] B. Yang, W. K. Chai *et al.*, "Cost-efficient nfv-enabled mobile edge-cloud for low latency mobile applications," *IEEE Trans. on Network and Service Management*, vol. 15, no. 1, pp. 475–488, 2018.

[19] J. Krarup and P. M. Pruzan, *Selected Families of Discrete Location Problems: Part III, the Plant Location Family*. Research Library, Faculty of Business, University of Calgary, 1977.

[20] S. K. Jacobsen, "Heuristics for the capacitated plant location model," *Eur. J. Oper. Res.*, vol. 12, no. 3, pp. 253–261, 1983.

[21] "Antennekaart," https://antennekaart.nl, accessed: 15/04/2023.

[22] Standard Performance Evaluation Corporation, "SPECpower results." [Online]. Available: https://www.spec.org/power_ssj2008/results/

[23] "5GPPP use cases and performance evaluation modeling," Accessed: 15/4/23. [Online]. Available: https://5g-ppp.eu/wpcontent/uploads/2014/02/5G-PPP-use-cases-and-performance-evaluation-modeling_v1.0.pd

[24] "METIS-II Mobile and Wireless Communications Enablers for Twenty–Twenty Information Society II," 2020, Accessed: 15/04/2023. [Online]. Available: https://metis-ii.5g-ppp.eu/