

Towards Establishing Intelligent Multi-Domain Edge Orchestration for Highly Distributed Immersive Services: A Virtual Touring Use Case

Tarik Zakaria Benmerar^{1*†}, Theodoros Theodoropoulos^{2†},
Diogo Fevereiro^{3†}, Luis Rosa^{3†}, João Rodrigues^{4†}, Tarik Taleb^{5†},
Paolo Barone^{6†}, Giovanni Giuliani^{6†}, Konstantinos Tserpes^{2†},
Luis Cordeiro^{3†}

^{1*}ICTFICIAL Oy, Finland.

²Harokopio University of Athens, Greece.

³OneSource, Portugal.

⁴Cyango, Portugal.

⁵Ruhr University Bochum, Germany.

⁶Hewlett Packard Enterprise, Italy.

*Corresponding author(s). E-mail(s): tarik.benmerar@ictficial.com;
Contributing authors: ttheod@hua.gr; duarte.fevereiro@onesource.pt;
luis.rosa@onesource.pt; joao.rodrigues@cyango.com; tarik.taleb@rub.de;
paolo.barone@hpe.com; giuliani@hpe.com; tserpes@hua.gr;
cordeiro@onesource.pt;

†These authors contributed equally to this work.

Abstract

Edge cloud technologies working in conjunction with AI-powered solutions can help surmount the challenges associated with the distributed execution of immersive services and contribute to delivering a positive end-user experience. Intelligent resource management, orchestration, and predictive systems can enhance service deployment, adapt to changing demands, and ensure seamless service operation. This paper introduces an innovative architectural paradigm that enables multi-domain edge orchestration for highly distributed immersive services by leveraging various AI solutions and technological tools to support multi-domain edge deployments. The proposed architecture is designed to function based on multi-level specification blueprints, separating high-level user-intent infrastructure definition from AI-driven orchestration and the final execution

plan. This architectural design enables the incorporation of AI solutions to be conducted in a modular manner. Furthermore, the Application Management Framework (AMF) provides a visual language and tool as an alternative to formal methods for creating intent blueprints. The proposed architecture is evaluated within the frame of an immersive virtual touring use-case scenario.

Keywords: Edge cloud, immersive service, orchestration, cluster, Kubernetes, centralized management, and decentralised management

1 Introduction

Extensively dispersed immersive services hold the promise of transforming various sectors, offering novel and inventive approaches to perceiving and engaging with our surroundings. For example, Virtual Reality (VR) and Augmented Reality (AR) can offer students interactive and captivating educational encounters. Similarly, VR simulations can assist in training professionals in diverse fields, including healthcare and aviation. Within the realm of entertainment, immersive services have revolutionized the gaming industry, enabling players to completely immerse themselves in virtual realms, while AR can enrich live events by superimposing interactive digital elements onto real-world settings.

Unfortunately, the delivery of immersive services at a high level of Quality of Service (QoS) hinges on the availability of extremely low latency and high bandwidth connections, as referenced in previous studies [1–4]. Existing scientific literature recommends that to ensure a satisfactory end-user experience, the end-to-end latency should not surpass 15 milliseconds, and the available bandwidth should scale up to 30 Gbps, as outlined in relevant research [5–7]. Moreover, potential glitches in task processing can disrupt service delivery and jeopardize the immersive experience’s integrity. Hence, it is imperative for such applications to possess fault-tolerance capabilities.

Additionally, immersive applications demand substantial computational resources involving intricate 3D models and high-quality graphics. However, embedding these essential computational resources within end-user devices would lead to bulky and costly setups, contradicting the core principles of immersive applications, which prioritize portability and affordability of end-user equipment, as discussed in [8]. Cloud computing presents a solution by offloading the computational load to remote resources, enabling end-user devices to remain portable and cost-effective. Nevertheless, cloud topologies face limitations in meeting immersive services’ ultra-low latency and high bandwidth requirements, primarily due to the complex networks that intervene between end-user devices and cloud servers.

The primary goal of edge computing is to minimize the volume of data sent to remote cloud servers, enabling data processing to occur in close proximity to the data sources. Consequently, edge architectures offer advantages such as quicker response times, enhanced data transfer speeds, and improved scalability and reliability. Consequently, deploying immersive services across a distributed cloud-edge infrastructure would be advantageous for application developers, contributing to the maintenance

of high-quality Quality of Service (QoS) provisioning, as noted in the study by Taleb (2022) [9].

However, a significant challenge remains unresolved: the efficient allocation of distributed tasks across the network and computational resources within the cloud-edge continuum, particularly within the requirements in terms of latency, bandwidth, scalability, and reliability inherent to distributed immersive applications. To address these challenges, Machine Learning (ML) and Deep Learning (DL) can play a pivotal role by offering intelligent resource management and orchestration systems capable of adapting to the evolving requirements of end users. For instance, Artificial Intelligence (AI)-driven load balancing and resource allocation algorithms can optimize service deployment across the cloud-edge infrastructure, accounting for variables such as network conditions, computational resources, and user demands. Additionally, AI-powered predictive analytics can be harnessed to foresee future demands and proactively allocate resources as needed [10], thus averting potential bottlenecks or service disruptions. Furthermore, AI-driven monitoring and fault detection systems can contribute to the seamless operation of services and the early identification of issues before they impact the end-user experience.

On top of the aforementioned AI solutions, Application Management Frameworks (AMFs) play a crucial role in the deployment of applications in edge computing. AMFs offer a systematic approach for creating, documenting, controlling, and implementing applications within an edge computing environment. The significance of AMFs lies in their capacity to establish a common language and framework that allows stakeholders to collaborate effectively in the design, execution, and management of applications. Through the adoption of standardized practices, AMFs can ensure uniformity, interoperability, and scalability throughout the system while also empowering stakeholders to make well-informed decisions regarding trade-offs between cost, performance, and other factors.

Moreover, it is imperative to integrate the necessary technological facilitators to streamline the orchestration and management of highly distributed immersive services, particularly as these deployments often span multiple domains. Therefore, these technological facilitators have to prioritize multi-domain edge orchestration. In line with this objective, this paper presents an innovative architecture that simplifies intelligent multi-domain edge orchestration for immersive services. The proposed architectural framework is designed to operate using multi-tier specification blueprints, distinguishing the high-level user-intent infrastructure definition from AI-powered orchestration and the ultimate execution strategy. This architectural approach facilitates the integration of AI solutions in a modular fashion. Additionally, the AMF offers a visual language and tool as an alternative to formal techniques for shaping intent blueprints. The viability of the proposed architecture is assessed within the context of an immersive virtual touring use-case scenario.

This work aims to expand upon the findings of one of our previous works [11] by providing a deeper and more detailed analysis of the intricacies of the proposed architecture. Towards achieving this goal, the remainder of this paper is organized in the following manner: Section 2 explores some research work relevant to intelligent multi-domain edge orchestration. Section 3 introduces the architecture of the envisioned

Table 1 Structure of this paper.

Section	Title
2	Related Work
3	Intelligent Multi-Domain Edge Orchestration
3.1	Native AI and Intent-driven Multi-domain Orchestration Architecture
3.2	From Application Intents to Infrastructure Blueprints
3.3	Service Deployment Planning
3.4	Decentralised and Centralised Cluster Management
3.5	AI-Driven Provisioning and Lifecycle Management
3.6	Infrastructure and Application Monitoring
3.7	Inter-Cluster Peering
3.8	Execution Graph-based Blueprint Orchestration
4	Immersive Service Use Case
4.1	Immersive Virtual Touring
4.2	Harnessing Metrics for Intelligent Orchestration
4.3	Application Management Framework
4.4	Architectural Evaluation
5	Conclusion

intelligent multi-domain edge orchestration system. Section 4 describes an immersive virtual touring use-case scenario to validate the proposed architectural paradigm, as well as a corresponding Application Management Framework (AMF) which provides a visual language and tool alternative to the formal approach for the intent blueprint. Finally, Section 5 summarizes the merits of this work. An overview of the paper structure is depicted in Table 1.

2 Related work

Edge computing is geared towards addressing the growing demands and requirements of the next generation of highly distributed applications [12]. Each cluster of edge nodes is responsible for processing data from multiple applications and is designed to handle the specific processing needs of these applications to reduce latency, improve data privacy, enable real-time decision-making, achieve high scalability and resilience, and allow better resource utilization. 3GPP SA6 [13] proposes an edge computing-based architecture for enabling Edge Applications (EdgeAPP). EdgeAPP is built on principles of application portability, service differentiation, flexible deployment and interworking with the 3GPP network. EdgeAPP specification discusses aspects such as service provisioning, registration, Edge Application Server (EAS) discovery or Service Continuity (i.e., maintaining a service in case of user mobility or migration).

Moreover, a trend is emerging in favour of multi-cloud architectures [14][15]. Such a new paradigm allows for harnessing the advantages offered by diverse cloud providers strategically distributed across geographical locations. Each provider offers unique functionalities while sidestepping the limitations of single-cloud and single-provider architectures. Nevertheless, orchestrating a multi-cloud environment remains a complex challenge. Previous studies have delved into diverse aspects of resource provisioning, resource selection, and the overall management of multi-cloud environments. In [16], the authors systematically reviewed cloud resource orchestration frameworks,

comparing their distinctive features, deployment models, interoperability or integration with external services and systems. One of their observations emphasises the relevance of application portability and the usage of standards for describing them. In 2013, the OASIS Foundation introduced Topology and Orchestration Specification for Cloud Applications (TOSCA) to promote the standardization of cloud-native applications and services [17]. TOSCA enables describing these applications using a universal YAML language that remains agnostic to specific orchestration solutions or tools. Since then, TOSCA specification has been used in several works. In [18], the authors present an architectural design that facilitates Kubernetes cluster federation and cloud applications, capitalizing on utilising TOSCA blueprints across diverse cloud providers. Other authors have explored the networking factors associated with multi-cluster architectures. In [19], the authors focused on the challenge of interconnecting multiple clusters (and clouds) and service mobility between them. They proposed using Network Service Mesh (NSM) operating at a Layer2-L3 level to interconnect the various clusters. A more integrated orchestration platform for multi-cloud deployments with policy-based scheduling, autoscaling, and cloud bursting capabilities was presented in [20]. It aligns with Kubernetes' core principles, seeking to optimize resource utilization, ensure application placement within available resources, adapt applications to changing traffic patterns, and avoid over-provisioning concerns.

ZSM (Zero-touch network and Service Management) is an end-to-end management reference architecture developed by ETSI to provide a flexible and automated [21] approach to managing services and infrastructure in a 5G network. It comprises six building blocks: Management Services, Management Functions, Management Domains, E2E Service Management Domain, Integration Fabric, and Data Services. The ZSM Management Services provide a standardized and consistent way to expose different management capabilities across a multi-domain deployment. Management Functions combine multiple capabilities to form broader management features. The Management Services are organized into Management Domains, where services can either be internal or exposed outside the domain. The ZSM framework also allows for a hierarchy of Management Domains, where multiple domains can be stacked on top of each other. Integration Fabrics facilitate communication between management functions. The Domain Integration Fabric connects services within the same domain, while the Cross-Domain Integration Fabric facilitates communication over different domains. Both fabrics are used as a communication bus and to register, discover, and invoke different supported services. Data Services allow for the decoupling and reusing of the same management data across distinct management services. Ultimately, ZSM can be seen as a strategy to move from automatic to autonomous (A2A) orchestration architectures considering mechanisms such as policy-driven, intent-based, network governance, network stability, reinforcement learning, and transfer learning [22]. In [23], ZSM enabling technologies and networking automation solutions are analysed to close the gap between mobile networking and ZSM. In [24], the authors explore the potential of cloud-native architectures for enhancing service development and resource management to meet different service requirements. In [25], a ZSM-inspired orchestrator was proposed for managing network services autonomously while profiling the performance of such services.

Following the advancements in AI methodologies, there have also been numerous attempts [26],[27] at incorporating them to build further ZSM developments in different aspects, such as multi-tenancy management, traffic monitoring, and architecture coordination. In [28]. ETSI details a list of relevant AI-enabling areas for AI-driven network management, such as Trustworthy Machine Learning, Decentralised Machine Learning, AI/ML model validation, Anomaly Management using AI/ML-based closed loops, ML model cooperation and federated learning.

Multi-domain E2E service lifecycle management can be split into three categories of processes: onboarding, fulfilment and assurance [29]. The first two deal with the aspects of the service bootstrapping (e.g., service onboarding, service activation, reconfiguration, decommissioning), whereas assurance processes are used to continuously (in a close-loop) monitor and guarantee processes are running as supposed and according to the expected Service Level Agreement (SLA) and QoS [30]. Likewise, Intent-driven architectures provide manifold benefits, including the promise to help to simplify the management of complex infrastructures such as 5G multi-vendor deployment scenarios as described in 3GPP specifications [31, 32]. Intents focus on what needs to be achieved regardless of the actual implementation or the underlying infrastructure details [33]. ETSI created the Experiential Network Intelligence (ENI) framework to enable networks to leverage the benefits of AI methodologies while ensuring they meet Quality of Service requirements [34]. The ENI Cognitive Architecture model involves a set of hierarchical closed control loops based on the Observe-Orient-Decide-Act model, with extensions to accommodate collaborative decision-making, learning, and policy management. These enhancements enable the system to adapt its behaviour according to changes in user needs, business goals, and environmental conditions. It operates in two modes: recommendation and command. The former functions as an assistant recommending actions, and the latter functions as an actual governing other management components.

There is also now a plethora of emerging tools and enablers focused on supporting the management of containers, Virtual Machines (VMs) and services across the edge and cloud environments. Open Network Automation Platform (ONAP) [35] is designed to automate the composition and creation of network services. Akraino Edge Stack [36] is a Linux Foundation project focused on creating a framework for edge computing, providing a set of blueprints and reference architectures that help developers build and deploy edge applications. ClusterAPI [37] is a Kubernetes project that aims to provide declarative APIs for cluster creation, management, and lifecycle management, simplifying the creation and management process of Kubernetes clusters in multiple cloud providers, on-premises data centres, and hybrid environments. Open Source MANO (OSM) [38] is an ETSI project that aims to provide a platform for deploying, managing, and monitoring virtual network functions (VNFs) and network services. Cloudify [39] is a multi-cloud management platform designed to automate and manage the deployment of complex applications and services across multiple clouds and data centres with support for hybrid and multi-cloud environments. OpenShift[40] is a container application platform built on top of Kubernetes that provides developers with an integrated environment for building, deploying, and scaling containerized applications.

Unfortunately, none of the aforementioned solutions are specifically designed to facilitate the intricacies that are intertwined with the multi-domain edge orchestration for highly distributed immersive services. This work aims to fill this research gap by proposing a novel architectural framework that considers the various aspects of the aforementioned technologies and solutions that, when combined, can facilitate multi-domain edge orchestration for highly distributed immersive services. In fact, the incorporation of the necessary technological enablers can enhance the organization and control of extensively distributed immersive services. This is particularly crucial given that these deployments often extend across various domains. Consequently, these technological enablers should prioritize orchestrating activities across multiple domains at the edge. In pursuit of this objective, this paper introduces an inventive architecture that simplifies intelligent multi-domain edge orchestration for immersive services. The proposed architectural framework is crafted to function using multi-tier specification blueprints. It distinguishes the high-level definition of user intent infrastructure from AI-driven orchestration and the eventual execution strategy. This architectural approach facilitates the modular integration of AI solutions. Finally, the AMF provides a visual language and tool as an alternative to formal techniques for shaping intent blueprints.

3 Intelligent Multi-Domain Edge Orchestration

This section discusses the key strands of the proposed intelligent multi-domain edge orchestration, namely the reference architecture, the blueprints to express applications and infrastructure, the service planning and deployment steps, the role of Native AI and AI-based mechanisms to fulfil the needs of immersive services, the monitoring and the core metrics and finally, the concept of inter-cluster peering to facilitate distributed application deployments.

3.1 Native AI and Intent-driven Multi-domain Orchestration Architecture

Edge computing and multi-domain architectures are two emerging technologies meant to disrupt how immersive services are built and delivered. Amongst others, they enable service deployments closer to users, a more efficient and, therefore, sustainable edge-cloud continuum utilization, and last but not least, heterogeneous infrastructure composition (i.e., no restrictions to a single provider or single cluster deployments). Nevertheless, there is a gap between immersive application developers' intentions and the expertise needed to maintain and orchestrate a (complex) multi-domain environment. Apart from the infrastructure, a deep understanding of cloud-native architectures, tools, mechanisms and protocols is needed. Managed solutions provide a step towards alleviating such complexity. Still, they typically fail to deliver an intuitive way of expressing the developer's intentions or do not include advanced features such as autonomous service deployment and lifecycle management, which are increasingly relevant in these scenarios.

Figure 1 depicts the proposed native AI and intent-driven multi-domain orchestration architecture conceived to support the service provisioning and life-cycle

management of highly distributed immersive services across a distributed edge-cloud infrastructure. Such architecture aims to empower immersive application developers with tools for (visually) expressing and composing their applications. Later, the proposed architecture aims to translate application blueprints into orchestration and execution blueprints, which are used to ensure the expected lifecycle of the application's components.

Such architecture was designed to: i) take into consideration immersive service expectations and intents; ii) abstract the virtualized physical infrastructure from applications-specific deployments; iii) take advantage of multi-domain, multi-stakeholder environments and exploit the full Edge-Cloud continuum; iv) incorporate the concept of Native AI orchestration capabilities (c.f. Section 3.5; v) energy efficiency and QoE optimization (e.g. by deciding the most suitable location for allocating resources, on-demand resource provisioning - including the cluster creation, or by continually monitoring and reacting to resource patterns) vi) service lifecycle automation leveraging the concept of Zero-Touch and automated closed loops.

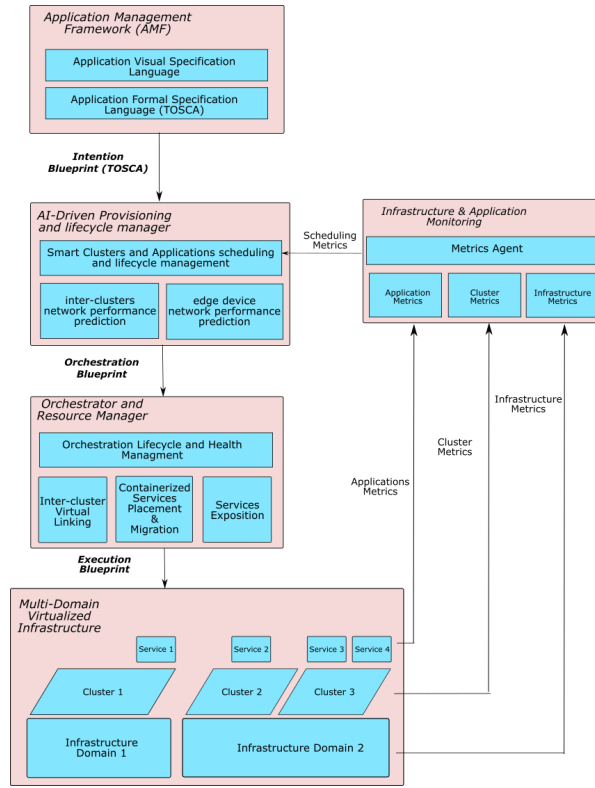


Fig. 1 Intelligent Multi-Domain Edge Orchestration architecture.

The proposed architecture is composed of five key substrates as follows:

- **Application Management Framework** - a user-friendly front-end UI for immersive application developers to compose their applications. Automatically translates component composition and definitions provided by developers into application intent blueprints in TOSCA format (c.f. Section 3.2). Provides the means to trigger application deployments via human interaction or via API (for deployments triggered by devices).
- **AI-Driven Provisioning and Life-cycle Manager** - includes the Native AI mechanisms for intelligently devising applications' best scheduling plans based on infrastructure characteristics (e.g., place services requiring GPU support on GPU-enabled locations) or based on optimization criteria (e.g. user-proximity, energy efficiency, security constraints). This substrate is also responsible for continuously predicting resource utilization to support proactive service and infrastructure management (e.g. scale-in/-out clusters and components on-demand, anticipate service migration needs).
- **Infrastructure and Application Monitoring** comprises the set of monitoring agents responsible for gathering and exposing application, cluster and infrastructure metrics.
- **Orchestrator and Resource Manager** - comprises the building blocks and primitives which allow enforcing the decisions (i.e., orchestration blueprints) into an execution plan (i.e., the execution blueprint). Orchestrator and Resource Manager allow seamless integration with different cloud and infrastructure providers by providing the means to create clusters across numerous domains transparently to end-users. Such clusters form a cohesive edge-cloud computing continuum, providing the flexibility to leverage multiple locations and select the most suitable one for each service component, allowing optimal resource utilisation and enhancing the deployed services' overall efficiency.
- **Multi-Domain Virtualized Infrastructure** is formed by aggregating available infrastructure providers and a list of existing clusters and application services.

3.2 From Application intents to infrastructure blueprints

Our proposed orchestration solution is built around three types of blueprints: User Intent, Orchestration and Infrastructure Blueprints. Together, they define different layers of details related to the application deployment. This allows the separation of concerns between what the end-user intends at a high level from the actual implementation and execution, including the AI-driven optimized decisions and the low-level infrastructure deployments and configurations.

3.2.1 User Intent Blueprint

At a high level, users describe their intention regarding the functional architecture of their applications regardless of the underlying infrastructure. This description provides opportunities for an intelligent scheduler to optimise networking and resources while respecting the initial user intent. Based on an extended version of the industry standard *OASIS TOSCA* (Topology and Orchestration Specification for Cloud Applications) [41], a blueprint specification is defined for the application deployment

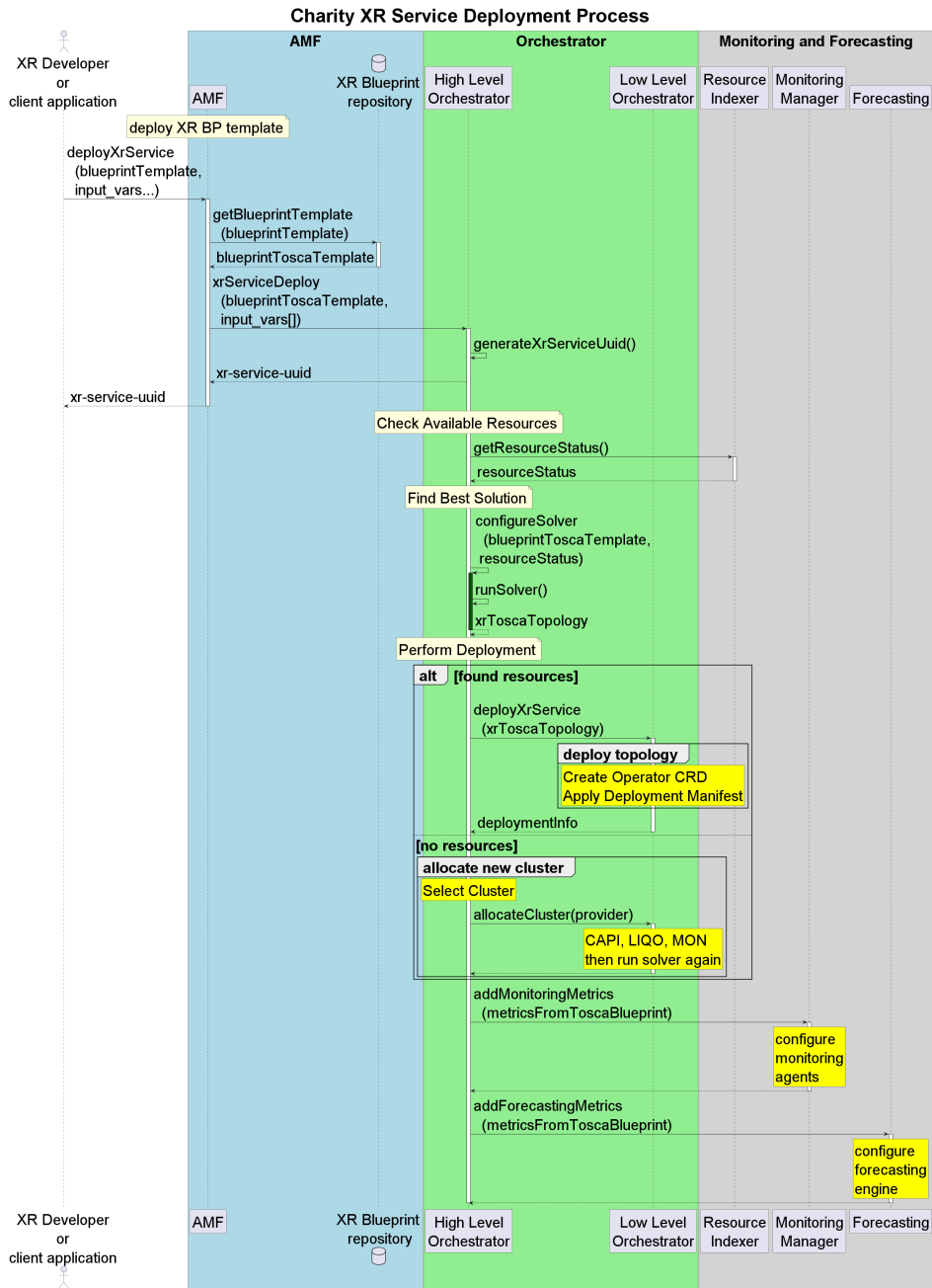


Fig. 2 High-level flow of operations for the deployment of an XR service, with cluster selection or creation and activation of monitoring and forecasting

model. The user intent blueprint includes a high-level view of the application represented as a composition of so-called *XR Service Enablers* - modular services packaged

into containers or Virtual Machines - defining user application services images and the connection points and virtual links between them. In addition, users can specify requirements in terms of resource needs (e.g., number of cores, RAM, GPU, storage), number of replicas, and expected Quality of Service (e.g., bandwidth and latency). Resource definitions drive the choices of the decision layer on the most suitable targets for infrastructure provisioning. On the other hand, QoS requirements define SLAs that the selected infrastructure must satisfy at runtime. Hence they are drivers for the monitoring and service lifecycle loops. For the blueprint definition phase, users describe their application from the AMF graphical front end, which guides them in defining the building blocks, their interconnections, and the requirements and the input parameters or environment variables that may be required at deployment time. The AMF then generates the related TOSCA representation for the application model. Figure 3 describes the high-level sequence of interactions between an XR Developer and the AMF to create an XR Service Blueprint.

For the deployment phase, the AMF front-end provides two modes: human and machine-to-machine interaction. Human interaction leverages the GUI front-end to select an application blueprint and deploy from it an application instance, with forms for manually entering input parameters. Machine-to-machine interaction leverages a REST API to be invoked by a device or a system to trigger the deployment of a specific blueprint. The overall operational flow for deploying an XR Application, comprising the orchestration phases for cluster selection or creation and the activation of monitoring and forecasting [42], is depicted in the sequence diagram in Figure 2.

3.2.2 Orchestration Blueprint

The TOSCA user intent blueprint defines the application at a higher level. It doesn't specify how the infrastructure and the services are created and which resources are used. The intelligent scheduler harnesses the TOSCA definition and the live monitoring data during application runtime to create and update the orchestration blueprint containing the detailed infrastructure and services provisioning. The orchestration blueprint structure is specified using a Kubernetes CRD (Custom Resource Definition) [43] and submitted to a management cluster. CRDs provide a way to extend Kubernetes with new kinds of resources. As detailed further, a CRD leverages an accompanying operator for the lifecycle management of the new type of resource. The orchestration blueprint uses a similar structure to the TOSCA User Intent blueprint but enriches it with the required details for the Kubernetes cluster provisioning. It specifies the infrastructure providers used for Kubernetes cluster provisioning with additional parameters such as the number of control planes and worker machines, deployment region to be used, machine images, etc.

Moreover, the TOSCA user intent blueprint only provides high-level hints of edge needs for guiding their provisioning. It does not specify how the edges will be created, how many will be required and where they need to be provisioned. On the contrary, the orchestration blueprint specifies all the Kubernetes-based edge clusters that will be provisioned with the required infrastructure parameters as any other Kubernetes cluster in the orchestration blueprint. It is important to note that the intelligent scheduler updates the edge definitions in the blueprint during application runtime

using the live monitoring data and the performance hints specified in the TOSCA user intent blueprint. The decision blueprint is divided into three sections as follows:

- **Clusters:** defines the Kubernetes clusters to be created. It specifies the cluster provider parameters and details (i.e., number of machines, deployment region, etc.).
- **Services:** defines the containerized services to be deployed. It specifies in which cluster the service will be deployed, the container image used, ports exposed, the number of replicas and other parameters required for correct execution (e.g. environment variables).
- **Links:** defines which services expositions across two clusters using secured virtual links. This allows strong multi-domain communication security without publicly exposing services over the internet.

While the blueprint parameters are currently fixed for the different sections, an extensible blueprint specification has to be considered in the long term to make the orchestration of different use cases viable.

3.2.3 Infrastructure Blueprints

Based on the orchestration blueprint, the initial application deployment and subsequently updated deployments are handled by a Kubernetes Operator. This later is a software extension to execute the orchestration blueprint corresponding to a Kubernetes Custom Resource instance. It is important to note that the Operator pattern and CRDs (Custom Resource Definitions) are the de facto pillars for extending Kubernetes functionalities.

Based on *clusters* section details in the orchestration blueprint, the operator set up the required third-party clusters bootstrapping blueprints required for their provisioning on the specified cloud/infrastructure provider. Currently, *ClusterAPI* was chosen as the provider for cluster setup. ClusterAPI provides manifold benefits, including the ability to instantiate and manage the lifecycle of Kubernetes on widely used cloud providers. *Services* section in the orchestration blueprint is used by the operator to set up the necessary Kubernetes deployment resources to the specified cluster. From the *links* section in the orchestration blueprint, the operator set up the *VPN* links through inter-cluster peering. *Ligo*, detailed later, is used as the third-party tool for this operation.

3.3 Service Deployment Planning

As explained earlier, the operator sets up the required clusters for the application services to be deployed. Regarding deployment planning, at least two approaches can be considered: different services and applications isolated in their own clusters or having them deployed unto the same cluster for consolidation purposes. In the first approach, every new service deployment requires tearing up a new cluster for the application beforehand. This process requires a certain amount of time, which adds to the application setup time. In the latter approach, the consolidation reduces the setup time. Nevertheless, it adds a significant amount of logic complexity and an elaborate security system that must be in place to guarantee complete isolation between services

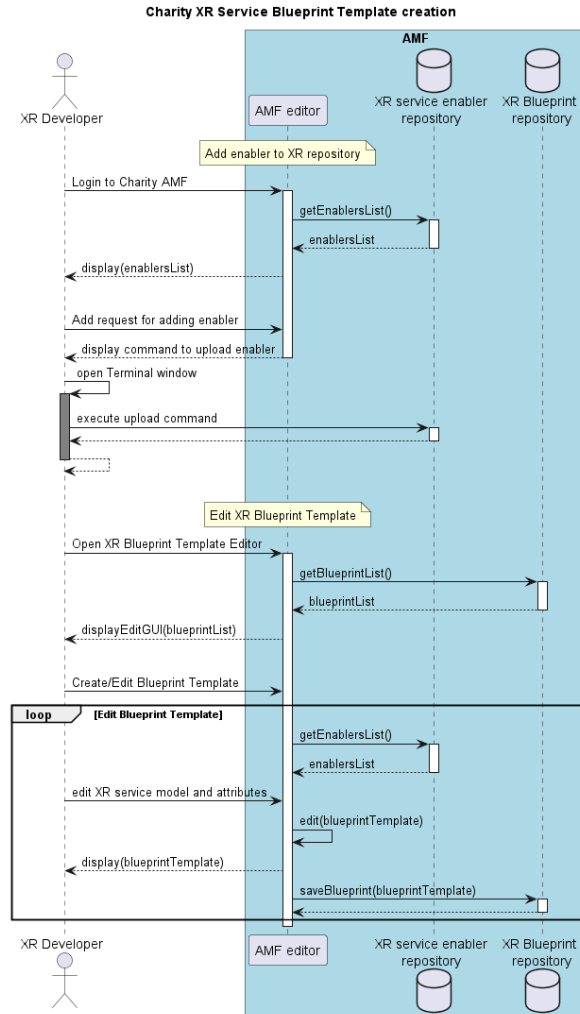


Fig. 3 Operations for editing a Blueprint Template for an XR Service

from different applications. For the sake of simplicity, the first approach was chosen, although improvements are planned to support additional deployment schemes, which, although more complex, can bring manifold benefits, including a more sustainable infrastructure utilization.

3.4 Decentralised and Centralised Cluster Management

The proposed solution involves the usage of distinct Kubernetes clusters for management and deploying application components. The orchestration components are deployed in a dedicated Kubernetes cluster for security, resource efficiency, and operational flexibility. This allows for better isolation of sensitive data and critical resources,

including the orchestration blueprints. Nevertheless, we can further consider different architectural strategies, considering how the orchestrator keeps track of application blueprint changes and synchronises their states with the actual infrastructure or how the different components are exposed. Two main approaches can be pursued: centralised or decentralised (cf. Fig. 4).

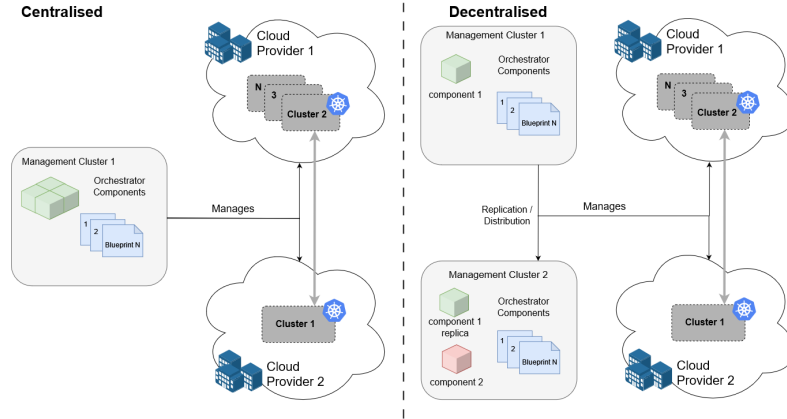


Fig. 4 Centralised(top) vs decentralised(bottom) multi-domain management clusters

In a centralised approach, a single Kubernetes cluster manages the multi-domain infrastructure. All the orchestration components and the application’s blueprints reside in the same cluster. Even considering the possibility of a multi-node cluster scenario, the management part can still be considered a Single Point of Failure (SPOF) at the cluster level. Such an approach can also lead to scalability issues as we increase the number of elements to manage (i.e., more applications and workload clusters). Moreover, the redundancy strategies are limited to the cluster scope (e.g., component replicas within the same cluster or high-availability Kubernetes cluster). On the other hand, maintaining and managing a single cluster is simpler as there are no multi-domain constraints, heterogeneity or inter-cluster communication to consider.

As opposed to that, in a decentralised approach, multiple management clusters are used. Those clusters are used for either replicating components, ensuring fault tolerance and availability, or distributing them for enhanced scalability and performance. In a replication approach, each replica can be seen as an independent entity containing one or more replicas of the various components. Whereas this approach allows for more than a single-point failure, it has the significant downside of requiring tight coordination among different parts, hence more complexity. For instance, application blueprint replicas require synchronization and consistency (e.g., using eventual or strong consistent strategies). Likewise, when a change is detected, a singular entity, such as a designated operator, orchestrates the conversion into infrastructure operations. This can be achieved through leader-based consensus algorithms or load-balancing individual tasks to various replicas. In that case, replication also contributes to the overall

scalability by allowing the parallelization and load distribution of operations across multiple domains.

On the other hand, a decentralised approach consisting of simply distributing the components (without replication) is also possible. Multiple clusters remove the barriers of single-cluster architectures and can be used to spread components across different domains, administratively or technology-wise. In such a scenario, a more strategic component placement can be considered for better resource efficiency, geographic diversity and isolation between environments (i.e., the clusters). Compared to multi-node Kubernetes architectures, it's worth noting that a multi-cluster approach represents a step further in infrastructure isolation. In the latter, component decentralization is not contingent upon a Kubernetes cluster's internal operations or Kubernetes nodes' synchronisation. All the coordination in that case is application-dependent. Still, the components spanning the various domains need to communicate with each other. A solution for that can be the usage of inter-cluster communication tools (refer to Section 3.7). Implementing a decentralised hybrid approach, combining replication and distribution, is indeed also possible. This approach allows us to account for a truly heterogeneous multi-domain environment, combining the benefits of replication and distribution. Nevertheless, as we move towards more decentralised approaches with all of their advantages, we must acknowledge that we are also introducing additional complexity and potential challenges in terms of synchronization.

3.5 AI-Driven Provisioning and Lifecycle Management

Highly distributed immersive applications in edge computing face several latency, bandwidth, reliability, and scalability challenges. These challenges can impact user experience and the overall performance of the application. AI solutions can be used for the lifecycle management of highly distributed immersive applications in edge computing. These solutions can significantly contribute towards optimizing the performance and reliability of these services throughout their lifecycle and ensure efficient resource allocation and management. In the context of the proposed architecture, the various AI solutions that shall be examined in this subsection correspond to the Deep Learning paradigm. Such solutions can be leveraged to accommodate the complexity associated with Multi-Domain Edge Orchestration for Highly Distributed Immersive Services. To achieve this goal, the authors of this work have identified two types of AI solutions. This taxonomy, which draws inspiration from the ENI specification from ETSI, was briefly explored in the Related Works section and is based on the role of AI in the context of the orchestration process.

The first type is indicative of AI solutions that use the available information to produce valuable insights that can be leveraged in the context of the orchestration process in the form of predictive analytics. This type describes a plethora of Deep Learning time-series forecasting[44] methodologies that are capable of performing accurate predictions regarding numerous critical factors such as network conditions, computational resources, and user demand. The orchestrator leverages these predictions. The second type is indicative of AI solutions designed to operate as orchestrators. To produce the various orchestration strategies, they examine a plethora of information, which

includes the aforementioned critical factors. This type describes various Deep Reinforcement Learning[45] methodologies that are in charge of functionalities such as task offloading, load balancing, and resource allocation.

Both types of AI solutions are implemented as parts of closed-control loops, similar to those described within ZSM and ENI specifications. As such, they play an integral role in the decision-making process and can contribute towards tackling the aforementioned challenges. Figure 5 illustrates a taxonomy that includes a plethora of AI solutions based on the challenges they can tackle.

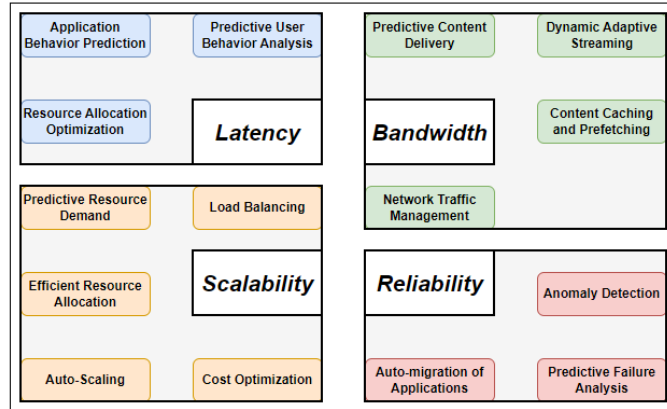


Fig. 5 A taxonomy that includes a plethora of AI solutions based on the challenge that they are capable of tackling.

3.5.1 Latency

Edge computing involves processing data as close as possible to the source or end-user devices instead of centralizing all processing in a remote data centre. This proximity to the data source minimizes the physical distance data must travel, significantly reducing latency. Edge computing, therefore, plays a crucial role in addressing latency concerns in immersive applications. However, it's important to note that distributing an application across multiple edge nodes can introduce latency due to data transfer between these nodes. The challenge here is to ensure that the distribution of the application and data is well-optimized to minimize these inter-node latencies.

Latency is a critical concern in immersive applications, where even the slightest delay can profoundly impact the user experience. In this context, latency refers to the time it takes for data to travel between the user's device and the processing unit, encompassing actions like rendering graphics, processing user inputs, and delivering real-time feedback. Edge computing has emerged as a powerful solution to reduce latency by processing data closer to the source, but it also introduces challenges related to the distribution of the application across multiple edge nodes. AI plays a vital role in mitigating these latency issues by proactively predicting user behaviour and application requirements. AI-Driven Latency Mitigation can be achieved via the following measures:

- **Predictive User Behavior Analysis:** AI can analyze and learn from user behaviour and interaction patterns. By understanding how users interact with the application, AI can predict their future actions [46]. For example, in a gaming application, if a user is engaged in a fast-paced action sequence, the AI can predict that they will need more processing power for rendering graphics and physics calculations in the near future. By predicting these actions, the AI can ensure that the necessary resources are pre-allocated, reducing the likelihood of latency during high-demand situations.
- **Application Behavior Prediction:** AI also extends its predictive capabilities to the application itself. It can monitor the application's behaviour and resource consumption patterns [47]. For instance, AI can detect when the application is about to enter a resource-intensive phase, such as rendering a complex 3D scene or processing a large dataset. By anticipating these processing requirements, AI can instruct the edge nodes to allocate additional computing resources in advance, ensuring smooth operation during resource-intensive tasks.
- **Resource Allocation Optimization:** With the ability to predict both user behaviour and application requirements, AI optimizes resource allocation in real-time. It can dynamically distribute computing power, memory, and other resources across the edge nodes to match the changing demands of the application [48]. For example, if AI anticipates a surge in user interactions that require immediate processing, it can allocate additional resources to the edge node responsible for handling those interactions.

3.5.2 Bandwidth

Bandwidth is critical to immersive applications, particularly those involving multimedia content, such as VR and AR. These applications require substantial bandwidth to stream high-quality content in real time. In the context of immersive applications, limited bandwidth in edge networks can lead to delays and interruptions in streaming multimedia content. To mitigate these challenges, AI can play a pivotal role in optimizing the use of available bandwidth. To address this issue, AI offers several strategies for optimizing bandwidth usage:

- **Predictive Content Delivery:** AI can analyze user behaviour, preferences, and historical data to predict the type of content that users are likely to access. By understanding user patterns, AI can preload relevant multimedia content onto edge devices [49]. This reduces the need for real-time streaming and minimizes data transfer requirements. Consequently, it conserves bandwidth and accelerates content delivery, reducing the risk of interruptions in the streaming experience.
- **Dynamic Adaptive Streaming:** AI can dynamically adjust the quality and resolution of multimedia content based on the available bandwidth and device capabilities [50]. When network bandwidth is limited, AI can reduce video resolution or compress audio, ensuring a smooth user experience. Conversely, when bandwidth improves, AI can enhance content quality, optimizing the streaming experience.
- **Content Caching and Prefetching:** AI can identify popular or trending content likely to be accessed by users. It can proactively cache or prefetch this content on

edge servers or devices, reducing the need for repeated downloads over the network [51]. This not only conserves bandwidth but also speeds up content access, especially for frequently requested items.

- **Network Traffic Management:** AI can analyze network traffic patterns and optimize data distribution across the edge network. It dynamically prioritizes and routes data based on real-time demand and network conditions, reducing congestion and ensuring efficient resource utilization [52].

3.5.3 Reliability

Reliability is a crucial concern in highly distributed immersive applications that operate within an edge computing environment. By virtue of their decentralised nature, these applications can be susceptible to network and node failures. The consequences of such failures can be disruptive, resulting in a degraded user experience or even service downtime. To address these reliability challenges, AI plays a pivotal role by continuously monitoring the performance and behaviour of the application and implementing proactive measures to ensure uninterrupted service. AI systems are equipped to monitor the performance and behaviour of the application in real-time. They continuously analyze many metrics, including network throughput, processing speed, resource utilization, and application response times. By doing so, AI can quickly detect any anomalies or deviations from the expected behaviour [53]. For example, if a node's performance starts to degrade or if there is a sudden increase in latency, AI can promptly identify these issues. Thus such systems are capable of enhancing reliability in edge-based immersive applications in the following ways:

- **Anomaly Detection:** AI leverages its analytical capabilities to detect anomalies in the application's performance [54]. When discrepancies are observed, such as an unusual surge in network traffic or a sudden spike in error rates, AI can raise alerts or take predefined actions to mitigate these issues. This rapid anomaly detection is vital in preventing minor problems from escalating into major failures.
- **Predictive Failure Analysis:** AI can go beyond mere anomaly detection and predict when a node is likely to fail. By analyzing historical data, hardware conditions, and the node's current state, AI can estimate the likelihood of a failure occurring in the near future. This proactive prediction enables the system to take preventive measures before failure happens [55].
- **Auto-migration of Applications:** In anticipation of an impending node failure, AI can orchestrate the seamless migration of the application to a different node. This proactive approach ensures continuity of service and minimizes disruption to users. The migration process involves transferring data, state, and processing tasks to a healthy node, all while ensuring data consistency and minimal downtime [56].

3.5.4 Scalability

Immersive applications are known for being resource-intensive. They require significant computing power, memory, and network bandwidth to deliver a seamless and engaging experience. Additionally, as the number of users grows, so does the demand for resources, making it essential to efficiently manage and allocate these resources to

ensure a consistently high-quality experience. AI is pivotal in addressing these resource management and scalability challenges in immersive applications.

- **Predictive Resource Demand:** AI leverages its predictive capabilities to estimate the number of users, their behaviour patterns, as well as the underlying resource usage [57]. By analyzing historical usage data and monitoring real-time user interactions, AI can make informed predictions about how many users are likely to access the application and what kind of activities they will engage in. For instance, in a VR application that involves multiple simultaneous users, AI can predict if there will be a surge in users during peak hours or special events.
- **Efficient Resource Allocation:** With predictive insights about user demand, AI can dynamically allocate resources more efficiently [58]. This involves provisioning the right amount of computing power, memory, and network resources to meet the anticipated demand. For example, if AI predicts a spike in user activity, it can allocate additional server capacity to handle the increased load, ensuring that the application remains responsive and performs optimally.
- **Auto-Scaling:** AI can automate the process of scaling resources up or down based on demand [59]. When the user count increases, AI can trigger the auto-scaling of additional servers or edge nodes to handle the load. Conversely, during periods of reduced demand, AI can release excess resources, optimizing cost efficiency.
- **Load Balancing:** Furthermore, AI can automate the process of load balancing across the various edge nodes based on a plethora of parameters [60]. Many immersive applications are distributed across multiple edge nodes for load balancing and redundancy. AI can optimize the allocation of resources across these nodes and dynamically route users to nodes with available resources, avoiding overloading any single node. Additionally, AI can monitor the performance of each node in real-time and redistribute workloads to ensure that no single node becomes a bottleneck.
- **Cost Optimization:** AI can also contribute to cost optimization by ensuring that resources are allocated in a cost-effective manner. It can balance the need for resource availability with cost constraints, helping organizations manage their cloud or edge computing expenses effectively [61].

Thus, within the frame of the proposed architectural paradigm, the two types of AI solutions do not operate independently but are instead envisioned to conduct their functionalities collaboratively. More specifically, the predictions/ insights produced by the first type of AI solutions can be leveraged by the orchestrating entities that belong to the second type. This enables the latter to devise more refined orchestration strategies that consider the future state of the multi-domain edge environment. Furthermore, the incorporation of the federated learning paradigm [62] into the aforementioned AI-driven functionalities is intertwined with a plethora of benefits in terms of privacy preservation, distribution of AI knowledge sharing, and enhanced learning efficiency in the context of distributed edge computing.

Federated learning [63] is an ML approach that enables training models across multiple decentralised / edge devices while keeping the data on those devices. Instead of sending data to a central server for training, the models are trained locally on the edge devices, and only the model updates or gradients are shared with a central

server for aggregation. This approach preserves data privacy and security, as sensitive data remains on the devices where it is generated. Edge devices contribute locally trained models, which are aggregated to create a global model. This process establishes cross-domain learning, where knowledge is shared without compromising privacy. The distributed nature of federated learning facilitates knowledge transfer from resource-rich to resource-limited devices, benefiting all devices. On top of that, training models on edge devices [64] reduces latency and allows real-time decision-making. Federated learning only exchanges model updates instead of raw data, thus minimizing communication overhead and conserving bandwidth. Finally, distributing the learning among edge devices [65] allows for horizontal scalability. As more edge devices join the federated learning process, the system can handle larger volumes of data and train more complex models without relying solely on centralized infrastructure.

3.6 Infrastructure and Application Monitoring

For the correct operation of the AI-driven provisioning and lifecycle management component, it must have access to historical and live monitoring data. Historical monitoring data are essential for correct network performance or workload predictions, particularly inter-cluster latency and bandwidth and edge devices network latency and application usage in our immersive experiences use-case. Live monitoring data act as real-time feedback to the smart scheduling decisions and determine whether the expected performances have been achieved or the infrastructure resources can still cope with the submitted workload.

Based on the defacto Kubernetes clusters monitoring tool **Prometheus** [66], the monitoring component creates a set of agents that aggregates all the historical and live monitoring data for the given application. Depending on a specific deployment layer performance we are interested in, different metrics agents can be deployed, namely: infrastructure metrics agents, cluster metrics agents, and application metrics agents.

3.6.1 Infrastructure Metrics Agents

All the application clusters are deployed unto the existing virtualized regional cloud infrastructures. The AI-driven provisioning and lifecycle manager harnesses historical data for the existing cloud regional infrastructure metrics to guide the cluster placement. Agents are deployed to the regional cloud infrastructures to gather the required metrics. These agents provide the resource data, latency, and historical bandwidth data between the cloud infrastructure regions. The provisioning and lifecycle manager can then make the required predictions for the placement and migrations of application services. Note that these agents can either be based on software deployed by the cloud provider or as part of a dedicated monitoring cluster independent of any application deployment.

3.6.2 Cluster Metrics Agents

Every cluster in the given application is able to handle the required deployed services workload while achieving the target performances. A set of cluster metrics agents are deployed into each cluster to continuously monitor the hardware resources (e.g.,

CPU, Memory) consumed by each node and deployed container, as well as network performance between clusters (e.g., latency and bandwidth). The provisioning and service lifecycle management component will then use these metrics to scale up or down the cluster nodes depending on the workload or for migrating services from one cluster to another to achieve better network latency or bandwidth.

3.6.3 Application Metrics Agents

Custom hints can be specified in the User Intent Blueprint as part of specific network or application workload performances. These hints are first aggregated from well-defined sources using a dedicated monitoring system [67] and, later, harnessed by AI-driven provisioning and lifecycle manager to optimise the intended hint criteria. A plugin system is put in place to achieve maximum flexibility in integrating the various application-specific metrics. Amongst others, this allows the case of communicating edge devices network performances and geolocalisation hints metrics. Metrics agents can also measure a particular application's Quality Of Service (QoS), such as a specific job of application queuing time, helping the user plan for service replication. Better, automated as part of our potential improvements to our system.

3.7 Inter-Cluster Peering

Nowadays, an immersive application consists of multiple micro-service components, which can benefit from being distributed across different clusters. Immersive application components highly depend on the capability to communicate with each other, regardless of whether they sit in the same or different locations (and clusters). As such, a transversal connectivity solution capable of enabling connectivity between clusters is increasingly required. Such a solution facilitates the deployment of cross-domain applications, enabling dynamic location-aware scheduling decisions (whether based on developer requirements or an AI-driven decision) independently from labour and time-consuming developer configurations. Several technologies, such as Liko or Submariner, promise to address such automated peering cluster connectivity and **service discovery** across Kubernetes clusters [68]. Liko, both free and open-source, is the solution adopted in our proposed architecture. Liko is designed to enable seamless connectivity among geographically distributed clusters (e.g., on-premises, edge or cloud). Liko relies on peer-to-peer secure (encrypted) connections between clusters to validate the identity clusters. Remote clusters are seamlessly abstracted through the concept of *virtual nodes* on the local cluster, allowing transparent communication between the peered clusters, regardless of the CNI plugin installed. Indeed, for bidirectional peerings, a virtual node is created in each cluster representing the resources the remote one provides. Moreover, Liko also brings the notion of *offloading* to reflect and execute workloads on top of those virtual nodes (e.g., namespaces, services and pods). This allows exposing services or even the execution of workloads in remote clusters. For instance, when a namespace is offloaded, Liko *extends* that namespace by creating a twin namespace in the remote cluster, enabling the pods and services to run

on that cross-cluster shared namespace. Figure 6 compares pod offloading versus service offloading. Both modes start with the peering of clusters (i.e., creating a dynamic VPN tunnel) and the creation of a shared namespace.

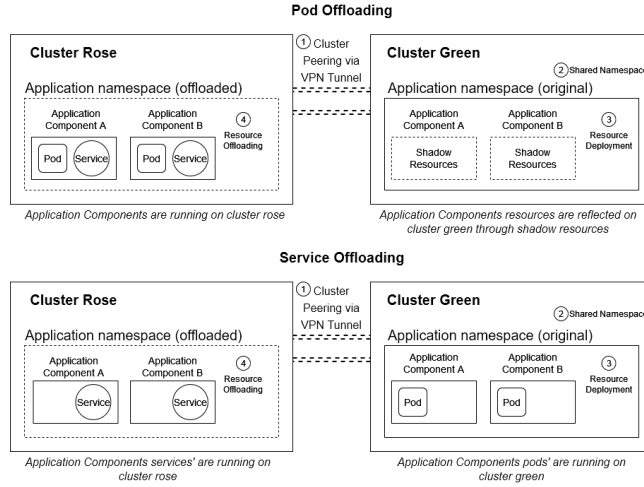


Fig. 6 Pod and Service Offloading comparison.

Nevertheless, the pod offloading strategy includes moving the actual execution of pods and the services to a peered cluster (e.g., in Figure 6, the application components are first deployed on the original Green Cluster and later executed in the Rose cluster). For instance, high-demand computing tasks, such as video processing or handling requests during peak traffic periods, can be easily moved to a (more suitable) cloud cluster. By offloading some of the application workloads to a cloud cluster, one can optimize the use of resources across the edge-cloud continuum, reducing costs and improving overall efficiency. Contrary, service offloading consists of exposing only the Kubernetes services on a remote cluster. In that case, the pod execution continues at the original cluster, and the pod deployment is performed from the beginning in the targeted cluster. The remaining components are also aware of the names of the services on the remote cluster.

3.8 Execution Graph-based Blueprint Orchestration

AI-driven provisioning and Lifecycle Manager set the execution blueprint to reflect the desired infrastructure state. The Orchestrator and Resource Manager choose the resources to initiate and in which order to achieve the target state. At the core of the Orchestrator and Resource Manager, a Kubernetes Operator watches the Orchestration Blueprint represented by a CRD (Custom Resource Definition) *spec* section and updates its *status* section to reflect the current infrastructure status. To achieve such flow, for every Orchestration Blueprint creation or update, an execution graph is generated describing the different steps and their dependencies to achieve the target state. In Figure 7, we illustrate an example of the initial version of such a Blueprint.

```

apiVersion: charity-project.eu/v1
kind: OrchestrationBlueprint
metadata:
  name: orchestrationblueprint1
spec:
  clusters:
  - provider: "provider1"
    name: "cluster1"
  - provider: "provider2"
    name: "cluster2"
  links:
  - "cluster1"
  - "cluster2"

```

Fig. 7 Initial Orchestration Blueprint Example.

The Blueprint describes an application of two clusters with two different infrastructure providers connected with Ligo (*links* subsection). The services are omitted for brevity. To achieve such orchestration, an execution graph is generated and illustrated in Figure 8. The dashed lines indicate the associated data for each node.

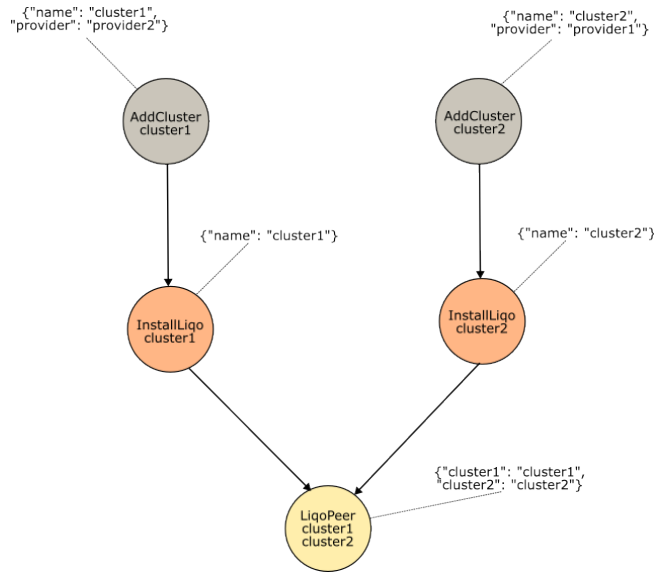


Fig. 8 Initial Execution Graph Example.

The resulting Execution Graph expresses the creation of the clusters with Ligo installation in each, which can be done in parallel, as shown. Once Ligo is installed in both clusters, peering can be set up between them. Now, let's suppose the orchestration blueprint is updated while the first cluster has been deployed. In Figure 9, we illustrate such a blueprint.

In this new orchestration blueprint, the first cluster has a new infrastructure provider (*provider2*). The *status* section contains the existing infrastructure state represented by the old first cluster properties. A new execution graph is then generated from this updated orchestration blueprint. A first approach is illustrated in Figure 10.

```

apiVersion: charity-project.eu/v1
kind: OrchestrationBlueprint
metadata:
  name: orchestrationblueprint1
spec:
  clusters:
    - provider: "provider2"
      name: "cluster1"
    - provider: "provider2"
      name: "cluster2"
  links:
    - "cluster1"
    - "cluster2"
status:
  clusters:
    - provider: "provider1"
      name: "cluster1"

```

Fig. 9 Updated Orchestration Blueprint Example.

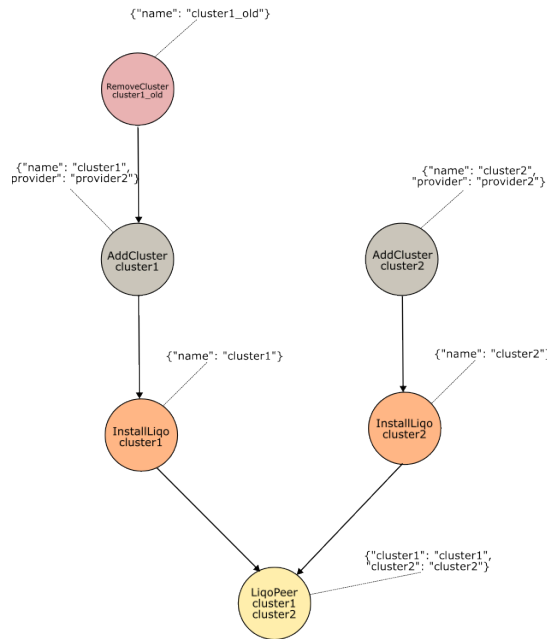


Fig. 10 First Approach of the updated Execution Graph Example.

In such an execution graph approach, the old version of the first class is removed before setting up the rest of the infrastructure. While such an approach is an acceptable one for achieving the goal state, the issue remains if ensuring the continuous working of the application is mandatory while transitioning to a new infrastructure state. Another approach is to consider the uninterrupted transition to the new infrastructure as illustrated in Figure 11.

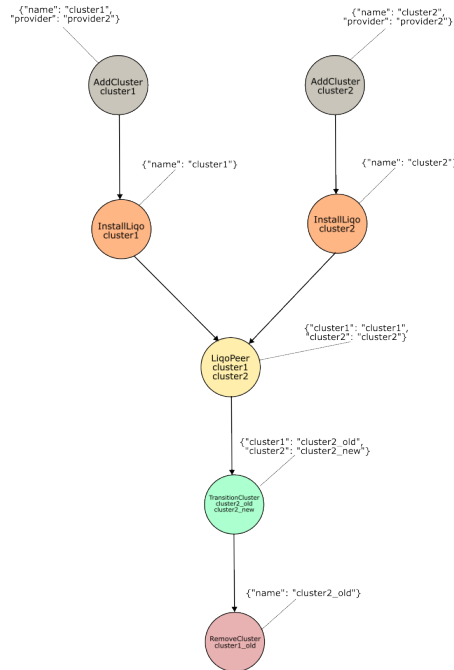


Fig. 11 Second Approach of the updated Execution Graph Example.

In this new execution graph, the new cluster with the new provider is constructed before the old cluster version is deleted. Moreover, a transition node is added to allow uninterruptible operation before final deletion. While the corresponding operation related to the transition step is beyond the paper content, the execution graph is extensible enough to cover all the possible scenarios. Once the execution graph is updated, Executors instances are created corresponding to nodes with no precedence and with an equivalent class. Hence, an AddClusterExecutor will be created for the corresponding AddCluster node. Once the executor finishes execution, the corresponding execution graph node is removed, the Orchestration Blueprint *status* section is updated, and new unconstrained nodes are selected to create executor instances. The process continues once all the execution graph nodes have been processed.

4 Immersive Service Use Case

4.1 Immersive Virtual Touring

One use case that can benefit from this architectural paradigm is Cyango Cloud Studio[69], a VR SaaS (Software as a service) that allows anyone to create Virtual Reality experiences. Cyango empowers businesses with a solution that allows them to explain, show, teach and sell directly in real-time with interactive 360^o video experiences. Cyango Cloud Studio targets content creators and marketing agencies requiring a seamless workflow for creating enhanced Virtual Reality experiences. Cyango Cloud

Studio delivers high-quality VR editing capabilities to content creators and high-quality 360 VR content to end-users. This content can be video, image, audio or 3D models.

Besides many features, Cyango allows four distinct use cases:

- **Real-time video streaming:** where the users can stream video and audio to many viewers using any recording device (e.g., a 360° camera).
- **Asset converting:** where users upload different kinds of assets and convert it to multiple quality levels that can be later adaptively loaded.
- **Video editing:** allows users to perform remote video editing without requiring a powerful ad-hoc machine.
- **Static video consuming:** where users can load and visualize the immersive experience with 360° videos using HLS protocol that adapts to different network speeds and devices.

The Cyango Cloud Studio provides a graphical interface for users to upload and edit their assets and build the virtual experience, as seen in Figure 12.

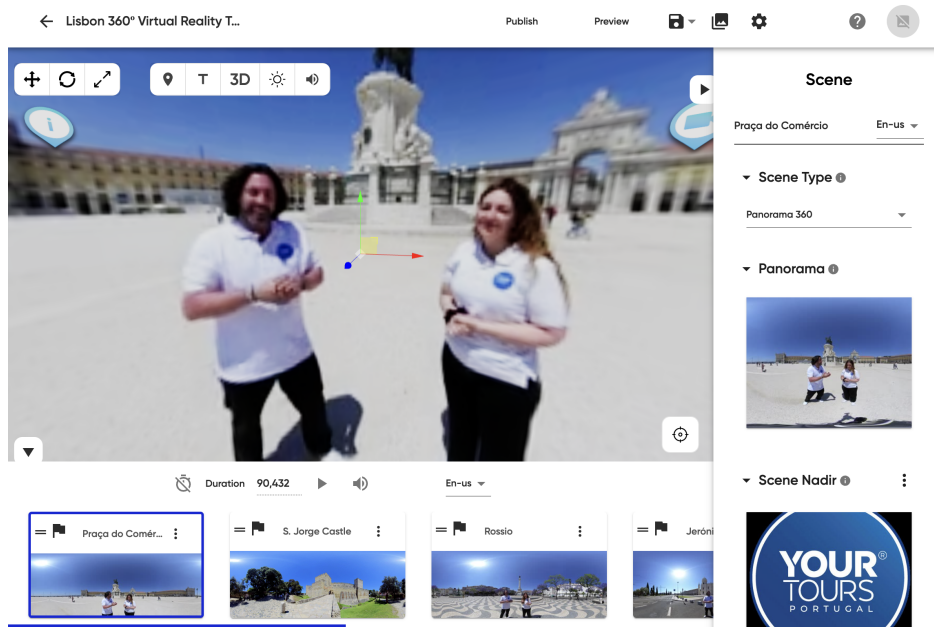


Fig. 12 Screenshot of Cyango Cloud Studio Web Interface.

The user can access such an interface in the browser where all the actions, like video converting and editing, are made. This makes it necessary to have a very low latency response in video editing. For instance, user edits on the browser must be reflected in (near) real-time in a way that feels like a fast response to the edit action. Moreover, Cyango Cloud Studio deployments consider the ability to scale and adapt the content delivery according to the number of concurrent users and achieve the best possible

QoS and QoE. This can be achieved with algorithms that create the optimal number of content quality levels, allowing streaming and adapting to situations where bandwidth and resources vary. From a developer perspective, there is also a need to quickly push new code to a versioned source code repository and seamless integration with CI/CD pipelines. Cyango Cloud Studio is based on WebXR and WebGL technologies and uses a micro-service architecture comprising several containerised components (cf., Figure 13).

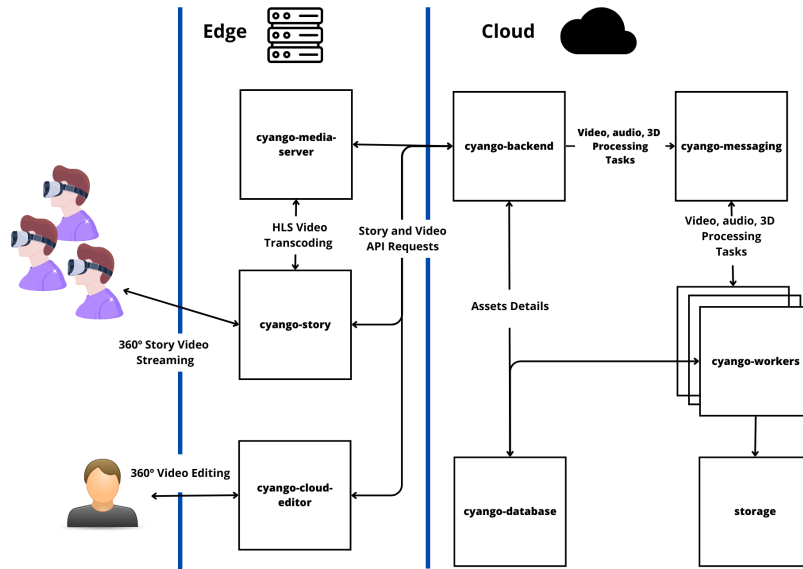


Fig. 13 Cyango Cloud Studio Components Architecture.

The users (i.e., content creators and VR experience consumers) interact with **cyango-story** and **cyango-cloud-editor** components, which are strategically placed in edge locations to minimize the latency of video editing and consumption. The **cyango-media-server** component also requires a strategic location for serving and performing real-time video/audio transcoding and live streaming. The service placement of them is implemented in a manner that maximizes the QoE of different users at different locations. Components **cyango-story** and **cyango-cloud-editor** use Three.js, a WebGL library abstraction for Javascript. This library allows video and image as textures in a 3D environment while allowing interactivity. It provides an immersive 3D experience that can be loaded on smartphones, desktops and VR headsets. The TOSCA-enabled custom orchestration platform described in 3.2.2 (supporting the TOSCA simple profile version 1.3 [70]) enables the selection of the most suitable locations of these components considering the specific hardware capabilities to improve the overall processing performance (e.g., GPU-enabled nodes). The **cyango-backend** component works as an API component that communicates and delegates processing tasks to **cyango-workers**. The **cyango-worker(s)** can be considered the most resource-expensive components as

they handle all the heavy tasks like converting the video and audio from any file extension to a standard HLS protocol playlist that can be consumed using an adaptive bitrate method. This component uses ffmpeg native libraries to convert audio, video, and other kinds of assets (e.g., images using the sharp library to manipulate the image and convert it to standard extensions that are readable as WebGL textures). Replicas of cyango-worker(s) must be replicated using a Horizontal Auto Scaling strategy. The cyango-messaging component acts as a messaging bus between cyango-backend and cyango-worker(s) for an asynchronous task-based processing schema. Finally, the cyango-database component stores all the stories, assets and details. The storage component stores the processed files, which users will later consume. These two storage components can also greatly benefit from an intelligent scheduling placement approach minimizing the network latencies when accessing and persisting the assets.

4.2 Harnessing Metrics for Intelligent Orchestration

This subsection is dedicated to exploring how various metrics that are closely associated with the performance of the immersive service use case can be harnessed. The collection of such metrics is crucial towards establishing intelligent orchestration. Figure 14 represents a deployment of a VR Tour Creator application considering a multi-cloud environment.

The scenario is composed of two clusters: one cluster for hosting the orchestrator components (the management cluster) and a second one used for the deployment of the VR Tour application components (the workload cluster). For the sake of simplicity, the VR Tour components were deployed in a single cluster. Such workload cluster is created and managed by the orchestrator. On top of the orchestrator components, the management cluster also includes the orchestration blueprints and the metrics-related components. The blueprints consist of application definitions and requirements. Meanwhile, the metrics collector endpoint represents the metrics aggregated from the various applications' exposed metrics endpoints. On the other hand, the workload cluster hosts the media server, the cloud editor and the remaining VR application components.

Moreover, the replicated VR Tour scenario intends to replicate how real users would consume content on-demand in a fast and seamless way. In this case, we recreated a scenario where content creators want to promote their services and products through an engaging live shopping experience. This scenario consists of a user that possesses a 360 camera which is capable of live streaming through an RTMP URL to a consumable VR experience for the potential buyers to access via any browser in any device, be it desktop, smartphone or XR headset (cf. Figure 14).

The first stage consists of a live-streaming scene that must be configured using the cloud-editor component, which generates an RTMP URL coupled with a unique stream key. After publishing the story to a public online URL, any end-user can start consuming it. In the second stage, the creator initiates the live streaming using a 360-degree camera specially designed for wired streaming. We used a high-resolution camera capable of streaming 4096x2048 pixels, 30 FPS and a 5 Mbps video bitrate. Such a stream was sent to the media server component residing in the workload cluster using the RTMP protocol. Later, this stream was consumed and exposed through the

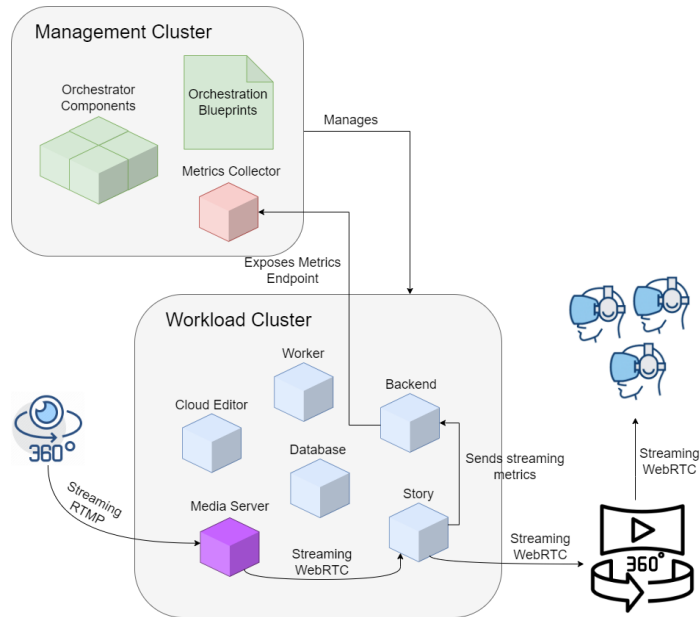


Fig. 14 Experimental Streaming Scenario

story component, enabling end-users to immerse themselves in the 360 VR experience. At the same time, upon establishing a new WebRTC connection within the story component, the collection of metrics was performed. The metrics encompass various facets of WebRTC communication and were systematically gathered and transmitted from the story component to a dedicated endpoint within the backend component. The story component uses an open-source library called `webrtc-stats` from `peermetrics` [71]. We measured the Available Incoming Bitrate from the video stream in bps, the Bytes Discarded On Send in bytes, the Bytes Received in bytes, the Bytes Sent in bytes, the Current Round Trip Time in milliseconds and the Total Round Trip Time in milliseconds. These metrics can define how good the user experience is when consuming the VR live stream and how the system performs. In particular, the round trip time, which can be influenced by:

- **Distance:** The span a signal must cover directly affects the time it takes for a request to travel from a user's browser to a server and back with a response.
- **Transmission Medium:** The type of pathway used to guide a signal, whether it's copper wiring or fibre optic cables, can influence the speed at which a request is sent to a server and relayed back to the user.
- **Number of Network Hops:** Whenever a signal encounters intermediate routers or servers, it encounters processing delays, which increase the Round-Trip Time (RTT). The greater the number of these signal "hops", the higher the RTT.
- **Traffic Levels:** RTT tends to rise during peak network congestion when there's a surge in traffic. Conversely, during periods of low network activity, RTT tends to decrease.

- **Server Response Time:** The time required for a designated server to respond to a request depends on factors such as its processing capacity, the volume of requests it's handling, and the complexity of the request. A longer server response time contributes to an increase in RTT.

Finally, the VR tour backend component processed the collected connection metrics and exposed them as a Prometheus endpoint. This facilitates subsequent analysis and visualization of the metrics, which are crucial for assessing the performance and quality of the WebRTC connection in the context of the 360-degree VR livestream. Figures 15 and 16 illustrate the obtained values observed in Prometheus UI.

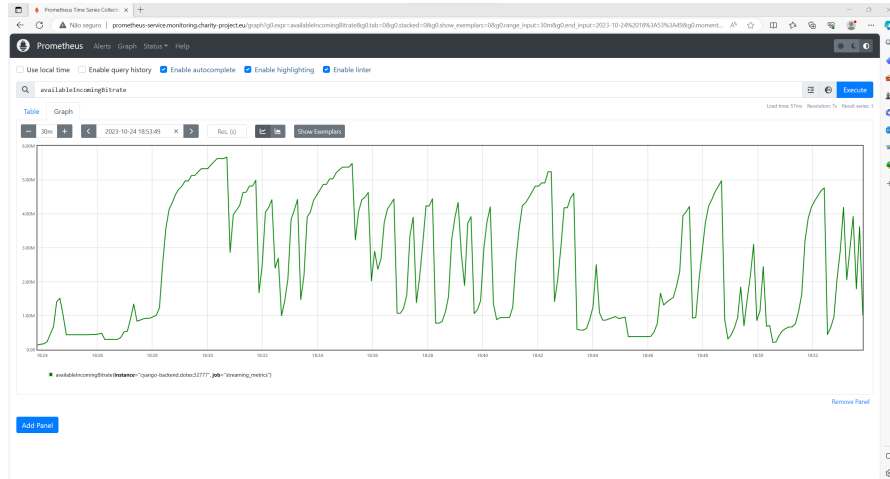


Fig. 15 Streaming Incoming Bitrate

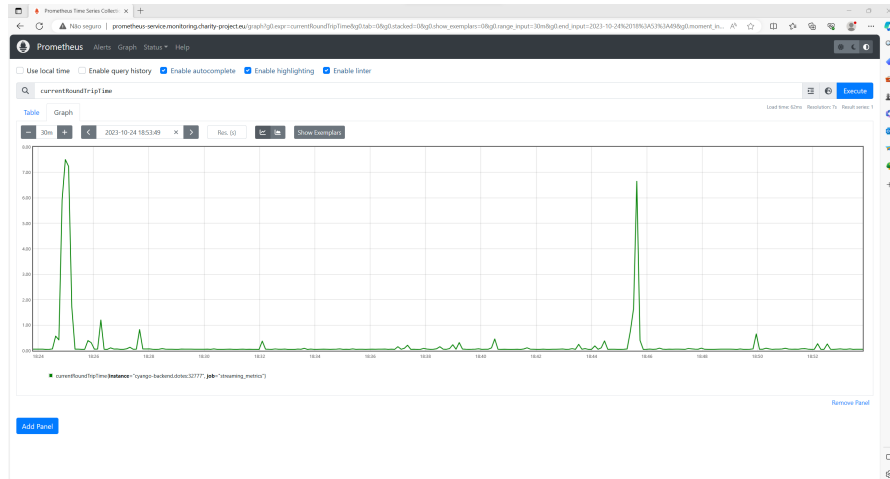
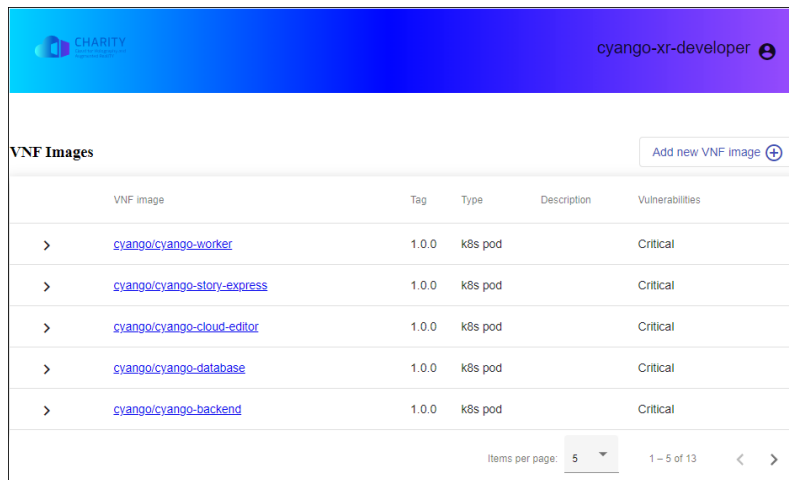


Fig. 16 Streaming Round Trip Time

4.3 Application Management Framework

Application Management Framework (AMF) offers immersive application developers an environment for defining and deploying highly interactive and collaborative next-generation services. AMF can be considered an entry point for immersive application developers, from which they define the modules of their applications and visually shape and compose them by specifying properties, parameters, relationships and requirements in terms of computational resources and expected QoS at runtime. This information is critical as it provides the algorithms used by the orchestration layer, to select the best topology for the deployment of the application components.

The modules are software artifacts packaged into containers or virtual machine images. The AMF provides a dedicated registry where immersive application developers upload their artifacts. A specific section in the user interface helps developers upload new images and presents them as a list of the available images (cf. 17).



The screenshot shows a web interface for managing VNF Images. At the top, there is a header with the CHARITY logo on the left and the user name 'cyango-xr-developer' on the right. Below the header, the section is titled 'VNF Images' and includes a button 'Add new VNF image'. The main content is a table with the following columns: 'VNF image', 'Tag', 'Type', 'Description', and 'Vulnerabilities'. The table lists five VNF images, each with a chevron icon to its left and a 'Critical' vulnerability status.

VNF image	Tag	Type	Description	Vulnerabilities
> cyango/cyango-worker	1.0.0	k8s pod		Critical
> cyango/cyango-story-express	1.0.0	k8s pod		Critical
> cyango/cyango-cloud-editor	1.0.0	k8s pod		Critical
> cyango/cyango-database	1.0.0	k8s pod		Critical
> cyango/cyango-backend	1.0.0	k8s pod		Critical

At the bottom of the table, there is a pagination control showing 'Items per page: 5' and '1 - 5 of 13'.

Fig. 17 AMF VNF Images management: list of VNF Images uploaded by Cyango developers for the Cyango Cloud Studio application

For every image in the list, by clicking on their name, users can get detailed information related to their build history (as shown in Figure 18) and, more importantly, to possible security concerns.

In fact, as the first step of application provisioning, AMF triggers a DevSecOps chain to check the uploaded images against security threats. A detailed report is generated (cf. 19), and pointers to descriptions about each security issue and possible resolutions are shown to the users (cf. 20).

Once the needed images are loaded into the AMF registry, the developers can define the application blueprints for their services visually in the AMF. The user interface guides the developers in the steps required to define the application components. Figure 21 shows how the AMF Blueprint Editor is used to model the Cyango application described in the previous section. Developers define the application name,

The screenshot shows the 'VnfImageDetails' page for the image 'cyango/cyango-cloud-editor'. The header includes the CHARITY logo and the user 'cyango-xr-developer'. The main content is divided into three sections: 'VnfImageDetails', 'Overview', and 'Additions'.

Name	Digest	Severity	Total	Fixable	Details
cyango/cyango-cloud-editor	sha256:e37ce27b4a31b016a25b5638ae09ae97d6893f1d52686c56d04814d7b4a4484	Severity: Critical	66	66	Critical: 4 High: 30 Medium: 30 Low: 2

Overview

Attribute name	Attribute value
architecture	amd64
author	NGINX Docker Maintainers <docker-maint@nginx.com>
config	["Cmd":["nginx","-g","daemon off;"],"Entrypoint":["docker-entrypoint.sh"],"Env":["PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"],"NGINX_VERSION=1.23.2","NJS_VERSION=0.7.7","PKG_RELEASE=1"],"Labels":{"maintainer":"NGINX Docker Maintainers <docker-maint@nginx.com>"},"ExposedPorts":{"80/tcp":[""],"StopSignal":"SIGQUIT"}]
created	2023-09-05T07:32:02.954640959Z
os	linux

Additions

Vulnerabilities | Build History

Created on	Command
8/9/22, 7:19 PM	/bin/sh -c #!nop) ADD file 2a949686d9886ac7c10582a6c29116d029d3077d0275e67e111870d53607725 in /
8/9/22, 7:19 PM	/bin/sh -c #!nop) CMD ["/bin/sh"]

Fig. 18 AMF VNF Images management: details about the build history for an image uploaded into the AMF registry

Additions

Vulnerabilities | Build History

Vulnerability	Severity	CVSS3	Package	Current version	Fixed in version
CVE-2023-1999 <small>Description: There exists a use after free/double free in libwebp. An attacker can use the ApplyFiltersAndEncode() function and loop through to free best.bw and assign best = trial pointer. The second loop will then return 0 because of an Out of memory error in VP8 encoder, the pointer is still assigned to trial and the AddressSanitizer will attempt a double free.</small>	High	nvd: 7.5 redhat: 7.5	libwebp	1.2.3-r0	1.2.3-r1
CVE-2023-4853	High	nvd: 8.8 redhat: 9.6	libwebp	1.2.3-r0	1.2.3-r2
CVE-2023-29491	High	nvd: 7.8 redhat: 7.8	ncurses-libs	6.3_p20220521-r0	6.3_p20220521-r1
CVE-2023-29491	High	nvd: 7.8 redhat: 7.8	ncurses-terminfo-base	6.3_p20220521-r0	6.3_p20220521-r1
CVE-2023-35945	High	nvd: 7.5 redhat: 7.5	nghttp2-libs	1.47.0-r0	1.47.0-r1
CVE-2022-41409	High	nvd: 7.5	pcrc2	10.40-r0	10.42-r0
CVE-2022-32221	Critical	nvd: 9.8 redhat: 4.8	curl	7.83.1-r3	7.83.1-r4
CVE-2023-23914	Critical	nvd: 9.1 redhat: 6.5	curl	7.83.1-r3	7.83.1-r6
CVE-2022-42915	High	nvd: 8.1 redhat: 7.5	curl	7.83.1-r3	7.83.1-r4
CVE-2022-42916	High	nvd: 7.5 redhat: 7.5	curl	7.83.1-r3	7.83.1-r4
CVE-2022-43551	High	nvd: 7.5 redhat: 7.5	curl	7.83.1-r3	7.83.1-r6
CVE-2023-27533	High	nvd: 8.8 redhat: 3.1	curl	7.83.1-r3	8.0.1-r0

Fig. 19 AMF VNF Images management: vulnerabilities report generated by the DevSecOps pipeline

description, version number and privacy level. Also, in this phase, it is possible to define global input parameters required at deployment time when launching an application from a blueprint.



Fig. 20 AMF VNF Images management: vulnerability details

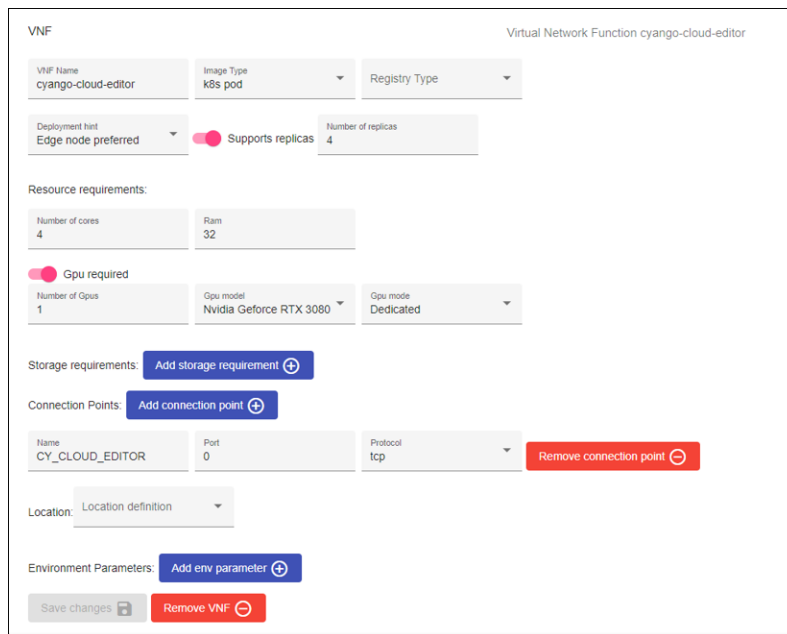


Fig. 21 AMF Blueprint Editor: sample of the definition of the Cyango Cloud Studio application module (VNF).

The second step is the definition of external devices or systems that are not directly managed by the orchestration platform but are required by the application. Therefore, they must be represented in the model to understand how to communicate with

them. An example of this category is an end-user mobile device which connects to the deployed application or a cloud service (e.g., Amazon S3) used by the application to gather some data. The next step is the definition of the set of modules (called Virtual Network Functions in the AMF) constituting the application, together with details on resource requirements in terms of cores, RAM, GPU, storage, the support for replicas, the input and environment parameters, and the connection points to communicate with other components. Connection points specify the port numbers and protocols for outgoing or incoming communication flow. The orchestrator later uses them to create the corresponding services and required connections between clusters. This is done for every module composing the application.

The final step is the definition of the virtual links (Figure 22) and the communication channels allowing modules to interact via their connection points. A dedicated form allows the user to select the available connection points defined on the VNFs in the previous steps and to select from a checkbox the ones that must be connected, hence defining the communication path between the modules. For every virtual link defined, users can set requirements for QoS related to bandwidth and latency. This can be done by filling values in a form or interacting with a slider. These two metrics are closely associated with service performance. Reliability and scalability, on the other hand, constitute desired emergent properties of highly distributed systems that support the ability of these systems to facilitate the various requirements in terms of latency and bandwidth. Consequently, users need to define only their requirements in terms of latency and bandwidth, while reliability and scalability are established via the use of the aforementioned AI solutions.

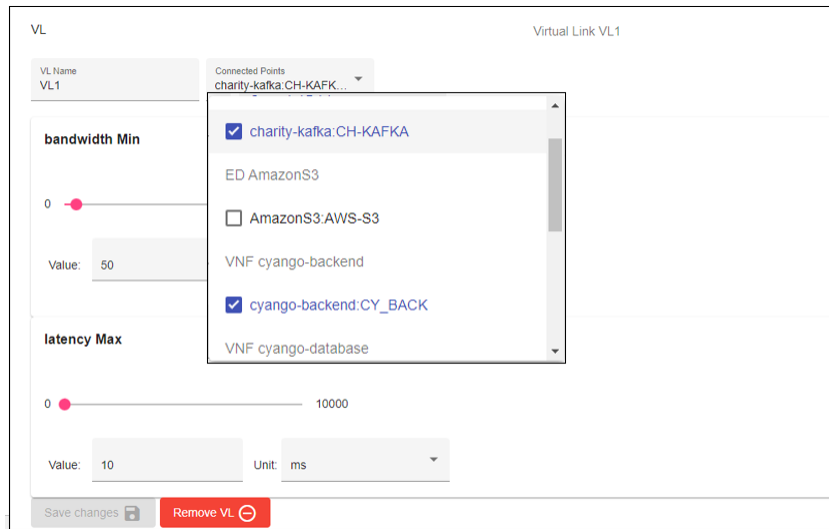


Fig. 22 AMF Blueprint Editor: sample of the definition of communication links and QoS properties.

Every time the users define a new item, a graphical representation is updated on the top side of the GUI so that the developers can have visual feedback on what they

are modelling. The final representation of the Cyango Cloud Studio model is shown in Figure 23.

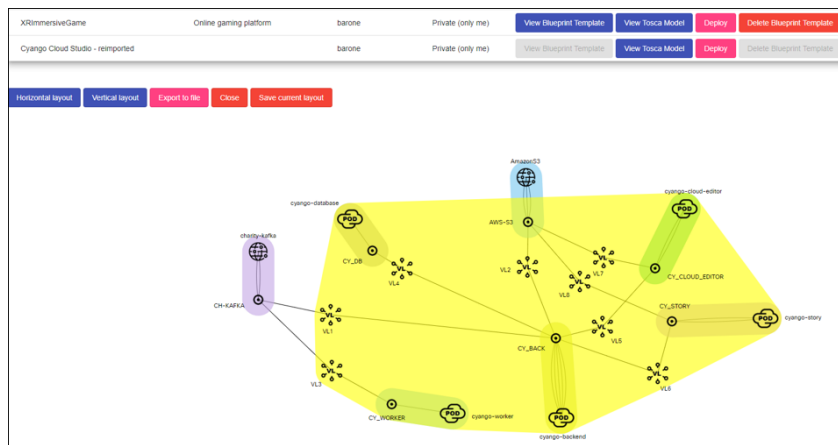


Fig. 23 AMF Blueprint Editor: model of Cyango Cloud Studio application.

Upon completion of the blueprint, the AMF generates a TOSCA representation that will be used at deployment time (cf. 24). To trigger a deployment request, developers will first click a "Manage application" button to access a form for specifying the input parameters modelled in the blueprint (cf. 25). After filling in the input values, they ask for the deployment by clicking a "Deploy new application" button. The AMF will then pass the input parameters and the TOSCA application model to the lower orchestration layers.

4.4 Architectural Evaluation

Existing standards and solutions like ETSI MANO and ZSM provide outstanding network-centric reference architectures to structure multi-domain Edge Orchestration. Nevertheless, such specifications remain highly complex to implement and lack high-level components to provide pervasive, scalable orchestration as a service to end users. In the same way, Platforms as a Service provide a scalable and simpler programmable interface to an underlying complex infrastructure compared to Infrastructure as a Service.

Our architecture proposes the concept of user intent blueprints as an application-centric programmable interface towards a full-featured multi-domain intelligent orchestration as a Service. The blueprint provides a declarative design at a very high level of the multi-domain infrastructure. Further care has been put into providing the visual specification tool AMF on top of the blueprint. The TOSCA user intent blueprint and AMF have been validated with use-case owners for completeness and usability. The AI alleviates a lot of the complexity of multi-domain orchestration through a smart fine-tuning of the application infrastructure details. Existing reviewed solutions don't define a detailed set of infrastructure decisions and the corresponding required

```

description: Cloud Studio Service
metadata:
  # The following fields are "normative" and expected in TOSCA
  template_name: Cyango Cloud Studio - reimported
  template_author: cyango-xr-developer
  template_version: ''
imports:
  - charity_custom_types_v08.yaml
topology_template:
  inputs:
  node_templates:
    charity-kafka:
      type: Charity.Component
      properties:
        name: charity-kafka
        deployment_unit: EXTERNAL
    AmazonS3:
      type: Charity.Component
      properties:
        name: amazons3
        deployment_unit: EXTERNAL
    cyango-backend:
      type: Charity.Component
      properties:
        name: cyango-backend
        deployment_unit: K8S_POD
        placement_hint: CLOUD
        image: repository.charity-project.eu/dotes/cyango-backend:beta
        environment:
          NODE_ENV: { get_input: NODE_ENV }
      requirements:
        - host: cyango-backendNode
    cyango-backendNode:
      type: Charity.Node
      node_filter:
        capabilities:
          - host:
              properties:
                num_cpus:
                  - equal: 0
                mem_size:
                  - greater_than: 0 MB

```

Fig. 24 AMF Blueprint Editor: excerpt of the TOSCA model generated for the Cyango use case.

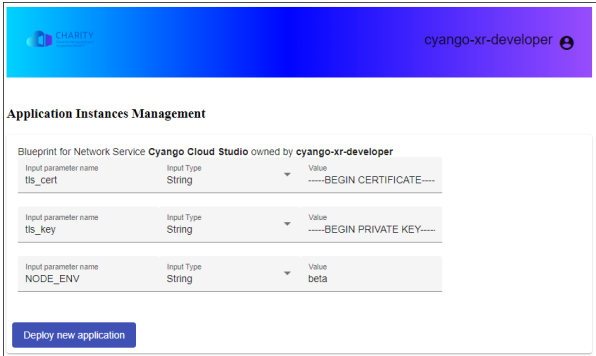


Fig. 25 AMF Application management: page with input parameters form for and button to trigger the application deployment towards the orchestration layer.

monitoring data. Our orchestration blueprint provides another level of declarative infrastructure design but with the more fine-grained tuning of the infrastructure details by an AI component. It is important to note that an AI can be plugged in without

any prior API adaptation compared to existing solutions, guaranteeing that the specification models are compatible (Graph-based, for example). Furthermore, we have identified monitoring data required for the intelligent conversion between the user intent and orchestration blueprints. Existing multi-domain orchestration architectures and tools specify a set of API endpoints and layered communication channels to set up and update the application infrastructure. Nevertheless, by not being based on an actual application infrastructure state specification, the execution plan is manually defined and ensured by the end user or a third-party automation tool. Any observed failure requires another level of management not covered in the reference architecture. Our Orchestration and Resource Manager provides a decoupling between the intelligent infrastructure orchestration decision and the infrastructure execution plan required during the application’s initial setup and its update during its lifetime. The operator nature of this component ensures that the proper application infrastructure state is ensured by following an execution plan that respects the agreed-upon Orchestration Blueprint. Moreover, any notable failure that deviates the infrastructure from the target state triggers a remediation plan by the component.

5 Conclusion

This publication presented a new design approach that can enable efficient management of immersive services across multiple domains at the edge, using a range of AI solutions and technology to support multi-domain edge deployments. Our new architecture proposes a new paradigm based around a set of multi-level specification blueprints which decouple the high-level user-intent infrastructure definition from the AI-driven orchestration and the final execution plan. Our Orchestration and Resource Manager component adheres to the operator pattern, facilitating the separation of the application infrastructure state from the associated execution plan, both for the initial setup and lifecycle management. The innovative ClusterAPI and Liqo have been harnessed as the main pillars for the execution plan operations. Moreover, we delved into the utilization of various AI techniques in component provisioning and lifecycle management, the architectural considerations of centralized versus decentralised orchestration approaches, and the decomposition of the execution plan into a detailed execution graph that describes the sequential steps and their interdependencies for attaining the desired infrastructure state. The AMF provides a visual language and tool alternative to the formal approach for the intent blueprint. We provided details about how developers can leverage the various aspects of AMF and validate them with the Immersive virtual touring use case owner. Finally, we recreated a distributed VR Tour application scenario and offered valuable insights into how the proposed architecture effectively addresses its orchestration requirements, spanning from the provisioning phase to the monitoring of application-specific metrics.

Acknowledgment

This research work has been supported by the CHARITY project that received funding from the EU’s Horizon 2020 program under Grant agreement No 101016509. This

paper reflects only the authors' view and the Commission is not responsible for any use that may be made of the information it contains.

References

- [1] Makris, A., Boudi, A., Coppola, M., Cordeiro, L., Corsini, M., Dazzi, P., Andilla, F.D., González Rozas, Y., Kamarianakis, M., Pateraki, M., Pham, T.L., Protopsaltis, A., Raman, A., Romussi, A., Rosa, L., Spatafora, E., Taleb, T., Theodoropoulos, T., Tserpes, K., Zschau, E., Herzog, U.: Cloud for holography and augmented reality. In: 2021 IEEE 10th International Conference on Cloud Networking (CloudNet), pp. 118–126 (2021). <https://doi.org/10.1109/CloudNet53349.2021.9657125>
- [2] Taleb, T., Nadir, Z., Flinck, H., Song, J.: Extremely interactive and low-latency services in 5g and beyond mobile systems. IEEE Communications Standards Magazine **5**(2), 114–119 (2021) <https://doi.org/10.1109/MCOMSTD.001.2000053>
- [3] Nadir, Z., Taleb, T., Flinck, H., Bouachir, O., Bagaa, M.: Immersive services over 5g and beyond mobile systems. IEEE Network **35**(6), 299–306 (2021) <https://doi.org/10.1109/MNET.121.2100172>
- [4] Yu, H., Taleb, T., Samdanis, K., Song, J.: Towards supporting holographic services over deterministic 6g integrated terrestrial & non-terrestrial networks. IEEE Network, 1–10 (2023) <https://doi.org/10.1109/MNET.133.2200509>
- [5] Boos, K., Chu, D., Cuervo, E.: Demo: Flashback: Immersive virtual reality on mobile devices via rendering memoization. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion. MobiSys '16 Companion, p. 94. Association for Computing Machinery, New York, NY, USA (2016). <https://doi.org/10.1145/2938559.2938583> . <https://doi.org/10.1145/2938559.2938583>
- [6] El Marai, O., Taleb, T., Song, J.: Ar-based remote command and control service: Self-driving vehicles use case. IEEE Network **37**(3), 170–177 (2023) <https://doi.org/10.1109/MNET.119.2200058>
- [7] Taleb, T., Sehad, N., Nadir, Z., Song, J.: Vr-based immersive service management in b5g mobile systems: A uav command and control use case. IEEE Internet of Things Journal **10**(6), 5349–5363 (2023) <https://doi.org/10.1109/JIOT.2022.3222282>
- [8] Theodoropoulos, T., Makris, A., Boudi, A., Taleb, T., Herzog, U., Rosa, L., Cordeiro, L., Tserpes, K., Spatafora, E., Romussi, A., *et al.*: Cloud-based xr services: A survey on relevant challenges and enabling technologies. Journal of Networking and Network Applications **2**(1), 1–22 (2022) <https://doi.org/10.33969/J-NaNA.2022.020101>

- [9] Taleb, T., Boudi, A., Rosa, L., Cordeiro, L., Theodoropoulos, T., Tserpes, K., Dazzi, P., Protopsaltis, A.I., Li, R.: Toward supporting xr services: Architecture and enablers. *IEEE Internet of Things Journal* **10**(4), 3567–3586 (2023) <https://doi.org/10.1109/JIOT.2022.3222103>
- [10] Theodoropoulos, T., Makris, A., Psomakelis, E., Carlini, E., Mordacchini, M., Dazzi, P., Tserpes, K.: Gnosis: Proactive image placement using graph neural networks & deep reinforcement learning. In: 2023 IEEE 16th International Conference on Cloud Computing (CLOUD), pp. 120–128 (2023). <https://doi.org/10.1109/CLOUD60044.2023.00022>
- [11] Benmerar, T.Z., Theodoropoulos, T., Fevereiro, D., Rosa, L., Rodrigues, J., Taleb, T., Barone, P., Tserpes, K., Cordeiro, L.: Intelligent multi-domain edge orchestration for highly distributed immersive services: An immersive virtual touring use case. In: 2023 IEEE International Conference on Edge Computing and Communications (EDGE), pp. 381–392 (2023). <https://doi.org/10.1109/EDGE60047.2023.00061>
- [12] Faticanti, F., Savi, M., De Pellegrini, F., Siracusa, D.: Locality-aware deployment of application microservices for multi-domain fog computing. *Computer Communications* **203**, 180–191 (2023) <https://doi.org/10.1016/j.comcom.2023.02.012>
- [13] 3GPP. TS 23.558: Architecture for enabling Edge Applications (2023)
- [14] Alonso, J., Orue-Echevarria, L., Casola, V., Torre, A.I., Huarte, M., Osaba, E., Lobo, J.L.: Understanding the challenges and novel architectural models of multi-cloud native applications—a systematic literature review. *Journal of Cloud Computing* **12**(1), 1–34 (2023) <https://doi.org/10.1186/s13677-022-00367-6>
- [15] Raj, P., Raman, A.: Automated Multi-cloud Operations and Container Orchestration, pp. 185–218. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78637-7_9 . https://doi.org/10.1007/978-3-319-78637-7_9
- [16] Tomarchio, O., Calcaterra, D., Di Modica, G.: Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks. *Journal of Cloud Computing* **9**, 49 (2020) <https://doi.org/10.1186/s13677-020-00194-7>
- [17] Bellendorf, J., Mann, Z.Á.: Specification of cloud topologies and orchestration using toasca: a survey. *Computing* **102**(8), 1793–1815 (2020) <https://doi.org/10.1007/s00607-019-00750-3>
- [18] Kim, D., Muhammad, H., Kim, E., Helal, S., Lee, C.: Tosca-based and federation-aware cloud orchestration for kubernetes container platform. *Applied Sciences* **9**(1) (2019) <https://doi.org/10.3390/app9010191>
- [19] Osmani, L., Kauppinen, T., Komu, M., Tarkoma, S.: Multi-cloud connectivity

- for kubernetes in 5g networks. *IEEE Communications Magazine* **59**(10), 42–47 (2021) <https://doi.org/10.1109/MCOM.110.2100124>
- [20] Tamiru, M.A., Pierre, G., Tordsson, J., Elmroth, E.: mck8s: An orchestration platform for geo-distributed multi-cluster environments. In: 2021 International Conference on Computer Communications and Networks (ICCCN), pp. 1–10 (2021). <https://doi.org/10.1109/ICCCN52240.2021.9522318>
- [21] ETSI GS ZSM 011: Zero-touch network and Service Management (ZSM); Intent-driven autonomous networks; Generic aspects (2023)
- [22] Liyanage, M., Pham, Q.-V., Dev, K., Bhattacharya, S., Maddikunta, P.K.R., Gadekallu, T.R., Yenduri, G.: A survey on zero touch network and service management (zsm) for 5g and beyond networks. *Journal of Network and Computer Applications* **203**, 103362 (2022) <https://doi.org/10.1016/j.jnca.2022.103362>
- [23] Coronado, E., Behraves, R., Subramanya, T., Fernández-Fernández, A., Siddiqui, M.S., Costa-Pérez, X., Riggio, R.: Zero touch management: A survey of network automation solutions for 5g and 6g networks. *IEEE Communications Surveys & Tutorials* **24**(4), 2535–2578 (2022) <https://doi.org/10.1109/COMST.2022.3212586>
- [24] Huang, S.-Y., Chen, C.-Y., Chen, J.-Y., Chao, H.-C.: A survey on resource management for cloud native mobile computing: Opportunities and challenges. *Symmetry* **15**(2) (2023) <https://doi.org/10.3390/sym15020538>
- [25] Nejabati, R., Moazzeni, S., Jaisudthi, P., Simenidou, D.: Zero-touch network orchestration at the edge. In: 2021 International Conference on Computer Communications and Networks (ICCCN), pp. 1–5 (2021). <https://doi.org/10.1109/ICCCN52240.2021.9522194>
- [26] Gallego-Madrid, J., Sanchez-Iborra, R., Ruiz, P.M., Skarmeta, A.F.: Machine learning-based zero-touch network and service management: a survey. *Digital Communications and Networks* **8**(2), 105–123 (2022) <https://doi.org/10.1016/j.dcan.2021.09.001>
- [27] Benzaid, C., Taleb, T.: Ai-driven zero touch network and service management in 5g and beyond: Challenges and research directions. *IEEE Network* **34**(2), 186–194 (2020) <https://doi.org/10.1109/MNET.001.1900252>
- [28] ETSI GS ZSM 012: Zero-touch network and Service Management (ZSM); Enablers for Artificial Intelligence-based Network and Service Automation (2022)
- [29] ETSI ZSM 008: Zero-touch network and Service Management (ZSM); Cross-domain E2E service lifecycle management (2022)
- [30] Korontanis, I., Tserpes, K., Pateraki, M., Blasi, L., Violos, J., Diego, F., Marin, E.,

- Kourtellis, N., Coppola, M., Carlini, E., *et al.*: Inter-operability and orchestration in heterogeneous cloud/edge resources: The accordion vision. In: Proceedings of the 1st Workshop on Flexible Resource and Application Management on the Edge, pp. 9–14 (2020). <https://doi.org/10.1145/3452369.3463816>
- [31] 3GPP. TR 28.312: Management and orchestration; Intent driven management services for mobile networks (2023)
- [32] 3GPP. TR 28.912: Study on enhanced intent driven management services for mobile networks (2023)
- [33] 3GPP. TR 28.812: Telecommunication management; Study on scenarios for Intent driven management services for mobile networks (2020)
- [34] Gutierrez-Estevez, D.M., Gramaglia, M., Domenico, A.D., Dandachi, G., Khatibi, S., Tsolkas, D., Balan, I., Garcia-Saavedra, A., Elzur, U., Wang, Y.: Artificial intelligence for elastic management and orchestration of 5g networks. *IEEE Wireless Communications* **26**(5), 134–141 (2019) <https://doi.org/10.1109/MWC.2019.1800498>
- [35] Linux Foundation: ONAP - Open Network Automation Platform. <https://www.onap.org/> (2023). (Accessed: 02 May 2023)
- [36] Linux Foundation: Akraino. <https://www.lfedge.org/projects/akraino/> (2023). (Accessed: 02 May 2023)
- [37] Cluster API: Kubernetes Cluster API. <https://cluster-api.sigs.k8s.io/> (2023). (accessed: 02 May 2023)
- [38] ETSI: OSM - Open Source MANO. <https://osm.etsi.org/> (2023). (accessed: 02 May 2023)
- [39] Cloudify: Bridging the Gap Between Applications and Cloud Environments. <https://cloudify.co/> (2023). (Accessed: 02 May 2023)
- [40] Redhat: Redhat - Openshift. <https://www.redhat.com/en/technologies/cloud-computing/openshift> (2023). (Accessed: 02 May 2023)
- [41] Tamburri, D.A., Heuvel, W.-J., Lauwers, C., Lipton, P., Palma, D., Rutkowski, M.: Tosca-based intent modelling: goal-modelling for infrastructure-as-code. *SICS Software-Intensive Cyber-Physical Systems* **34**(2), 163–172 (2019) <https://doi.org/10.1007/s00450-019-00404-x>
- [42] Theodoropoulos, T., Makris, A., Kontopoulos, I., Maroudis, A.-C., Tserpes, K.: Multi-service demand forecasting using graph neural networks. In: 2023 IEEE International Conference on Service-Oriented System Engineering (SOSE), pp. 218–226 (2023). <https://doi.org/10.1109/SOSE58276.2023.00033>

- [43] Yilmaz, O.: Extending the Kubernetes API, pp. 99–141. Apress, Berkeley, CA (2021). https://doi.org/10.1007/978-1-4842-7095-0_4 . https://doi.org/10.1007/978-1-4842-7095-0_4
- [44] Lim, B., Zohren, S.: Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A* **379**(2194), 1–14 (2021) <https://doi.org/10.1098/rsta.2020.0209>
- [45] Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* **34**(6), 26–38 (2017) <https://doi.org/10.1109/MSP.2017.2743240>
- [46] Theodoropoulos, T., Maroudis, A.-C., Violos, J., Tserpes, K.: An encoder-decoder deep learning approach for multistep service traffic prediction. In: 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService), pp. 33–40 (2021). <https://doi.org/10.1109/BigDataService52369.2021.00010>
- [47] Theodoropoulos, T., Makris, A., Kontopoulos, I., Violos, J., Tarkowski, P., Ledwoń, Z., Dazzi, P., Tserpes, K.: Graph neural networks for representing multivariate resource usage: A multiplayer mobile gaming case-study. *International Journal of Information Management Data Insights* **3**(1), 100158 (2023) <https://doi.org/10.1016/j.jjimei.2023.100158>
- [48] Fang, C., Zhang, T., Huang, J., Xu, H., Hu, Z., Yang, Y., Wang, Z., Zhou, Z., Luo, X.: A drl-driven intelligent optimization strategy for resource allocation in cloud-edge-end cooperation environments. *Symmetry* **14**(10) (2022) <https://doi.org/10.3390/sym14102120>
- [49] Zhang, Y., Li, Y., Wang, R., Lu, J., Ma, X., Qiu, M.: Psac: Proactive sequence-aware content caching via deep learning at the network edge. *IEEE Transactions on Network Science and Engineering* **7**(4), 2145–2154 (2020) <https://doi.org/10.1109/TNSE.2020.2990963>
- [50] Behraves, R., Rao, A., Perez-Ramirez, D.F., Harutyunyan, D., Riggio, R., Boman, M.: Machine learning at the mobile edge: The case of dynamic adaptive streaming over http (dash). *IEEE Transactions on Network and Service Management* **19**(4), 4779–4793 (2022) <https://doi.org/10.1109/TNSM.2022.3193856>
- [51] Narayanan, A., Verma, S., Ramadan, E., Babaie, P., Zhang, Z.-L.: Deepcache: A deep learning based framework for content caching. In: Proceedings of the 2018 Workshop on Network Meets AI & ML. NetAI’18, pp. 48–53. Association for Computing Machinery, New York, NY, USA (2018). <https://doi.org/10.1145/3229543.3229555> . <https://doi.org/10.1145/3229543.3229555>
- [52] Theodoropoulos, T., Kafetzis, D., Violos, J., Makris, A., Tserpes, K.: Multi-agent deep reinforcement learning for weighted multi-path routing. In: Proceedings of

- the 3rd Workshop on Flexible Resource and Application Management on the Edge. FRAME '23, pp. 7–11. Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3589010.3594888> . <https://doi.org/10.1145/3589010.3594888>
- [53] Theodoropoulos, T., Makris, A., Violos, J., Tserpes, K.: An automated pipeline for advanced fault tolerance in edge computing infrastructures. In: Proceedings of the 2nd Workshop on Flexible Resource and Application Management on the Edge. FRAME '22, pp. 19–24. Association for Computing Machinery, New York, NY, USA (2022). <https://doi.org/10.1145/3526059.3533623> . <https://doi.org/10.1145/3526059.3533623>
- [54] Ma, W.: Analysis of anomaly detection method for internet of things based on deep learning. *Transactions on Emerging Telecommunications Technologies* **31**(12), 3893 (2020) <https://doi.org/10.1002/ett.3893>
- [55] Theodoropoulos, T., Violos, J., Tsanakas, S., Leivadeas, A., Tserpes, K., Varvarigou, T.: Intelligent proactive fault tolerance at the edge through resource usage prediction. *ITU Journal on Future and Evolving Technologies* **3**(3), 761–778 (2022) <https://doi.org/10.52953/ehjp3291>
- [56] Chen, W., Chen, Y., Wu, J., Tang, Z.: A multi-user service migration scheme based on deep reinforcement learning and sdn in mobile edge computing. *Physical Communication* **47**, 101397 (2021) <https://doi.org/10.1016/j.phycom.2021.101397>
- [57] Al-Asaly, M.S., Bencherif, M.A., Alsanad, A., Hassan, M.M.: A deep learning-based resource usage prediction model for resource provisioning in an autonomic cloud computing environment. *Neural Computing and Applications*, 1–18 (2021) <https://doi.org/10.1007/s00521-021-06665-5>
- [58] Xiao, Z., Hu, S.: Dscaler: A horizontal autoscaler of microservice based on deep reinforcement learning. In: 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1–6 (2022). <https://doi.org/10.23919/APNOMS56106.2022.9919994>
- [59] Violos, J., Tsanakas, S., Theodoropoulos, T., Leivadeas, A., Tserpes, K., Varvarigou, T.: Intelligent horizontal autoscaling in edge computing using a double tower neural network. *Computer Networks* **217**, 109339 (2022) <https://doi.org/10.1016/j.comnet.2022.109339>
- [60] Liu, Q., Xia, T., Cheng, L., Eijk, M., Ozcelebi, T., Mao, Y.: Deep reinforcement learning for load-balancing aware network control in iot edge systems. *IEEE Transactions on Parallel and Distributed Systems* **33**(6), 1491–1502 (2022) <https://doi.org/10.1109/TPDS.2021.3116863>
- [61] Theodoropoulos, T., Makris, A., Korontanis, I., Tserpes, K.: Greenkube: Towards

- greener container orchestration using artificial intelligence. In: 2023 IEEE International Conference on Service-Oriented System Engineering (SOSE), pp. 135–139 (2023). <https://doi.org/10.1109/SOSE58276.2023.00023>
- [62] Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., Gao, Y.: A survey on federated learning. *Knowledge-Based Systems* **216**, 106775 (2021) <https://doi.org/10.1016/j.knosys.2021.106775>
- [63] Li, L., Fan, Y., Tse, M., Lin, K.-Y.: A review of applications in federated learning. *Computers & Industrial Engineering* **149**, 106854 (2020) <https://doi.org/10.1016/j.cie.2020.106854>
- [64] Chen, J., Ran, X.: Deep learning with edge computing: A review. *Proceedings of the IEEE* **107**(8), 1655–1674 (2019) <https://doi.org/10.1109/JPROC.2019.2921977>
- [65] Wang, Y., Guo, L., Zhao, Y., Yang, J., Adebisi, B., Gacanin, H., Gui, G.: Distributed learning for automatic modulation classification in edge devices. *IEEE Wireless Communications Letters* **9**(12), 2177–2181 (2020) <https://doi.org/10.1109/LWC.2020.3016822>
- [66] Cloud Native Computing Foundation: Prometheus. <https://prometheus.io> (2023). (Accessed: 02 May 2023)
- [67] Korontanis, I., Makris, A., Theodoropoulos, T., Tserpes, K.: Real-time monitoring and analysis of edge and cloud resources. In: *Proceedings of the 3rd Workshop on Flexible Resource and Application Management on the Edge. FRAME '23*, pp. 13–18. Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3589010.3594892> . <https://doi.org/10.1145/3589010.3594892>
- [68] Iorio, M., Risso, F., Palesandro, A., Camiciotti, L., Manzalini, A.: Computing without borders: The way towards liquid computing. *IEEE Transactions on Cloud Computing* **11**(3), 2820–2838 (2023) <https://doi.org/10.1109/TCC.2022.3229163>
- [69] Cyango: Cyango - Virtual Reality, AR & Digital Transformation Studio. <https://www.cyango.com/> (2023). (Accessed: 8 December 2023)
- [70] OASIS: Tosca simple profile version 1.3. <https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/os/TOSCA-Simple-Profile-YAML-v1.3-os.pdf> (2020)
- [71] Peermetrics: Peermetrics. <https://github.com/peermetrics/webrtc-stats> (2023). (Accessed: 16 October 2023)

-
[



Tarik Zakaria Benmerar received his Engineer degree in computer engineering from USTHB-Algiers, in 2010 and master's degree in networks and distributed systems from USTHB-Algiers, in 2011. He received his PhD degree in parallelism and cloud computing (SaaS) applied for cerebral connectivity using diffusion MRI, at USTHB-Algiers in 2019. He is currently a senior lecturer at USTHB, and a research engineer at ICTIFICIAL, Oy. His current research interests include cloud computing/edge/IoT architecture and orchestration, WebRTC-based streaming architectures and web browser-based parallelism.



Tarik Taleb (Senior Member, IEEE) received the B.E. degree (with distinction) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a Full Professor at Ruhr University Bochum, Germany. He was a Professor with the Center of Wireless Communications, University of Oulu, Oulu, Finland. He is the founder of ICTIFICIAL Oy, and the founder and the Director of the MOSA!C Lab, Espoo, Finland. From October 2014 to December 2021, he was an Associate Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. Prior to that, he was working as a Senior Researcher and a 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. Before joining NEC and till March 2009, he worked as an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, in a lab fully funded by KDDI. From 2005 to 2006, he was a Research Fellow with the Intelligent Cosmos Research Institute, Sendai. Taleb has been directly engaged in the development and standardization of the Evolved Packet System as a member of the 3GPP System Architecture Working Group. His current research interests include AI-based network management, architectural enhancements to mobile core networks, network softwarization and slicing, mobile cloud networking, network function virtualization, software-defined networking, software-defined security, and mobile multimedia streaming.



Theodoros Theodoropoulos received the Eng. Diploma degree from the School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece. He is currently pursuing the Ph.D. degree with the Department of Informatics and Telematics, Harokopio University of Athens, Kallithea, Greece. He has been working as a Research Engineer with the Harokopio University of Athens for the last three years. During this time, he had the chance to work at several Research and Development projects and to author numerous scientific publications. His main research interests include deep learning, graph neural networks, deep reinforcement learning, and cloud & edge computing.



Konstantinos Tserpes is currently transitioning from a faculty position at Harokopio University of Athens to a role at the National Technical University of Athens. He holds a PhD in the area of Distributed Systems from the school of Electrical and Computer Engineering of the National Technical University of Athens (2008). His research interests revolve around efficient computing, optimizing computing systems to support novel application classes. He is working at the intersection of research areas such as cloud/edge computing, distributed systems and ML/AI. He has been involved in several EU and National funded projects leading research for solving issues related to scalability, interoperability, fault tolerance, and extensibility in application domains such as multimedia, e-governance, post-production, finance, e-health and others. He has co-authored more than 150 articles in scientific conference proceedings and journals.



Giovanni Giuliani, Master Architect at Hewlett Packard Enterprise Italy, graduated in Electronics Engineering with honors in 1982. He has 40 years industry experience in R&D, IT consulting and System Integration in major Computer Companies, as well as university teaching. Master Architect at Hewlett Packard Enterprise, delivering several complex projects in Manufacturing, Government, Finance and Mobility and leading various Cloud Computing technologies related initiatives.

Contributor to several EC funded research projects and Technical Coordinator in a sequence of 3 Green-IT research projects.



Paolo Barone, Senior Solution Architect at Hewlett Packard Enterprise Italy, received the M. Sc. in Computer Science with full marks in 1999 from the State University of Milan. He has 24 years of working experience in Italian and international IT Consulting firms. In Hewlett Packard Enterprise since 2006, he is working in the Advisory & Professional Services unit, covering the role of Senior Solution Architect. He participated to Hewlett Packard Enterprise initiatives related to both commercial and research programs in the areas of eHealth, DevOps, Cloud Computing, mobile technologies, advanced Internet applications and technologies, energy saving.



Diogo Fevereiro is a junior researcher specializing in cloud computing, network, and service management. He completed his Bachelor of Science in Computer Science from Universidade de Coimbra, where he demonstrated a strong aptitude in topics regarding cloud resources management and orchestration. Building on this foundation, he pursued his Master's degree in 2023 at Universidade de Coimbra, delving deeper into such subjects.

Throughout his academic journey, Diogo has shown a keen interest in cutting-edge technologies and their practical applications. His research efforts have resulted in several publications in renowned academic journals and conference proceedings, showcasing his analytical prowess and innovative thinking. These publications have contributed valuable insights to the field of cloud focused on multi-domain cloud-edge resources orchestration solutions.