POLITECNICO DI TORINO
Repository ISTITUZIONALE

Edge-to-Cloud Continuum Made Real

Original
Edge-to-Cloud Continuum Made Real / Galantino, S., Marino, J., Risso, F., Braghin, S., Nedoshivina, L., Fernando Skármeta Gómez, A., Siracusa, D.. - In: COMPUTER. - ISSN 0018-9162. - 59:3(2026), pp. 51-59. [10.1109/MC.2025.3607210]

Availability:
This version is available at: 11583/3002809 since: 2026-02-23T08:36:56Z

Publisher:
IEEE

Published
DOI:10.1109/MC.2025.3607210

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

19 November 2026

Edge-to-Cloud Continuum Made Real

Stefano Galantino, Jacopo Marino, Fulvio Risso, *Politecnico di Torino, Torino, Italy*
Stefano Braghin, Liubov Nedoshivina, *IBM Research, Dublin, Ireland*
Antonio Fernando Skármeta Gómez, *University of Murcia, Murcia, Spain*
Domenico Siracusa, *University of Trento, Trento, Italy*

Abstract—Despite the number of projects and initiatives aiming at creating the so-called computing continuum, surprisingly no commonly adopted definitions exist so far. This paper proposes a definition of the computing continuum that relies on three transparency properties, namely orchestration, communication, and resource availability. Then, it analyzes the most suitable open-source technologies that can provide a foundation to the transparent computing continuum, and it presents how, thanks to the FLUIDOS architecture and implementation, this concept has been brought into the real world.

The traditional siloed approach to computing, characterized by isolated and domain-specific solutions (e.g., edge, cloud), presents significant challenges in addressing the diverse and dynamic requirements of modern applications. This approach limits flexibility, scalability, and interoperability, crucial for applications such as real-time processing [1] and data analytics [2]. The computing continuum, instead, envisions a seamless integration of heterogeneous computing layers, enabling dynamic and efficient resource utilization across geographical and operational boundaries. Despite its potential to revolutionize modern computing, the lack of a universally accepted definition hinders its standardization and adoption.

To bridge this gap, this paper proposes a refined definition of the computing continuum, grounded in three fundamental transparency properties: *orchestrations*, *communication*, and *resource availability*. These properties draw inspiration from the seminal concept of *liquid computing* [3] and have been further developed within the FLUIDOS Horizon Europe project. FLUIDOS (Flexible, scaLable, secUre, and decentralIseD Operating System) is a decentralized meta-OS that leverages the orchestration capabilities of Kubernetes. This is augmented by the multi-cloud and multi-cluster abstractions provided by Liqo [3], enabling each node to seamlessly share computational resources across continuum members while consistently enforcing the three transparency properties. Beyond establishing the continuum infrastructure, FLUIDOS also manages applications deployed within this environment, maximizing the user-defined execution goals.

To address the challenges associated with the siloed-based computing landscape and the absence of a unified definition, this paper has three main objectives: (*i*) It proposes a precise definition of the computing continuum based on three transparency properties that collectively ensure seamless integration of heterogeneous computing resources. (*ii*) It introduces FLUIDOS, a decentralized meta-operating system designed to instantiate these transparency properties in real-world environments. (*iii*) It evaluates the effectiveness of FLUIDOS through experimental validation, demonstrating its ability to enable dynamic and efficient workload execution across the continuum.

INDUSTRY LANDSCAPE AND MOTIVATIONS

Here, we highlight applications constrained by traditional siloed computing, which cannot meet their diverse requirements within isolated environments.

[R1] Real-time Applications require immediate data processing to make decisions within a few milliseconds [1]. To ensure such delays, these applications must operate at the far edge, where data is typically produced. However, high availability of resources is a critical concern, i.e., ensuring that if a hosting cluster becomes unreachable, a new healthy instance of the application must be deployed on a nearby site.

XXXX-XXX © IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

TABLE 1: Industrial Gaps and Continuum Advantages Across Domains

Sector	Common Practice	Deficiency	FLUIDOS Advantage
Information Technology [R2][R3]	Hybrid deployments across cloud and on-prem infrastructure, with static multi-cloud orchestration	Limited edge integration; no dynamic workload reallocation across domains	Intent-based orchestration and seamless scaling across edge, on-prem, and cloud
Automotive [R1][R2][R3][R4]	In-vehicle edge processing combined with cloud-based model training and fleet analytics	Inability to support real-time task migration across domains; service handover lacks automation	Mobility-aware offloading and dynamic orchestration across vehicle, roadside, and telco-edge nodes
Healthcare [R2][R3]	Hospital-edge processing with strict privacy compliance; cloud for archival and research analytics	Siloed data domains prevent collaboration; regulatory barriers limit workload mobility	Federated orchestration with privacy-aware resource negotiation and data sovereignty enforcement
Telecommunications [R1][R3][R4]	Centralized management of RAN, MEC, and core infrastructure; isolated service domains	Limited flexibility for tenant-based services; no cross-site orchestration in real time	Decentralized orchestration and dynamic continuum slicing across edge and metro nodes
Energy and Utilities [R1][R2]	Site-level controllers and cloud dashboards for monitoring; limited edge analytics	Delayed control decisions; lack of cross-domain resource visibility	Distributed orchestration and edge-to-cloud integration for predictive maintenance and grid resilience

[R2] Data-Intensive Applications rely on vast amounts of data that may need processing, filtering, and storage [2]. The availability of GPU accelerators in this case is of paramount importance for statistical data analysis or machine learning models for inference.
[R3] Cross-Domain Data Sensitive Applications handle sensitive data that must adhere to strict privacy regulations. As a result, they need localized processing to ensure that the data is stored and processed according to specific regulations. This all relates to the concept of *data spaces* [4], [5], which refers to building complex federated ecosystems for data sharing to enable participants to pool, access, process, use, and share data trustfully and transparently.
[R4] Highly Mobile and Decentralized Applications deal with the mobility of users, and, consequently, the mobility of the application following the same user [6]. This process is similar to the handover process in telecommunications, in which a cellular transmission is seamlessly migrated from one base station to another.

Despite extensive investment and progress in sectors such as *Information Technology (IT)*, *automotive*, and *healthcare*, current edge-to-cloud computing practices remain fragmented and domain-specific. Within the FLUIDOS project, bridging the gap between these heterogeneous industry demands and the dynamic execution capabilities offered by the continuum is of paramount importance. To this end, Table 1 presents a cross-sector analysis of domains that may benefit the most from the FLUIDOS continuum. Each sector is mapped with the distinct application requirements outlined previously, revealing a recurring need for support of real-time and data-intensive workloads. Notably, privacy and security emerge as critical concerns, particularly for applications involving sensitive user data or those operating in shared, multi-tenant infrastructures.

THE CONTINUUM PILLARS

The computing continuum may seem like an existing reality. For instance, a single Kubernetes cluster spanning all layers may all use cases. However, it lacks localized intelligence, as all physical nodes are managed by the same control plane. Therefore, this section highlights three key features that current approaches still lack and are enforced in FLUIDOS.

Orchestration transparency

When microservices are set up in the current silo-based computing environment, each must be deliberately configured to be placed in a specific location, such as an edge data center versus the cloud. The location for each component is thus predetermined and fixed; changes to the location of any component are not permitted without initiating a new deployment phase. As a result, any potential dynamic optimization that could be performed during operation is complicated or even impossible. In contrast, within liquid computing, an intent-based interface ensures that each microservice is launched in the most suitable location and allows for dynamic adjustments as needed. Therefore, all services benefit from a unified deployment and control point, while FLUIDOS ensures services start in the optimal location based on the requirements, the current state of the infrastructure, and the required

security and privacy constraints.

Resource availability transparency

With today's technology, a microservice can use only the resources within its own cluster. As a result, a service might face disruptions due to an inability to access available resources located in other areas of the continuum, despite their availability. Resource scarcity is unlikely in cloud data centers, however, the challenge becomes more pronounced with the limited resource pools found at the edge, where typically only a few servers are available. The liquid computing addresses this challenge by facilitating the creation of a virtual computing space that extends across multiple physical domains. This allows a service initiated within this virtual space to utilize resources from any contributing domain, regardless of their physical locations. Thus, in FLUIDOS, a service can scale fluidly based on resource availability across the entire virtual infrastructure. For instance, it could result in one instance running at the telco edge and another in the cloud, effectively erasing the rigid boundaries of clusters, of paramount importance also for future 6G networks.

Communication transparency

The way microservices communicate varies depending on whether they are part of the same communication domain or not. Internal cluster communications use different mechanisms (like Kubernetes' ClusterIP service) compared to those needed for external communications (such as Kubernetes' LoadBalancer services). This means services need to be specifically set up to interact with each other based on their location, adding complexity to any potential re-deployment effort. Some technologies, such as message brokers, can mitigate this issue; however, these solutions require shared communication mechanisms, which may not be feasible due to limitations of the publish/subscribe model (e.g., latency) or reliance on other protocols. The concept of liquid computing introduces a solution by creating a virtual cluster that extends over multiple physical clusters, with applications operating within this virtual environment. As a result, communication between services within this virtual space is as straightforward as if they were located in the same physical cluster, eliminating the need for complex and potentially error-prone configuration.

THE PATHWAY TO THE CONTINUUM

We outline here the most relevant technologies that can be used to build the continuum, differentiating among *production-ready* and *research-oriented* projects. For each technology, we indicate which pillars it satisfies. It is worth mentioning that, although theoretically the full set of pillars might be enforced by a combination of more than one solution, this might not always work in practice due to inherent incompatibilities among them (e.g., *KubeEdge* does not integrate with *Cilium Cluster Mesh*).

Production-ready open-source projects

KubeEdge. KubeEdge [7] extends Kubernetes orchestration to edge devices, enabling device management, metadata synchronization, and application deployment across cloud and edge. KubeEdge facilitates efficient interaction with devices and sensors connected to edge nodes, enabling localized data collection and processing while maintaining synchronization with the cloud. However, KubeEdge has been designed for specific communication patterns (i.e., edge-to-cloud and cloud-to-edge). Therefore, it lacks the support for broader multi-cluster scenarios (e.g., edge-to-edge). *Orchestration transparency: ✓, Resource availability transparency: ✗, Communication transparency: ✗.*

Karmada. Karmada (Kubernetes Armada)[1] is designed to manage applications across multiple Kubernetes clusters without requiring changes to the applications. Karmada provides centralized multi-cluster management, automated deployment, high availability, failure recovery, and traffic scheduling across hybrid and multi-cloud scenarios. While Karmada facilitates cluster management under a master-worker model, it does not allow for equal-level federation of clusters (i.e., peer-to-peer). Moreover, Karmada lacks a built-in network fabric for inter-cluster communication, relying instead on external tools like Submariner[2]. *Orchestration transparency: ✓, Resource availability transparency: ✓, Communication transparency: ✗.*

Liqo. Liqo [3] enables dynamic and seamless Kubernetes multi-cluster management across heterogeneous environments, including on-premises, cloud, and edge. Liqo introduces four primary capabilities: peering, offloading, network fabric, and storage fabric. Peering establishes relationships between clusters. Offloading allows for workload distribution across remote clusters while maintaining full Kubernetes API compliance. The network fabric enables inter-cluster pod communication, even in overlapping network spaces, while the storage fabric postpones storage binding to optimize data locality. As a result, when peered with

[1] https://karmada.io.
[2] https://submariner.io.

Liqo, a remote Kubernetes cluster is collapsed into a *virtual node* visible through the local Kubernetes API, allowing for seamless cluster management. Despite some weaknesses in terms of automation, as the peering process must be performed (*i*) manually from the infrastructure operator, and (*ii*) based on a priori knowledge (i.e., it does not provide resource discovery), Liqo shows the broader coverage in terms of the above-mentioned transparency pillars. Hence, Liqo has been selected as the enabling technology of FLUIDOS, extending the functionalities well beyond such limitations, while inheriting the excellent scalability properties shown in [3]. By abstracting every remote *site* (i.e., remote *cluster*) as a single *virtual node*, Liqo makes it possible to scale to thousands of connected *sites*, potentially exceeding the usual node-based scalability limits of a single Kubernetes cluster (i.e., 5000 nodes). *Orchestration transparency: ✓, Resource availability transparency: ✓, Communication transparency: ✓.*

Cilium Custer Mesh. Cilium Cluster Mesh[3] creates a network mesh for seamless communication between pods across multiple Kubernetes clusters. However, it does not support propagation of *services*; it requires all clusters to coordinate IP addresses, as it does not support overlapping IPs or NAT-based address translation, and mandates the use of the Cilium CNI in all clusters, limiting interoperability with other CNIs or, in general, with multi-owner infrastructures. *Orchestration transparency: ✗, Resource availability transparency: ✗, Communication transparency: ✓.*

Research-oriented projects

Cloudlets. Mobile cloud computing addresses the resource limitations of mobile devices by offloading tasks to the cloud, but high WAN latency makes it unsuitable for real-time applications. To overcome this, cloudlets (trusted, resource-rich devices near mobile users [8]) enable low-latency execution by rapidly deploying virtual machines for specific tasks [9]. Cloudlets can operate as ad-hoc networks, dynamically adjusting to node availability, or as elastic virtualized infrastructures that scale resources as needed. Still, the approach lacks a dedicated network fabric to scale the applicability to multi-cluster environments. *Orchestration transparency: ✓, Resource availability transparency: ✗, Communication transparency: ✗.*

FLEDGE. FLEDGE [10] is a container orchestration system designed for resource-constrained edge devices while maintaining compatibility with Kubernetes. It integrates a modified Virtual Kubelet and a VPN to connect edge devices with Kubernetes clusters. Unlike a standalone cluster, edge devices do not run a full Kubernetes setup but instead rely on management from a cloud-based Kubernetes cluster. It lacks a proper network fabric needed for broader multi-cluster scenarios (e.g., edge-to-edge). *Orchestration transparency: ✓, Resource availability transparency: ✗, Communication transparency: ✓.*

Decentralized Kubernetes Federation Control Plane. L. Larsson et *al.* [11] propose a distributed and decentralized control plane for the Kubernetes federation. The approach foresees distributed federation controllers in each cluster, enabling local management of federated resources while maintaining global consistency through a shared database of conflict-free replicated data types (CRDTs). The shared CRDT database allows clusters to redistribute workloads dynamically based on resource availability. However, its scalability properties are unclear, and it does not provide an underlying network fabric. *Orchestration transparency: ✓, Resource availability transparency: ✗, Communication transparency: ✗.*

FLUIDOS ARCHITECTURE

The FLUIDOS software stack is composed of four main layers: (*i*) Vanilla operating systems abstract the underlying hardware capabilities. Currently, FLUIDOS is compatible with all major Linux distributions, ROS, as well as some embedded OSs. (*ii*) Kubernetes supports workload execution and introduces a uniform layer on top of different infrastructures. (*iii*) Liqo, which brings in a multi-cluster abstraction on top of Kubernetes, enables seamless offloading of workloads and transparent communication from one cluster to another. At the same time, it provides the three computing pillars described previously. (*iv*) FLUIDOS, which implements the other required Meta-OS capabilities.

FLUIDOS's main component is the *node*, which represents the abstraction of a set of compute resources in the continuum. Specifically, a FLUIDOS node may abstract a Kubernetes cluster with multiple worker nodes or degenerate to a single worker node. FLUIDOS nodes can interact *vertically* and/or *horizontally*. The vertical interaction introduces aggregation and hierarchical scaling into the picture. A FLUIDOS *domain* can be created by aggregating multiple nodes, hence exporting a virtual space that is the union of the resources (and services) of the composing nodes. On the other hand, the horizontal interaction enables the creation of a "stretched" FLUIDOS domain among peers, which can share their resources and services,

[3] https://docs.cilium.io/en/stable/network/clustermesh/intro.

or part of them, based upon a set of policies.

FLUIDOS nodes communicate with each other using REAR (REsource Advertisement and Reservation) [12], a novel protocol designed to enable entities within the continuum to advertise their resources and facilitate the reservation of those assets by consumers or applications. Currently, five different classes of assets are supported: Kubernetes slices (i.e., a slice of computing resources of a Kubernetes cluster), VMs, Services (i.e., implementing the so-called Software-as-a-Service), Data, and Sensors. Building upon the negotiation with the REAR protocol, FLUIDOS then enforces the creation and management of the agreed-upon resources in the node. FLUIDOS nodes then rely on the *Network Manager* to discover other FLUIDOS nodes and maintain their metadata. This discovery mechanism ensures that nodes can identify potential resource providers, allowing them to participate effectively in the REAR advertisement and reservation.

All this serves as an enabling technology for the *meta-orchestrator* [13], the FLUIDOS component in charge of dynamically [14] adjusting the shape of the local cluster, acquiring resources from other FLUIDOS nodes, in response to the submitted workload from the user. Specifically, the meta-orchestrator receives requests to deploy an application in the form of *intents*, which specify user requirements for the execution of the application. The intents allow to specify requirements beyond what is traditionally supported by Kubernetes, allowing the specification of performance characteristics – such as latency, throughput or available bandwidth between endpoints –, security and compliance, and energy efficiency or carbon emission constraints. Upon receiving application definition and associated intents, the meta-orchestrator will identify a *template* [15] of resource suitable for executing such workload in accordance with the user-defined constraints. The meta-orchestrator will then trigger the REAR protocol to identify the most suitable cluster to satisfy the user intent. After REAR returns the set of suitable clusters, the meta-orchestrator will then use this information to further refine the selection of the most suitable candidate with respect to characteristics that are shared through REAR [12]. This approach allows for a greater and agile specification of user requirements, alongside the implementation of a flexible mechanism for optimal selection of the cluster providing the resources to execute a given application [13].

Addressed research challenges

We now outline the major research challenges steaming from the FLUIDOS liquid computing, bridging the gap with known challenges [16].

Decentralization. The computing continuum inherently involves highly decentralized infrastructures. As a result, each entity in the continuum must be able to autonomously decide how to shape the boundaries of its own continuum. FLUIDOS leverages decentralized orchestration strategies by combining the REAR protocol and the meta-orchestrator to enforce a peer-to-peer-based interaction for resource management. However, a centralized approach is still supported through the vertical interaction previously discussed.

Data privacy and sovereignty. As data flows across heterogeneous domains, ensuring data privacy and sovereignty becomes paramount. With regulations like GDPR enforcing strict control over data crossing geographic or administrative boundaries, the need for well-defined privacy policies to govern data is becoming increasingly critical. To this end, resources negotiated with the REAR protocol embed information on compliance with certain regulations. Therefore, upon purchasing resources from other entities, clients know whether the processing and the corresponding data are compliant with the requested regulations.

Cross-domain authentication and authorization. The continuum involves interactions between multiple administrative domains , making authentication and authorization complex. To this end, FLUIDOS introduces a distributed authentication/authorization approach based on Distributed Ledger Technology (DLT) to validate the resources available in the continuum. In addition, FLUIDOS is based on self-sovereign identity (SSI) IdM where data isolation is achieved by placing control of data through verifiable credentials, which allow entities in the continuum to selectively share specific pieces of information without revealing their entire identity or exposing sensitive data. As a result, this approach provides runtime security by protecting against unauthorized access, data breaches, and manipulation of credentials and interactions during runtime. The strong source of trust provided by the DLT is used to further minimize risks in adversarial environments, as the policies requested for authorization are verifiable during runtime and can be checked by all participants.

Extension beyond Kubernetes. While Kubernetes is a dominant solution for edge/cloud container orchestration, it faces limitations at the far edge, where heterogeneous and resource-constrained devices are prevalent. FLUIDOS leverages KubeEdge in these cases, with the capability to share services and support transparent communications also in the above environment.

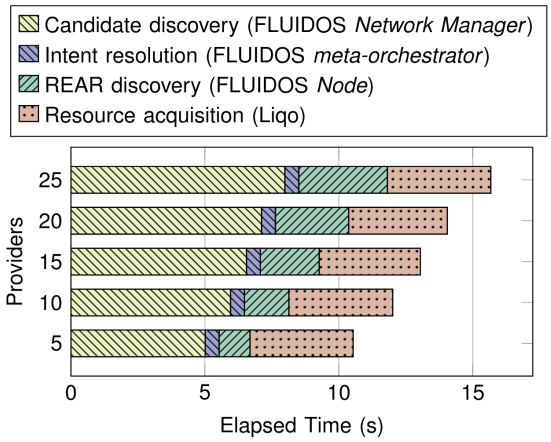


FIGURE 1: Time required to identify and acquire resources from suitable cluster candidates.

EXPERIMENTAL EVALUATION

In the following, we first evaluate the time required to identify and acquire resources from suitable candidates, we then explore a potential usage scenario for leveraging the newly acquired resources.

Resource acquisition

The time required to acquire resources in the continuum encompasses different stages as depicted in Figure 1. Specifically, the *Candidate discovery* time is defined as the time required to identify all available nodes in the continuum. Starting from a fleet of 25 initial nodes, the discovery time is measured as the time required by a single node to discover the first N other nodes, where $1 \leq N \leq 24$. The entire process is handled by the Network Manager, which supports two technologies: multicast and Distributed-Hash-Table-based (DHT)[4]. The former targets Local Area Networks (LANs), enabling the discovery of nodes connected to the same physical network, e.g., robots or networks of drones connected to the same 5G cell. In contrast, the latter targets Wide Area Networks (WANs), allowing FLUIDOS to support geographically distributed infrastructures. In the following, we will focus on the multicast-based discovery. Intents are then submitted to the FLUIDOS meta-orchestrator, which performs an *Intent resolution* phase to identify the resources that can satisfy the specific intent. Then, a *REAR discovery* phase probes for the available resources in the continuum, and, once the perfect candidate is identified, a *Resource acquisition* phase extends the local Kubernetes resource pool using Liqo.

A linear correlation emerges in Figure 1 for the *Candidate discovery* time, with the shortest discovery time recorded as 5 seconds for a five-node system (i.e., discovering and discovered nodes) and approximately 7.5 seconds to discover all other nodes. It is worth mentioning that the discovery phase is performed periodically, and results are cached locally; hence, this time is not strictly needed to be able to start the other phases. Similarly, the *REAR discovery* phase scales linearly with the number of nodes because it directly correlates with the number of sent messages. Instead, both the *Intent resolution* and the *Resource acquisition* phases are not correlated with the number of providers as (*i*) the meta-orchestrator adopts a recommendation system to identify the most appropriate set of resources to satisfy the user intent (i.e., without considering the availability in the continuum), and (*ii*) Liqo performs the resource acquisition only with the selected candidate. Overall, this entirely automated process converges in the worst case within approximately 15 seconds. Moreover, the observed linear trend in elapsed time suggests favorable scalability characteristics for FLUIDOS, favored also by the *vertical* interaction supported by the FLUIDOS nodes to replicate hierarchical structures.

Application offloading

Once FLUIDOS nodes become aware of the surrounding environment in the continuum, enhanced policies can be enforced to select the best candidate to peer with (using FLUIDOS terminology), based on the requested intent. This test leverages a FLUIDOS use case, in which a battery-constrained robot might dynamically offload part of the computation to nearby edge nodes or another (inactive/charging) robot while moving, hence reducing the energy drained from the local battery. The offloadable application consists of two components: navigation and obstacle detection. The navigation component is responsible for computing the path that allows the robot to autonomously travel from point A to point B. The obstacle detection component analyzes the camera feed to recognize and avoid obstacles during navigation.

The battery data is obtained by reading the output of the robot's Battery Management System (BMS), which measures the power consumption of the entire system, including sensors and actuators (such as wheel motors). Figure 2 depicts the resource usage for a single robot in case only local onboard processing is possible, and when the FLUIDOS-enabled liquid computing allows for task offloading on nearby nodes.

[4]DHT repository: https://gitlab.com/pi-lar/neuropil.

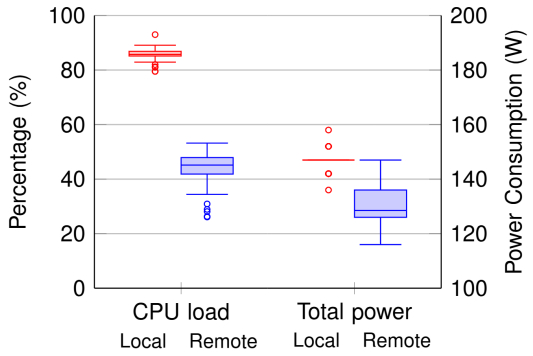


FIGURE 2: Robot dynamics when only local processing is permitted, and when the offloading is enabled.

Specifically, the left side represents the CPU usage measured on the robot, whereas the right side shows the resulting power consumption. When the robot is in offloading mode, the CPU load is almost halved: the motion planning is executed remotely, while the logic required to interact with sensors and actuators cannot be moved and hence still runs locally. The reduction in CPU usage also results in a non-negligible decrease in the power consumption for the entire robot (∼15%), given that other components (e.g., wheels, etc.) continue to drain power as usual. This significantly increases the duration of the battery and, consequently, the overall operation time.

Further considerations

This example illustrates an instance of FLUIDOS' dynamic orchestration capabilities: from decentralized discovery and intent resolution to runtime resource acquisition and cross-node deployment without operator intervention. These steps collectively constitute a fully operational computing continuum stack. While our evaluation focuses on a robotic offloading scenario, the core mechanisms are domain-agnostic and directly applicable to other sectors identified in Table 1:

- In **automotive** use cases, the same intent resolution logic can be used to migrate services as vehicles move between network cells.
- In **healthcare**, privacy-compliant service placement can be achieved by attaching regulatory metadata to ensure compliance in data manipulation.
- In **telecommunications**, dynamic peering and resource federation supports elastic continuum slicing in 6G scenarios.
- In **energy and utility**, distributed orchestration and edge-to-cloud integration ensure grid monitoring and automatic management, improving resiliency.

CONCLUSIONS

This paper introduced FLUIDOS, a meta-operating system that addresses key challenges in decentralized orchestration, resource sharing, and dynamic scaling in the computing continuum through deployment, communication, and resource availability transparency. Experimental results confirmed its effectiveness in optimizing resource utilization and energy efficiency. Future work will focus on short-term goals such as automated resource pricing, accounting for fluctuations in resource availability, SLA-aware negotiation and enforcement to guarantee the agreed-upon QoE for applications, and automated resource management for the different devices in the continuum. Longer-term directions, instead, will mostly focus on interoperability across heterogeneous domains to further solidify the continuum's role in modern computing, ultimately leading to a production-ready deployment.

ACKNOLEDGEMENTS

This work was supported by European Union's Horizon Europe research and innovation programme under grant agreement No 101070473, project FLUIDOS (Flexible, scaLable, secUre, and decentralIseD Operating System). This research was conducted as part of Jacopo Marino Ph.D. programme, under the financing of the Piano Nazionale di Ripresa e Resilienza (PNRR) and the NextGenerationEU initiative. Finally, we would like to thank Matteo Cicilloni for his support.

REFERENCES

1. S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque *et al.*, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. 23rd Int. Conf. Archit. Support Program. Lang. Oper. Syst.* Association for Computing Machinery, 2018, p. 751–766.
2. M. Z. Khan, S. Harous, S. U. Hassan, M. U. Ghani Khan, R. Iqbal, and S. Mumtaz, "Deep unified model for face recognition based on convolution neural network and edge computing," *IEEE Access*, vol. 7, pp. 72 622–72 633, 2019.
3. M. Iorio, F. Risso, A. Palesandro, L. Camiciotti, and A. Manzalini, "Computing without borders: The way towards liquid computing," *IEEE Trans. on Cloud Comput.*, vol. 11, no. 03, pp. 2820–2838, 2023.
4. L. Nagel, J. J. Hierro, E. Perea, D. Lycklama, C. Mertens *et al.*, "Design principles for data spaces: Position paper," E. ON Energy Research Center, Tech. Rep., 2021.

5. J. Marino, L. Camiciotti, F. Cheinasso, A. Olivero, and F. Risso, "Enabling compute and data sovereignty with infrastructure-level data spaces," in *Proc. 3rd Eclipse Secur. AI Archit. Model. Conf. Cloud Edge Continuum*. Association for Computing Machinery, 2023, p. 77–85.
6. L. Belcastro, F. Marozzo, A. Orsino, D. Talia, and P. Trunfio, "Edge-cloud continuum solutions for urban mobility prediction and planning," *IEEE Access*, vol. 11, pp. 38 864–38 874, 2023.
7. Y. Xiong, Y. Sun, L. Xing, and Y. Huang, "Extend cloud to edge with kubeedge," in *IEEE/ACM Symp. Edge Comput. (SEC)*. IEEE Computer Society, 2018, pp. 373–377.
8. M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.
9. T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: bringing the cloud to the mobile user," in *Proc. 3rd ACM Workshop Mobile Cloud Comput. Serv.*, ser. MCS '12. Association for Computing Machinery, 2012, p. 29–36.
10. T. Goethals, F. De Turck, and B. Volckaert, "Extending kubernetes clusters to low-resource edge devices using virtual kubelets," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2623–2636, 2022.
11. L. Larsson, H. Gustafsson, C. Klein, and E. Elmroth, "Decentralized kubernetes federation control plane," in *13th IEEE/ACM Int. Conf. Utility Cloud Comput. (UCC)*, 2020, pp. 354–359.
12. S. Galantino, E. Albanese, N. Asadov, S. Braghin, F. Cappa *et al.*, "Building the cloud continuum with REAR," in *10th IEEE Int. Conf. Netw. Softwarization (NetSoft)*. IEEE, 2024, pp. 67–72.
13. S. Braghin and L. Nedoshivina, "MIMO: a framework for extensible and flexible intent-based workload meta-orchestration," in *Proceedings of the 2nd International Workshop on MetaOS for the Cloud-Edge-IoT Continuum*, 2025, pp. 1–6.
14. J. M. Bernabe' Murcia, E. C. Martinez, A. M. Zarca, S. Braghin, and A. Skarmeta, "Cross-domain telemetry architecture: Real-time metrics in the computing continuum," in *Int. Conf. Mobile Internet Secur.*, 2024.
15. L. Nedoshivina, K. Levacher, K. Fraser, A. Halimi, and S. Braghin, "AI-driven workload management in meta os," in *Proc. 1st Int. Workshop MetaOS Cloud-Edge-IoT Continuum*, ser. MECC '24. Association for Computing Machinery, 2024, p. 14–20.
16. M. Tortonesi, "The compute continuum: Trends and challenges," *Computer*, vol. 58, no. 03, pp. 105–108, 2025.

Stefano Galantino is a Researcher at Politecnico di Torino, Torino, Italy. His research interests include cloud computing, orchestration, and energy-efficient computing. Galantino received his Ph.D. in Computer Engineering from Politecnico di Torino. Contact him at stefano.galantino@polito.it.

Jacopo Marino is a Ph.D. student at Politecnico di Torino, Torino, Italy. His research interests include cloud computing, orchestration of multi-cloud environments, and robotics at the edge. Marino received his Master's Degree in Computer Engineering from Politecnico di Torino. Contact him at jacopo.marino@polito.it.

Fulvio Risso is a Full Professor at Politecnico di Torino, Torino, Italy. His research interests include high-speed and flexible network processing, edge/fog computing, and network function virtualization. Risso received his Ph.D. in Computer Engineering from Politecnico di Torino. Contact him at fulvio.risso@polito.it.

Stefano Braghin is a Senior Research Engineer at IBM Research, Dublin, Ireland. His research interests include Security and Privacy of Distributed Systems, AI Security and Privacy, and Cryptography. Braghin received his Ph.D. in Computer Science from University of Insubria. Contact him at stefanob@ie.ibm.com.

Liubov Nedoshivina is a Research Engineer at IBM Research, Dublin, Ireland. Her research interests include AI security, data privacy, and AI-driven orchestration. Nedoshivina received her Master's degree in Computational Science from ITMO University and LUT University. Contact her at liubov.nedoshivina@ibm.com.

Antonio Fernando Skármeta Gómez is a Full Professor at University of Murcia, Murcia, Spain. His research interests include networking, security, and IoT. Skármeta received his Ph.D. in Computer Science from University of Murcia. Contact him at skarmeta@um.es.

Domenico Siracusa is an Associate Professor at the University of Trento, Trento, Italy. His research interests include infrastructure security, service orchestration, in-network processing. He received his Ph.D. in Information Technology from Politecnico di Milano. Contact him at domenico.siracusa@unitn.it.