ESC: An Emulation Framework for the Spatial Computing Continuum

Ivan Sanchez-Cordero
FPU Predoctoral Fellow
University of Extremadura
Caceres, Spain
ivansc@unex.es

Angel Macias
Researcher
University of Extremadura
Caceres, Spain
amaciba@unex.es

Miguel A. Garcia-Sevilla
Senior Programmer
University of Extremadura
Caceres, Spain
mgarciajh@alumnos.unex.es

Juan Luis Herrera
Assistant Professor
University of Extremadura
Caceres, Spain
jlherrerag@unex.es

Sergio Laso
Industrial PhD
Gloin S.L.
Caceres, Spain
slasom@unex.es

Cristina Vicente-Chicote
Tenured Associate Professor
University of Extremadura
Caceres, Spain
cristinav@unex.es

Juan A. Fraire
ISFP Researcher
INSA Lyon - INRIA
Lyon, France
juan.fraire@inria.fr

Oscar Ledesma
Associate Professor
International University of La Rioja
Logroño, Spain
oscar.ledesma.garcia@unir.net

David Bernal
PhD Candidate
International University of La Rioja
Logroño, Spain
david.bernalaguera@unir.net

Julian Fernandez
Co-Founder - Lead Researcher
FOSSA Systems S.L.
Madrid, Spain
julian@fossa.systems

Jaime Galan-Jimenez
Associate Professor
University of Extremadura
Caceres, Spain
jaime@unex.es

Javier Berrocal
Full Professor
University of Extremadura
Caceres, Spain
jberolm@unex.es

***Abstract*—The Computing Continuum (CC) is increasingly employed because it enhances Quality of Service (QoS). Likewise, the relevance of Space Computing is rising thanks to the reinforcement it provides to terrestrial infrastructures against catastrophes. Even if the expansion of the CC paradigm with new layers placed within satellites or further celestial bodies seems like a natural evolution, it certainly does not lack challenges. In fact, adding non-terrestrial layers to the CC could not only imply a significant increase in deployment costs but also make it harder to ensure certain QoS requirements. Therefore, to ensure the viability of mixed earth-space CC infrastructures, the development of new tools for evaluating these architectures *a priori* remains essential, given the enormous expenditure that an entire deployment on a real infrastructure would entail. To accomplish this goal, this paper presents ESC (Emulator for the Spatial Continuum): an emulation tool that enables the design and evaluation of software behavior in these mixed architectures while avoiding the costs associated with real deployments.**

***Index Terms*—Computing Continuum, Space Computing, Microservices, Emulation**

I. Introduction

Nowadays, the use of distributed applications across the Computing Continuum (CC) has been increasing [1], leveraging its different layers to minimize total computing load and improve Quality of Service (QoS). Despite these benefits bestowed by the CC, terrestrial infrastructures responsible for its implementation can be affected by adverse environmental conditions, such as earthquakes or hurricanes [2]. The Spatial CC (SCC) paradigm emerged in response, among other factors, to these traditional (terrestrial) CC limitations [3]. The SCC allows the distribution of computing resources beyond Earth to decentralize processes and reduce dependence on terrestrial systems. SCC thus expands the CC by adding layers in which devices deployed to space, such as Low Earth Orbit (LEO) satellites or servers established on other celestial bodies, can also be used for computing. Public interest in this matter is rising, with corporations such as Starlink operating around 9,400 LEO satellites [4]. Consequently, other companies dedicated to the SCC are emerging, such as FOSSA Systems and Sateliot [5], aiming to apply the Internet of Things (IoT) across space.

SCC is laid out as a distributed infrastructure, in which many independent computing elements are arranged over a network and coordinate with one another to perform one or more tasks and achieve a goal. To make full use of this distributed infrastructure, it is necessary to leverage a software architecture that supports a distributed deployment. Microservice Architecture (MSA) is a preferred approach for distributed applications because it enables the partitioning of software into modules called microservices [6]. MSA-based applications can distribute their workload across multiple devices. Microservices can be deployed independently on different machines, while they can be composed to communicate over the network to perform larger tasks. In contrast, traditional monolithic applications can be deployed on a single machine, whereas MSA-based applications can distribute their workload across multiple devices. Therefore, MSA is an ideal architecture for applications running in the SCC.

ISBN 978-3-903176-82-9 © 2026 IFIP

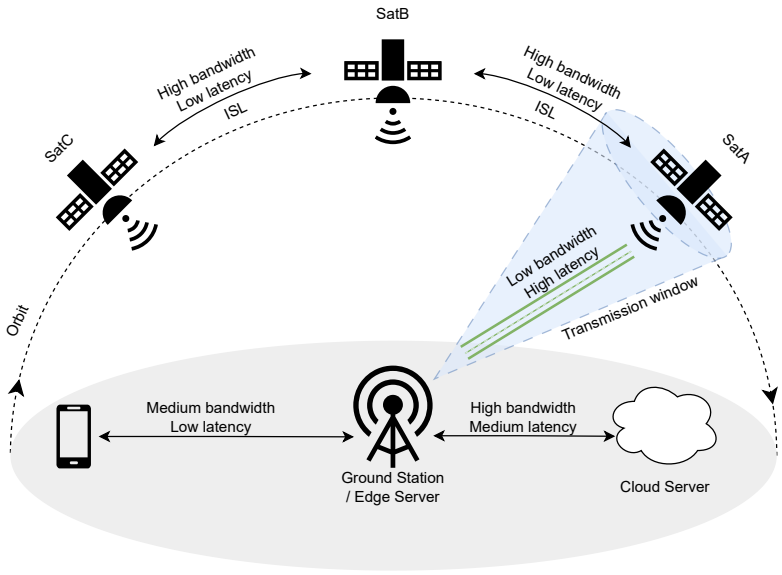

Fig. 1. Operation of the transmission window in an example SCC scenario.

The software architecture is not the only relevant aspect for SCC applications. QoS remains essential for deploying IoT in intensive domains, such as industry or medicine, where requirements are highly stringent [7], [8]. This means the coordination of tasks within SCC for these contexts must be highly effective, which may require several rehearsals. There is thus a need to test the performance and validity of every microservice, since the cost of deploying and updating a microservice on LEO satellites is very high, and the potential costs if the satellite lacks the resources to execute the microservice correctly are significant. Thus, it is necessary for a system that allows developers to test the validity and QoS of their application without incurring the high monetary cost and time consumption of direct testing on a real SCC infrastructure.

This paper presents **ESC**: Emulator for the Spatial Continuum, an emulator capable of deploying a virtual SCC infrastructure running a real MSA-based application. **ESC** allows for the emulation of devices in layers that are outside of planet Earth, where both LEO satellites and other celestial bodies could be included. The fundamental aspect of **ESC** is providing support to developers by reducing the cost of testing the validity and QoS of their applications in the SCC, compared to deploying real devices in space and microservices for them. The main contribution of **ESC** is its orientation towards MSA, which, as far as we know, makes it the first emulator to support Spatial MSA.

The rest of the article is organized as follows: Section II details the fundamental model of the SCC on which **ESC** is based. Section III motivates the proposal through a related use case. Section IV describes the behavior and elements that make up the architecture, as well as the remaining open challenges to be addressed by the emulator. Finally, Section V concludes the paper and addresses some future work.

II. System Model

Because of the variability in these communications both in space and between space and Earth [9], [10], bandwidth and latency are inversely and directly proportional, respectively, to the current distance between nodes, with these changes accentuating as they draw near. Traditionally, the terrestrial CC has a mostly static infrastructure, with small to no variability in the position, availability and properties of nodes and links. In contrast, SCC proposes a different schema in which, albeit the terrestrial part is static, every link regarding spatial nodes is variable and volatile.

To understand the model employed for **ESC** emulation, it is necessary to detail three crucial concepts that cause the variability and volatility: the infrastructure of ground stations, the earth-satellite connection, and the links between satellites. The remainder of this section details each of these aspects and how **ESC** manages them.

The first concept in the **ESC** model is the ground station. Ground stations are nodes in terrestrial infrastructure that can communicate wirelessly with spatial layers. These stations are generally connected to the Internet as shown in Figure 1, since managed requests are often received remotely. There can be one or more ground stations per area, depending on the area's circumstances and the network topology. As network nodes, they can communicate with each other via routers and satellites.

The latter connections, between terrestrial and spatial infrastructures, are performed between a ground station and a nearby satellite, and follow the transmission window model. The transmission window between a satellite and a ground station is the portion of the satellite's orbit during which the satellite is within the Line of Sight (LoS) of the ground station.

For example, Figure 1 shows SatA within the transmission window of the station while SatB and SatC have not reached it yet. Furthermore, for a satellite to be within LoS of a ground station, it must keep an elevation angle within an acceptable threshold and be within a minimum distance of the ground station [10]. According to the transmission window model, a ground station and a satellite can only be considered to be connected, and thus able to transmit information, when the satellite is in its transmission window.

Before going into detail on satellite technology, another concept that is worth clarifying regarding the recommended model for **ESC** is the behavior induced by MSA. Adopting this variety of architectures eases understanding of some ordinary uses of SCC; moreover, it is also influenced by this variability and volatility. For this reason, the composition of these microservices in **ESC** is managed using workflows. Workflows are pipelines of consecutive microservices, that is, each microservice sends its output as input for the next in line. At the end of the pipeline, the last microservice can send the results of the full workflow execution to the device that requested it.

Furthermore, satellites are not always within LoS of a ground station throughout a workflow due to orbital dynamics: other satellites can reach the same position in LoS over time, and there may be periods when no satellites are in LoS with this station. To mitigate the issues arising from this problem and make use of the capabilities provided by spatial laser communications, LEO satellites can have Inter-Satellite Links (ISLs) at their disposal to complete these workflows successfully. ISLs consist of direct wireless links to other satellites at close range, so they are not always available, similar to LoS with stations. For example, in Figure 1, SatA can only contact SatC through SatB because of their arrangement. ISL technology enables satellites to communicate with other satellites nearby to exchange information when it is required, and hence, allows them to reach a satellite that is currently in LoS to continue or finish the workflow.

Figure 1 illustrates an example outlining these concepts. This scenario has a mobile phone acting as User Equipment connected to a ground station that is co-located with an edge server. This edge server is connected to the internet and, as a result, has access to the cloud. The ground station can connect to LEO satellites, with SatA being the only one in LoS. As shown, the connection between the phone and the station is stable, as well as through the edge and the cloud. Links among satellites and transmission windows are inherently volatile. It can be observed that the link between the phone and the edge station has a lower latency than the link reaching the cloud, although it has less bandwidth. ISLs have much better bandwidth and latency, on the other hand. However, these values are significantly worse within the transmission window for various reasons, e.g., the distance the information must traverse. Therefore, if a workflow in this scenario requires offloading computation to a satellite, SatA could transmit information to SatB via ISL if SatA leaves the transmission window, since SatB will eventually reach SatA's location.

Although terrestrial and spatial infrastructures can operate independently, there are benefits to combining their respective advantages. Stability from terrestrial networks, along with the greater coverage and independence from Earth enabled by spatial networks, are some crucial features this fusion allows. Some places around the world that are isolated from global infrastructure can reach it via a satellite network if they own a ground station, e.g., Antarctica.

To enable the deployment of MSA-based applications in these architectures, satellite nodes must also execute microservices. Some satellites are equipped to perform computational operations in addition to their communication capabilities. At the same time, they remain isolated devices in outer space with rechargeable solar batteries that must be periodically turned off, unlike standard servers. This restriction affects their availability, meaning that there may be cases where transmission when in LoS with a station is not possible. Furthermore, it is really difficult for them to undergo maintenance if they become damaged. Because they are more limited, they do not perform well with some demanding microservices. Therefore, to reach complete SCC it is preferable to distribute computing tasks between space and Earth, and not rely solely on one kind.

III. Motivations

To illustrate the need for **ESC**, this section presents a motivational use case on Earth observation for disaster monitoring. This use case is based on the analysis by Haloho *et al.* [11] on the use of LEO satellites across various aspects of monitoring and data analysis. It is noteworthy that the use of **ESC** is not limited to this specific use case.

This Earth monitoring use case aims to monitor different areas to detect fires and ensure a prompt response. This can be done by analyzing sensor data and satellite images [11]. This application provides two main benefits: on the one hand, if the fire is addressed soon after it starts, it is easier to control and prevent its spread. On the other hand, quick reaction can also minimize the destructive effects of the fire in the affected area. Hence, it is necessary to leverage an infrastructure that can provide short response times, minimizing the time between detecting a possible fire and notifying the relevant services.

Initially, the CC may seem like an appropriate infrastructure. The CC can provide low latency for the fire detection applications if its microservices are adequately distributed. However, the CC can only provide service to areas with a stable Internet connection, such as urban areas. This does not cover rural areas, which are not only affected by fire, but the effects of such disasters are disproportionately more impactful in them [12]. To cover these areas, the traditional CC can be expanded to an SCC infrastructure, with LEO satellites providing coverage to these rural areas.

To leverage the SCC, the QoS the application obtains is important for ensuring a prompt response. This affects, on the one hand, the types of satellites used: more powerful satellites can lead to shorter response times by reducing execution time, but can also be more costly. The number of satellites and their network connections are also key: if a satellite detects

a fire but can only communicate with the ground every 12 hours, the response time is significantly increased. As satellite connections are intermittent by design, the microservices must be prepared to handle possible disconnections and partial data. The distribution of microservices between satellites and ground devices is also important for the QoS achieved. Finally, the cost of updating software on satellites can be high; hence, ensuring the correctness of the microservices deployed to satellites is also important.

Traditionally, these aspects are evaluated in a real SCC infrastructure, either by renting an existing LEO satellite constellation and ground CC, which represents a high economic cost, or by deploying a new constellation, which is even more costly. In the CC, this issue was solved by using emulators, which allow testing applications and their QoS with a fraction of the economic cost of a real infrastructure [13]. These emulators, however, do not support LEO satellites. **ESC** addresses this need by offering a complete SCC emulator, bridging the gap between ground and space infrastructure.

IV. ESC: Emulator for the Spatial Continuum

This section describes, firstly, **ESC**'s architecture and how it allows replicating real SCC systems and, secondly, the main challenges associated with the development of the emulation tool.

A. ESC Architecture

There is a dichotomy regarding how to imitate a system's functioning [3]: through *simulation*, that is, skipping implementation details to simplify the execution procedure; or through *emulation*, which replicates operating procedures in both hardware and software. This last alternative prioritizes accuracy but often leads to worse performance due to the added complexity. The approach chosen by **ESC** prefers precise outcomes (emulation), providing an emulated environment which allows the execution of real code.

ESC is proposed as an extension of *CCSIM* [13], a CC emulation framework that provides a stable basis, as well as already being compatible with most of the planned architecture. This way, *CCSIM* users can easily transfer their architectures to **ESC**, including new layers and different kinds of spatial devices to their systems.

ESC employs Kathará [14] to deploy the virtual architecture, defining a network composed of custom containers to ensure each component's features remain the same as in the physical twin. Since Docker containers comprise elements from the scenario, both software and hardware perspectives, and consider each device's CPU cores and RAM, the system can be considered a precise emulation.

The workflow employed by **ESC**'s architecture is illustrated by Figure 2. Firstly, the scenario configuration is retrieved from a file that specifies information on microservices, devices, network topologies, ground stations, transmission windows, and the orbital dynamics of the considered satellites. This input file is then processed by the Config Parser component, which extracts applicable configurations for **ESC** from its contents. Those configurations include containers to be deployed for the scenario, virtual networks among nodes, and instructions requiring coordination to maintain consistency across nodes in orbit. Once the configuration has been obtained, the scenario is deployed via the Terraform Deployer component within a virtualized environment on a remote machine, where execution is maintained for the specified duration. In the meantime, the Data Collector stores the container log files along with various QoS metrics for the scenario so that the deployment can be objectively evaluated. This last step is crucial because of its many applications, from testing microservice orchestration strategies for these environments to simply measuring the capacity that a specific infrastructure can provide.

As shown in Figure 3, the same procedure as *CCSIM* is employed for terrestrial topologies inside the carried emulation. This means that a new container is generated for each separate IoT device or router that participates in the transmission, and for each deployed microservice, an additional container is added to ensure compatibility with Docker images. Nonetheless, spatial nodes present a series of singularities that must be considered to ensure proper behavior for **ESC** because of their constant movement across their respective orbits. Therefore, an orchestrator has been attached to the architecture, being in charge of managing corresponding links depending on the location of the devices, turning on when they are in line of sight and being turned off when they are no longer in reach. The orchestrator is also responsible for managing all spatial links over time to ensure accurate latency and bandwidth.

To address the orchestrator's management, it is planned to preprocess trajectories for every satellite in the network before execution, obtained from official data on existing constellations. This way, positions and changes can be estimated in advance for each experiment to determine the modifications the orchestrator must command, and the results can be saved to skip this step in subsequent experiments with the same scenario.

B. Open challenges for ESC

There are many pending challenges to ensure **ESC**'s proper development and expansion, summarized in Table I along with some seemingly feasible answers for them.

V. Conclusions and future works

The SCC paradigm promises improved QoS and greater resilience against catastrophic events through its inherent communication characteristics. However, the deployment of these systems proves very costly, which is especially problematic for testing and retrieving QoS data. This work proposes **ESC** to cover these needs, providing an emulation tool to allow developers to evaluate microservice orchestration within mixed earth-space CC systems, without incurring in expenses from deploying in the real system.

Regarding future works, it is expected to extend the emulator to include Kubernetes compatibility. This would facilitate microservice replication and enable advanced network management functions to test scenarios that prioritize robustness

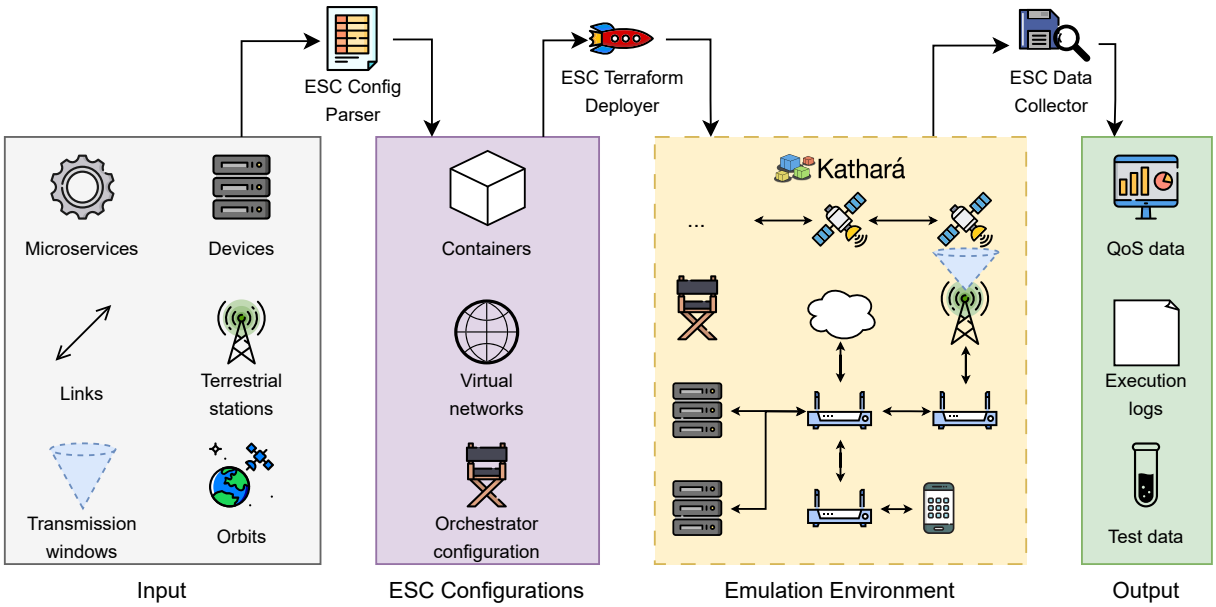

Fig. 2. Architecture diagram of **ESC**.

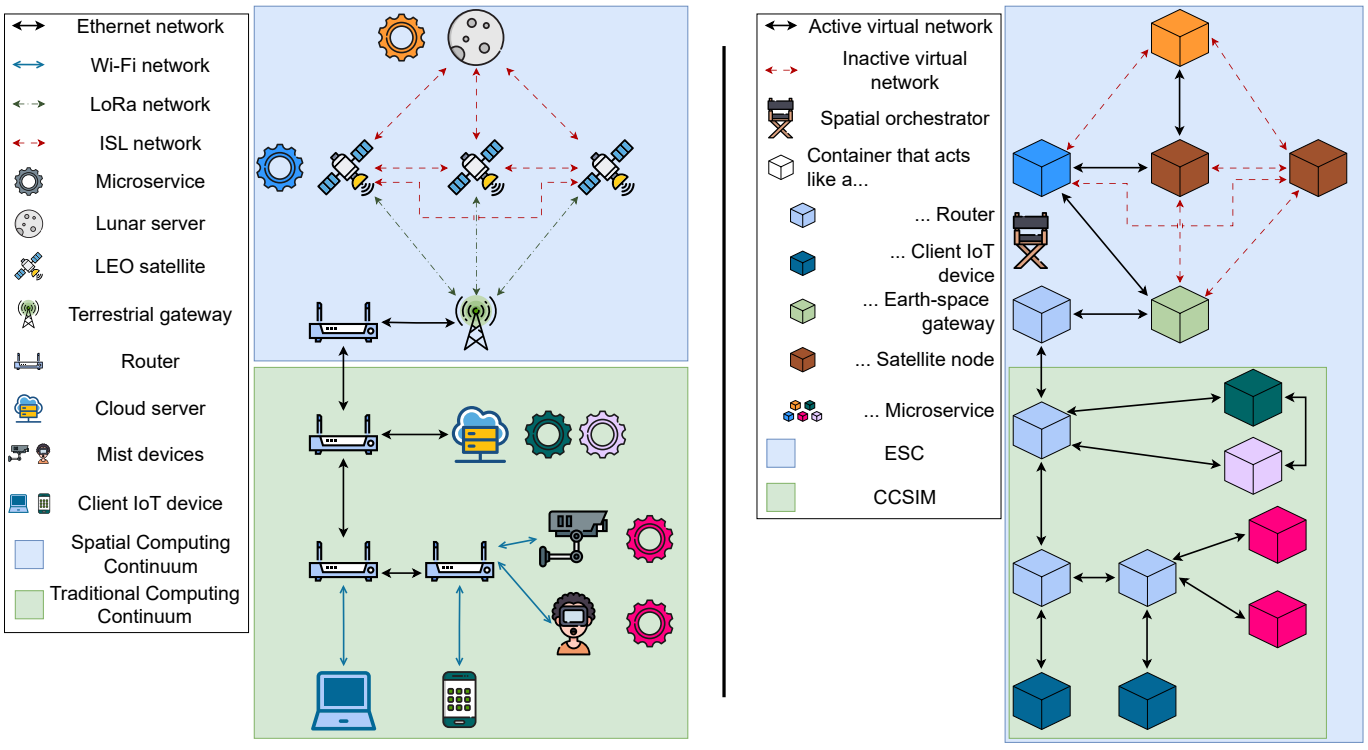

Fig. 3. Behaviour of **ESC** in an example scenario.

TABLE I
MAIN CHALLENGES FOR ESC ALONG WITH SOME PROMISING SOLUTIONS

Title	Description	Possible solutions
Management of data sources for the physical twin	Required information from satellites and the uploading method must be determined, with the update frequency as well.	Study of satellite query functions, search for an approximation of frequency that allows balance between likelihood and complexity.
Addition of physical satellite properties	Addition of physical properties, like energy consumption or solar batteries state.	Queries upon battery physical registries based on elapsed time to approximate.
Satellite hardware emulation	Emulation of hardware sensors, such as image recording or temperature registries.	Temporary employment of synthetic data, use of a predictor trained from the physical twin.

against system failures. Another expected future work is the adjustment of bandwidth and latency based on the simulated distance between satellite nodes according to orbital dynamics to improve the accuracy of the results provided. Finally, there are plans for extending **ESC** through plug-ins that provide a solution to the presented open challenges.

ACKNOWLEDGMENT

This activity has been co-financed 85% by the European Union, the European Regional Development Fund and the Regional Government of Extremadura. Managing Authority: Ministry of Finance. Grant File Number: **GR24099**. The work has also been partially subsidized by the grants FPU24/04302 and PTQ2024-013825 funded by MICIU/AEI/10.13039/50100011033.

REFERENCES

[1] L. F. Bittencourt, R. Rodrigues-Filho, J. Spillner, F. De Turck, J. Santos, N. L. da Fonseca, O. Rana, M. Parashar, and I. Foster, "The computing continuum: Past, present, and future," *Computer Science Review*, vol. 58, p. 100782, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013725000589

[2] D. Genkina, "Data Centers on the Moon: Genius or Foolhardy?" 2025, (Accessed on 23/02/2026). [Online]. Available: https://spectrum.ieee.org/data-center-on-the-moon

[3] T. W. Pusztai, J. Hisberger, C. Marcelino, and S. Nastic, "Stardust: A Scalable and Extensible Simulator for the 3D Continuum," in *2025 IEEE International Conference on Edge Computing and Communications (EDGE), IEEE EDGE 2025, Helsinki, Finland, July 7-12, 2025*, R. N. Chang, C. K. Chang, J. Yang, N. Atukorala, D. Chen, S. Helal, S. Tarkoma, Q. He, T. Kosar, C. A. Ardagna, F. M. Awaysheh, V. Hilt, and Y. Simmhan, Eds. IEEE, 2025, pp. 44–53. [Online]. Available: https://doi.org/10.1109/EDGE67623.2025.00014

[4] M. Jackson, "Starlink Secure Approval to Launch 7,500 Faster LEO Broadband Satellites," 2026, (Accessed on 31/03/2026). [Online]. Available: https://tinyurl.com/mrx5sr47

[5] M. Kaczmarek, "Spain's Stellar Ascent: Inside the Boom of Its Space and Satellite Industry," 2025, (Accessed on 31/03/2026). [Online]. Available: https://tinyurl.com/zne326na

[6] L. De Lauretis, "From monolithic architecture to microservices architecture," in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2019, pp. 93–96. [Online]. Available: https://doi.org/10.1109/ISSREW.2019.00050

[7] N. Banavase Venkategowda and L. F. Bittencourt, "Service Placement in the Computing Continuum for Delay-Sensitive IoT/IIoT Applications," *Journal of Network and Systems Management*, vol. 33, no. 4, p. 89, 2025. [Online]. Available: https://doi.org/10.1007/s10922-025-09964-7

[8] U. Islam, M. N. Alatawi, A. Alqazzaz, S. Alamro, B. Shah, and F. Moreira, "A hybrid fog-edge computing architecture for real-time health monitoring in IoMT systems with optimized latency and threat resilience," *scientific reports*, vol. 15, no. 1, p. 25655, 2025. [Online]. Available: https://doi.org/10.1038/s41598-025-09696-3

[9] O. Ledesma, P. Lamo, and J. A. Fraire, "Trends in LPWAN Technologies for LEO Satellite Constellations in the NewSpace Context," *Electronics*, vol. 13, no. 3, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/3/579

[10] C. J. Rojas Milla, "Design, Emulation, and Control of Edge Computing Systems for the Space Cloud in LEO Satellite Networks," Ph.D. dissertation, Università di Genova, 2026. [Online]. Available: https://hdl.handle.net/11567/1285557

[11] L. S. Haloho and A. A. Supriyadi, "Utilization of satellite technology in communication systems, disaster monitoring, border surveillance, and military intelligence: A literature review," *Remote Sensing Technology in Defense and Environment*, vol. 1, no. 1, pp. 36–44, 2024. [Online]. Available: https://doi.org/10.61511/rstde.v1i1.2024.842

[12] S. Wang, M. Zhang, X. Huang, T. Hu, Q. C. Sun, J. Corcoran, and Y. Liu, "Urban–rural disparity of social vulnerability to natural hazards in Australia," *Scientific reports*, vol. 12, no. 1, p. 13665, 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-17878-6

[13] P. Rodríguez, S. Laso, J. Berrocal, P. Fernández, A. Ruiz-Cortés, and J. M. Murillo, "Computing Continuum Simulator: A comprehensive framework for continuum architecture evaluation," *SoftwareX*, vol. 30, p. 102156, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711025001232

[14] G. Bonofiglio, V. Iovinella, G. Lospoto, and G. Di Battista, "Kathará: A container-based framework for implementing network function virtualization and software defined networks," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–9. [Online]. Available: https://doi.org/10.1109/NOMS.2018.8406267