IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 July 2026, accepted 28 July 2026, date of publication 31 July 2026, date of current version 6 August 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3719135

SURVEY

Edge–Cloud Orchestration for IoT Services: Technologies, Architectures, and Research Directions

CLAUDIO MARCHE, (Member, IEEE)
Department of Electrical and Electronic Engineering (DIEE), University of Cagliari, 09123 Cagliari, Italy
National Telecommunication Inter-University Consortium (CNIT), Research Unit of Cagliari, 09123 Cagliari, Italy
e-mail: claudio.marche@unica.it

This works has been supported by Fondazione di Sardegna, research project "DITTO - DIgital Twin as The bridge between meTaverse and internet Of things", CUP F23C25000300007, call 2023.

ABSTRACT The advent of 6G networks and the rapid growth of Internet of Things (IoT) are revolutionizing the telecommunication sector, integrating edge-cloud systems with vast amounts of data from IoT and AI techniques. These advancements make these systems essential in managing and delivering a multitude of services, offered by smart cities and industrial domains, being just a few among the many possible application scenarios. Specifically, this shift introduces complexities in orchestrating services and managing available resources while addressing challenges such as reducing latency, growing bandwidth, ensuring trust, and integrating different technologies. In this sense, this review explores the recent approaches of the state of the art focused on service orchestration in IoT edge-cloud environments, concentrating on architectures and methodologies that enable resource allocation and service management. Furthermore, it examines platforms for orchestration development, highlighting their characteristics and contributions. Finally, emerging concerns such as network topology and trust, which previous surveys have often overlooked, are discussed together with the most relevant research directions.

INDEX TERMS Internet of Things, edge-cloud continuum, service orchestration, 6G networks.

I. INTRODUCTION

In recent years, the convergence of edge and cloud computing has gained significant attention, especially in the context of the Internet of Things (IoT) and the evolution of communication networks towards 6G [1]. The increasing amount of data generated by IoT devices and the need for low-latency and high-bandwidth services for recent applications have exposed the limitations of traditional centralized cloud infrastructures [2]. Several architectures have emerged to address these new requirements, distributing services across the continuum between cloud and edge nodes, located closer to end devices [3]. These approaches have enabled a vast set of new services, but have also introduced a new level of complexity. In this scenario, service orchestration is essential to ensure that services are managed efficiently [4]. In general, orchestrating services in edge-cloud architecture is more challenging than in traditional cloud settings. Edge nodes often offer fewer computational capabilities, and network conditions can vary significantly with respect to the cloud. These factors demand solutions that should be able to operate in dynamic environments [5].

Unlike previous surveys, which typically focus on specific aspects such as edge or fog orchestration, AI methods, or Kubernetes-based solutions, this review provides an integrated perspective on IoT service orchestration across the edge–cloud continuum. In particular, the proposed study analyzes the problem of orchestration in the IoT scenario and combines four elements that are not adequately addressed together in earlier literature: i) a multi-dimensional taxonomy for classifying orchestration architectures, ii) a comparative analysis of recent representative works, iii) a discussion

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Quan.

of enabling platforms and their practical characteristics, and iv) an explicit focus on emerging concerns. For this reason, the paper is intended to discuss the gap between conceptual orchestration models and technology choices, while also highlighting the research directions that are most relevant for IoT edge–cloud systems. In fact, researchers who want to design an orchestration solution for these systems lack an integrated reference that connects the architectural and decision choices to the available platforms and to the concerns that are becoming central, such as network topology, trust, and sustainability. This survey aims to fill this gap, providing a perspective that links these aspects and derives from them the most relevant research directions.

In this context, this paper presents a review of the recent service orchestration in edge-cloud environments, which focuses on the diffusion of IoT services. Starting from the proposal of a taxonomy, the manuscript analyzes a set of recent orchestration architectures, focusing on their structural design, decision mechanisms, and the objectives they pursue. Moreover, an examination of how these architectures support service management is offered, highlighting their characteristics. The main contributions of this work are as follows:

- The paper proposes a taxonomy to analyze orchestration solutions in edge-cloud architectures, covering dimensions such as architectural structure, resources, orchestration mechanisms, allocation methods, objectives, application domains, and other provided functionalities.
- The taxonomy has been applied to the most widely recognized papers from the recent literature, providing an analysis of their characteristics.
- An analysis of the main enabling platforms is offered, discussing their strengths and limitations, and the metrics through which they are assessed.
- A set of open research directions is derived from the analysis, including emerging concerns that previous surveys treat only marginally.

The rest of the paper is organized as follows: Section II introduces the basic concepts of cloud-edge architectures and orchestration, Section III reviews existing surveys and outlines their contributions and limitations, Section IV discusses the main orchestration platforms, Section V presents the taxonomy and applies it to the selected architectures, Section VI offers a comparative discussion, highlights open research challenges and, finally, Section VII concludes the paper.

II. CONCEPTS

This section introduces concepts of cloud-edge architectures, highlighting general characteristics of service orchestration and the metrics used to evaluate the performance.

A. PRINCIPLES OF CLOUD-EDGE ARCHITECTURES

In recent years, cloud edge architectures have emerged to address the limitations of traditional cloud computing, particularly in applications requiring low latency and high bandwidth [6]. These architectures define a computing paradigm where processing capabilities are distributed between centralized cloud and decentralized edge nodes. Leveraging on both types of resources, i.e. cloud and edge, makes workload allocation possible, reducing reliance on distant data centres [7]. Their hierarchical structure represents the fundamental principle of these architectures. At the highest level, the cloud provides computational power and large storage, ensuring centralized coordination. At the lower levels, edge nodes, such as gateways and fog devices, perform intermediate processing and pre-filtering of data before transmitting it to the cloud [8].

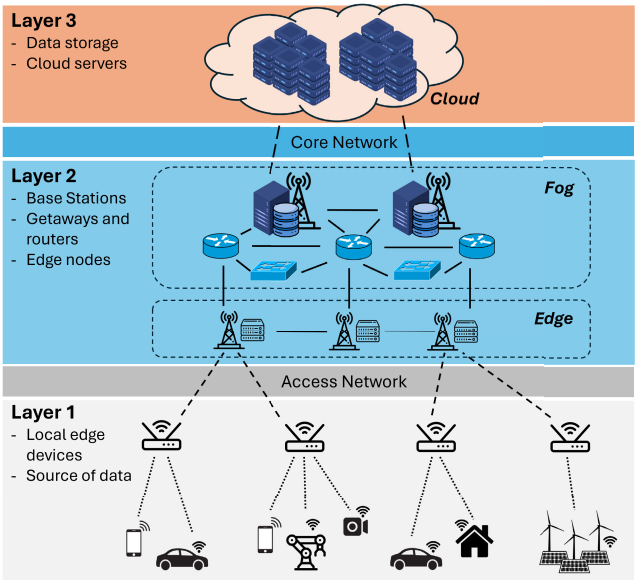

FIGURE 1. Generic cloud-edge architecture.

In this sense, the Figure 1 illustrates a general cloud edge architecture, structured into three levels: cloud, edge-fog, and device layers, interconnected by network segments. At the bottom, the Layer 1 constitutes the primary source of data, encompassing IoT devices such as smartphones, vehicles or industrial robots. These constrained devices connect to the upper layers through an Access Network, which provides the radio or wired connectivity between endpoints and the computing infrastructure above [9]. Layer 2 represents the intermediate level, organized into two sublayers. The Edge consists of base stations with computing servers that provide proximity processing for sensitive services, typically deployed one or a few network hops from end devices. Above it, the Fog includes gateways, routers, and dedicated nodes offering aggregation capacity and coverage. Fog nodes are located between the cloud and the edge, enabling execution of tasks from IoT devices to cloud data centers [10]. A core network segment then connects this intermediate layer to the cloud. At the cloud layer, data centres provide computational power and storage, while the second layer serves as an intermediary between the cloud and end devices. In this layer, nodes can perform localized data

processing, reducing the need for frequent communication with the cloud and minimizing latency. Each tier of this continuum offers a distinct set of processing capabilities depending on its proximity to the data source: the device and edge layers handle low latency; the fog layer supports more intensive but moderately latency sensitive tasks; and the cloud provides virtually unlimited computational power and storage [9]. Usually, this hierarchy is further characterized in terms of distance, noting that edge nodes operate within a kilometer from the data source, far-edge deployments cover distances of 1 to 100 kilometers, and cloud infrastructure operates on a global scale [11].

An important aspect is the communication that interconnects cloud and edge components. Modern networking paradigms, including 5G and the emerging 6G technologies, along with software-defined networking (SDN) as a network management paradigm, facilitate data exchange between edge devices and cloud servers, ensuring distribution and resource utilization [12], [13]. The integration of these technologies allows applications to determine whether computations should be executed at the edge or to the cloud based on factors such as latency, network congestion or energy consumption. Moreover, by distributing processes across multiple layers, these architectures mitigate the points of failure and improve fault tolerance, essential for today applications, such as industrial automation, healthcare, smart city environments, and other critical sectors [14].

As cloud-edge architectures evolve, their role in supporting computing paradigms becomes significant. The convergence of cloud and edge computing gives inputs for developing systems capable of more real-time processing and adaptive decision-making. In this context, the adoption of layered architectures brings concrete benefits for IoT deployments: fog computing reduces latency by processing data near the network edge, while edge computing ensures data handling with lower transmission delays compared to cloud models [15].

B. SERVICES' ORCHESTRATION

In recent cloud-edge architectures, services play a fundamental role in enabling distributed applications, as they represent the execution units of applications [16]. In architectures involving cloud and edge computing, particularly in IoT scenarios, services identify tasks such as data processing, storage and analytics; so, to ensure optimal performance and resource utilization, they must be orchestrated efficiently [17]. In general, service orchestration refers to coordinating the deployment, execution and management of services across the cloud-edge continuum. Unlike traditional cloud-based orchestration, where resources are centrally managed, service orchestration aims to ensure the distribution of processes and service continuity across multiple entities [18].

Service orchestration encompasses several important functions [19]. Resource provisioning and process scheduling determine where services should be deployed and split into tasks assigned to available resources. Moreover, service composition ensures the integration of complex services, while management takes care of monitoring and fault recovery [20]. Unlike cloud data centres, which offer uniform resources in controlled environments, edge nodes vary in processing and energy availability. This requires decentralized orchestration strategies that incorporate task migration and light virtualization techniques, such as microservices [21]. Security and privacy are also important considerations. Edge nodes often operate in less secure environments, increasing vulnerabilities. Orchestration techniques must integrate encryption, identity management, and trust mechanisms [22].

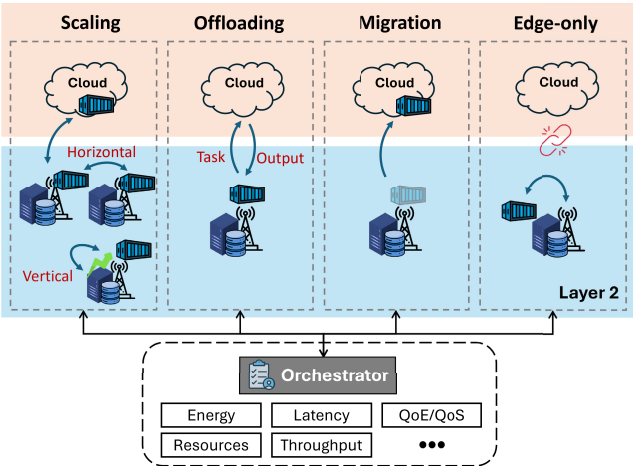

FIGURE 2. Orchestration strategies.

The main challenge of service orchestration lies in determining where and how to execute application components, i.e. services, since edge nodes have limited capacity and computation capabilities. Mechanisms must therefore balance responsiveness and efficiency under dynamic conditions, adapting to the requirements of each specific scenario [23]. In this context, different provisioning models are adopted to allocate services once orchestration decisions are made. Figure 2 illustrates four strategies over the cloud and edge-fog tiers. In the scaling strategy, instances of a service are replicated and distributed to follow the variation of the demand, through horizontal scaling, by adjusting the number of instances (e.g., increasing with more instances through the edge or the cloud), and vertical scaling, by resizing the resources assigned to a single instance [24]. In the offloading strategy, a node delegates a task to a more capable node, which executes it and returns only the result, so that the computation does not run on the source [19]. This is a task decision on whether and where a workload should be executed, and recent advances have shifted it from static choices towards dynamic ones taken at runtime according to network and device conditions. The migration strategy, instead, relocates an already running service together with its state between nodes, preserving service continuity when the operating conditions change [25]. Finally, in the edge-only

strategy, all computation is confined to the edge, minimizing delay and ensuring resilience when the connectivity to the cloud is intermittent or unavailable. It is important to note that offloading, migration and scaling are not alternatives to orchestration, but rather among the operations that orchestration coordinates as part of the whole system. As shown at the bottom of Figure 2, the selection among these strategies is operated by the orchestrator on the basis of a set of performance metrics, such as latency and resource usage, which are analyzed in detail in the next subsection.

C. PERFORMANCE METRICS

As anticipated by the orchestrator block in Figure 2, the choice of an orchestration strategy relies on a set of performance metrics, which are now described in detail. The performance of these architectures is typically evaluated through a set of metrics that assess efficiency and resource utilization [26].

Among the most used, latency remains central, as it measures the time required for data to travel between edge nodes and cloud servers, including both processing and response times. In IoT scenarios, latency is often further specified as end-to-end delay, jitter, or deadline miss ratio, since many applications (e.g., industrial automation or connected vehicles) are sensitive to these variations. Throughput complements latency by reflecting the rate at which data can be transferred across the system, which is important in services characterized by large traffic.

Resource utilization is another important metric, evaluating how computational and network resources are allocated across nodes. It is often coupled with scalability, which expresses the ability of orchestration mechanisms to maintain performance as the number of devices, services, or users increases. Scalability in IoT is particularly challenging because it must account not only for demand but also for the heterogeneity of devices. Availability and fault tolerance extend performance evaluation by measuring system continuity under failures, with mechanisms such as replication or migration.

Moreover, in recent years, additional metrics have gained importance. Energy consumption has become another primary concern, as orchestrating services close to the edge can increase performance but often raises power usage, especially when replicas are involved. In this sense, a related metric is sustainability, with works emphasizing the need to consider the environmental impact and even the carbon footprint [27]. Furthermore, two emerging metrics are represented by security and trust, recognized not only as qualitative requirements but also as performance factors: ensuring secure and reliable data exchange introduces computational and communication overhead that must be balanced against latency and throughput [28]. Finally, more user-centric indicators, such as Quality of Service (QoS) and Quality of Experience (QoE), are important to capture the perceived performance of applications, particularly in scenarios involving interaction or multimedia streaming.

III. EXISTING SURVEYS

In recent years, cloud-edge orchestration has attracted attention, and several surveys have addressed it from different perspectives. As follows, the most relevant surveys are organized into groups that reflect their focus, and it is discussed how the works present deficiencies with respect to the scope of the present study.

A first group of surveys addresses orchestration from the perspective of edge and fog computing foundations. Three works fall into this category, presented in [19], [24], and [29]. The first work analyses the orchestration of edge computing for IoT, focusing on task offloading, resource allocation, and network virtualisation, and it organises management and orchestration techniques through a taxonomy. The second work studies orchestration in edge and fog environments and derives a generic architecture from the literature, classifying fog orchestration techniques by control topology, layers, and management functions. The third work explores edge architectures for IoT, categorising them by data placement, orchestration services, and security, and proposing representative usage scenarios for edge computing. While these surveys establish solid foundations, they remain related to edge and fog and, although they include the cloud layer, they do not adopt a continuum perspective; moreover, they give limited attention to emerging orchestration concerns, such as trust, that fall outside their original scope.

Furthermore, another group of surveys adopts a technology perspective, centred on container platforms and their adaptation to the edge. Two works are representative, presented in [22] and [30]. The first work examines the use of Kubernetes for cloud-edge orchestration in smart-city settings, reviewing scheduling metrics and network topology, and discussing limitations such as container registry placement and fault tolerance. The second work focuses on cloud-native techniques adapted to edge computing, analysing containerised tooling and deployment methodologies for constrained environments while discussing reliability and infrastructure heterogeneity. These surveys offer concrete insight into enabling platforms, but their scope is only related to specific technologies; they do not generalise into orchestration mechanisms, and they miss important factors as new open issues, such as trust and service feedback evaluation.

A third group is organised around classification, proposing taxonomies that are more focused on the orchestration mechanisms. Two works adopt this perspective, presented in [28] and [8]. The first work targets the cloud-to-things continuum and introduces a taxonomy built on deployment models and orchestration strategies, identifying gaps such as the absence of standardised application descriptions and the limited interoperability across platforms. The second work centres on the role of artificial intelligence in orchestration for IoT, classifying architectures and data-processing techniques and emphasising machine learning approaches together with

TABLE 1. Comparison of existing surveys on cloud-edge orchestration.

Ref.	Focus	Multi dimension Taxonomy	Continuum Perspective	Platform Analysis	Main Deficiencies
[8]	Cloud-edge orchestration and AI-powered data processing for IoT	✓	✗	✗	AI/data-processing focus; no platform analysis; open issues centred on AI rather than on the orchestration mechanism
[10]	Computation offloading in the edge-cloud continuum; decentralised and federated solutions	✓	✓	✗	Bounded to the offloading operation with a decentralised and federated focus; no platform analysis
[11]	Edge-cloud continuum, developer-centric	✗	✓	✓	No multi-dim. orchestration taxonomy; limited attention to emerging research directions (e.g., topology, trust)
[19]	Management and orchestration of edge computing for IoT (offloading, caching, resource management)	✓	✗	✗	Edge-centric framing, not the full continuum; limited link between taxonomy and emerging open issues (e.g., topology, trust)
[22]	Kubernetes for cloud-edge orchestration (smart cities)	✗	✗	✓	Bounded to Kubernetes; no cross-platform taxonomy; open issues confined to K8s limitations
[27]	Sustainable orchestration of containerised applications	✓	✓	✗	Sustainability-specific; no platform analysis; open issues related only to the green-orchestration scope
[28]	Cloud-to-things orchestration; scalability, security	✓	✓	✓	Broad coverage but limited treatment of recent open issues
[24]	Orchestration in fog computing; generic architecture	✓	✗	✗	Fog-centric; no enabling-platform analysis; future directions only loosely connected to recent open issues
[29]	Edge-computing architectures for IoT	✗	✗	✗	Taxonomies of edge architectures and IoT apps, but not a multi-dimensional orchestration taxonomy; no continuum view; limited research-direction analysis
[30]	Cloud-native techniques adapted to edge computing	✗	✓	✓	Technology-bounded; no orchestration taxonomy; narrow discussion of open research directions
[31]	IoT-edge-cloud continuum systems; use cases, open issues	✗	✓	✗	Use-case driven; no orchestration taxonomy or platform analysis; recent works and open directions only partially covered

open issues on latency, privacy and resource allocation. Although both provide classifications, their taxonomies privilege deployment and learning aspects, and they pay limited attention to the emerging research directions.

A distinct line of work focuses on a single orchestration operation, that of computation offloading. The work presented in [10] surveys offloading in the edge-cloud compute continuum, with a specific focus on decentralised and federated execution models. A multi-dimensional taxonomy is proposed, classifying the solutions by architecture, in terms of communication pattern and consensus protocol, by strategy, in terms of operating type, resource allocation, and algorithm, by optimisation objective, and by evaluation environment. While this survey adopts a continuum perspective and a taxonomy, its scope remains bounded to the offloading operation and to decentralised execution; it does not extend to the set of orchestration mechanisms considered here, nor to the analysis of enabling platforms.

A further perspective concerns environmental sustainability. Regarding this aspect, in [27] the authors investigate the orchestration of containerised applications with attention to energy consumption and carbon footprint, categorising approaches by technological and model aspects and discussing the trade-offs with QoS as well as the need for open datasets and testbeds. The survey raises sustainability to a primary concern, yet it remains specific to the green scope and does not extend to enabling platforms or to a further set of orchestration mechanisms.

Finally, a more recent group embraces the complete edge-cloud continuum as the focus of analysis. Two surveys are representative, presented in [11] and [31]. The first work offers a developer perspective, detailing the architecture of the continuum and the nomenclature of its layers, evaluating public and open-source platforms, and providing comparative tools and benchmarking strategies oriented to real deployments. The second work surveys IoT-edge-cloud continuum systems, discussing their status, use cases, and open issues across the layers. These works are the closest to the present study; however, neither applies a multi-dimensional taxonomy, and the emerging concerns highlighted in this work, among them the network topology and trust, are treated only marginally.

Table 1 summarises the analysed surveys along five dimensions: their main focus, the presence of a multi-dimensional taxonomy, the adoption of a continuum perspective, the analysis of enabling platforms, and their main

deficiencies. From a quantitative view, the comparison reveals an imbalance: among the eleven surveys, six present a multi-dimensional taxonomy, six adopt a continuum perspective, and four analyse enabling platforms, while only one of them combines all three. Even in that case, the emerging research directions that are relevant for the continuum remain largely unaddressed, among which the network topology and trust are only two examples. In contrast, these aspects are analysed in this study. A continuum perspective is adopted, and a multi-dimensional taxonomy is applied to a set of recent architectures. The main enabling platforms are analysed together with their characteristics, and the discussion is extended to the open research directions.

IV. ENABLING TECHNOLOGIES

As illustrated in the previous sections, cloud-edge architectures make use of several platforms to manage services dynamically. These platforms enable coordination, optimize resource allocation, and enable the migration of services across the layers. As follows, the main platforms are illustrated; the aim is not to examine in depth all the platforms in the state of the art, but to focus exclusively on illustrating the most well-known and used.

Kubernetes represents the most popular orchestration platform, originally developed by Google and now maintained by the Cloud Native Computing Foundation (CNCF) [32]. Its modular control plane, composed of the API Server, Controller Manager, and Scheduler, automates the deployment, scaling, and management of containerized applications across distributed clusters. The Scheduler assigns pods, which are the smallest deployable units, to nodes based on resource availability and predefined constraints, while controllers ensure consistency between the desired and current system state. The cluster state is preserved in a key-value store, while an autoscaler supports horizontal scaling by adjusting the number of pods according to CPU and memory utilisation. On each worker node, an agent manages the lifecycle of the pods, whereas a proxy opens the ports and forwards the traffic towards the deployed services. Kubernetes provides abstractions for networking, giving each pod a unique IP and exposing services through load balancing. A further element of flexibility is the modularity of the control plane, since the default scheduler can be replaced and additional labels can be attached to the nodes to enrich the information used during placement. Although primarily designed for centralized cloud environments, distributions such as K3s and adaptations for constrained infrastructures have extended its applicability to edge deployments. Nevertheless, several challenges remain, e.g. native Kubernetes lacks awareness of network topology, and cannot fully exploit real-time metrics for scheduling. In its default configuration, placement decisions rely on static resource demand and supply and pods are scheduled individually, which limits the suitability of the platform in edge scenarios sensitive to latency.

KubeEdge extends Kubernetes' capabilities to edge computing environments [33]. On the cloud side, the CloudHub acts as a WebSocket server that manages connections, caches data, and delivers messages to the edge, while the EdgeController extends the default Kubernetes controllers to handle metadata and synchronize configurations. On the edge side, the EdgeHub maintains communication with the CloudHub and ensures state consistency. An important feature of KubeEdge is its support for the MQTT protocol, enabling publish and subscribe communication between IoT devices and the orchestration layer. This exchange relies on a centralized MQTT broker that collects the messages from the clients and acts as a gateway for the sensors. Through this architecture, KubeEdge aims to enable deployment, monitoring, and management of containerized services close to data sources, reducing latency and supporting IoT applications. A capability specifically designed for edge deployments is the handling of unstable cloud-edge connectivity, since the synchronization between the two sides is performed only under stable network conditions. The framework also integrates a mesh component that provides service-to-service communication and discovery across the layers of the continuum, distributing the network metadata to operate under complex network conditions. Its scalability has been demonstrated in deployments orchestrating up to one million pods across one hundred thousand edge nodes. Finally, the interaction with constrained devices is mediated by dedicated mappers that support industrial protocols such as Bluetooth and Modbus. Beyond KubeEdge, other projects extend Kubernetes towards the edge. One of these is represented by OpenYurt, an open-source platform initiated by Alibaba that brings the orchestration to edge nodes while retaining the standard Kubernetes APIs [34]. Its design is generally non intrusive, since the edge capabilities are added through components on top of a cluster rather than by rewriting the native ones, which eases the adoption for those already familiar with Kubernetes. A central aspect is the edge autonomy, which allows the nodes to keep operating and to maintain the local services even when the connection to the cloud control plane is temporarily lost. Comparative evaluations report that both KubeEdge and OpenYurt preserve the system state and the transit of messages during network outages, improving resiliency under the intermittent connectivity that is common at the edge, although these mechanisms increase the resource consumption and lengthen the recovery time with respect to more minimalist distributions.

Furthermore, other two well-known platforms are represented by OpenFaaS [35] and Cloud Foundry [36]. OpenFaaS is an open-source framework that brings serverless computing to container orchestration environments, with native support for Kubernetes. It allows developers to package functions as containers, which are then deployed and managed by the platform. The architecture is composed of a gateway that routes requests, a set of function containers, and a processing mechanism for handling queued events. Functions can be triggered through multiple sources, including HTTP requests and IoT device events. OpenFaaS aims to reduce

latency by executing functions closer to data sources and can be integrated with edge platforms such as KubeEdge. This allows the deployment of analytics and preprocessing directly at the edge, before forwarding aggregated results to the cloud. Besides Kubernetes, the framework can also be executed on constrained environments through a light runtime that manages the functions without requiring a full cluster. The energy behaviour of the platform has also been characterized, observing that the lifecycle of a serverless function spans a startup, a runtime, and an idle stage; while the startup involves the loading of the container images and the required libraries, the largest share of the energy consumption occurs during the runtime stage and is due to the running functions rather than to the platform itself, the idle stage being the least demanding. Its simplicity has led to adoption in IoT and smart city applications. On the other hand Cloud Foundry is a platform-as-a-service (PaaS) designed to abstract the complexity of managing container orchestration, offering an environment for the deployment of applications. Unlike Kubernetes, which requires container and cluster management, Cloud Foundry provides a runtime that automates build, deployment, scaling, and lifecycle management of applications. Applications are packaged with their dependencies using the buildpack model, without direct interaction with the orchestration layer. Recent adaptations have integrated Cloud Foundry with Kubernetes clusters, leveraging Kubernetes abstractions while maintaining the PaaS simplicity for developers. However, its reliance on centralized components makes it less suited to decentralized scenarios. A further serverless solution, related to Open-FaaS, is Knative, an open-source platform built on top of Kubernetes [37]. Its functionality is organized into a serving component and an eventing component: the first relies on an autoscaler that adjusts the number of instances according to the incoming requests and can scale an application down to zero when no traffic is received, whereas the second follows a model aligned with the CloudEvents specification, a vendor standard that defines a format for describing events, to favour interoperability. Relying on a declarative configuration model inherited from Kubernetes, Knative aims to provide a portable serverless layer whose applications can be moved across different providers.

Another important orchestration framework is OpenNebula, which focuses on virtual machine orchestration rather than containerized services [38]. Built as an open-source cloud management platform, OpenNebula enables the deployment and migration of virtualized services across cloud and edge infrastructures. Its support for federated cloud management allows organizations to interconnect different providers, while exposing functions for optimizing resource consumption and reducing costs. OpenNebula has also been extended with initiatives such as ONEedge5G, which introduce forecasting and resource optimization for 5G edge infrastructures. In this extension, a separation is drawn between the management of the resource lifecycle and the orchestration layer, in which workload forecasting and anomaly detection are combined with a manager that performs horizontal and vertical autoscaling. On this basis, the placement of the virtual resources is formulated as an optimization problem and solved to map them onto the edge servers according to criteria such as the minimization of the number of active servers, the balancing of the load, and the reduction of latency violations. Despite its flexibility, its reliance on VMs may limit its integration with modern container-based architectures, where Kubernetes is more commonly adopted. Moreover, StarlingX represents another platform tailored to distributed edge environments [39]. Developed under the Open Infrastructure Foundation, it integrates multiple open-source components, including Kubernetes, OpenStack, Ceph, and fault-management services. StarlingX supports both virtual machines and containers, enabling deployments in sectors such as telecommunications, industrial IoT, and aerospace. Important features include redundancy, monitoring, and service failover. More in detail, the platform exposes dedicated configuration, host, fault, and service management components, while a software management service enables upgrades and the migration of the workloads off a node during maintenance. A further component, built on top of OpenStack, coordinates from a central cloud the deployment and configuration of the distributed edge sites. StarlingX aims to address the challenges of geographically dispersed infrastructures where service continuity is essential.

A different class of platforms addresses the orchestration of network functions rather than generic services. In this context, Open Source MANO (OSM) is an open-source project developed by ETSI that implements the management and orchestration component of the NFV architectural framework, acting as an end-to-end network service orchestrator [40]. The lifecycle of the virtualised network functions is coordinated by interacting with one or more virtual infrastructure managers, such as OpenStack, which instantiate the functions and the virtual links that connect them into network services. The deployment is driven by descriptors and configuration models that automate the instantiation and the subsequent operations, while the interaction with software-defined networking controllers extends the coordination to the transport network. These characteristics make OSM particularly suited to telecommunication and 5G scenarios, where network services must be deployed and managed across distributed infrastructures.

Finally, Baetyl is an open-source edge framework initiated by Baidu and hosted by the Linux Foundation Edge [41]. It is designed to extend cloud capabilities to edge devices, providing services such as message routing, stream processing, and AI inference. Baetyl supports offline operation, ensuring that applications can continue to run even when connectivity to the cloud is disrupted. Its architecture, running in containers and supporting Kubernetes integration, aims to enable the deployment of edge services. Real-world use cases, such as environmental monitoring systems on devices like Raspberry Pi, demonstrate its ability to reduce latency and packet loss

TABLE 2. Comparison of enabling technologies for cloud–edge orchestration.

Platform	Strengths	Limitations	Use Cases	Metrics considered
Kubernetes [32]	Most popular container orchestration; modular control plane; scalability mechanisms	Limited awareness of network topology; limited use of real-time network metrics	General-purpose orchestration across cloud and edge clusters	Latency; Throughput; Resource usage (CPU, Memory, Storage); Scalability
KubeEdge [33]	Extends Kubernetes towards edge sites; device/edge connectivity and cloud-edge coordination	Additional edge components; potential overhead/scalability limits in large deployments	Smart cities, industrial IoT, real-time monitoring	Latency; Throughput; Resource usage (CPU, Memory, Storage); Scalability
OpenYurt [34]	Non-intrusive extension of Kubernetes to the edge; standard APIs; edge autonomy and message preservation under interruptions	Higher resource consumption and longer recovery time; added maintenance complexity	Edge sites with intermittent connectivity; resilient IoT deployments	Recovery time; Resource usage (CPU, Memory); Availability
OpenFaaS [35]	Lightweight serverless; event-driven functions; integrates with container platforms	Cold-start overhead; less suitable for long-running services	Real-time analytics, preprocessing, anomaly detection at the edge	Latency; Throughput; Success rate; Resource usage (CPU, Memory)
Knative [37]	Serverless layer on Kubernetes; event-driven via CloudEvents; request-based autoscaling with scale-to-zero	Oriented to synchronous HTTP workloads; inherits Kubernetes complexity	Event-driven IoT, microservices, data processing	Latency; Request rate; Concurrency; Scalability
Cloud Foundry [36]	Developer-friendly PaaS abstraction; automated lifecycle management	Centralized components can be limiting for highly decentralized edge sites	Enterprise applications, hybrid cloud deployment	Latency; Request rate; Throughput; Resource usage (CPU, Memory, Storage)
OpenNebula [38]	VM orchestration for geo-distributed edge; automation and optimization extensions (e.g., ONEedge5G)	VM-based approach may be heavier than containers; integration complexity	Telecom/5G edge infrastructures, distributed edge sites	Latency; SLA compliance; Resource usage (CPU, Memory, Storage); Forecasting accuracy
StarlingX [39]	Integrated HA edge-cloud stack; fault management and redundancy	Complex architecture; higher resource overhead than minimalist frameworks	Telecommunications, aerospace	Availability; Recovery time; Resource usage (CPU, Memory, Storage)
OSM [40]	ETSI NFV management and orchestration; end-to-end network service orchestration; multi-VIM support (e.g., OpenStack)	NFV/telco; setup complexity and VIM interoperability	Telecommunications, 5G network services	VNF deployment time; Resource usage (CPU, Memory); Availability
Baetyl [41]	Lightweight edge framework; offline operation; stream processing and AI inference	Smaller ecosystem; fewer standardized benchmarks than Kubernetes-based stacks	Environmental monitoring, AI-enabled IoT on constrained devices	Latency; Connectivity resilience; Message reliability

compared to cloud processing. Baetyl has emerged as a lightweight framework for IoT applications at the edge.

Table 2 summarizes the main enabling technologies for cloud–edge orchestration, comparing their strengths, limitations, and typical use cases. Moreover, it adds a view on the main metrics used to assess these platforms.

V. EDGE-CLOUD ORCHESTRATION: PROPOSED ARCHITECTURES

This section analyzes the most recent representative edge-cloud orchestration architectures proposed in the recent literature. The analysis is structured around a classification to compare different approaches and their important aspects. To this end, a taxonomy is introduced, and its aspects are described to analyse the approaches.

A. TAXONOMY

The proposed taxonomy includes seven dimensions, which highlight the design aspects of the proposal. The first dimension concerns the architectural structure, which describes how orchestration entities are distributed across the infrastructure. In this sense, common models include monolithic or layered designs, hierarchical topologies and federated schemes. The second dimension focuses on resource control, i.e., where decisions are handled. This may be centralized, distributed, or follow a hybrid model combining both categories. Moreover, a third aspect is the type of orchestration, referring to the logic used to manage services across the architecture. This includes AI orchestration and systems based on policy or events. The fourth dimension concerns the resource allocation method, distinguishing between static strategies, where deployment decisions are taken a priori, and dynamic ones that adapt to changing conditions. In addition, each architecture typically targets a specific objective, such as minimizing latency, reducing energy consumption or optimizing resource utilization; this is described by the main objective dimension. Furthermore, the application scenario is another relevant dimension: usually orchestration strategies are tailored to specific domains such as smart cities, e-health or industrial automation. Finally, the taxonomy accounts for the functionalities provided by each orchestration solution. These include capabilities such as service monitoring, migration or scaling.

Figure 3 summarizes the taxonomy, illustrating the classification dimensions used in this study. These seven dimensions have been selected to capture where orchestration

FIGURE 3. Proposed Cloud-Edge Orchestration Taxonomy.

is organized, how decisions are made, how resources are adapted at runtime, which goals are optimized, in which domains the solutions are applied, and which operational capabilities they expose. Each of the following subsections will analyze one of these dimensions in detail, discussing how the architectures align with the proposed classification.

B. PAPER COLLECTION METHODOLOGY

The works analyzed in this survey were collected from Scopus and Web of Science by restricting the publication years to 2020-2026 and by using the query reported in Listing 1. The search string is designed to cover the terminology adopted in this research area. For this reason, the setting block does not include only the edge-cloud formulations, but also the continuum terms that are now common in the literature, such as cloud and compute continuum, and continuum computing. The IoT term is not imposed as a mandatory term in the query: many contributions that address IoT services adopt a specific domain (e.g., 5G, vehicular, industrial, or smart-grid) and do not report the term among their title, abstract, or indexed keywords; enforcing it in the query would discard relevant works. Similarly, the operation block is not limited to orchestration and resource allocation, but is extended to placement, offloading, migration, provisioning, and service distribution, so that works addressing equivalent operations under different names are also retrieved. The query is reported as follows:

```
(("edge cloud" OR "edge-cloud" OR "cloud-edge" OR
    "cloud-to-edge" OR "cloud continuum" OR "edge
    cloud continuum" OR "edge-cloud continuum" OR
    "compute continuum" OR "computing continuum"
    OR "continuum computing" OR "multi-tier") AND
    (orchestration OR "resource allocation" OR
    scheduling OR placement OR offloading OR
    migration OR provisioning OR "service
    distribution"))
```

LISTING 1. Search String used in Scopus and Web of Science.

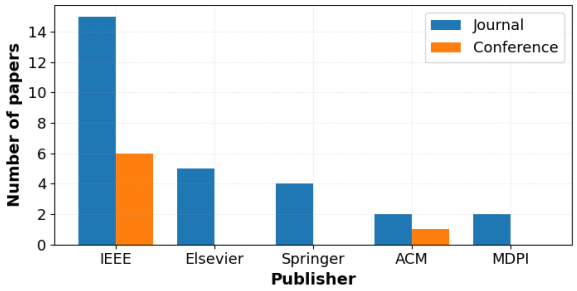

FIGURE 4. Distribution of the analyzed works by publisher and venue type.

The query retrieved 4990 records from Scopus and 3801 from Web of Science. After merging the two sets and removing duplicates, both within each database and across the two, 5587 unique records have been retained. These records have then been screened by title and abstract for relevance to IoT services orchestrated across the edge-cloud continuum, and 902 records have been retained. A work is included when it targets the edge-cloud continuum, describes an orchestration mechanism, and concerns IoT services. A work is excluded when it is conceptual without an orchestration mechanism, when it is confined to a single tier without orchestration across the continuum, or when it is not peer-reviewed. Citation counts, when considered, are used only as a secondary criterion and never as an inclusion threshold, so that recent works are not penalized. These records have then been assessed in full text, prioritizing the approaches described in top journals, resulting in 29 eligible studies.

The analysis is complemented by a search based on citations, in which the reference lists and the citing papers of the already included works have been examined to retrieve further relevant contributions that the keyword query does not capture. This step added 6 works to the 29 obtained from the database search, resulting in the 35 works retained for the comparative analysis.

Figure 4 reports the distribution of the selected papers by publisher and venue type.

C. ARCHITECTURAL STRUCTURE

The architectural structure dimension analyzes how orchestration is organized across the edge-cloud continuum. Based on the recent state-of-art, four main categories can be identified: monolithic, layered, hierarchical, and federated. The placement of the orchestration logic determines where decisions are taken, how the system scales, and how it tolerates the failure of nodes. The four categories are ordered by an increasing distribution of the orchestration logic, ranging from a single decision point to multiple domains, with a trade-off between the simplicity of coordination and the autonomy granted to the lower tiers.

In monolithic architectures, orchestration is handled by a single entity, typically located in the cloud. Edge nodes

act as passive units, with no decisions or autonomy. This approach simplifies orchestration and is often easier to deploy. A clear example of this category is presented in [42], where orchestration is managed by a cloud scheduler that extends the Hadoop YARN architecture to prioritize jobs without involving edge control.

Layered architectures, on the other hand, distribute orchestration responsibilities across different tiers, typically separating coordination in the cloud from execution at the edge. In this model, orchestration decisions may still be centralized, but the infrastructure is explicitly structured into layers. For example, the ECCO system introduced in [43] uses orchestration to define and manage service pipelines, while edge nodes are responsible for executing only specific functions. A similar architecture is proposed in [44], where an edge controller performs workload prediction, while computation is distributed across several edge nodes.

Hierarchical architectures refine the layered architectures by introducing multiple levels of orchestrators with different scopes. These architectures often feature local, regional, and global controllers that collaborate to achieve orchestration. Decisions can be made at lower levels, while upper layers provide guidance and coordination. The architecture described in [45] follows this model, with each application being assigned its own scheduler, operating independently. Another representative example is given in [46], where an orchestration system combines global workload planning with local telemetry, enabling autonomous management of services deployed on edge appliances.

Finally, federated architectures enable multiple orchestration domains, such as different edge or cloud sites. These architectures are relevant in large or community deployments where control must be distributed among independent nodes or domains. In [47], the framework demonstrates an orchestration where nodes negotiate resource usage and scheduling in a peer-to-peer manner. Similarly, in [48], the orchestration is deployed across multiple providers that act independently. Each provider operates its own controller, and collaboration occurs through coupled interactions. A related approach is presented in [49], where a framework coordinates several independent cluster orchestrators, enabling multi-cluster deployment and runtime optimization of distributed applications.

D. RESOURCE CONTROL

The resource control dimension focuses on how decisions regarding the allocation of services are taken; in specific, it characterizes whether orchestration is centralized, distributed, or hybrid. While the architectural structure describes how the orchestration entities are arranged, this dimension characterizes where the decision authority resides. The distinction affects the optimality of the decisions, the communication overhead, and the resilience to the failure of the controller. Centralized control favours a global view, distributed control favours scalability and fault isolation, and hybrid control seeks a balance between the two.

In centralized control, all decisions are taken by a single orchestrator with global visibility. An example is presented in [50], where a centralized agent deployed in the cloud selects both the location and the services that should be used. Edge and end devices execute the decisions but do not contribute to the orchestration. This model simplifies coordination but creates a single point of control and potential attacks. On the other hand, in distributed control, each node makes independent orchestration decisions. For instance, the architecture described in [51] enables vehicles to coordinate task allocation through interactions between them, without relying on a central entity. Such autonomy improves scalability and resilience, though at the cost of more complex coordination. A similar decentralized approach is presented in [52], where each user device runs its deep reinforcement learning agent and decides whether to execute each task locally, at the edge, or in the cloud, without coordinating with the other devices. Finally, hybrid control combines centralized control with localized autonomy; it leverages the benefits of both approaches: the global view of centralized control and the decisions of the decentralized edge controllers. In [53], the orchestration combines a critic in the enterprise cloud, which evaluates the global scheduling state, with distributed actors at the edge sites that optimize and apply the scheduling autonomously according to their local conditions.

E. ORCHESTRATION MECHANISM

This dimension analyzes the mechanism used for the orchestration decisions. Different approaches have been used in the state of the art; among them, the most used rely on mechanisms such as AI models, algorithms based on policies or events, optimization functions, or game-theoretic strategies. These categories differ in how a placement or scheduling decision is reached: a policy-based mechanism applies predefined rules or events; an optimization-based mechanism computes the solution of a formulated objective under constraints; an AI-based mechanism learns the decision from data or past experience; and a game-theoretic mechanism derives it from the interaction among several actors.

Regarding orchestration based on AI, these approaches leverage learning techniques that infer the placement and scheduling decisions from data or past experience to perform in dynamic environments. In [50], a reinforcement learning agent learns the offloading policy and the selection of the deep learning model to be used. Deep reinforcement learning is instead adopted in [53], where an actor-critic algorithm drives the task scheduling of a large industrial IoT. Similarly, in [54], supervised machine learning models (e.g., decision trees and random forests) are used to guide selection and network configuration based on runtime metrics. In the same way, in [55], the incoming tasks are first classified by urgency and computational weight,

and a reinforcement learning algorithm then performs the scheduling and load balancing across the three tiers to reduce the energy consumption of the nodes. Moreover, the mechanisms based on policies rely on predefined rules or events to drive placement decisions, so that the resulting placement follows directly from the declared logic, without optimizing any objective function. For instance, in [56], services are orchestrated using Kubernetes policies such as node affinity and YAML manifests. These approaches are easier to manage but are typically less flexible in non-static environments. On the other hand, orchestration based on optimization functions expresses the placement and the resource management as an objective to be optimized, such as the minimization of latency, energy, or cost, subject to a set of constraints; the decision is then obtained by solving this problem through exact or heuristic methods. In [57], resource allocation is modelled as a satisfaction problem, where a centralized SDN controller uses two optimization functions to satisfy Service Level Agreements (SLAs) while optimizing for energy and bandwidth efficiency. Another optimization-based approach is described in [58], where a fuzzy logic method selects the target layer for offloading and a scheduling algorithm maps the incoming IoT requests onto fog and cloud resources, balancing the load while reducing latency and energy consumption. A further optimization-based approach is presented in [59], where an orchestrator placed on top of the virtual infrastructure managers solves the placement as a virtual network embedding problem through a greedy heuristic, with attention to the scalability of the platform. Similarly, in [25], the orchestration across edges is driven by agreements between the orchestration entities together with the maximization of utility functions that bound the resources each domain can use. Finally, mechanisms based on game theory model the problem as interactions between rational agents aiming to reach a global equilibrium. The system in [60] adopts a community energy trading scheme where users first engage in a Bayesian game for local trading and then use an optimization method to coordinate energy storage and pricing.

F. ALLOCATION METHOD

This dimension refers to how services are allocated. Specifically, three classes have been identified: static, semi-dynamic, and dynamic allocation. Static methods define all decisions a priori, during deployment. Semi-dynamic approaches allow for limited reconfiguration, often based on thresholds or manual interventions. Dynamic methods continuously adapt resource usage based on runtime metrics and feedback. This dimension captures the aspect of orchestration, that is, whether the allocation is decided or revised. The three classes reflect a trade-off between the stability and the low overhead of fixed deployments and the responsiveness of adaptation, which requires continuous monitoring and incurs an additional cost.

In static allocation, service assignment is fixed; an example is presented in [18], where microservices are deployed on edge nodes using predefined configurations, and service migration is triggered manually in case of failure. The orchestration relies on containerized services assigned to specific nodes, and in case of failure, migration is handled manually through redeployment procedures described in their smart city testbed. Moreover, semi-dynamic methods offer some level of reconfiguration, typically based on simple conditions or rules. In [61], an initial static deployment is complemented by a resource allocation method that adjusts, within predefined limits, how communication and computational resources are assigned between cloud and edge nodes according to the monitored load. Finally, dynamic allocation enables adaptation based on several metrics. In [62], resource usage is tuned through autonomic controllers at the edge. These controllers integrate monitoring, analysis, planning, and execution loops that adjust resource usage according to latency and energy metrics, with runtime relocation of services when thresholds are exceeded. Dynamic allocation is also adopted in [63], where a deep reinforcement learning scheduler decides at runtime whether each task is executed at the edge, in the cloud, or through a combination, adapting the decisions to the observed waiting time and quality-of-service requirements. A solution supporting both static and dynamic deployment is instead presented in [64], where applications are deployed across the continuum, and a runtime mechanism enables the migration of a microservice between the fog and the cloud while preserving data continuity.

G. OBJECTIVE

This dimension analyzes the main objectives pursued by the orchestration proposal. The most common goals include reducing latency, optimizing energy consumption, increasing security, and improving resource efficiency. The targeted objective drives the design of the mechanism and the selection of the metrics used to evaluate it. In most proposals, the objectives are multiple and partially conflicting, so that they are pursued jointly, either by combining them into a single goal or by optimizing one of them while treating the others as constraints.

Some approaches focus primarily on reducing latency to enable real-time processing. In [65], the system orchestrates service function chains to minimize service delay. By employing a quantum machine learning model, the orchestrator ensures that processing occurs close to data sources. The objective is explicitly to minimize service delay, ensuring that data are processed close to their sources. A similar emphasis on latency is found in [66], where a traffic scheduling mechanism processes critical IoT data streams, improving latency and throughput while reducing energy consumption. Other works emphasize reliability as the most important goal. In [67], the proposed system orchestrates isolated network slices. The main objective is to ensure secure services even under adversarial conditions, such as

attacks, while maintaining compliance with SLA constraints. A further popular objective is to improve utilization and cost efficiency. In [68], the focus is on vision applications. The orchestrator partitions tasks across devices to minimize latency and execution costs while satisfying service constraints. The system evaluates different scheduling strategies to determine the most efficient use of available computing resources. Energy efficiency is instead the primary objective in [69], where an orchestrator distributes the computation across the IoT, edge, fog, and cloud layers to reduce energy consumption while meeting the latency requirements of IoT systems.

H. APPLICATION SCENARIO

This dimension analyzes the application domains targeted by orchestration solutions. Typical scenarios include industrial automation, telecommunication infrastructures, mobility, smart cities, healthcare, and energy systems. The domain determines the requirements that the orchestration must satisfy, such as the latency, the criticality of the services, and the mobility of the nodes. Some proposals are instead designed independently of a specific domain and are therefore reported as generic.

Among the analyzed works, many focus on industrial automation. An example is provided by [23], in which the proposed architecture enables the deployment of digital twins for manufacturing processes. By combining edge resources with cloud coordination, the system supports real-time control close to the shop floor, while maintaining synchronization with global management functions. Another relevant domain is telecommunication infrastructure. In [33], the orchestration framework addresses the deployment of 5G core functions across distributed clusters. The system adopts Kubernetes orchestration and evaluates multi cluster deployments, with edge sites managing local control while the cloud provides scalable processing capabilities. Furthermore, mobility and vehicular applications represent another important scenario. In [70], a hybrid framework orchestrates predictive analytics for smart mobility services. Forecasting tasks are allocated at the edge for short-term responsiveness and in the cloud for more accurate long-term predictions. Smart cities also represent a targeted scenario. In [71], an orchestration platform extends Kubernetes with a custom scheduler and continuous monitoring to manage both critical and non-critical services, ensuring that the timing constraints of the critical ones are respected. A multi-domain deployment is addressed in [72], where a framework based on Kubernetes integrates continuous integration and deployment practices to operate IoT services, validated in three domains, namely smart cities, e-health, and smart manufacturing. Other orchestration solutions extend to healthcare and energy systems, where orchestration enables scenarios such as medical data analytics and community energy trading.

I. OTHER FUNCTIONALITIES

In addition to service placement and management, orchestration architectures often provide additional functionalities. These features play an important role in maintaining service continuity. These distinguish a complete orchestration solution from a simple placement engine. They are often interdependent, since monitoring typically provides the information on which scaling, migration, and quality-of-service optimization rely.

Indeed, one of the most common functionalities is *monitoring*, which enables the collection of metrics related to status, resource usage and network performance. Monitoring is often used to inform decisions related to scaling, migration, or SLA compliance. Moreover, *migration* and *rescheduling* allow services to be moved or reassigned at runtime in response to load changes or failures. These mechanisms contribute to reliability and improve resource utilization. Furthermore, *scaling* refers to the capability of the system to adjust the allocated resources. This includes both horizontal scaling, by increasing or reducing the number of services, and vertical scaling, by adjusting the resources assigned to a single service. Several architectures also include mechanisms for *Quality-of-Service* or *Quality-of-Experience optimization*. These may involve control loops to adjust the system behaviour to meet user expectations. Finally, other important functionalities include support for *fault tolerance*, such as service replication or failure detection, and *security management*, which involves authentication and access control.

In the next section, a summary of the analyzed papers will be provided, showing which of these functionalities are implemented in each solution.

VI. DISCUSSION AND OPEN CHALLENGES

The analysis presented in Table 3 highlights the variety of approaches adopted in recent years for orchestrating services across edge cloud infrastructures. Although these solutions share a common goal, i.e. management of resources to support distributed services, they differ in how they address orchestration aspects. From an architectural perspective, monolithic solutions appear as isolated cases, while most works adopt layered or hierarchical structures; on the other hand, federated setups are used when coordination must span independent providers. Layered designs are typical of platforms where cloud coordination and edge execution are explicitly separated, whereas hierarchical proposals introduce local, regional, and global orchestrators with distinct scopes (representative examples include mobility and automation). Federated architectures appear when domains retain their own controllers and collaborate through different peers, which is the case in several telco or cross-provider frameworks [33], [48].

Moreover, the resource control approach often reflects this architectural structure. Centralized control is still common in

TABLE 3. Comparison of the analyzed orchestration architectures.

Ref.	Architectural Structure	Resource Control	Orchestration Mechanism	Allocation Method	Main Objectives	Application Scenario	Other Functionalities
[23]	Layered	Hybrid	AI-based (Deep learning prediction)	Dynamic	Latency, Resource	Industrial Automation	MO, SC
[42]	Monolithic	Centralized	Policy-based (Job prioritization)	Dynamic	Resource Efficiency, QoS Optimization	Generic	MO, QSE
[59]	Hierarchical	Hybrid	Optimization-based (Greedy heuristic)	Dynamic	Latency, Resources, Scalability	Generic	MO, MI, SC, QSE
[18]	Layered	Centralized	Policy-based (Predefined configuration)	Static	Latency, Resources	Smart Cities	MO, MI, SC, QSE
[33]	Federated	Hybrid	Policy-based (Declarative Kubernetes)	Static	Latency, Resources	Telco	MO, SC, QSE
[50]	Layered	Centralized	AI-based (Reinforcement learning)	Dynamic	Latency, Resources, Energy	Smart Cities, Healthcare	MO, SC, QSE
[61]	Layered	Hybrid	Optimization-based (Resource allocation)	Semi-dynamic	Latency, Resources	Energy Systems	MO, QSE
[65]	Layered	Centralized	AI-based (Quantum machine learning)	Dynamic	Latency, Resources	Generic	MO, QSE
[48]	Federated	Hybrid	Optimization-based (Online optimization)	Dynamic	Resources, Cost	Generic	MO, SC
[25]	Hierarchical	Hybrid	Optimization-based (Utility maximization)	Dynamic	Latency, Resources	Mobility	MO, SC, QSE
[46]	Hierarchical	Hybrid	Optimization-based (Workload planning)	Dynamic	Resources, Energy, Latency	Mobility	MO, MI, SC, QSE, FT
[43]	Layered	Hybrid	Policy-based (Service chaining)	Semi-dynamic	Latency, Resources, Safety	Mobility	MO, MI, QSE, FT
[62]	Hierarchical	Hybrid	Optimization-based (Autonomic control loops)	Dynamic	Energy, Resources, Latency	Industrial Automation	MO, MI, SC, QSE, FT
[54]	Hierarchical	Hybrid	AI-based (Supervised learning)	Dynamic	Latency, Resources, Reliability	Telco	MO, SC, QSE, FT
[56]	Layered	Centralized	Policy-based (Kubernetes policies)	Semi-dynamic	Latency, Resources, Scalability	Generic	MO, SC, FT
[44]	Layered	Centralized	Optimization-based (Admission control)	Dynamic	Latency, Resources	Smart Cities	MO, QSE
[70]	Layered	Hybrid	AI-based (Forecasting models)	Dynamic	Latency, Resources, QoS/QoE	Mobility	MO, QSE
[51]	Federated	Distributed	Policy-based (Vehicle coordination)	Dynamic	Latency, Resources	Mobility	MO, MI, SC
[47]	Federated	Distributed	Policy-based (Peer-to-peer negotiation)	Dynamic	Latency, Resources, Scalability	Generic	MO, MI, SC
[57]	Layered	Centralized	Optimization-based (Constraint satisfaction)	Dynamic	Resources, Energy, SLA Compliance	Industrial Automation	MO, SC, QSE
[45]	Hierarchical	Distributed	Policy-based (Per-application rules)	Semi-dynamic	Latency, QoS	Generic	MO, MI, SC, QSE
[53]	Federated	Hybrid	AI-based (Actor–Critic)	Dynamic	Latency, Energy, Resources	Industrial Automation	MO, QSE
[67]	Hierarchical	Centralized	Optimization-based (Resource optimization)	Dynamic	Latency, Resources, Security, Energy	Generic	MO, SC, QSE, SM
[60]	Hierarchical	Hybrid	Game-theoretic (Bayesian game)	Dynamic	Latency, Resources, Efficiency, Cost	Energy Systems	MO, QSE
[68]	Layered	Centralized	Optimization-based (Task scheduling optimization)	Semi-dynamic	Latency, Resources, Cost	Computer Vision	MO, SC, QSE
[71]	Layered	Centralized	Policy-based (Kubernetes scheduler)	Dynamic	Latency, Security, QoS	Smart Cities	MO, SC, QSE, FT
[55]	Layered	Centralized	AI-based (Deep reinforcement learning)	Dynamic	Latency, Energy, Resources	Generic	MO, SC, QSE
[69]	Layered	Hybrid	AI-based (Federated reinforcement learning)	Dynamic	Energy, Latency, Scalability	Generic	MO, SC, QSE
[63]	Layered	Centralized	AI-based (Deep reinforcement learning)	Dynamic	Latency, QoS/QoE, Resources	Generic	MO, MI, SC, QSE
[58]	Layered	Centralized	Optimization-based (Fuzzy logic and heuristic)	Dynamic	Latency, Energy, Resources	Generic	MO, SC
[49]	Federated	Hybrid	Optimization-based (Multi-level optimization)	Dynamic	Resources, Cost, Latency	Generic	MO, MI, SC, QSE
[64]	Layered	Centralized	Policy-based (Declarative deployment)	Dynamic	QoS/QoE, Resources	Smart Agriculture	MO, MI
[66]	Layered	Hybrid	Policy-based (Priority scheduling)	Dynamic	Latency, Resources, Energy	Industrial Automation	MO, SC
[52]	Layered	Distributed	AI-based (Actor–Critic)	Dynamic	QoS/QoE, Energy, Latency	Generic	MO, QSE
[72]	Layered	Hybrid	Policy-based (Kubernetes deployment)	Dynamic	Resources, QoS, Scalability	Smart Cities, Healthcare and Industry	MO, SC, FT

Legend: MO – Monitoring; MI – Migration; SC – Scaling; QSE – QoS/QoE Optimization; FT – Fault Tolerance; SM – Security Management.

cloud-centric designs where a single scheduler with global visibility can manage the system; the table reports this in cloud-based approaches that select services while edge nodes simply execute. Many works, however, opt for hybrid

control to combine a global view with localized autonomy for adaptation at the edge. Distributed control is selected when nodes must coordinate directly, e.g. vehicles negotiating task assignment or peer nodes brokering resources, without a single controller. Notably, the federated approaches never rely on a purely centralized controller, while layered architectures are paired either with centralized or hybrid control depending on how much decision power is retained at the edge [51], [53].

The table also indicates how the decision logic is heterogeneous and how different orchestration mechanisms exist. Policy-based mechanisms rely on rules, constraints, or events (e.g., placement policies and affinities with Kubernetes) and are common when the orchestration is declarative. AI-based mechanisms are adopted when the platform must react to changing conditions; the approaches include Reinforcement Learning (RL), deep RL, and supervised models. Optimization-based proposals encode placement and scheduling as explicit problems with objectives and constraints, used both at the network or controller level and within specific applications; a game-theoretic scheme appears where interactions among participants are modeled directly. These choices are operated especially in dynamic mode, considering the allocation method, where placement and resource usage are adjusted at runtime; in the table, dynamic allocation typically occurs with hybrid or distributed control and with functionalities such as QoS/QoE optimization [70], reflecting the need for continuous feedback and adaptation. On the other hand, semi-dynamic approaches are used when bounded reconfiguration is sufficient, often triggered by simple conditions such as threshold or fault detection, and are used to restore service without redesigning the deployment [43]; in these cases, the controller maintains a global mechanism but permits adjustments at the edge. Static configurations remain only where deployments follow predefined plans and recovery is manual with fixed templates; here, orchestration focuses on enforcing the declared layout rather than following control loops. Specifically, the allocation mode in the table aligns with the resource control: dynamic is preferred when the platform exposes runtime signals; semi-dynamic appears when stability and simplicity are prioritized; static persists in a few scoped or specific settings [18].

Furthermore, regarding the main objectives, latency reduction and efficient use of resources are the primary aims, which explains the prevalence of runtime adaptation. Energy becomes explicit when it is intrinsic to the problem (e.g., edge controllers or power scenarios), and it appears with growing frequency among the most recent works, anticipating the sustainability concern discussed among the open challenges; security, instead, emerges only where resilience is central. Cost is explicitly modeled as an objective when policies trade off execution time against monetary expense. These objective profiles align with the application scenarios: mobility and vehicular proposals prioritize latency and service continuity, hence they adopt hierarchical or hybrid control with dynamic allocation to control variation [70]; telco works usually stress reliability, motivating federated approaches and coordination between cloud/edge orchestration and network control [33], [54]; industrial automation targets deterministic response near the real world, reflecting the latency and efficiency needs that drives placement; on the other hand, energy approaches bring energy and cost to the foreground and manage orchestration with market coordination [60].

Finally, the other functionalities column clarifies the external features that are implemented. Monitoring is common and provides the feedback needed for scaling, migration, and SLA checks. QoS/QoE optimization and scaling are widely reported and match the dynamic allocation choices. Migration and rescheduling appear when runtime relocation is part of the design rather than an exceptional path, while explicit fault-tolerance and security-management features are highlighted in a smaller set of approaches, typically where access control is handled within the orchestration layer itself ([43], [46]).

Overall, the table indicates a preference for multi-tier designs, i.e. layered or hierarchical, and, when coordination spans different providers, federated solutions. Control is typically hybrid, combining a global view with local autonomy; fully distributed control appears when peers must negotiate directly, while centralized control is still frequent in cloud-centric designs where a single orchestrator is sufficient. Orchestration mechanisms vary but are generally run with dynamic allocation supported by monitoring, scaling, and QoS/QoE optimization. Semi-dynamic approaches cover reconfiguration on thresholds or faults, and static modes persist only in specific deployments. On objectives, latency and efficient resource use dominate; energy becomes explicit when intrinsic to the setting, security emerges only a few times, where isolation is required, and cost is modeled when policies trade time against expense. These objective profiles align with the scenarios reported: mobility favors hierarchical/hybrid approaches with runtime reconfiguration, telco emphasizes federation and cluster coordination, industrial automation keeps decisions near the devices, and energy systems pair orchestration with grid coordination. Functionally, monitoring is ubiquitous, QoS/QoE and scaling are widely implemented, while migration, fault tolerance, and security management appear when explicitly required by the use case.

However, although recent architectures demonstrate progress in the research, several research directions remain open:

- *Interoperability across architectures and multiple domains*: Despite recent advances, orchestration remains fragmented along vertical constraints: service, policies, and interfaces are often specific to platforms. In this context, Digital Twins (DTs) are emerging as a unifying abstraction to represent real-world entities and processes under a single software identity that encapsulates state and control [73]. If treated as the main entity, a DT can serve as the unit of orchestration: a DT

can be placed, migrated, replicated, paused or resumed, and retired under explicit rules, while preserving SLA constraints. To make this effective in practice, research should define DT orchestration models that specify state schema, update cadence, interfaces for sensing and actuation, data and privacy requirements, independently of the runtime. Equally important is a control plane that maintains namespace, discovery, identity, and authorization, together with operations and semantics for instances. Moreover, portability is essential, since intents should have the same meaning across different orchestrators and be translated by each platform. This, in turn, requires metrics with shared semantics, so that monitoring remains comparable. In practice, the community needs conformance tests and reference adapters for the main stacks to ensure that a given DT specification behaves the same everywhere.

- *Topology network*: Although many orchestration proposals target layered or hierarchical edge-cloud systems, the underlying network structure is often treated only implicitly. In practice, these infrastructures frequently follow a hierarchical organization, where edge nodes are connected through intermediate aggregation points and upper layers toward the cloud. In such settings, congestion is not uniform across the infrastructure, and the most critical bottlenecks may emerge on shared links. However, current orchestration approaches still tend to base placement and migration decisions mainly on computational resource availability or on the distance to the requester, while giving limited attention to the structural properties of the network. This could produce several inefficiencies: heuristics that select the node closest to the requester tend to concentrate the instances at the leaf level and to saturate its capacity, while the intermediate aggregation nodes, although reachable from several leaves, are left unused. At the same time, the distribution of the instances across the branches is rarely aligned with the distribution of the demand, so that low-demand branches are over-provisioned while high-demand ones remain under-served, and decisions driven by the distance to a fixed reference node, such as the cloud, bias the placement toward that node regardless of where the requests originate. A relevant research direction is therefore the design of orchestration mechanisms that explicitly incorporate the topology of the infrastructure into the decision process. This includes taking into account the position of nodes, the communication distance, the load on the shared links, and the effect of placements across different branches of the hierarchy, even to take advantage of the network to serve more users simultaneously. A more explicit treatment of these aspects may improve service placement.
- *Deployment realism*: Most evaluations still assume one central cloud and a set of identical edge nodes connected by stable networks. Real systems instead span multiple sites and providers, with real links and heterogeneous hardware. Research should move to testbeds that mirror this reality: multiple clusters with their own controllers, links with variable latency, loss, and jitter, disconnections and later reconnections, and a mix of nodes running different runtimes. A further aspect of realism concerns the cost of the orchestration operations themselves, which are frequently modeled as if they were instantaneous. A placement requires pulling the image, initialising the runtime, and loading the state, while a migration additionally requires checkpointing the state, transferring it, and restoring the execution at the destination, with delays in the order of hundreds of milliseconds and of one second or more, respectively. When this reconfiguration overhead is neglected, its effect is hidden in the average behaviour and emerges only at runtime, in particular at start-up, when instances initialised in the cloud are relocated toward the edge, and whenever new instances are deployed on demand, since the requests served during the reconfiguration experience an additional delay that can cause requirement violations. Moreover, placement and migration should account for network capacity and failures, service discovery should use consistent naming, and observability should expose a common set of metrics and traces so monitoring has the same meaning across providers. To make results comparable, testbeds should be reproducible, include traffic generators, and provide fault injection. At the same time, many simulators are available but real deployments remain scarce; simulation results should be validated on pilots. The community would benefit from open deployments with setup scripts, so behavior can be reproduced and controllers compared under the same network constraints.
- *Transparency of AI orchestration*: Many proposals delegate placement to learning orchestrators, but the decision process is often vague and insufficiently evaluated. Research should make decisions inspectable and reproducible: record the rationale for each action. Ablation studies are needed to isolate the contribution of each component, together with stress tests; results should report criteria and constraints. Evaluation must include non-learning baselines and quantify convergence time and efficiency, not just average latency improvements. Authors should disclose preprocessing and decision to enable exact replays, and the conditions under which the orchestrator degrades, if any exists.
- *Scalability and computational complexity*: Placement and scheduling across cloud–edge are complex problems; as the number of services, sites, links, and constraints grows, the decision space explodes and orchestration latency becomes critical. Many works rely on iterative solvers, but few report complexity, cost, or memory usage. Research should make the problem model explicit and report cost or bounds, measure time and quantify overhead. A complementary and often

overlooked aspect is the cost that the orchestration logic itself imposes on the nodes responsible for the decisions, since iterative solvers and learning agents consume computation and memory that, when the orchestrator runs on constrained edge nodes, compete with the very services it is meant to support; this footprint is rarely characterised and the orchestration cost is seldom reported as a metric. Moreover, experiments should include growing loads, multiple scenarios, and multiple constraints and report comparable KPIs, so that scalability limits are explicit.

- *Benchmarking and datasets*: Comparative evaluation is weak because public setups are rare. Research should define shared scenarios with inputs and outputs: topologies, link profiles and event schedules that include failures. Datasets should cover both IoT pipelines and provide targets (SLAs, budgets) together with resource limits. To make experiments reproducible, datasets should ship traffic and configuration bundles (topology, limits). Moreover, datasets should be released in raw form, with documented schemas, and the same traces should be replayable in both simulators and real testbeds, so that proposals can validate results first in simulation and then on pilots under the same conditions.
- *Trust models*: Security is occasionally reported (identity, access, encryption), but trust is almost never modeled or evaluated, even though orchestration decisions depend on how much a site, device, or data source can be relied upon over time [74]. The orchestrators should keep trust views derived from observed outcomes (SLA, integrity of produced data, continuity of operation) and use them to place and migrate instances. When an instance moves across sites or providers, the receiving domain should obtain the current trust state and supporting evidence so that eligibility can be checked before activation. Because trust evolves, policies need clear update rules that prevent trust degradation. Policies should also define how trust is computed and used. In practice, this could be computed by keeping separate trust views for data sources, deriving them from observed outcomes, and weighing negative events. Trust should also be maintained for hosts and services, not only for data sources, and needs decay rules. In general, trust should be tied to stable identities and evaluated for each entity type. For data sources, this means tracking integrity checks and incident history; for services, tracking failure rates and reliability of services. When services move, the trust state should travel with them, since migrations need to include the current trust value and the state of entities that have managed them.
- *Sustainability*: Energy is often treated as a secondary metric, while orchestration should make it an important objective. Decisions on placement need to account for device power and the carbon intensity of the supplying grid; this requires telemetry and models that include both compute costs, since moving data can consume more energy than processing it locally. Policies should expose trade-offs with latency and cost: locate stages that exchange large volumes, avoid migrations, and schedule work when power is cheaper. Evaluation must report how energy and carbon are measured and aggregated, and present comparable KPIs such as energy per request and total consumption. Finally, sustainability should be controllable at runtime, so that results are reproducible and the impact of green decisions on performance is explicit.

VII. CONCLUSION

Edge–cloud computing has expanded the deployment of IoT services beyond centralized data centers, but it also introduces variability that traditional cloud practices do not address. This review set out to clarify how orchestration sustains IoT services in practice, structuring the analysis through seven dimensions and applying it to recent architectures. The following conclusions outline the research directions that, in our view, are most urgent for the next generation of edge–cloud orchestrators.

The analysis shows a clear preference for multi-tier designs and, when coordination spans independent providers, federated solutions. Control is typically hybrid, combining a global view with localized autonomy, while centralized control remains frequent when a single scheduler suffices and distributed control is reserved for peer coordination. Decision mechanisms are heterogeneous, yet most systems operate under dynamic allocation supported by monitoring, scaling, and QoS/QoE optimization functions. Latency reduction and efficient resource use dominate the objective set, energy becomes explicit in power scenarios, while security surfaces only where isolation is central and cost is modeled when time–expense trade-offs are essential. Functionally, monitoring is ubiquitous, QoS/QoE optimization and scaling are widespread, while migration, fault tolerance, and security management appear when explicitly required by the use case.

From these findings, several directions emerge for the next generation of orchestrators: (i) interoperable models and descriptors, especially oriented to DT, that carry placement and migration across different platforms and domains; (ii) testbeds and pilots that reflect real deployments; (iii) transparent evaluation of learning-based controllers, with ablation and stress tests; (iv) explicit treatment of scalability and orchestration cost; (v) trust as a constraint with measurable impact; and (vi) sustainability as an equally important objective.

REFERENCES

[1] Q. He, J. Lin, H. Fang, X. Wang, M. Huang, X. Yi, and K. Yu, "Integrating IoT and 6G: Applications of edge intelligence, challenges, and future directions," *IEEE Trans. Services Comput.*, vol. 18, no. 4, pp. 2471–2488, Jul. 2025.

[2] M. Banafaa, I. Shayea, J. Din, M. Hadri Azmi, A. Alashbi, Y. Ibrahim Daradkeh, and A. Alhammadi, "6G mobile communication technology: Requirements, targets, applications, challenges, advantages, and opportunities," *Alexandria Eng. J.*, vol. 64, pp. 245–274, Feb. 2023.

[3] C. Marche, V. Popescu, L. Atzori, and M. Nitti, "Context-aware itinerary planning in smart tourism: IoT-enabled design and tourist profiling," *IEEE Internet Things J.*, vol. 13, no. 15, pp. 34146–34157, Aug. 2026.

[4] E. E. Uslu, B. Ozturk, B. Gorkemli, M. Soyturk, S. B. Cihan, A. R. Gunduzhev, and O. Dagdeviren, "Next-generation management and orchestration for 6G: Emerging trends, architectures, and challenges," *IEEE Open J. Commun. Soc.*, vol. 7, pp. 2222–2269, 2026.

[5] R. Swetha, J. Thriveni, and K. R. Venugopal, "Resource utilization-based container orchestration: Closing the gap for enhanced cloud application performance," *Social Netw. Comput. Sci.*, vol. 6, no. 3, p. 191, Feb. 2025.

[6] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[7] B. Varghese, N. Wang, J. Li, and D. S. Nikolopoulos, "Edge-as-a-service: Towards distributed cloud architectures," in *Parallel Computing is Everywhere*. Amsterdam, The Netherlands: IOS Press, 2018, pp. 784–793.

[8] Y. Wu, "Cloud-edge orchestration for the Internet of Things: Architecture and AI-powered data processing," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12792–12805, Aug. 2021.

[9] H. F. Shahid, B. Akdemir, J. Islam, I. Ahmad, I. Ahmad, and E. Harjula, "IoT service orchestration in edge–cloud continuum with 6G: A review," *IEEE Internet Things J.*, vol. 13, no. 10, pp. 20339–20358, May 2026.

[10] J. Pournazari, A. Ullah, A. Al-Dubai, and X. Liu, "Computation offloading in the edge-to-cloud compute continuum: A survey of federated architectural solutions," *Cluster Comput.*, vol. 28, no. 13, p. 839, Nov. 2025.

[11] L. Belcastro, F. Marozzo, A. Orsino, D. Talia, and P. Trunfio, "Navigating the edge-cloud continuum: A state-of-practice survey," *IEEE Access*, vol. 14, pp. 40622–40647, 2026.

[12] V. Ziegler, H. Viswanathan, H. Flinck, M. Hoffmann, V. Räisänen, and K. Hätönen, "6G architecture to connect the worlds," *IEEE Access*, vol. 8, pp. 173508–173520, 2020.

[13] A. Dogra, R. K. Jha, and S. Jain, "A survey on beyond 5G network with the advent of 6G: Architecture and emerging technologies," *IEEE Access*, vol. 9, pp. 67512–67547, 2021.

[14] L. L. Piras, C. Marche, S. Quattropani, G. Sciddurlo, and M. Nitti, "Leveraging social IoT to improve orchestration of digital twin networks," in *Proc. IEEE 11th World Forum Internet Things (WF-IoT)*, Dec. 2025, pp. 1–6.

[15] S. Ranjbaran, M. Amadeo, C. Marche, G. Ruggeri, A. Sinha, and M. Nitti, "Cloud-edge resource management and migration: Leveraging online learning for digital twin re-placement," in *Proc. IEEE 10th World Forum Internet Things (WF-IoT)*, Nov. 2024, pp. 1–6.

[16] S. Painuly, S. Sharma, and P. Matta, "Future trends and challenges in next generation smart application of 5G-IoT," in *Proc. 5th Int. Conf. Comput. Methodologies Commun. (ICCMC)*, Apr. 2021, pp. 354–357.

[17] S. Batewela, P. Ranaweera, M. Liyanage, E. Zeydan, and M. Ylianttila, "Addressing security orchestration challenges in next-generation networks: A comprehensive overview," *IEEE Open J. Comput. Soc.*, vol. 6, pp. 669–687, 2025.

[18] L. Roda-Sanchez, C. Garrido-Hidalgo, F. Royo, J. L. Maté-Gómez, T. Olivares, and A. Fernández-Caballero, "Cloud–edge microservices architecture and service orchestration: An integral solution for a real-world deployment experience," *Internet Things*, vol. 22, Jul. 2023, Art. no. 100777.

[19] Y. Chiang, Y. Zhang, H. Luo, T.-Y. Chen, G.-H. Chen, H.-T. Chen, Y.-J. Wang, H.-Y. Wei, and C.-T. Chou, "Management and orchestration of edge computing for IoT: A comprehensive survey," *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14307–14331, Aug. 2023.

[20] D. Hästbacka, J. Halme, L. Barna, H. Hoikka, H. Pettinen, M. Larrañaga, M. Björkbom, H. Mesiä, A. Jaatinen, and M. Elo, "Dynamic edge and cloud service integration for industrial IoT and production monitoring applications of industrial cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 498–508, Jan. 2022.

[21] K. Kaur, F. Guillemin, and F. Sailhan, "Container placement and migration strategies for cloud, fog, and edge data centers: A survey," *Int. J. Netw. Manage.*, vol. 32, no. 6, p. 2212, Nov. 2022.

[22] S. Böhm and G. Wirtz, "Cloud-edge orchestration for smart cities: A review of kubernetes-based orchestration architectures," *EAI Endorsed Trans. Smart Cities*, vol. 6, no. 18, p. e2, May 2022.

[23] Q. Nie, D. Tang, C. Liu, L. Wang, and J. Song, "A multi-agent and cloud-edge orchestration framework of digital twin for distributed production control," *Robot. Computer-Integrated Manuf.*, vol. 82, Aug. 2023, Art. no. 102543.

[24] B. Costa, J. Bachiega, L. R. de Carvalho, and A. P. F. Araujo, "Orchestration in fog computing: A comprehensive survey," *ACM Comput. Surveys*, vol. 55, no. 2, pp. 1–34, Feb. 2023.

[25] N. Slamnik-Kriještorac, G. M. Yilma, M. Liebsch, F. Z. Yousaf, and J. M. Marquez-Barja, "Collaborative orchestration of multi-domain edges from a connected, cooperative and automated mobility (CCAM) perspective," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2001–2020, Apr. 2023.

[26] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research," *Internet Things*, vol. 12, p. 100273, Dec. 2020.

[27] M. Gaglianese, J. Soldani, S. Forti, and A. Brogi, "Green orchestration of cloud-edge applications: State of the art and open challenges," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Sep. 2023, pp. 250–261.

[28] A. Ullah, T. Kiss, J. Kovács, F. Tusa, J. Deslauriers, H. Dagdeviren, R. Arjun, and H. Hamzeh, "Orchestration in the cloud-to-things compute continuum: Taxonomy, survey and future directions," *J. Cloud Comput.*, vol. 12, no. 1, pp. 1–29, Sep. 2023.

[29] S. Hamdan, M. Ayyash, and S. Almajali, "Edge-computing architectures for Internet of Things applications: A survey," *Sensors*, vol. 20, no. 22, p. 6441, Nov. 2020.

[30] R. Vaño, I. Lacalle, P. Sowiński, R. S-Julián, and C. E. Palau, "Cloud-native workload orchestration at the edge: A deployment review and future directions," *Sensors*, vol. 23, no. 4, p. 2215, Feb. 2023.

[31] P. Gkonis, A. Giannopoulos, P. Trakadas, X. Masip-Bruin, and F. D'Andria, "A survey on IoT-edge-cloud continuum systems: Status, challenges, use cases, and open issues," *Future Internet*, vol. 15, no. 12, p. 383, Nov. 2023.

[32] S. Böhm and G. Wirtz, "Towards orchestration of cloud-edge architectures with kubernetes," in *International Summit Smart City 360°*. Springer, 2021, pp. 207–230.

[33] O.-M. Ungureanu, C. Vladeanu, and R. Kooij, "Collaborative cloud–edge: A declarative API orchestration model for the NextGen 5G core," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Aug. 2021, pp. 124–133.

[34] D. Yakubov and D. Hästbacka, "Comparative analysis of lightweight Kubernetes distributions for edge computing: Security, resilience and maintainability," in *Proc. Eur. Conf. Service-Oriented Cloud Comput.*, 2025, pp. 96–104.

[35] A. Alhindi, K. Djemame, and F. B. Heravan, "On the power consumption of serverless functions: An evaluation of OpenFaaS," in *Proc. IEEE/ACM 15th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2022, pp. 366–371.

[36] X. Zhang, "A fine-grained task scheduling mechanism for digital economy services based on intelligent edge and cloud computing," *J. Cloud Comput.*, vol. 12, no. 1, p. 30, Mar. 2023.

[37] N. Kaviani, D. Kalinin, and M. Maximilien, "Towards serverless as commodity: A case of knative," in *Proc. 5th Int. Workshop Serverless Comput.*, Dec. 2019, pp. 13–18, doi: 10.1145/3366623.3368135.

[38] R. Moreno-Vozmediano, R. S. Montero, E. Huedo, and I. M. Llorente, "Intelligent resource orchestration for 5G edge infrastructures," *Future Internet*, vol. 16, no. 3, p. 103, Mar. 2024.

[39] M. Abuibaid, A. H. Ghorab, A. Seguin-Mcpeake, O. Yuen, T. Yungblut, and M. St-Hilaire, "Edge workloads monitoring and failover: A StarlingX-based testbed implementation and measurement study," *IEEE Access*, vol. 10, pp. 97101–97116, 2022.

[40] M.-I. Csoma, B. Koné, R. Botez, I.-A. Ivanciu, A. Kora, and V. Dobrota, "Management and orchestration for network function virtualization: An open source MANO approach," in *Proc. 19th RoEduNet Conf., Netw. Educ. Res. (RoEduNet)*, Dec. 2020, pp. 1–6.

[41] Y. Teng, J. Cui, and W. Jiang, "Research on application of edge computing in real-time environmental monitoring system," *J. Physics: Conf. Ser.*, vol. 2010, no. 1, Sep. 2021, Art. no. 012157.

[42] T. Yeh and S. Yu, "Realizing dynamic resource orchestration on cloud systems in the cloud-to-edge continuum," *J. Parallel Distrib. Comput.*, vol. 160, pp. 100–109, Feb. 2022.

[43] V. Cozzolino, J. Ott, A. Y. Ding, and R. Mortier, "ECCO: Edge-cloud chaining and orchestration framework for road context assessment," in *Proc. IEEE/ACM 5th Int. Conf. Internet-Things Design Implement. (IoTDI)*, Apr. 2020, pp. 223–230.

[44] G. Papathanail, I. Fotoglou, C. Demertzis, A. Pentelas, K. Sgouromitis, P. Papadimitriou, D. Spatharakis, I. Dimolitsas, D. Dechouniotis, and S. Papavassiliou, "COSMOS: An orchestration framework for smart computation offloading in edge clouds," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2020, pp. 1–6.

[45] A. Orive, A. Agirre, H.-L. Truong, I. Sarachaga, and M. Marcos, "Quality of service aware orchestration for cloud–edge continuum applications," *Sensors*, vol. 22, no. 5, p. 1755, Feb. 2022.

[46] F. Guim, T. Metsch, H. Moustafa, T. Verrall, D. Carrera, N. Cadenelli, J. Chen, D. Doria, C. Ghadie, and R. G. Prats, "Autonomous lifecycle management for resource-efficient workload orchestration for green edge computing," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 571–582, Mar. 2022.

[47] A. Pires, J. Simão, and L. Veiga, "Distributed and decentralized orchestration of containers on edge clouds," *J. Grid Comput.*, vol. 19, no. 3, p. 36, Sep. 2021.

[48] X. Shao, G. Hasegawa, M. Dong, Z. Liu, H. Masui, and Y. Ji, "An online orchestration mechanism for general-purpose edge computing," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 927–940, Mar. 2023.

[49] N. Filinis, I. Dimolitsas, D. Spatharakis, E. Fotopoulou, I. Tzanettis, C. Vassilakis, A. Zafeiropoulos, and S. Papavassiliou, "A synergetic meta-orchestration framework for distributed application deployments in the computing continuum," in *Proc. 1st Int. Workshop MetaOS Cloud-Edge-IoT Continuum*, Apr. 2024, pp. 21–27.

[50] S. Shahhosseini, D. Seo, A. Kanduri, T. Hu, S.-S. Lim, B. Donyanavard, A. M. Rahmani, and N. Dutt, "Online learning for orchestration of inference in multi-user end-edge-cloud networks," *ACM Trans. Embedded Comput. Syst.*, vol. 21, no. 6, pp. 1–25, Nov. 2022.

[51] S. Mittal, R. K. Dudeja, R. S. Bali, and G. S. Aujla, "A distributed task orchestration scheme in collaborative vehicular cloud edge networks," *Computing*, vol. 106, no. 4, pp. 1151–1175, Apr. 2024.

[52] G. Nieto, I. de la Iglesia, U. Lopez-Novoa, and C. Perfecto, "Deep reinforcement learning techniques for dynamic task offloading in the 5G edge-cloud continuum," *J. Cloud Comput.*, vol. 13, no. 1, p. 94, May 2024.

[53] Z. Zhang, F. Zhang, Z. Xiong, K. Zhang, and D. Chen, "LsiA3CS: Deep-reinforcement-learning-based cloud–edge collaborative task scheduling in large-scale IIoT," *IEEE Internet Things J.*, vol. 11, no. 13, pp. 23917–23930, Jul. 2024.

[54] E. Zeydan, J. Mangues-Bafalluy, and Y. Turk, "Intelligent service orchestration in edge cloud networks," *IEEE Netw.*, vol. 35, no. 6, pp. 126–132, Nov. 2021.

[55] H. F. Shahid, J. Islam, I. Ahmad, and E. Harjula, "Optimizing resource-aware service orchestration in edge-cloud continuum," in *Proc. IEEE Intell. Mobile Comput. (MobileCloud)*, Sep. 2025, pp. 44–50.

[56] D. Ermolenko, C. Kilicheva, A. Muthanna, and A. Khakimov, "Internet of Things services orchestration framework based on Kubernetes and edge computing," in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (ElConRus)*, Russia, Jan. 2021, pp. 12–17.

[57] J. Okwuibe, J. Haavisto, I. Kovacevic, E. Harjula, I. Ahmad, J. Islam, and M. Ylianttila, "SDN-enabled resource orchestration for industrial IoT in collaborative edge-cloud networks," *IEEE Access*, vol. 9, pp. 115839–115854, 2021.

[58] A. Mahapatra, S. K. Majhi, K. Mishra, R. Pradhan, D. C. Rao, and S. K. Panda, "An energy-aware task offloading and load balancing for latency-sensitive IoT applications in the fog-cloud continuum," *IEEE Access*, vol. 12, pp. 14334–14349, 2024.

[59] B. Sonkoly, D. Haja, B. Németh, M. Szalay, J. Czentye, R. Szabó, R. Ullah, B.-S. Kim, and L. Toka, "Scalable edge cloud platforms for IoT services," *J. Netw. Comput. Appl.*, vol. 170, Nov. 2020, Art. no. 102785.

[60] X. Li, C. Li, X. Liu, G. Chen, and Z. Y. Dong, "Two-stage community energy trading under end-edge-cloud orchestration," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 1961–1972, Feb. 2023.

[61] J. Li, Y. Deng, W. Sun, W. Li, R. Li, Q. Li, and Z. Liu, "Resource orchestration of cloud-edge–based smart grid fault detection," *ACM Trans. Sensor Netw.*, vol. 18, no. 3, pp. 1–26, Aug. 2022.

[62] I. Petri, O. F. Rana, L. F. Bittencourt, D. Balouek-Thomert, and M. Parashar, "Autonomics at the edge: Resource orchestration for edge native applications," *IEEE Internet Comput.*, vol. 25, no. 4, pp. 21–29, Jul. 2021.

[63] L. R. Raju, M. V. K. Reddy, S. R. Surukanti, G. Sudhakar, M. V. S. Sarma, and A. Adepu, "Intellischeduler: An edge-cloud computing environment hybrid deep learning framework for task scheduling based on learning," *Sci. Reports*, vol. 16, p. 11219, Feb. 2026.

[64] F. B. Oliveira, M. Di Felice, and C. Kamienski, "Iotdeploy: Deployment of IoT smart applications over the computing continuum," *Internet Things*, vol. 28, Dec. 2024, Art. no. 101348.

[65] Y. Liu, H. Lu, X. Li, D. Zhao, W. Wu, and G. Lu, "A novel approach for service function chain dynamic orchestration in edge clouds," *IEEE Commun. Lett.*, vol. 24, no. 10, pp. 2231–2235, Oct. 2020.

[66] F. Habeeb, K. Alwasel, A. Noor, D. N. Jha, D. AlQattan, Y. Li, G. S. Aujla, T. Szydlo, and R. Ranjan, "Dynamic bandwidth slicing for time-critical IoT data streams in the edge-cloud continuum," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 8017–8026, Nov. 2022.

[67] J. Tang, J. Nie, Z. Xiong, J. Zhao, Y. Zhang, and D. Niyato, "Slicing-based reliable resource orchestration for secure software-defined edge-cloud computing systems," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2637–2648, Feb. 2022.

[68] D. Lan, A. Taherkordi, F. Eliassen, L. Liu, S. Delbruel, S. Dustdar, and Y. Yang, "Task partitioning and orchestration on heterogeneous edge platforms: The case of vision applications," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7418–7432, May 2022.

[69] A. Shankar, S. Mumtaz, J. J. P. C. Rodrigues, P. Karthikeyan, and V. Sarveshwaran, "Energy-efficient task orchestration in the edge-cloud continuum using deep reinforcement and federated learning for sustainable IoT," *IEEE Trans. Ind. Informat.*, vol. 22, no. 7, pp. 5724–5735, Jul. 2026.

[70] V. R. Chintapalli, K. Kondepu, A. Sgambelluri, A. Franklin, B. R. Tamma, P. Castoldi, and L. Valcarenghi, "Orchestrating edge- and cloud-based predictive analytics services," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2020, pp. 214–218.

[71] R. Rosmaninho, D. Raposo, P. Rito, and S. Sargento, "Edge-Cloud continuum orchestration of critical services: A smart-city approach," *IEEE Trans. Services Comput.*, vol. 18, no. 3, pp. 1381–1396, May/Jun. 2025, doi: 10.1109/TSC.2025.3568251.

[72] K. Petrakis, E. Agorogiannis, G. Antonopoulos, T. Anagnostopoulos, N. Grigoropoulos, E. Veroni, A. Berne, S. Azaiez, Z. Benomar, H. Kakoulidis, M. Prasinos, P. Sotiriades, P. Mavrothalassitis, and K. Alexopoulos, "Enhancing DevOps practices in the IoT–edge–cloud continuum: Architecture, integration, and software orchestration demonstrated in the COGNIFOG framework," *Software*, vol. 4, no. 2, p. 10, Apr. 2025.

[73] M. Amadeo, C. Marche, G. Ruggeri, S. Ranjbaran, and M. Nitti, "Composing digital twins for Internet of Everything applications: A user-centric perspective," in *Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom)*, Jul. 2024, pp. 73–78.

[74] C. Marche and M. Nitti, "Towards trustworthy digital twins collaboration in the Internet of Things: An overview of essential design guidelines," *Comput. Netw.*, vol. 272, Nov. 2025, Art. no. 111733.

CLAUDIO MARCHE (Member, IEEE) received the M.Sc. degree in telecommunication engineering and the Ph.D. degree in electronic and computer engineering from the University of Cagliari, in 2018 and 2023, respectively. Since graduation, he has been working as a Researcher with the Department of Electrical and Electronic Engineering, University of Cagliari. He is currently an Assistant Professor with the University of Cagliari, where he is a Member of the Net4U Research Group. He is currently a Member of the Editorial Board for the IEEE Internet of Things Journal and the Frontiers in Energy Efficiency Journal. His current research interests include the Internet of Things (IoT), social networks, cloud-edge computing, and trust management.