
White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Leeds.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/238997/

Version: Accepted Version

Proceedings Paper:
Shinde, S.S., Naseh, D., DeCola, T. et al. (2025) A Distributed Task Allocation Methodology for Edge Computing in a LEO Satellite IoT Context. In: Advanced Satellite Mobile Systems (ASMS). 2025 12th Advanced Satellite Multimedia Systems Conference and the 18th Signal Processing for Space Communications Workshop (ASMS/SPSC), 26-28 Feb 2025, Sitges, Spain. Institute of Electrical and Electronics Engineers (IEEE). ISBN: 979-8-3315-2238-4. ISSN: 2329-7093. EISSN: 2326-5949.

https://doi.org/10.1109/asms/spsc64465.2025.10946045

This is an author produced version of a conference paper accepted for publication in Advanced Satellite Mobile Systems (ASMS) made available via the University of Leeds Research Outputs Policy under the terms of the Creative Commons Attribution License (CC-BY), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.





A Distributed Task Allocation Methodology for Edge Computing in a LEO Satellite IoT Context

Swapnil Sadashiv Shinde*, David Naseh†, Tomaso DeCola‡ and Daniele Tarchi§
*CNIT - University of Florence Research Unit, 50139 Firenze, Italy, email: swapnil.shinde@cnit.it
†Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi",
University of Bologna, 40136 Bologna, Italy, email: david.naseh2@unibo.it
‡Institute of Communications and Navigation, German Aerospace Center (DLR),
Weßling, Germany, Email: tomaso.decola@dlr.de
§Department of Information Engineering, University of Florence, 50139 Firenze, Italy, email: daniele.tarchi@unifi.it

***Abstract*—The Internet of Things (IoT) is one of the most promising applications in the field of computer networking. Edge computing is a computationally efficient method for processing user data in a terrestrial-satellite hybrid environment, where each device is connected exclusively through a low-elevation (LEO) satellite. This paper focuses on an IoT context, introducing methodologies to effectively manage the computation-communication trade-off by strategically distributing processing tasks across various satellites. In particular, an adaptive load balancing approach is considered for efficient utilization of satellite resources. The proposed method can be implemented in a distributed manner, enabling each satellite to evaluate its task handling capacity and forward tasks if it is beyond its capability. The numerical results demonstrate the effectiveness of the proposed method compared to conventional fixed allocation and cloud processing methodologies.**

***Index Terms*—Edge Computing, Internet of Things, Low Earth Orbit Satellite Networks, Load Balancing, Cloud Computing**

I. Introduction

In recent years, edge computing has evolved into a mature concept predominantly utilized in terrestrial networks, driven by significant efforts in industry, scientific research, and standardization. The fundamental idea revolves around bringing storage and computation resources much closer to end users to improve performance, addressing potential latency and security issues that can still persist in cloud computing access [1].

Edge computing has primarily been viewed as a framework aimed at reducing latency for certain services, typically limited to terrestrial facilities. However, increasing attention to satellite systems, along with advancements in satellite processing and the integration of satellite and terrestrial networks, now allows edge computing in hybrid setups or exclusively through satellites. This progression has introduced new applications, making satellite edge computing appealing both for academic study and industrial advancement [2].

Among various potential application scenarios, the Internet of Things (IoT) application appears to be very promising. This is due to its ability to establish a remote accessible and global network, where IoT nodes can utilize processing nodes to offer more sophisticated services. In Figure 1, a schematic representation of the IoT scenario in which multiple entities interact among them.

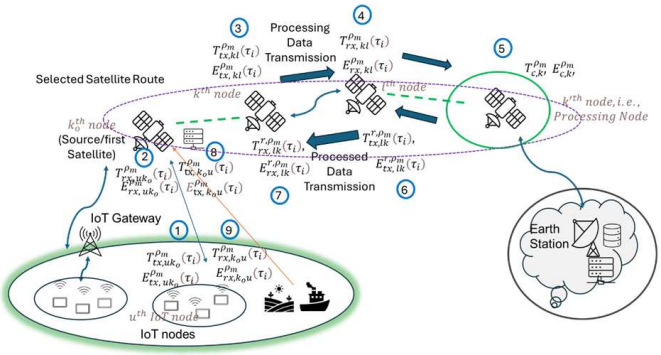


Fig. 1. Integrated Satellite-Terrestrial Edge Computing Scenario

In recent years, the literature has explored the feasibility of deploying edge computing solutions within satellite environments. The study in [3] introduces a three-tier hybrid cloud-edge computing architecture within Low Earth Orbit (LEO) satellite networks, offering diverse computing resources to global users. The focus is on optimizing computation offloading to minimize ground users' energy consumption, subject to coverage time and LEO satellite computation constraints. In [4], the emphasis shifts to the Space-Air-Ground Integrated Network (SAGIN) for Internet of Vehicles (IoV) applications, proposing a SAGIN-IoV edge cloud framework leveraging Software-Defined Networking (SDN) and Network Function Virtualization (NFV) to efficiently manage multiple communication networks. The work in [5] examines a satellite edge computing system that employs a game-theoretic strategy to offload computation, considering the transient nature of terrestrial-satellite links. Meanwhile, [6] explores an edge computing LEO network, offering benefits for IoT users beyond cloud solutions and discussing the challenges of space computing. In [7], an Edge Computing framework is introduced for terrestrial-satellite integrated IoT systems, with a key emphasis on reducing the energy consumption of IoT devices. Lastly, [8] presents a strategy for joint computation offloading and network selection to reduce latency and energy expenses in combined terrestrial and non-terrestrial IoT networks.

The optimization of task allocation in satellite constella-

tions, with the objective of equitably distributing processing loads, remains an open issue currently under investigation in the research community. This paper focuses primarily on an IoT context, introducing methodologies to effectively manage the communication-computing trade-off by strategically distributing processing tasks across various satellites. Four strategies are examined. The Nearest-Satellite-based strategy and the Cloud Computing strategy operate in a rigid manner examining two opposite cases of IoT data processing by assigning all tasks to the nearest satellite or the cloud computing facility, respectively. These can serve as benchmarks for evaluating system performance. Concurrently, two adaptive load balancing approaches are considered for allocating the tasks over a shortest path connecting the IoT users to terrestrial cloud facility. In the Random Allocation strategy, tasks are randomly allocated among nodes along the path between the task generation node and the cloud. Conversely, the Maximum Capacity Allocation aims to maximize the allocation capacity of the satellite nodes connecting the IoT users to terrestrial cloud facility acting as a backup when the system becomes fully occupied.

II. Satellite-based IoT Scenario

The Satellite-based IoT paradigm aims to enable seamless satellite connectivity for terrestrial IoT devices, each of which is associated with a Low Earth Orbit (LEO) satellite. Data transmission occurs directly between satellite and terrestrial nodes, as well as among nodes within the satellite network. LEO satellites possess the capability to process computational tasks offloaded by terrestrial IoT nodes; however, there are situations where it becomes necessary to exploit additional LEO satellites or remote cloud infrastructure. Moreover, complexities arise when task processing is delegated to another LEO satellite or remote cloud facility, necessitating additional routing processes.

Satellites currently improve IoT applications, characterized by two primary use cases with distinct impacts on satellites:

- Satellite-terrestrial hybrids for wide-area IoT service
- Standalone satellites for local IoT service

The former acts as a *gap-filler* for terrestrial NB-IoT, ensuring service continuity appealing for long-range user equipment mobility (e.g., vehicles). In contrast, the latter treats the satellite as a self-contained system, incorporating all functional and technical components of NarrowBand IoT (NB-IoT) within its network domain. In the following of the work, we will focus on the second case, where satellites are used as direct links for IoT nodes.

This direct link scenario is designed for applications deployed in regions lacking terrestrial communications (e.g., desert, rural, maritime) or when rapid deployment of a vertical IoT application is needed, minimizing reliance on terrestrial infrastructure and configurations. A stable connection to terrestrial networks is uncertain, as feeder links may be absent depending on the design of the LEO segment. Service delivery can occur via scheduled transmission times (provided that UEs are aware of LEO dynamics) or on demand through satellite access. The communication model involves interconnected sensors collecting local data and transmitting it to a central point via satellite support.

III. System Model

Consider a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ distributed within a specified rectangular area $\mathcal{A}$ with dimensions $D_x \times D_y$. These users are assumed to be randomly placed within this space. The system operates in a discrete-time framework, with the network parameters assumed to remain constant over each period of time τ. Each τ_i represents the ith time interval, denoted by $\tau_i = \{\forall t | t \in [i\tau, (i+1)\tau]\}$. The generic mth user is assumed to be able to communicate over a bandwidth $B_m^j(\tau_i)$ with a specified satellite s_j. Each user $u_m \in \mathcal{U}$ is considered active in every time interval with probability p_a, during which it generates a computation task request $\rho_m(\tau_i)$. This task is specified by the tuple $\langle D_{\rho_m}, D_{\rho_m}^r, \Omega_{\rho_m}, T_{\rho_m} \rangle$, which defines a task that has a size of D_{ρ_m} Bytes, expected to produce an output of size $D_{\rho_m}^r$ Bytes, requiring Ω_{ρ_m} CPU execution cycles, and having a maximum permitted execution latency of T_{ρ_m}.

In the deployment of a LEO satellite constellation, a collection of satellite nodes $\mathcal{S} = \{s_1, \ldots, s_j, \ldots, s_N\}$ is considered. These satellites are assumed to be positioned within the same shell, maintaining a uniform altitude of h_s from the Earth's surface. The constellation is organized into multiple orbital planes, with P_S indicating the total number of these planes. Satellites are distributed such that each plane approximately contains N/P_S satellites. Within each orbital plane, satellites are presumed to be evenly distributed at an inter-satellite separation denoted as d_{IS}. Additionally, each orbital plane is spaced apart from its neighbors by an interplane distance d_{IP}. Each satellite node has a computational capacity of c_j floating-point operations per second (FLOPS) per CPU cycle, with a CPU operating frequency of f_j and $\mathcal{L}_j$ cores, and an intersatellite communication bandwidth set at B_j. A ground station is integrated into the system, located in a predetermined position (x_C, y_C), and shares a location with a Cloud Computing facility. The facility is characterized by a processing capability of c_c FLOPS for each CPU cycle, operating at a CPU frequency of f_c, with $\mathcal{L}_c$ cores. In addition, it maintains a bandwidth of B_c for connectivity with any of the satellites within the LEO constellation.

A. Task Computation Model

The formulas representing the temporal and energetic consumption for executing the ρ_mth task on a generic kth device, which may include satellite nodes or cloud computing systems, are expressed as:

$$T_{c,k}^{\rho_m} = \frac{\Omega_{\rho_m}}{c_k^{\rho_m} f_k^{\rho_m}} \tag{1}$$

$$E_{c,k}^{\rho_m} = \frac{\Omega_{\rho_m}}{c_k^{\rho_m} f_k^{\rho_m}} P_{c,k}, \tag{2}$$

where $c_k^{\rho_m}$ denotes the number of floating-point operations per second, $f_k^{\rho_m}$ the frequency per CPU cycle allocated for

the ρ_mth task, and $P_{c,k}$ the computational power for the kth device [9].

B. Task Communication Model

The transmission duration and energy consumption for data exchange between arbitrary nodes k and l for the processing task ρ_m can be expressed as the aggregate of the duration of transmission and the duration of propagation of the signal:

$$T_{tx,kl}^{\rho_m}(\tau_i) = \frac{D_{\rho_m}}{r_{kl}(\tau_i)} + \frac{d_{kl}}{c} \tag{3}$$

In this equation, d_{kl} represents the physical separation between nodes k and l, while c denotes the speed of light. The energy expenditure is given as:

$$E_{tx,kl}^{\rho_m}(\tau_i) = \frac{D_{\rho_m}}{r_{kl}(\tau_i)} Pt_k \tag{4}$$

Here, $r_{kl}(\tau_i)$ signifies the data rate of the communication link between the nodes, and Pt_k stands for the transmission power of node k. Analogously, the time and energy required at node l for receiving a task of size D_{ρ_m} from node k are described by:

$$T_{rx,kl}^{\rho_m}(\tau_i) = \frac{D_{\rho_m}}{r_{kl}(\tau_i)} + \frac{d_{kl}}{c} \tag{5}$$

$$E_{rx,kl}^{\rho_m}(\tau_i) = \frac{D_{\rho_m}}{r_{kl}(\tau_i)} Pr_l \tag{6}$$

where Pr_l is the power consumed during data reception. The channel between k and l is presumed to exhibit symmetry.

In this study, we model the channel characteristics between the kth and lth nodes at the ith time interval according to the model proposed in [10]. The link gain is represented by

$$h_{k,l}(\tau_i) = \beta_0 \cdot d_{k,l}^{\theta^l}(\tau_i)$$

where $d_{k,l}(\tau_i)$ denotes the separation between node k and l at the ith interval. Here, β_0 indicates the power gain of the reference channel at a distance of 1 meter, and θ^l indicates the path loss exponent for the communication link between nodes k and l. The expression for the channel capacity is formulated from Shannon's capacity theorem and is given by:

$$r_{kl}(\tau_i) = b_k^{\rho_m}(\tau_i) \cdot \log_2\left(1 + \frac{Pt_k \cdot h_{k,l}(\tau_i)}{N_0}\right), \quad \forall k, l \tag{7}$$

where Pt_k stands for the transmission power of node k, $b_k^{\rho_m}(\tau_i)$ represents the allocated bandwidth for communication, and $N_0 = N_T b_k^{\rho_m}(\tau_i)$ defines the power of thermal noise, calculated from the noise power spectral density N_T.

C. Task Offloading Process

We assume the capability of task offloading within the system, allowing an operation initiated at one point to be processed by any node within the network. When the mth user decides to offload its task to any capable node, referred to as the generic kth node, including options like satellite and cloud computing nodes, the associated time and energy expenses for this offloading and obtaining the results can be expressed as follows:

$$T_{m,k}^{\text{off}}\left(\tau_i^{\text{off}}\right) = \sum_{l\in\mathcal{P}} T_{tx,l}^{\rho_m}\left(\tau_i^{\text{off}}\right) + T_{c,k}^{\rho_m}\left(\tau_i^{\text{off}}\right) + \sum_{l\in\mathcal{P}} T_{rx,l}^{r,\rho_m}\left(\tau_i^{\text{off}}\right) \tag{8}$$

$$E_{m,k}^{\text{off}}\left(\tau_i^{\text{off}}\right) = \sum_{l\in\mathcal{P}} E_{tx,l}^{\rho_m}\left(\tau_i^{\text{off}}\right) + \sum_{l\in\mathcal{P}} E_{rx,l}^{\rho_m}\left(\tau_i^{\text{off}}\right) + E_{c,k}^{\rho_m}\left(\tau_i^{\text{off}}\right) \tag{9}$$

In these equations, the summation indicates the aggregation of terms related to transmission, reception, and computation processes. It should be emphasized that the time and energy calculations incorporate every participating node in the offloading sequence, from the originating node to the destination. The set $\mathcal{P}$ comprises all intermediary nodes on the pathway between the source and the chosen edge computing node, with their selection strategy subject to future-defined policies. In these formulations, energy usage when nodes are idle is ignored, predicated on the assumption that nodes are continuously engaged, whether in the task at hand or in alternative activities. Thus, the energy evaluated specifically pertains to the incremental usage resulting from the offloading process.

IV. Problem Formulation

In the subsequent analysis, we explore scenarios in which distinct users simultaneously generate multiple requests. Each request may only be processed by one satellite at any given moment, indicating that each task is an indivisible entity and cannot be partitioned between different nodes. Each request is encapsulated within a single task and, consequently, assigned to one node. The allocation strategy, influenced by multiple requests, is dictated not only by the system's current state and the requisite conditions of each request but also by how different requests are collectively assigned. To achieve this, the assignment of tasks to each edge computing satellite node entails solving a multidimensional problem, requiring tasks to be assigned collectively while adhering to each task's specific requirements.

The objective of the analyzed system is to accurately identify the node capable of hosting the processing task. This requires a combined effort to minimize both the delay and energy consumption, allowing the chosen node to balance between energy use and processing time. The problem can be formally expressed as:

$$\min_{k,\mathcal{P}} \sum_m \left(\alpha_1 \cdot T_{m,k}^{\text{off}}\left(\tau_i^{\text{off}}\right) + \alpha_2 E_{m,k}^{\text{off}}\left(\tau_i^{\text{off}}\right)\right) \tag{10}$$

Here, k refers to the index of the node encompassing both satellite and cloud computing nodes, which are capable of jointly optimizing energy and time. Meanwhile, $\mathcal{P}$ denotes the path from the source to the node k selected for the computation offload. The optimal selection of this node involves not only selecting the node itself but also determining the most efficient

path to it. The method for selecting k will be elaborated later, in the context of benchmarks and our proposed load balancing approach. The strategy for selecting paths relies on the Dijkstra algorithm. This algorithm inherently calculates the optimal routes between two satellite nodes by evaluating distance measures, although it is beyond the scope of the optimization problem.

Given the constraints in link capacity and processing power, the former minimization problem is defined by two conditions:

$$\begin{cases} \sum_l r_{lk} \leq R_k \\ \sum_{\rho_m \in s_k} c_k^{\rho_m} \leq C_k \end{cases} \tag{11}$$

where R_k denotes the upper limit of data throughput for the node k, and C_k represents the upper limit of the processing capacity available for node k.

V. Load Balancing Solutions

Current scientific research highlights the transition from cloud computing to distributed computing, but satellite-based distributed data processing solutions are still rare. Satellites are frequently utilized as relays or central nodes for data handling, rather than collaboratively processing ground user data. Furthermore, research often highlights user-side energy consumption in distributed architectures, overlooking energy use by satellites and clouds. In IoT applications, studies tend to address generic IoT devices, which lack tailored service-specific data processing. Additionally, the joint optimization of Radio Resource Allocation and offloading processes, critical for resource-constrained satellites, is not adequately explored.

Expanding on earlier analyses, the devised load balancing strategies are efficient at distributing processing tasks to different satellite nodes, whether to the nearest node or the cloud node. When the system is tasked with handling multiple requests, an effective load balancing algorithm is essential. Consequently, two algorithms have been devised to demonstrate the effectiveness of load balancing compared to conventional fixed allocation methodologies. These are termed Random Allocation (RA) and Maximum Capacity Allocation (MCA). The RA strategy uses exclusively satellite resources, while the MCA approach initially uses distributed satellite resources before resorting to the cloud. Nevertheless, in this model, we do not account for the potential additional costs of using cloud resources, presuming these resources are beyond our direct management.

A. Random Allocation

In the context of RA, the principle is to allocate a processing task at random to a satellite node. To avoid any arbitrary selection, especially of nodes distant from the source node, we confine this random choice to satellites located along the route from the closest node, serving as an ingress node, to the cloud facilities. This strategy stems from the idea that, when distributing the load across multiple nodes, it is advantageous to distribute the load across all potential nodes, including cloud facilities. Consequently, this algorithm operates alongside the routing algorithm, allowing any intermediate node along the designated path to be chosen for processing tasks. Despite its straightforwardness and the absence of targeted optimization, the algorithm effectively distributes the workload among the nodes. Unlike the fixed approach, where nearby nodes rapidly become congested when multiple tasks are generated, or the exclusive use of cloud facilities that could result in increased delays, this method seeks to balance the load efficiently.

B. Maximum Capacity Allocation

MCA encompasses a methodology that aims to efficiently distribute workload between nodes, using the pathway that connects the ingress satellite node to the cloud facility. The core concept is that the closest node normally takes precedence, as it tends to reduce latency. However, this preference is valid only until the node reaches its capacity; once the task load surpasses this pre-established threshold, the node is bypassed in favor of the next available node en route to the cloud. The procedure is uniformly applied to all processing tasks, employing each node up to its designated capacity, which remains consistent but adjustable between nodes. As a safeguard, should all nodes along the path be at full capacity, the cloud facility acts as a fallback option, drawing upon its presumed limitless processing capabilities, unaffected by the task count. Algorithm 1 outlines the proposed maximum capacity allocation approach in detail. The procedure begins by assigning all users to the proximal satellite node for the transmission of IoT data in space (lines 1-6). Subsequently, the transmission distance between each satellite node and the terrestrial cloud facility is determined (lines 7-9). Thereafter, the shortest path between each input satellite, which is the satellite to which IoT users are assigned, and the cloud facility is identified utilizing the Dijkstra algorithm, based on the calculated distance metrics (lines 11-13). The subsequent step involves determining the cost associated with the satellite nodes for receiving data from IoT users (lines 14-16). The load balancing mechanism is then implemented, allowing each user to host a certain number of users. The process begins from the nearest satellite node, referred to as the input satellite, and progresses towards the cloud facility. Users are allocated to a particular satellite if capacity permits; otherwise, user data is redirected to the subsequent satellite along the path (lines 17-23). Following the generation of user allocation policies through the maximum allocation approach, a cost analysis is performed by evaluating the energy and latency costs (lines 24-27).

C. Benchmark Solutions

To facilitate comparison, we implement two reference solutions. These benchmarks represent traditional data processing scenarios. The solutions examined comprise a cloud-centric processing model in which user data is transmitted through satellite nodes to terrestrial cloud infrastructure. Alternatively, we examine the nearest satellite processing to enable data handling close to its source.

Algorithm 1 Maximum Capacity Allocation.

Input: $\mathcal{U}, \mathcal{S}, c$
Output: $\mathcal{E}, \mathcal{D}$

for all $u \in \mathcal{U}$ **do**
  for all $s \in \mathcal{S}$ **do**
    $d(s,u)$ ▷ the distance between s and u
  end for
end for
$s^* \leftarrow u^*$ if $d(s^*,u^*) = \min d(s,u)\ \forall s,u$
for all $s \in \mathcal{S}$ **do**
  $d(s,c)$ ▷ the distance between s and c
end for
$s' \leftarrow c$ if $d(s',c) = \min d(s,c)\ \forall s$
for all $s^* \neq 0$ **do** ▷ for each satellite with at least one associated user
  $P(s^*,s') = \text{DIJIKSTRA}(s^*,s')$ ▷ $P(s^*,s')$ is the shortest path between s^* and s'
end for
for all $s^* \neq 0$ **do** ▷ for each satellite with at least one associated user
  $D \mathrel{+}= d_{rx}(s^*) + d_{prop}(s^*)$ ▷ The first delay term considers the delay for receiving the task from any associated user (i.e., reception, propagation)
end for
for all $u^* \in \mathcal{U}$ **do**
  for all $s \in P(s*,s")$ **do**
    if $\|s \leftarrow u^*\| < U$ ▷ if the users allocated to s^* are below a certain number U
      $s \leftarrow u^*$ ▷ assign the user to that satellite
      break
  end for
end for
for all $s \in P(s^*,s)$ **do**
  $E \mathrel{+}= E_{rx}(s) + E_p(s) + E_{tx}(s)$ ▷ The Energy is the sum of all the energy terms spent by each satellite in the path up to the selected s satellite (i.e., reception, processing, transmission)
  $D \mathrel{+}= d_{tx}(s) + d_{prop}(s) + d_p(s)$ ▷ The Delay is the sum of all the delay terms spent by each satellite in the path up to the selected s satellite (i.e., transmission, propagation, processing)
end for

1) Cloud-based Processing Approach: Cloud computing infrastructures frequently serve as processing centers for user data in various wireless communication environments. Satellite nodes relay data to terrestrial cloud facilities located in various locations on Earth to process information from remote users. Due to their extensive resources, the use of cloud computing infrastructures for data processing constitutes a computationally efficient strategy relative to edge-based distributed computing models. However, the long transmission distances involved can lead to substantial data transfer delays. In satellite-driven scenarios, satellite nodes can collect data via direct links or gateway stations. Subsequently, the data is relayed through a satellite network to the nearest cloud facility.

2) Nearest Satellite-based Processing Approach: In the scenario under review, satellite nodes, which include either the targeted ground users or the specific area of interest, serve as collection points for user data. To investigate the feasibility of processing user data while minimizing transmission costs, we propose a data processing approach based on proximity to the nearest node. Here, user data handling is assigned to the closest satellite terminals, in order to lower overall transmission expenses. However, the presence of multiple users coupled with the limited computational capacity of the satellite nodes could lead to increased computational costs.

VI. Numerical Results

In this Section, the numerical results obtained through computer simulations are included. In order to evaluate the performance of the system, a Python-based simulator has been developed. All the numerical results are obtained by varying the number of active nodes corresponding to the nodes that have a task to be processed. Since we consider having an activity factor equal to 0.1, it corresponds to say that the total number of nodes is around 10 times that used in the abscissa. The activity factor is a value that models the fact that despite having a much higher number of IoT nodes, only a fraction of them are active (i.e., generate tasks) at any instant. The simulator is based on Python and runs on the Google Colab platform allowing us to develop the system in a collaborative way. Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs.

The numerical analysis was performed using hypothetical satellite constellations, defined by the following parameters. It is essential to note that these parameters specify a satellite orbital configuration that does not globally cover the Earth's surface; rather, it concentrates on the European region. The number of satellites is chosen to ensure efficient coverage in this area. It should be noted that utilizing satellites positioned on the Earth's antipodal side for Edge Computing is considered impractical, which renders this limited strategy logical.

In a satellite constellation, there are 3 orbital planes, each hosting 5 satellites. Satellites maintain an orbital radius of $1000\,\mathrm{km}$, with intersatellite separation and interplane separation measuring $500\,\mathrm{km}$ and $200\,\mathrm{km}$, respectively. For message transmissions, each satellite requires $1\,\mathrm{W}$ of power, which underscores that energy consumption is equated to the product of transmission duration and this power figure, while the power demand for receiving messages is $0.8\,\mathrm{W}$ per satellite. Satellites assign $1\,\mathrm{W}$ to computation duties on board, where the energy consumption corresponds to the computation time multiplied by this figure. The scenario considers a variable range of ground IoT nodes, specifically 50 to 300, dispersed homogeneously throughout a $25\,000\,\mathrm{km}^2$ region. The communication bandwidth allocated to each satellite for ground connections is assumed to be $100\,\mathrm{MHz}$, distributed among all users connected to a particular satellite. Each node generates tasks of size $2\,\mathrm{MB}$, with a computational requirement of 1000

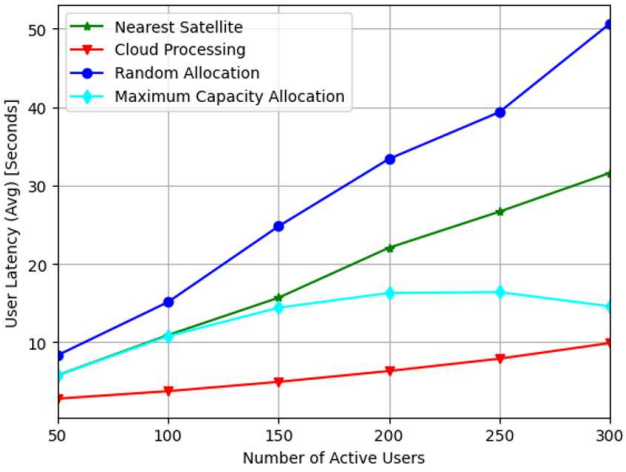


Fig. 2. Average per IoT node latency

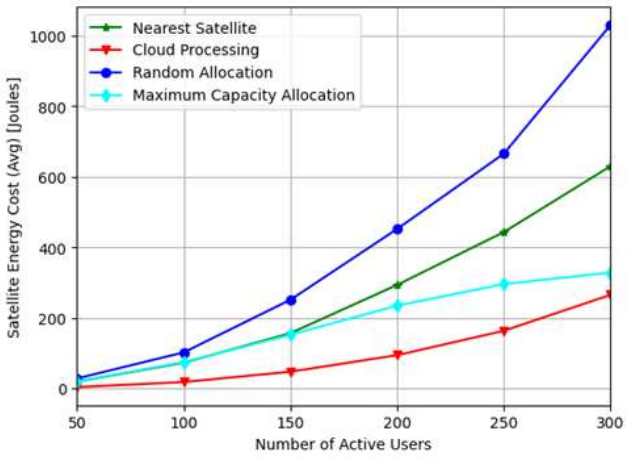


Fig. 3. Average Satellite Total Energy

FLOPS per bit, as indicated for these tasks [11]. The feeder link to the cloud facility offers a communication bandwidth of 20 MHz for each IoT message directed to the cloud, while its computational capacity reaches 10 GFLOPS per IoT task. Both the ground-to-satellite and inter-satellite communication links provide a bandwidth of 100 MHz. The computational capacity of the satellite nodes themselves is rated at 30 GFLOPS.

In Figure 2, the mean total latency is illustrated, comprising the latency costs of communication, propagation, and computation. With cloud computing, increasing user numbers leads to increased total latency. This is because, although computation latency in cloud systems is negligible, communication latency escalates with more users accessing cloud resources, raising total latency values. When the load is balanced, especially under maximum capacity allocation, latency is minimized. As user numbers grow, many favor cloud services for data handling due to their ample resources, resulting in almost zero processing costs. Consequently, beyond a certain point, the computation latency decreases. In contrast, the nearest-satellite strategy suffers from higher latency, as nearby satellites are overloaded with user tasks. The random allocation method performs the poorest, exhibiting the maximum latency due to inadequate load distribution, assigning users to limited satellites, resulting in congestion and increased computation latency. The simulation presupposes a direct path from the data source to the satellite node linked to the cloud facilities, facilitating cloud access. Thus, user data are channeled through an identical satellite en route to the cloud facility. Within the RA method, this escalates the chances of users choosing the same satellite, i.e., the one leading to the cloud, escalating computation latencies.

Figure 3 shows the average total energy use of satellites, combining communication and computation energy, as the number of active nodes varies. Among the strategies evaluated, random allocation results in the highest energy consumption due to ineffective load distribution, causing some satellites to become congested and incur higher latency and energy costs. The Nearest-Satellite strategy ranks second in energy consumption because it depends on one satellite to handle tasks, thereby increasing its computation load. In contrast, the load-balancing strategy, especially with Maximum Capacity Allocation, effectively spreads tasks across satellites, reducing energy use. Cloud computing, while minimal in the impact of satellite computation energy, sees an increase in communication energy with more users, leading to a rise in total average energy consumption.

To enhance our understanding of the implications for energy consumption, we also assessed the energy consumption per bit. This metric enables a more fair comparison across various methodologies and scenarios involving packets and tasks of differing sizes. Figure 4 presents an evaluation of the energy spent per bit. The observed trend parallels that depicted in the preceding figure, where average values were considered. In particular, the energy expended per bit in communication is less than that in computation. Furthermore, it is evident that the proposed Maximum Capacity Allocation, beyond a certain data production threshold, benefits energetically from offloading processes to the cloud.

Figure 5 presents a comparison of total energy and latency across four different methodologies, analyzed under a simulated scenario with 200 users. In particular, the random satellite allocation strategy results in suboptimal outcomes for certain users, showing increased energy usage and latency. This variability arises from the uneven distribution of users, leading to disparities in latency and energy expenses. Conversely, the Maximum Capacity Allocation technique excels in latency reduction due to its load balancing, offering a more strategic satellite selection than random allocation. Furthermore, solutions that incorporate Cloud Processing provide superior performance relative to the other methods analyzed.

Figure 6 illustrates the number of satellites used for computational tasks when applying various solution methodologies. In particular, cloud processing solutions exclude satellite nodes from being involved in processing operations. In contrast, the load-balancing strategy effectively distributes the computa-

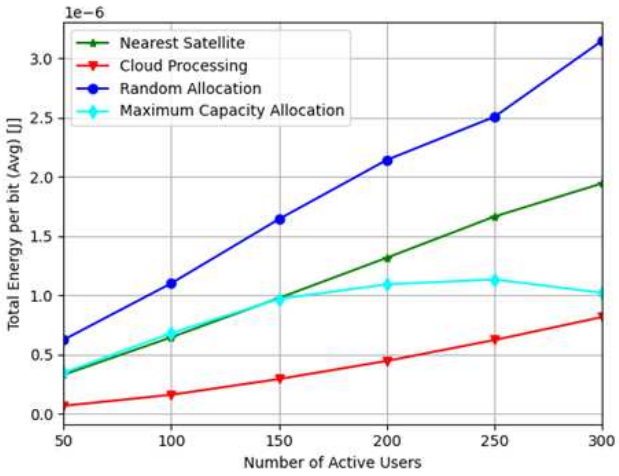


Fig. 4. Average Satellite Total Energy per bit

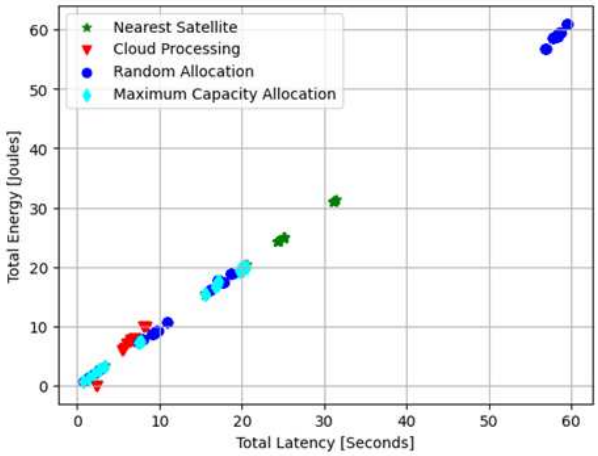


Fig. 5. Energy vs Latency

tional workload across a larger number of satellites compared to the random satellite selection method.

VII. Conclusions

The preceding analysis, in conjunction with numerical data, clarifies the influence of load-balancing strategies on Edge Computing within a terrestrial-satellite IoT framework. Significantly, various parameters associated with edge computing facilities and user demands were observed to have a significant impact on the numerical results. The proposed heuristic load-balancing method not only meets, but frequently exceeds traditional benchmarks while also demonstrating adaptability to diverse configurations. The proposed methodology has reduced complexity, allowing it to be implemented in a distributed manner, allowing each satellite to assess its task handling capacity and delegate tasks if they exceed its capabilities.

Acknowledgment

This work was supported by the European Space Agency under the framework of EDEGCOLB (“Joint communication and edge computing load balancing techniques for satellite networks”) project, ESA Contract n. 4000140850/23/NL/FGL. The view expressed herein can in no way be taken to reflect the official opinion of the European Space Agency.

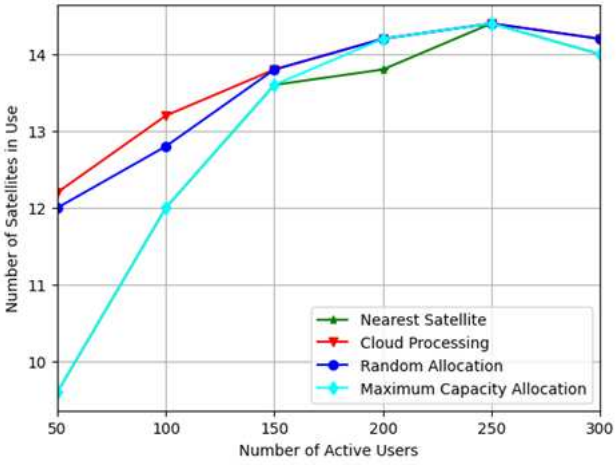


Fig. 6. Satellites in use

References

[1] G. Carvalho, B. Cabral, V. Pereira, and J. Bernardino, “Edge computing: current trends, research challenges and future directions,” *Computing*, vol. 103, pp. 993–1023, 2021.

[2] Z. Lai, H. Li, Q. Wu, Q. Ni, M. Lv, J. Li, J. Wu, J. Liu, and Y. Li, “Futuristic 6G pervasive on-demand services: Integrating space edge computing with terrestrial networks,” *IEEE Veh. Technol. Mag.*, vol. 18, no. 1, pp. 80–90, 2023.

[3] Q. Tang, Z. Fei, B. Li, and Z. Han, “Computation offloading in LEO satellite networks with hybrid cloud and edge computing,” *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9164–9176, 2021.

[4] B. Cao, J. Zhang, X. Liu, Z. Sun, W. Cao, R. M. Nowak, and Z. Lv, “Edge-cloud resource scheduling in space-air-ground-integrated networks for internet of vehicles,” *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5765–5772, 2022.

[5] Y. Wang, J. Yang, X. Guo, and Z. Qu, “A game-theoretic approach to computation offloading in satellite edge computing,” *IEEE Access*, vol. 8, pp. 12 510–12 520, 2020.

[6] D. Lioprasitis, A. Priovolos, G. Gardikis, S. Pantazis, S. Costicoglou, A. Perentos, E. Hadjioannou, M. Georgiades, A. Phinikarides, A. Fornés-Leal, R. Gonzalez-Usach, C. E. Palau, and M. Esteve, “Satellite edge computing for 5G rural applications,” in *2021 IEEE International Mediterranean Conference on Communications and Networking (MediterCom)*, 2021, pp. 1–2.

[7] Z. Song, Y. Hao, Y. Liu, and X. Sun, “Energy-efficient multiaccess edge computing for terrestrial-satellite internet of things,” *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14 202–14 218, 2021.

[8] S. S. Shinde and D. Tarchi, “Network selection and computation offloading in non-terrestrial network edge computing environments for vehicular applications,” in *2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, 2022, pp. 1–8.

[9] ——, “Collaborative reinforcement learning for multi-service internet of vehicles,” *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2589–2602, 2023.

[10] X. Gu and G. Zhang, “Energy-efficient computation offloading for vehicular edge computing networks,” *Computer Communications*, vol. 166, pp. 244–253, 2021.

[11] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. Salinas Monroy, “Multi-objective computation sharing in energy and delay constrained mobile edge computing environments,” *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2992–3005, 2021.