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

Tarik Zakaria Benmerar*, Theodoros Theodoropoulos†, Diogo Fevereiro‡, Luis Rosa‡, João Rodrigues§, Tarik Taleb¶, Paolo Barone||, Konstantinos Tserpes†, and Luis Cordeiro‡

*ICTFICIAL Oy, Finland; †Harokopio University of Athens, Greece; ‡OneSource, Portugal; §Cyango, Portugal; ¶University of Oulu, Finland; ||Hewlett Packard Enterprise, Italy

Emails: tarik.benmerar@ictficial.com, ttheod@hua.gr, duarte.fevereiro@onesource.pt, luis.rosa@onesource.pt, joao.rodrigues@cyango.com, tarik.taleb@oulu.fi, paolo.barone@hpe.com, tserpes@hua.gr, cordeiro@onesource.pt

***Abstract*—Edge cloud technologies in tandem with AI-enabled solutions can contribute to overcoming the challenges that pertain the distributed execution of immersive services and contribute towards providing a positive experience for the end-users. Intelligent resource management, orchestration, and prediction systems can optimize the deployment of services, adapt to changing demands, and ensure that the services are running smoothly. This paper introduces a novel architectural paradigm capable of facilitating multi-domain edge orchestration for highly distributed immersive services by incorporating a plethora of AI solutions and technological enablers that can support multi-domain edge deployments. The proposed architecture is designed to operate on the basis of multi-level specification blueprints, which decouple the simple high-level user-intent infrastructure definition from the AI-driven orchestration and the final execution plan. The Application Management Framework (AMF) offers a visual language and tool that can be used as an alternative to a formal method for creating the intent blueprint. In the frame of this work, the latter is validated by an immersive virtual touring use-case scenario.**

***Index Terms*—Edge cloud, immersive service, orchestration, cluster, Kubernetes, centralized management, and decentralized management**

I. Introduction

Highly distributed immersive services have the potential to revolutionize different industries by providing new and innovative ways of experiencing and interacting with the world. For instance, Virtual Reality (VR) and Augmented Reality (AR) can provide students with interactive and engaging learning experiences. Likewise, VR simulations can help train professionals in various industries, such as healthcare and aviation. In entertainment, immersive services have transformed the gaming industry by allowing players to fully immerse themselves in virtual worlds, while AR can enhance live events by overlaying interactive digital elements on real-world environments.

Unfortunately, to maintain a high level of Quality of Service (QoS), immersive services depend on extremely low latency and high bandwidth services [1]–[4]. Scientific literature suggests that for an end-user to have a satisfactory experience, the end-to-end latency should not exceed 15ms, and the available bandwidth should scale up to 30 Gbps [5]–[7]. In addition, faults in task processing can potentially disrupt service delivery and compromise the integrity of immersive experiences. Therefore, it is crucial for applications of this nature to possess fault tolerance capabilities. Furthermore, immersive applications are computationally intensive, requiring complex 3D models and high-quality graphics. Incorporating essential computational resources into end-user equipment would result in bulky and expensive setups, which goes against the principles of immersive applications that require end-user devices to be portable and cost-effective [8]. Cloud computing can shift the computational burden to remote resources, allowing end-user devices to remain mobile and affordable. However, cloud topologies cannot fully cope with the ultra-low latency and high bandwidth requirements of immersive services, since arbitrary complex networks intervene between end-user devices and cloud servers.

Edge computing aims to reduce the amount of data transmitted to remote clouds and allows data processing near the data sources. As a result, edge topologies provide faster response times, higher transfer rates, and better scalability and availability. Thus, running immersive services in a distributed manner across the cloud-edge fabric would benefit application developers and help maintain high-level QoS provisioning [9].

On the downside, the optimal allocation of distributed tasks across the network and compute resources of the cloud-edge continuum remains an open challenge, especially under the latency, bandwidth and fault tolerance constraints of the distributed immersive applications. Machine Learning (ML) / Deep Learning (DL) can help address these challenges by providing intelligent resource management and orchestration systems that can adapt to the changing demands of the end users. For example, Artifical Intelligence (AI)-powered load balancing and resource allocation algorithms can optimize the deployment of services across the cloud-edge fabric, taking into account factors such as network conditions, computational resources, and users' demands. Moreover, AI-powered predictive analytics can be used to anticipate future demand and proactively allocate resources accordingly, avoiding potential bottlenecks or service disruptions. AI-powered monitoring and fault detection systems can also help ensure that the services are running smoothly and identify any issues before they affect the end-users' experiences. On top of the aforementioned AI solutions, Application Management Frameworks (AMFs) are an essential tool for deploying applications on edge computing. AMFs provide a structured approach to designing, documenting, and managing an application and deploying it on an edge computing system. The importance of AMFs lies in their ability to provide a common language and framework for stakeholders to collaborate on the design, implementation, and management of applications. By using a standardized approach, AMFs can help ensure consistency, interoperability, and scalability across the system, while also enabling stakeholders to make informed decisions about trade-offs between cost, performance, and other factors. Finally, it is essential to incorporate the required technological enablers that are capable of facilitating the orchestration & management of highly distributed immersive services. This type of deployments typically spans across multiple domains. To that end, it is of paramount importance for these technological enablers to focus on multi-domain edge orchestration. In this vein, this paper showcases a novel architectural paradigm that facilitates intelligent multi-domain edge orchestration for immersive services.

The remainder of this article is organized in the following manner: Section II explores some research work relevant to intelligent multi-domain edge orchestration. Section III introduces the architecture of the envisioned intelligent multi-domain edge orchestration system. Section IV describes an Application Management Framework (AMF) which provides a visual language and tool alternative to the formal approach for the intent blueprint, as well as an immersive virtual touring use-case scenario that is used to validate the proposed architectural paradigm. Finally, Section V summarizes the merits of this work.

II. Related work

Edge computing is geared towards addressing the growing demands and requirements of the next generation of highly distributed applications [10]. 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, with the aim of reducing latency, improving data privacy, enabling real-time decision-making, achieving high scalability and resilience, and allowing better resource utilization. In [11], 3GPP SA6 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).

ZSM (Zero-touch network and Service Management) is an end-to-end management reference architecture developed by ETSI to provide a flexible and automated [12] 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 [13]. Indeed, following the advancements in AI methodologies, there have been numerous attempts [14], [15] at incorporating them to build further ZSM developments in different aspects, such as multi-tenancy management, traffic monitoring, and architecture coordination. In [16]. ETSI details a list of relevant AI-enabling areas for AI-driven network management, such as Trustworthy Machine Learning, Decentralized 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 [17]. 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 [18]. 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 [19], [20]. Intents focus on what needs to be achieved regardless of the actual implementation or the underlying infrastructure details [21]. 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 [22]. 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.

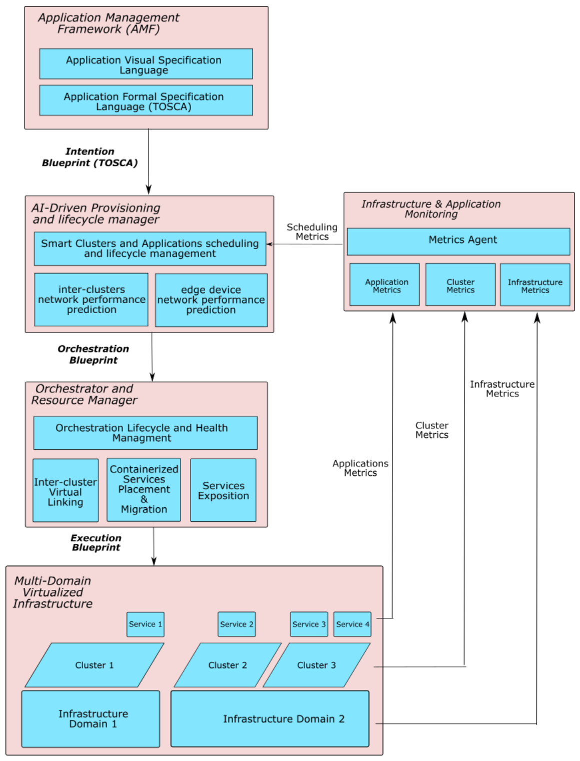
There is also now a plethora of emerging tools and enablers focused on supporting the management of containers, Virtual Machines (VMs) and services across the edge and cloud environments. Open Network Automation Platform (ONAP) [23] is designed to automate the composition and creation of network services. Akraino Edge Stack [24] 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 [25] 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) [26] is an ETSI project that aims to provide a platform for deploying, managing, and monitoring virtual network functions (VNFs) and network services. Cloudify [27] 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 [28] is a container application platform built on top of Kubernetes that provides developers with an integrated environment for building, deploying, and scaling containerized applications.

III. 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.

A. 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



Application Management Framework (AMF)
Application Visual Specification Language
Application Formal Specification Language (TOSCA)
Intention Blueprint (TOSCA)
AI-Driven Provisioning and lifecycle manager
Smart Clusters and Applications scheduling and lifecycle management
inter-clusters network performance prediction
edge device network performance prediction
Scheduling Metrics
Infrastructure & Application Monitoring
Metrics Agent
Application Metrics
Cluster Metrics
Infrastructure Metrics
Orchestration Blueprint
Orchestrator and Resource Manager
Orchestration Lifecycle and Health Managment
Inter-cluster Virtual Linking
Containerized Services Placement & Migration
Services Exposition
Infrastructure Metrics
Cluster Metrics
Applications Metrics
Execution Blueprint
Multi-Domain Virtualized Infrastructure
Service 1
Service 2
Service 3
Service 4
Cluster 1
Cluster 2
Cluster 3
Infrastructure Domain 1
Infrastructure Domain 2

Fig. 1. Intelligent Multi-Domain Edge Orchestration architecture.

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 III-E; 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.

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 III-B). 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 allows 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.

B. 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.

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) [29], a blueprint specification is defined for the application deployment model. The user intent blueprint includes a high-level view of the application represented as a composition of modular services (packaged into containers, Virtual Machines, etc.), 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, latency and jitter). 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, 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. 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.

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) [30] 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 plane 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. 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 should be considered in the long term to make the orchestration of different use cases viable.

*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. We should 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 deployments resources to the specified cluster. From the *links* section in the orchestration blueprint, the operator set up the *VPN* links through inter-cluster peering. *Liqo*, detailed later, is used as the third-party tool for this operation.

C. Service Deployment Planning

As explained earlier, the operator sets up the required clusters unto which the application services are planned 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 elaborated security system which must be in place to guarantee complete isolation between services 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.

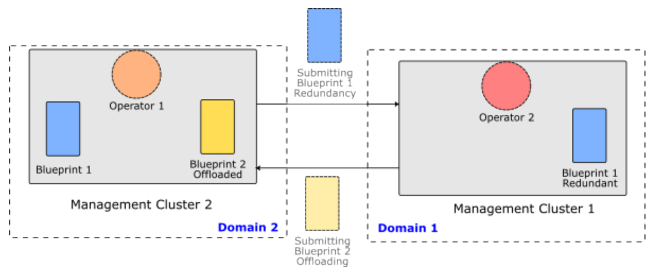


Fig. 2. Offloading and redundancy in decentralized multi-domain management clusters.

D. Cluster Management Approaches: Decentralized vs Centralized

It is important to note that an operator needs to run in a Kubernetes cluster like any other application. The operator is strategically placed in a dedicated management cluster for the envisioned multi-domain orchestration solution. Likewise, the blueprints are deployed inside that cluster as CRDs, allowing the operator to keep track of changes in the application blueprints and synchronise their states with the actual infrastructures states. Two approaches are possible for deploying the management cluster, the operator and the application's blueprints. These approaches can be centralised or decentralised. In the centralised approach, a single management cluster hosts the operator and all the application blueprints. A significant increase in submitting new or updated application blueprints in this centralised approach can lead to scalability and resilience issues. In this case, the management cluster itself can be seen as a Single Point of Failure (SPOF), and any strategy for having redundancy should occur at the cluster level (e.g., Kubernetes High-Availability multi-node cluster setup). Such a centralised approach allows simplified management of the deployed applications and the orchestration components. In the decentralised option, on the other hand, multiple management clusters are deployed, with each having its own operator and sharing the application blueprints (See Figure 2. In such a decentralised approach, both application blueprints offloading and redundancy between clusters are possible with the declarative nature of the blueprints.

E. 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 architectural paradigm, many AI solutions may 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 is inspired by ENI's modus operandi, was briefly explored in the Related Works section, and is based on the role of the 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 [31] 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 Reinforcement Learning [32] methodologies that perform 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. As such, they play an integral role in the decision-making process and can contribute towards tackling the aforementioned challenges in the following ways:

- **Latency**: In immersive applications, even small delays can affect the user experience significantly. Edge computing can help reduce latency by processing data closer to the source. However, the distribution of the application across multiple edge nodes can introduce additional latency. AI can help mitigate this by predicting the behaviour of the users [33] and the application to anticipate the processing requirements and allocate resources accordingly [34].
- **Bandwidth**: Immersive applications require high bandwidth for streaming multimedia content. However, the limited bandwidth in edge networks can cause delays or interruptions in streaming. AI can optimize the use of available bandwidth by predicting the content users are likely to access. That content is then preloaded in the edge devices to reduce the amount of data that needs to be transferred [35].
- **Reliability**: Highly distributed immersive applications in edge computing can be vulnerable to network and node failures. AI can help ensure reliability by monitoring the performance and behaviour of the application in real-time and detecting any anomalies or failures [36]. AI can also help to predict when a node is likely to fail [37] and migrate the application to a different node to ensure continuity of service [38].
- **Scalability**: Immersive applications can be resource-intensive, and as the number of users increases, the demand for resources also increases. AI can help manage the demand for resources by predicting the number of users, their behaviour, and the subsequent resource demand [39] to allocate resources more efficiently. AI can also help optimize the allocation of resources across multiple edge nodes [40] to ensure that the application can scale up or down as needed.

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 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 distributed edge computing.

Federated learning [41] is a ML approach that enables training models across multiple decentralized / 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 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 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.

F. Infrastructure and Application Monitoring

For the correct operation of the AI-driven provisioning and lifecycle management component, it must have access to both 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** [42], 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.

*1) **Infrastructure Metrics Agents:*** All the applications 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 infrastructures 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.

*2) **Cluster Metrics Agents:*** Every cluster in the given application should be able to handle the required deployed services workload while achieving the target performances. A set of cluster metrics agents are deployed unto 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) **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 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.

G. 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 Liqo or Submariner, promise to address such automated peering cluster connectivity and **service discovery** across Kubernetes clusters [43] [44]. Liqo, both free and open-source, is the solution adopted in our proposed architecture. Liqo is designed to enable seamless connectivity among geographically distributed clusters (e.g., on-premises, edge or cloud). Liqo 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, Liqo 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, Liqo *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 3 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.

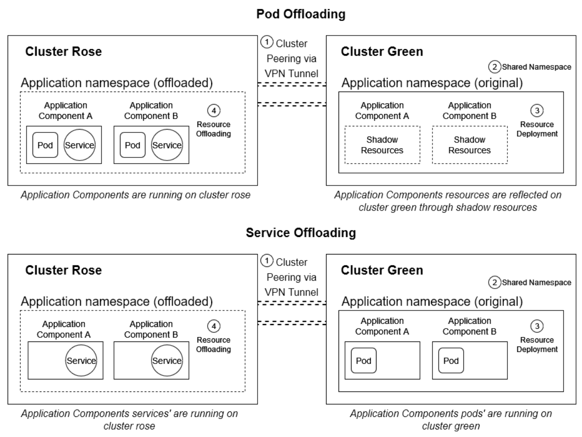


Fig. 3. Pod and Service Offloading comparison.

Nevertheless, the pod offloading strategy includes moving the actual execution of pods and the services to a peered cluster(e.g., in Figure 3, the application components are first deployed on the original Green Cluster and later executed in the Rose cluster). For instance, high-demanding 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 remains in the original cluster, and the pod deployment should be performed from the beginning in the targeted cluster. The remaining components should also be aware of the names of the services on the remote cluster.

IV. Immersive Service Use Case

A. Immersive Virtual Touring

One use case that can benefit from this architectural paradigm is Cyango Cloud Studio, a VR SaaS (Software as a service) that allows anyone to create Virtual Reality experiences. Cyango empowers businesses with a solution to allow them to explain, show, teach and sell directly in real-time with interactive 360º video experiences. Cyango Cloud Studio targets content creators and marketing agencies requiring a seamless workflow for creating enhanced Virtual Reality experiences. Cyango Cloud Studio focus on delivering 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 360º 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 360º 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 4.

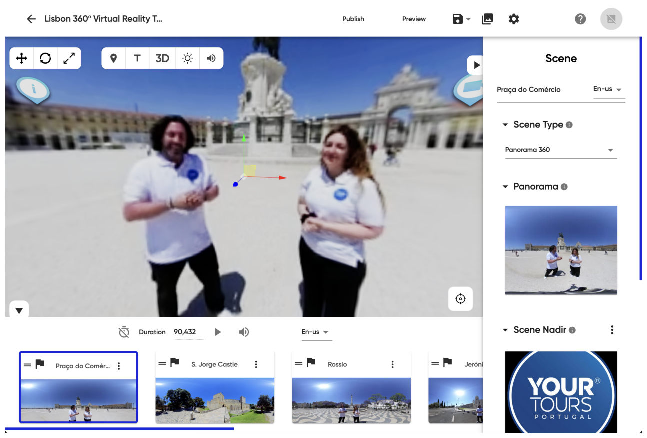


Fig. 4. Screenshot of Cyango Cloud Studio Web Interface.

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 it feels like a fast response to the edit action. Moreover, Cyango Cloud Studio deployments should consider the ability to scale and adapt the content delivery accordingly to the number of concurrent users and guarantee the best QoS and QoE. From a developer perspective, there is also a need for quickly pushing new code to a versioned source code repository and seamless integration with CI/CD pipelines. Cyango Cloud Studio is based on WebXR, WebGL technologies and uses a micro-service architecture comprising several containerised components (cf., Figure 5).

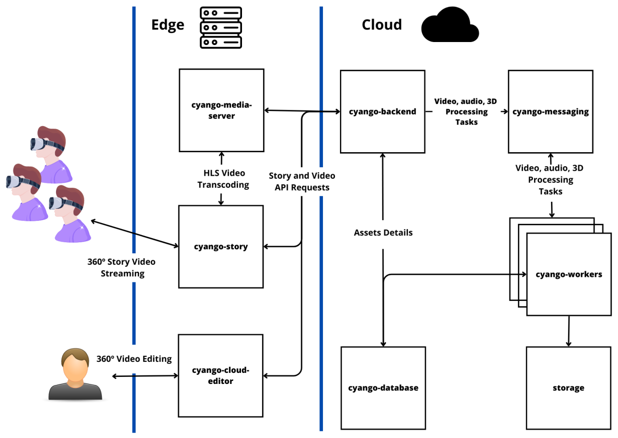


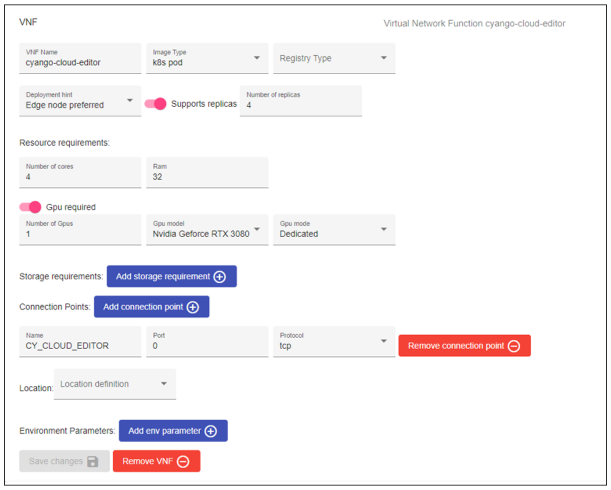
Fig. 5. Cyango Cloud Studio Components Architecture.

The users (i.e., content creators and VR experience consumers) interact with cyango-story and cyango-cloud-editor components which should be strategically placed in edge locations to minimize the latency of video editing and consuming. The cyango-media-server component also requires a strategic location for serving and performing real-time video/audio transcoding and livestreaming. The service placement of the three should maximize 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. An orchestration platform should support choosing 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.

B. 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 and relationships. The modules are software artefacts packaged into container or virtual machine images. The AMF provides a dedicated registry where immersive application developers upload their artefacts. Then, 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, and pointers to descriptions about each security issue and possible resolutions are shown to the users. 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 6 shows how the AMF Blueprint Editor is used to model the Cyango application described in the previous section. Developers start by defining the application name, 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.

The second step is the definition of external devices or systems that are indeed 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.


VNF
Virtual Network Function cyango-cloud-editor

VNF Name: cyango-cloud-editor	Image Type: k8s pod	Registry Type
Deployment hint: Edge node preferred	Supports replicas	Number of replicas: 4

Resource requirements:

Number of cores: 4	Ram: 32

Gpu required

Number of Gpus: 1	Gpu model: Nvidia Geforce RTX 3080	Gpu mode: Dedicated

Storage requirements:
Connection Points:

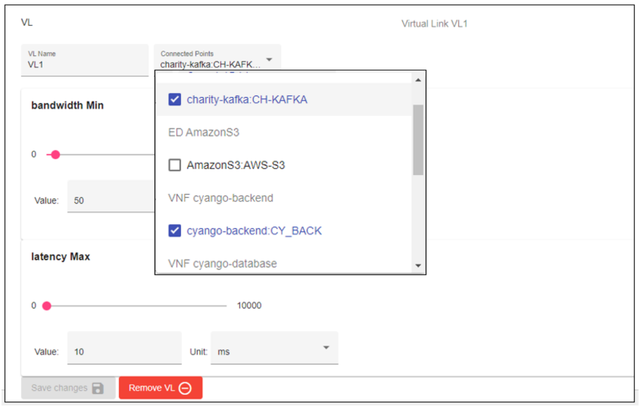
Name: CY_CLOUD_EDITOR	Port: 0	Protocol: tcp

Location: Location definition
Environment Parameters:

Fig. 6. AMF Blueprint Editor: sample of the definition of the Cyango Cloud Studio application module (VNF).

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 7), 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, latency and jitter. This can be done by filling values in a form or interacting with a slider.


VL
Virtual Link VL1

VL Name: VL1
Connected Points: charity-kafka:CH-KAFK...

- ☑ charity-kafka:CH-KAFKA
- ED AmazonS3
- ☐ AmazonS3:AWS-S3
- VNF cyango-backend
- ☑ cyango-backend:CY_BACK
- VNF cyango-database

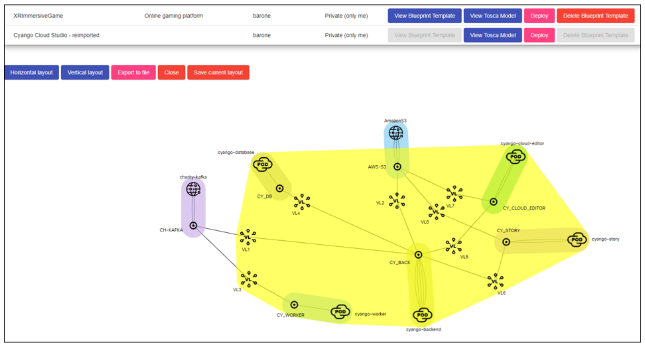
bandwidth Min
0
Value: 50

latency Max
0 10000
Value: 10 Unit: ms

Fig. 7. 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 8.

Upon completion of the blueprint, the AMF generates a



XRImmersiveGame	Online gaming platform	barone	Private (only me)
Cyango Cloud Studio - reimported		barone	Private (only me)

Fig. 8. AMF Blueprint Editor: model of Cyango Cloud Studio application.

TOSCA representation that can be used at deployment time, together with input parameters, for a deployment request to the lower orchestration layers.

```
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. 9. AMF Blueprint Editor: excerpt of the TOSCA model generated for the Cyango use case.

C. 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 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 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, guaranteed 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.

V. Conclusion

This article 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 enables to support multi-domain edge deployments. Our new architecture proposes a new paradigm based around a set of multi-level specification blueprints which decouples the simple high-level user-intent infrastructure definition from the AI-driven orchestration and the final execution plan. The innovative ClusterAPI and Liqo have been harnessed as the main pillars for the execution plan operations. The AMF provides a visual language and tool alternative to the formal approach for the intent blueprint. This later has been validated by the Immersive virtual touring use case owner. Our Orchestration and Resource Manager component follows the operator pattern, which allows the decoupling of the application infrastructure state from the corresponding execution plan for the initial setup and remediation plan during a failure.

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] A. Makris, A. Boudi, M. Coppola, L. Cordeiro, M. Corsini, P. Dazzi, F. D. Andilla, Y. G. Rozas, M. Kamarianakis, M. Pateraki, *et al.*, "Cloud for holography and augmented reality," in *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*, pp. 118–126, IEEE, 2021.

[2] T. Taleb, Z. Nadir, H. Flinck, and J. Song, "Extremely interactive and low-latency services in 5g and beyond mobile systems," *IEEE Communications Standards Magazine*, vol. 5, no. 2, pp. 114–119, 2021.

[3] Z. Nadir, T. Taleb, H. Flinck, O. Bouachir, and M. Bagaa, "Immersive services over 5g and beyond mobile systems," *IEEE Network*, vol. 35, no. 6, pp. 299–306, 2021.

[4] H. Yu, T. Taleb, K. Samdanis, and J. Song, "Towards supporting holographic services over deterministic 6g integrated terrestrial & non-terrestrial networks," *IEEE Network*, 2023.

[5] K. Boos, D. Chu, and E. Cuervo, "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, (New York, NY, USA), p. 94, Association for Computing Machinery, 2016.

[6] O. El Marai, T. Taleb, and J. Song, "Ar-based remote command and control service: Self-driving vehicles use case," *IEEE Network*, 2022.

[7] T. Taleb, N. Sehad, Z. Nadir, and J. Song, "Vr-based immersive service management in b5g mobile systems: A uav command and control use case," *IEEE Internet of Things Journal*, 2022.

[8] T. Theodoropoulos, A. Makris, A. Boudi, T. Taleb, U. Herzog, L. Rosa, L. Cordeiro, K. Tserpes, E. Spatafora, A. Romussi, *et al.*, "Cloud-based xr services: A survey on relevant challenges and enabling technologies," *Journal of Networking and Network Applications*, vol. 2, no. 1, pp. 1–22, 2022.

[9] T. Taleb, A. Boudi, L. Rosa, L. Cordeiro, T. Theodoropoulos, K. Tserpes, P. Dazzi, A. I. Protopsaltis, and R. Li, "Toward supporting xr services: Architecture and enablers," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3567–3586, 2022.

[10] F. Faticanti, M. Savi, F. De Pellegrini, and D. Siracusa, "Locality-aware deployment of application microservices for multi-domain fog computing," *Computer Communications*, vol. 203, pp. 180–191, 2023.

[11] TS 23.558, "Architecture for enabling edge applications," Mar. 2023.

[12] ETSI GS ZSM 011, "Zero-touch network and service management (zsm); intent-driven autonomous networks; generic aspects," Feb. 2023.

[13] M. Liyanage, Q.-V. Pham, K. Dev, S. Bhattacharya, P. K. R. Maddikunta, T. R. Gadekallu, and G. Yenduri, "A survey on zero touch network and service (zsm) management for 5g and beyond networks," *Journal of Network and Computer Applications*, p. 103362, 2022.

[14] J. Gallego-Madrid, R. Sanchez-Iborra, P. M. Ruiz, and A. F. Skarmeta, "Machine learning-based zero-touch network and service management: A survey," *Digital Communications and Networks*, vol. 8, no. 2, pp. 105–123, 2022.

[15] C. Benzaid and T. Taleb, "Ai-driven zero touch network and service management in 5g and beyond: Challenges and research directions," *IEEE Network*, vol. 34, no. 2, pp. 186–194, 2020.

[16] ETSI GS ZSM 012, "Zero-touch network and service management (zsm); enablers for artificial intelligence-based network and service automation," Dec. 2022.

[17] ETSI GS ZSM 008, "Zero-touch network and service management (zsm); cross-domain e2e service lifecycle management," July 2022.

[18] I. Korontanis, K. Tserpes, M. Pateraki, L. Blasi, J. Violos, F. Diego, E. Marin, N. Kourtellis, M. Coppola, E. Carlini, *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.

[19] TR 28.312, "Management and orchestration; intent driven management services for mobile networks," Apr. 2023.

[20] TR 28.912, "Study on enhanced intent driven management services for mobile networks," Mar. 2023.

[21] TR 28.812, "Telecommunication management; study on scenarios for intent driven management services for mobile networks," Mar. 2020.

[22] D. M. Gutierrez-Estevez, M. Gramaglia, A. D. Domenico, G. Dandachi, S. Khatibi, D. Tsolkas, I. Balan, A. Garcia-Saavedra, U. Elzur, and Y. Wang, "Artificial intelligence for elastic management and orchestration of 5g networks," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 134–141, 2019.

[23] Linux Foundation, "Onap - open network automation platform." https://www.onap.org/. (accessed: 02.05.2023).

[24] Linux Foundation, "Akraino." https://www.lfedge.org/projects/akraino/. (accessed: 02.05.2023).

[25] Cluster API, "Kubernetes cluster api." https://cluster-api.sigs.k8s.io/. (accessed: 02.05.2023).

[26] ETSI, "Osm - open source mano." https://osm.etsi.org/. (accessed: 02.05.2023).

[27] Cloudify, "Bridging the gap between applications and cloud environments." https://cloudify.co/. (accessed: 02.05.2023).

[28] Redhat, "Redhat - openshift." https://www.redhat.com/en/technologies/cloud-computing/openshift. (accessed: 02.05.2023).

[29] D. A. Tamburri, W.-J. Van den Heuvel, C. Lauwers, P. Lipton, D. Palma, and M. Rutkowski, "Tosca-based intent modelling: goal-modelling for infrastructure-as-code," *Sics software-Intensive cyber-Physical systems*, vol. 34, pp. 163–172, 2019.

[30] O. Yilmaz, "Extending the kubernetes api," in *Extending Kubernetes: Elevate Kubernetes with Extension Patterns, Operators, and Plugins*, pp. 99–141, Springer, 2021.

[31] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[32] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[33] T. Theodoropoulos, A.-C. Maroudis, J. Violos, and K. Tserpes, "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, IEEE, 2021.

[34] J. Violos, S. Tsanakas, T. Theodoropoulos, A. Leivadeas, K. Tserpes, and T. Varvarigou, "Intelligent horizontal autoscaling in edge computing using a double tower neural network," *Computer Networks*, vol. 217, p. 109339, 2022.

[35] Y. Zhang, Y. Li, R. Wang, J. Lu, X. Ma, and M. Qiu, "Psac: Proactive sequence-aware content caching via deep learning at the network edge," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2145–2154, 2020.

[36] T. Theodoropoulos, A. Makris, J. Violos, and K. Tserpes, "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*, pp. 19–24, 2022.

[37] T. Theodoropoulos, J. Violos, S. Tsanakas, A. Leivadeas, K. Tserpes, and T. Varvarigou, "Intelligent proactive fault tolerance at the edge through resource usage prediction," *arXiv preprint arXiv:2302.05336*, 2023.

[38] W. Chen, Y. Chen, J. Wu, and Z. Tang, "A multi-user service migration scheme based on deep reinforcement learning and sdn in mobile edge computing," *Physical Communication*, vol. 47, p. 101397, 2021.

[39] T. Theodoropoulos, A. Makris, I. Kontopoulos, J. Violos, P. Tarkowski, Z. Ledwoń, P. Dazzi, and K. Tserpes, "Graph neural networks for representing multivariate resource usage: A multiplayer mobile gaming case-study," *International Journal of Information Management Data Insights*, vol. 3, no. 1, p. 100158, 2023.

[40] C. Fang, T. Zhang, J. Huang, H. Xu, Z. Hu, Y. Yang, Z. Wang, Z. Zhou, and X. Luo, "A drl-driven intelligent optimization strategy for resource allocation in cloud-edge-end cooperation environments," *Symmetry*, vol. 14, no. 10, p. 2120, 2022.

[41] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[42] Cloud Native Computing Foundation, "Prometheus." https://prometheus.io. (accessed: 02.05.2023).

[43] M. Iorio, F. Risso, A. Palesandro, L. Camiciotti, and A. Manzalini, "Computing without borders: The way towards liquid computing," pp. 1–17, 2022.

[44] L. Osmani, T. Kauppinen, M. Komu, and S. Tarkoma, “Multi-cloud connectivity for kubernetes in 5g networks,” *IEEE Communications Magazine*, vol. 59, no. 10, pp. 42–47, 2021.