2025 IEEE 33rd International Conference on Network Protocols (ICNP) | 979-8-3315-0376-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICNP65844.2025.11192462

inCoord: Intent-based Coordination in the Multi-domain Cloud-Edge Continuum

Andrea Morichetta [1], Juan Brenes [2], Mikhail Kolobov [2], Djawida Dib [3], Thijs Metsch [3], Anna Lackinger [1], Cveta Capova [1], Hui Song [4], Rustem Dautov [4], Ahmed Khalid [5], Sigmund Akselsen [6], Arne Munch-Ellingsen [6], Schahram Dustdar [1]
[1] *Distributed Systems Group, TU Wien*, Austria; [2] *Nextworks*, Pisa, Italy; [3] *Intel Labs, Intel Corporation*, Germany;
[4] *SINTEF*, Oslo, Norway; [5] *Dell Research, Dell Technologies*, Cork, Ireland; [6] *Telenor ASA*, Norway.

***Abstract*—The computing continuum aims to break the isolation of edge and cloud computing, creating a smooth and heterogeneous infrastructure surface for deploying applications. However, the continuum is practically fragmented, with infrastructure managed in isolation by various parties in a vertical dimension, i.e., edge, fog, and cloud layers, and horizontally, i.e., computing, network, and storage domains. This scenario negatively impacts the fulfillment of the application objectives. We propose *inCoord*, an intent-aware solution that enables the creation of a unified computing continuum by coordinating its instances to fulfill application objectives. Each instance represents a cluster of (compute, storage, or network) nodes with their own manager component. Traditionally, systems take low-level actions on these instances. In contrast, inCoord learns their emerging behaviors and adapts their managers' objectives to fulfill the application's intents. Here, through a Reinforcement-Learning-based Proof of Concept, we show the potential of this system to understand emerging behavior and manage multi-domain, independently managed instances.**

***Index Terms*—computing continuum, reinforcement learning, coordination, intent-based management, multi-domain**

I. Introduction

The promise of the computing continuum paradigm is to offer a seamless interface to the distributed infrastructure that spans from the edge to the cloud. However, in operational terms, a distributed infrastructure is still structured as a composition of clusters of nodes, i.e., *instances*, handled by different parties. Furthermore, whereas the focus is typically on computing nodes, managing real infrastructures also requires managing the network and storage components. Therefore, aspiring to have complete **control** over it is unrealistic. Existing solutions, like "Sky Computing" [1] or other commercial [1] [2] [3] ones, aim to build a unified interface for application owners. They offer the illusion of deploying on a continuous infrastructure, hiding how it is composed. While helpful, modeling the computing continuum as an orchestration abstraction layer does not solve the concrete problem of accurately capturing and handling the applications' real-time dynamics.

Taking the application owner's perspective, a key challenge they have in the computing continuum is to identify the right infrastructure **requirements** to guarantee a reliable runtime for their service. Intent-based management aims to solve this concern by letting users define high-level objectives and translating them into concrete infrastructure requirements. However, such frameworks typically support one single domain, such as the network [2] or, recently, the compute [3], [4]. The two presented problems are orthogonal, but both intersect at the same point: managing the computing continuum. The first concerns *infrastructure control*, and the second *application management*; both need to abstract away the inherent complexity of the computing continuum and create a smooth abstraction of it.

We address these gaps by introducing *inCoord*, an intent-aware solution for harmonizing infrastructure instances in the computing continuum. Given an application deployed in a set of instances, *inCoord* receives application-level and instance-level intents (or objectives). Each instance manager optimizes its strategy to guarantee its intent fulfillment. From a vantage point, *inCoord* captures emerging behaviors and their impact on the overall application intent. Consequently, it uses this information to adjust the instance-level intents, triggering an update of the instance managers' strategies. The novelty of this approach lies in not requiring a direct management of the infrastructure. *inCoord* makes it possible to rely on independently managed instances, leaving to the instance managers the freedom to deploy their strategies. Furthermore, in this way, *inCoord* creates a proper computing continuum abstraction. Indeed, it does not differentiate between in-premise or third-party nodes, nor network, compute, or storage domains, nor between edge or cloud layers. Finally, it allows for a holistic, self-adaptive strategy to harmonize the computing continuum, requiring minimal human intervention. *inCoord*'s principles make the selection of the self-adaptive method flexible, opening up the research of the best approach.

In this paper, we offer a detailed definition of our architecture, including how to obtain concrete application and instance objectives from high-level instructions, and how to adapt them at runtime. Furthermore, we implement a Proof of Concept to showcase *inCoord*'s potential. We set up a simulated environment with a video streaming use case, using Reinforcement Learning (RL), in particular Proximal Policy

This work has been supported by the European Union's Horizon Europe research and innovation program under grant agreement No. 101135576 (INTEND).

[1]Red Hat OpenShift: https://www.redhat.com/en/technologies/cloud-computing/openshift

[2]meshStack (meshCloud): https://www.meshcloud.io/en/product/core/

[3]IBM Cloud Pak: https://www.ibm.com/products/cloud-pak-for-integration

Optimization (PPO) for the self-adaptive strategy. The results show that *inCoord* has the potential to allow simple but effective harmonizing of instances in the computing continuum, given a high-level application intent.

The paper is organized as follows. In Section II we review the state of the art. Section III depicts an example of intent definition for the computing continuum, later evaluated with our use case. In Section IV we introduce the *inCoord* architecture and its mechanisms. Section V presents the results of applying *inCoord* in a simulated use case, while Section VI concludes the paper.

II. Related Work

Recent efforts in the field have investigated the use of intent-driven orchestration as a method for managing complex and dynamic computing environments. One approach [5] introduces the concept of incorporating tolerations into intents, allowing for greater flexibility in how objectives and expectations are met. This method aims to improve orchestration outcomes by enabling systems to continue operating effectively under varying or suboptimal conditions, which the approaches in this paper can leverage.

A broader examination of intent-based orchestration is provided in [3], [6], and [7]. In [3] the authors outline its core principles and associated benefits. These include abstraction from low-level control, improved automation, and enhanced scalability across distributed systems. The study emphasizes the utility of expressing high-level goals to facilitate more efficient and responsive orchestration processes. In [6] LLMs are used as the central decision-making entity to address violations of intent-driven performance criteria. It subsequently automates deployment reconfigurations based on the LLMs' recommended corrective actions. [7] proposes a generic approach for managing intent-driven orchestration of distributed applications. A hierarchical decision-making scheme is employed, where compute and network orchestration components collaboratively manage parts of the application. Expanding on these concepts, [8] presents a framework in which intents provide semantic context to resource providers. This additional context enables dynamic adaptation of resource behavior in response to changing conditions, contributing to more autonomous and context-aware orchestration to tackle sustainability challenges. [9] addresses the optimization problem for serverless applications based on high-level intents. A Reinforcement Learning-driven auto-scaling mechanism is used to optimize resource utilization. Additionally, a scheduling optimizer is proposed to minimize link utilization, transformation costs, and power consumption for efficient function placement. Together, these works support the shift toward modular and federated coordination models, where declarative specifications, rather than procedural control, guide the operation of heterogeneous systems across the computing continuum.

Several researchers have offered contributions to the coordination of intents through automated agents. Some solutions [10], [11] focus on using Reinforcement learning methods in the networking domain, with network service assurance as a first-class citizen. Another research stream uses RL for computing environments. While, e.g., Song et al. [12] propose an intent-based cognitive continuum for IoT-Edge-Cloud environments, other works focus on serverless orchestration [13] or distributed application management [14]. In contrast, our work focuses on cross-domain coordination, where reinforcement learning is used not within a single infrastructure domain, but to mediate and align objectives across compute, network, and storage domains simultaneously.

It is worth highlighting that from the standardization perspective, several standards and industry specifications, such as ETSI ZSM [15], 3GPP [16] and [17] and TMForum [18] propose intent-based management architectures with varying focus and approaches. However, Common to all is the definition of two roles (intent owner and intent handler), with high-level APIs for intent submission and status reporting. This article follows this approach and details how the computing continuum management can coordinate across different domains via standard intent-management APIs.

III. Example Intent

A video streaming service that delivers content to users over the internet, like Netflix, requires efficient processing of video frames and handling of large volumes of data to ensure high-quality playback. For instance, such a service can have a business intent of processing video frames in under 100 milliseconds. A coordination service's role is to disassemble this application-level intent into instance-level intents, as shown in Figure 1. In the network domain, this can be translated to an intent for network slices with latency below 50 milliseconds and bandwidth greater than 10Mbps. In the compute domain, the intent can be specified as a requirement to process video frames within 30 milliseconds. In the storage domain, the intent can focus on ensuring rapid access to video content with a read latency of less than 10 milliseconds and maintaining data availability at 99.99%. To accommodate real-world variability, occasional deviations from these targets can be tolerated, such as allowing processing times to reach 110 milliseconds during short periods of peak demand, while ensuring that 99% of the time, the processing remains under 100 milliseconds. This toleration is crucial for maintaining service quality without over-provisioning resources.

The way these intents are met is proprietary to each domain provider. The network provider can leverage knowledge about the available network resources in each area of the computing continuum to estimate the boundaries of the latency that can be provided in each area. The compute provider can use various scaling and resource allocation strategies to handle the computational demands efficiently. The storage provider can implement caching and replication techniques to enhance availability and reduce read latency. Instance managers continuously provide feedback to the coordination service, allowing for dynamic adjustment based on real-time data. Scoring functions can be employed to evaluate how well each domain is meeting its specific intent. This mechanism guides optimization efforts to ensure that the application-level intent is consistently achieved.

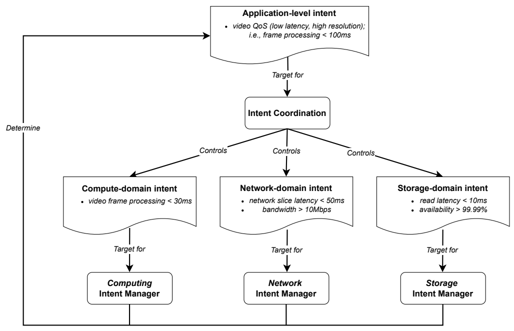


Fig. 1: Example of intent coordination.

IV. Intent coordination across domains

Here, we display the architecture for *inCoord*. The goal is to build a flexible and self-adaptive solution for coordinating multi-domain infrastructure instances, based on intents. In complex and hierarchical systems, such as the computing continuum [19], [20], switching from "how" to "what" is essential. Our key idea is to achieve that by only controlling the instance-level intents. We tune these intents, as knobs, to fulfill the application-level intent and handle conflicts across infrastructure domains. This approach makes the solution flexible and independent of intents and infrastructures, as it automatically triggers strategy adaptations for the involved instance managers.

A. Framework and methodology for coordination across domains

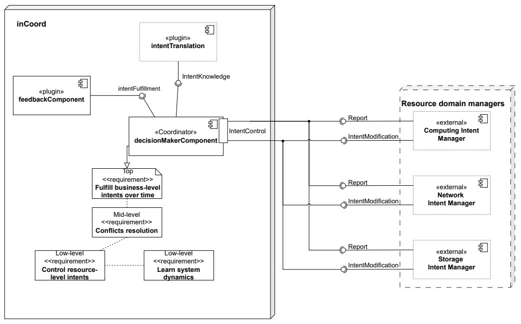


Fig. 2: Component diagram of *inCoord*'s main architecture and interfaces.

To have a clear overview of the *inCoord* role in a multi-domain computing infrastructure, we can analyze the main components depicted in Figure 2. *inCoord* operates based on high-level instructions translated and decomposed into actionable intents. This set contains at least one application-level intent that reflects the objective for the target application, and at least one instance-level intent for each domain instance. A dedicated component ("`intentTranslation`") performs this translation through an external tool or an internal plugin in *inCoord*. In [21] we implemented a dedicated "Intent-to-Learning" plugin that, through a multi-agent system built on Large Language Models (LLMs), translates application-level intents into executable Reinforcement Learning (RL) environments. The "`intentFulfillment`" holds the procedure, whether it is a rule or a more complex structure, such as an RL reward function, to evaluate whether the coordination successfully fulfills the application-level intent. This feedback can be produced by an external tool during the intent translation phase or manually constructed by a domain expert. Again, structuring it as a plugin leaves it open to adapting to every use case. In fact, at runtime, *inCoord* relies on telemetry from the infrastructure manager instances and application, continuously evaluating each in compliance with the defined instance-level and application-level intents, respectively. This information facilitates real-time decision-making.

The "`decisionMakerComponent`" is the *inCoord* core element, as it holds the strategy's logic. By learning the system's dynamics, this component decides at runtime whether to adjust the intent for each instance. The update actions consist of forwarding the new instance-level intents to each domain manager instance so that, in turn, they can make low-level decisions on the infrastructure. These recurrent adjustments have two main effects. The first one is to tailor the instance-level intents to the real requirements of the application. Even if carefully defined through some expert knowledge or application profiling, it is still possible that the intents might be different. For example, the computing demand might be relaxed, allowing the application owner to save on infrastructure costs. The second important aspect is resolving conflicts in a multi-domain, possibly multi-provider infrastructure, where conditions can change at runtime.

We envision *inCoord* interoperability through the definition of APIs. The API enables requirements "Ingestion," i.e., where intent-translation tools can submit both application- and instance-level intents; furthermore, *inCoord* offers reporting, where it shares current choices and metrics. Event-driven notifications could allow *inCoord* to receive real-time updates, e.g., violations, following a subscription model. For example, the framework proposed by TM Forum enables the reception of reports associated with the intents.

Overall, the *inCoord* architecture aligns with the TMForum proposed intent-management architecture, which on the one side enables the application owners to receive periodic reports regarding the fulfillment of the intent objectives, and on the other side provides mechanisms to determine the estimated value an intent handler can guarantee for a certain objective [22]. This latter is done through the *best intent report expectation* proposed in [22]. In practice, this allows *inCoord* to establish the boundaries for the application- and instance-level intents that impact the different resource level domains. For example, the application-level end-to-end latency of the video-streaming service needs to be balanced and coordinated between the latency of the compute resource domain (produced by the video stream processing), and that of the network connecting the application to the end-users. It is worth highlighting that both the application-level intent and internal coordination policies help determine tolerance for fulfilling a specific objective. This best-effort boundary estimation aligns with how *inCoord* continuously adjusts and re-aligns instance-level objectives across domains. In particular, the ability of *inCoord* to dynamically probe feasibility and adapt to real-time telemetry mirrors the intent of the BEST/PROPOSAL practice. The API structure and runtime loop within *inCoord* can incorporate such feedback by requesting boundaries from resource domains and refining goals accordingly, without overstepping domain-specific autonomy.

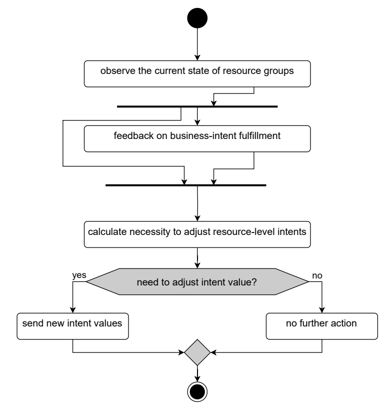

observe the current state of resource groups
feedback on business-intent fulfillment
calculate necessity to adjust resource-level intents
yes
need to adjust intent value?
no
send new intent values
no further action

Fig. 3: High-level workflow of the *inCoord* component.

Workflow For the *inCoord* runtime, we envision a control loop that improves its decision-making over time, allowing for self-adapting characteristics. Figure 3 summarizes the activity and highlights the major steps. In particular, the deployed method should learn which instance-level intent values are more appropriate. To do so, *inCoord* monitors whether each instance manager has breached its intent. If so, that likely means that it should adjust the targets. Furthermore, and most importantly, it should check the impact on the application-level intent, i.e., the application performance. Based on that, the mechanism should reason whether some updates are necessary and of what magnitude. Downstream, it should be able to understand its impact on the overall equilibrium. Therefore, the *inCoord* mechanism should be able to learn from previous experience and develop an understanding of the system dynamics.

V. Evaluation

To demonstrate the practical applicability of our coordination framework, we present a simplified use case based on the intent defined in Section III. The primary stakeholder in this scenario is a video streaming service provider whose business objective is to deliver consistent high-definition content to end users. The target objective is to maintain the service's quality of experience (QoE) without reducing the video quality (set at 1440p). For this use case, we limit the coordination to two manager instances, one for the network domain and the other for the computing one. Hence, the coordination

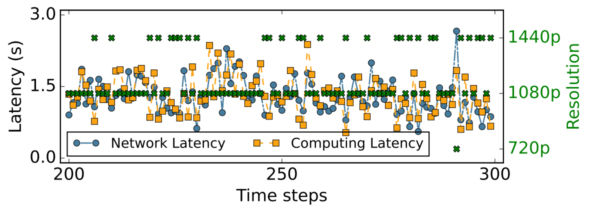


Fig. 4: Excerpt of the training dataset, with network and computing latencies and corresponding video quality.

agent must learn to balance resource allocation across these domains, understanding that improvements in one area might compensate for limitations in another.

Dataset: We create two datasets, the training and testing, with 1,000 and 10,000 elements, respectively. These synthetic datasets are based on aggregated performance data from an actual streaming service to ensure that our experiments reflect real-world operational patterns. Each data entry captures the multidimensional nature of streaming performance through key metrics: total *network latency*, which represents end-to-end network delay; *total computing latency*, which encompasses all necessary operations such as video encoding and processing overhead. These two can influence the *throughput*, thus reflecting available bandwidth for content delivery and, as a consequence, the *resolution*, i.e., the target metric. In the following Figure 4, we report a brief excerpt of the generated network and computing latencies for the training set, and the resulting resolution associated with these values. It is evident how large network and computing latencies lead to performance degradation, forcing a sub-par resolution of 720p.

A. *inCoord Mechanism Implementation*

In the context of self-adaptive systems, researchers and practitioners have developed several solutions, where they demonstrated the usefulness of Reinforcement-learning-based approaches [23]–[26] in multi-objective coordination strategies. However, even though there has been a focus on defining the best thresholds for auto-scaling using RL solutions [27]–[29], with promising results, no one has applied this whole approach to multi-domain use cases. With *inCoord*, we aim at achieving a self-adaptive coordination strategy for multi-domain infrastructure instances. In particular, for this evaluation, we compare three approaches.

1) **MLP-based DQN:** Based on the Deep Q-Network architecture proposed by DeepMind (2013), using deep neural networks for policy approximation.
2) **PPO:** Proximal Policy Optimization, introduced by OpenAI (2017), is a policy gradient method that restricts large policy updates for training stability.
3) **GRPO:** Group Relative Policy Optimization extends PPO by comparing an action to a sampled group of alternatives, emphasizing relative advantage.

The *inCoord* agent goal is to maintain the target streaming QoE by controlling latency intents for network and computing

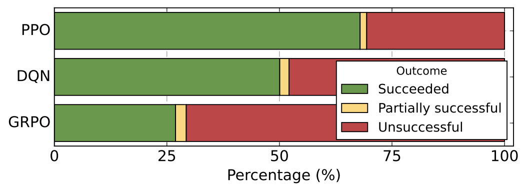
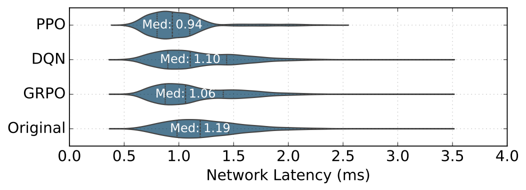
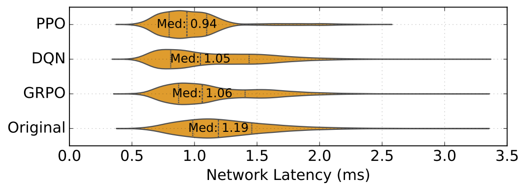


Fig. 5: Performance of RL models in maintaining 1440p video resolution



(a) Network latency change with each RL method.



(b) Computing latency change with each RL method.

Fig. 6: Violin plots showing the change introduced by the three tested RL solutions.

domain managers. The agent can decide to keep the intents as-is, to tighten either the network or computing objective, or both. The new latency target is the result of an adaptive step mechanism s that dynamically adjusts the magnitude of latency reductions based on current throughput R_c conditions. Rather than applying fixed reduction values, we employ an exponential scaling factor that becomes more aggressive when throughput is below target levels and more conservative when adequate throughput is achieved. We control the adaptation with the equation $s = \text{reduct}_{max} \times e^{\left(-\alpha \times \left(\frac{R_c}{R_t}\right)\right)}$ where α controls the aggressiveness of the reduction, reduct_{max} defines the maximum possible reduction step, R_c is the current throughput and the target throughput R_t is set to 15,000 kbps to support 1440p streaming.

B. Results

We evaluate the trained models on 10,000 unseen test entries, measuring their ability to maintain the target QoE. Outcomes are classified as success (target met), partial success (correct action direction but insufficient magnitude), or failure (no effective correction). As shown in Figure 5, PPO performs best, with over 70% successful actions, indicating its strong suitability for this type of coordination task. GRPO underperforms, likely due to poor adaptation to this scenario, calling for redesign or further tuning. DQN achieves moderate performance (around 50%), but may benefit from more sophisticated architectures or hyperparameter tuning.

The effectiveness of PPO is depicted in Figure 6a and Figure 6b, where it is immediately visible how the latency values are successfully reduced. The violin plots show the distribution of data points on variables. Each violin is drawn using a kernel density estimate of the underlying distribution. PPO is improving both computing and network latency, showing promising results in understanding the actions needed to guarantee better video quality. Notably, while PPO and GRPO exhibit similar latency profiles, the DQN behaves differently between the two. This difference is highlighted by Figure 7. There, we depict the distribution of actions taken by each method. We can see how, interestingly, DQN is performing a more diverse set of operations, by deciding to change either computing or network latency more frequently than the other two methods. In contrast, GRPO is very conservative, leading to significantly more violations than the other methods. Overall, PPO achieves good results by “simply” deciding to decrease both network and computing latencies. While this guarantees the best results, more tests are necessary to investigate solutions that offer more diversified actions, as this can help better solve conflicts. Finally, it is interesting to look at how the two more comparable models, and the ones for which we plan to perform more accurate tests, work during training. To do so, we analyze the cumulative reward. This inspection is standard in RL performance evaluation and allows us to see if and how an RL algorithm learns, over time, to pick the best actions. We depict the result in Figure 8. Surprisingly, while PPO clearly has a progressive improvement, our DQN algorithm seems to suffer during learning.

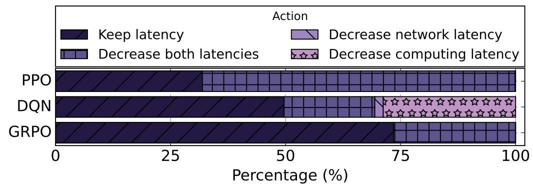


Fig. 7: Stacked bar chart of recommended actions for the three tested RL algorithms.

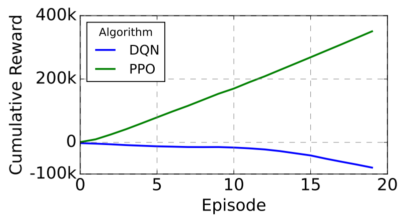


Fig. 8: Cumulative reward evolution for DQN and PPO over 20 episodes.

Takeaways These results show our approach’s capability to adapt in a dynamic environment, adjusting the infrastructure to maintain the application-level intent fulfilled. At the same time, the evaluation highlights the need to invest effort in adjusting the RL strategies. Both DQN and PPO would benefit from precise fine-tuning, which we intend to include in future work. Given the flexible architectures, we plan to introduce strategies based on other principles at a later stage, e.g., the promising active inference [30]. Finally, we aim to test the generalizability of our inCoord-based approach across different problems, incorporating additional instances from diverse domains, such as storage.

VI. Conclusion

In this paper, we presented *inCoord*, an intent-aware solution for harmonizing infrastructure instances in the multi-domain computing continuum. *inCoord*, conversely from other approaches, does not require direct infrastructure management. On the contrary, it allows instance manager to deploy their strategies. The control takes place by updating the instance-level intents, thus prompting changes in the managers' strategies. We offered a detailed definition of the *inCoord* architecture, together with a Proof of Concept. The simulation results in a video streaming use case show a low number of violations, highlighting the proposed solution's self-adaptive potential, given its capability to adapt quickly to the environment dynamics. In future work, we plan to extend this solution to more challenging and diverse use cases and include more diverse domain instances.

References

[1] I. Stoica and S. Shenker, "From cloud computing to sky computing," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, pp. 26–32, 2021.

[2] A. Leivadeas and M. Falkner, "A survey on intent-based networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 625–655, 2022.

[3] T. Metsch, M. Viktorsson, A. Hoban, M. Vitali, R. Iyer, and E. Elmroth, "Intent-driven orchestration: Enforcing service level objectives for cloud native deployments," *SN Comput. Sci.*, vol. 4, no. 3, p. 268, 2023.

[4] A. Morichetta, N. Spring, P. Raith, and S. Dustdar, "Intent-based management for the distributed computing continuum," in *2023 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 239–249, IEEE, 2023.

[5] M. D. Leite de Souza and T. Metsch, "Advancing orchestration: Leveraging distance functions to meet application intents," in *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*, pp. 82–87, 2024.

[6] N. Akbari, J. Grundy, A. Cheema, and A. N. Toosi, "Intentcontinuum: Using llms to support intent-based computing across the compute continuum," *arXiv preprint arXiv:2504.04429*, 2025.

[7] A. Zafeiropoulos, E. Fotopoulou, C. Vassilakis, I. Tzanettis, C. Lombardo, A. Carrega, and R. Bruschi, "Intent-driven distributed applications management over compute and network resources in the computing continuum," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, pp. 429–436, IEEE, 2023.

[8] T. Metsch and A. Hoban, "Breaking barriers: From black box to intent-driven, user-friendly power management," in *Proceedings of the 14th International Conference on Smart Cities and Green ICT Systems, SMARTGREENS 2025*, pp. 24–35, SCITEPRESS, 2025.

[9] N. Filinis, I. Tzanettis, D. Spatharakis, E. Fotopoulou, I. Dimolitsas, A. Zafeiropoulos, C. Vassilakis, and S. Papavassiliou, "Intent-driven orchestration of serverless applications in the computing continuum," *Future Generation Computer Systems*, vol. 154, pp. 72–86, 2024.

[10] S. K. Perepu, J. P. Martins, R. S. S, and K. Dey, "Intent-based multi-agent reinforcement learning for service assurance in cellular networks," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, (Rio de Janeiro, Brazil), pp. 2879–2884, IEEE, Dec. 2022.

[11] C. V. Nahum, V. H. L. Lopes, R. M. Dreifuerst, P. Batista, I. Correa, K. V. Cardoso, A. Klautau, and R. W. Heath, "Intent-Aware Radio Resource Scheduling in a RAN Slicing Scenario Using Reinforcement Learning," *IEEE Transactions on Wireless Communications*, vol. 23, pp. 2253–2267, Mar. 2024.

[12] H. Song, A. Soylu, and D. Roman, "Towards Cognitive Self-Management of IoT-Edge-Cloud Continuum based on User Intents," in *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, (Vancouver, WA, USA), pp. 1–4, IEEE, Dec. 2022.

[13] N. Filinis, I. Tzanettis, D. Spatharakis, E. Fotopoulou, I. Dimolitsas, A. Zafeiropoulos, C. Vassilakis, and S. Papavassiliou, "Intent-driven orchestration of serverless applications in the computing continuum," *Future Generation Computer Systems*, vol. 154, pp. 72–86, May 2024.

[14] A. Zafeiropoulos, E. Fotopoulou, C. Vassilakis, I. Tzanettis, C. Lombardo, A. Carrega, and R. Bruschi, "Intent-Driven Distributed Applications Management Over Compute and Network Resources in the Computing Continuum," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, (Pafos, Cyprus), pp. 429–436, IEEE, June 2023.

[15] "Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 3: Advanced Topics," Group Report GR ZSM 009-3 V1.1.1, ETSI, Sophia Antipolis, France, Aug. 2023.

[16] 3GPP, "3GPP TS 28.312: Telecommunication management; Study on management and orchestration of 5G networks and network slicing; Stage 1," Technical Specification 28.312, 3rd Generation Partnership Project (3GPP), 2023. Accessed: 2025-07-11.

[17] 3GPP, "3GPP TS 28.912: Telecommunication management; Study on architecture for next generation system," Technical Specification 28.912, 3rd Generation Partnership Project (3GPP), 2020. Accessed: 2025-07-11.

[18] TM Forum, "TMF921A Intent Management API Profile," Technical Specification TMF921A, TM Forum, Morristown, NJ, USA, Mar. 2022. Published 31 Mar 2022; TM Forum Approved.

[19] A. Morichetta, V. C. Pujol, and S. Dustdar, "A roadmap on learning and reasoning for distributed computing continuum ecosystems," in *2021 IEEE International Conference on Edge Computing (EDGE)*, pp. 25–31, IEEE, 2021.

[20] A. Morichetta, A. Lackinger, and S. Dustdar, "Cohabitation of intelligence and systems: Towards self-reference in digital anatomies," in *2024 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 102–110, IEEE, 2024.

[21] C. Capova, A. Morichetta, A. Lackinger, and S. Dustdar, "Intent translation for multi-domain cloud continuum management," in *2025 IEEE 33rd International Conference on Network Protocols (ICNP)*, IEEE, 2025.

[22] TM Forum, "Tr291c: Proposal of best intent model," tech. rep., TM Forum, 2022.

[23] G. Minelli and M. Musolesi, "Comix: A multi-agent reinforcement learning training architecture for efficient decentralized coordination and independent decision-making," *arXiv preprint arXiv:2308.10721*, 2023.

[24] P.-A. Dragan, A. Metzger, and K. Pohl, "Towards the decentralized coordination of multiple self-adaptive systems," in *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pp. 107–116, IEEE, 2023.

[25] C. Kröher, L. Gerling, and K. Schmid, "Control action types-patterns of applied control for self-adaptive systems," in *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 32–43, IEEE, 2023.

[26] N. Nguyen, D. Nguyen, J. Kim, G. Rizzo, and H. Nguyen, "Decentralized coordination for multi-agent data collection in dynamic environments," *IEEE Transactions on Mobile Computing*, 2024.

[27] S. Horovitz and Y. Arian, "Efficient cloud auto-scaling with sla objective using q-learning," in *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 85–92, IEEE, 2018.

[28] F. Rossi, V. Cardellini, F. L. Presti, and M. Nardelli, "Dynamic multi-metric thresholds for scaling applications using reinforcement learning," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1807–1821, 2022.

[29] T. Tournaire, H. Castel-Taleb, and E. Hyon, "Efficient computation of optimal thresholds in cloud auto-scaling systems," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 8, no. 4, pp. 1–31, 2023.

[30] T. Parr, G. Pezzulo, and K. J. Friston, *Active inference: the free energy principle in mind, brain, and behavior.* MIT Press, 2022.