Toward Supporting XR Services: Architecture and Enablers

Tarik Taleb, *Senior Member, IEEE*, Abderrahmane Boudi, Luis Rosa, Luis Cordeiro, Theodoros Theodoropoulos, Konstantinos Tserpes, Patrizio Dazzi, Antonis I. Protopsaltis, and Richard Li

***Abstract*—Emerging cross-reality (XR) applications, including holography, augmented, virtual, and mixed reality, are characterized by unprecedented requirements for Quality of Experience (QoE), largely exceeding those currently attainable. To cope with these requirements, noticeable efforts and a number of initiatives are ongoing to enhance the current communications technologies, especially in the direction of supporting ultralow latency and increased bandwidth. This work proposes an architecture that puts together the key enablers to support future XR applications, highlighting the shortcomings of existing technologies and leveraging the ongoing innovations. It demonstrates the feasibility of the proposed architecture by describing the processes driving the platform with relevant use case scenarios, and mapping the envisioned functionality to existing tools.**

***Index Terms*—5G and beyond, AR, cross reality (XR), edge computing, holography, immersive services, network function virtualization (NFV), network slicing, New IP, open RAN (O-RAN), VR.**

I. Introduction

As 5G communication systems are being rolled out, we observe high expectations at 5G, such as the processing of data silos to provide real-time feedback within nanoseconds, leveraging multiaccess edge computing (MEC) capabilities. The deployment and standardization activities of 5G networks are accordingly intensified. Despite these efforts, 5G is in a relatively early stage of adoption, and it is not expected to be in a position to support high data rates, and highly reliable communication links for advanced mobile cross-reality (XR) applications.

Emerging XR applications, including augmented/virtual/mixed reality and holography, do not rely only on fixed networks, but also on technologies that allow user mobility. They demand unprecedented Quality of Experience (QoE) requirements. Technology enablers, in this case, are directly dictated by QoE requirements of the end-users' XR applications. In fact, studies show (e.g., [1]) that for an acceptable user experience (no motion sickness) with high fidelity, the end-to-end latency should be less than 15 ms and the bandwidth should scale up to 30 Gb/s. To provide an indication, streaming in VR applications requires 10× more bandwidth than a 4K video. These figures exceed by far those currently attained. It is indicative that infrastructure and application providers currently respond to the increased demand of digital services by throttling services or reducing their quality.[1]

6G connectivity promises to tackle such QoE requirements with offers of low latency communications and ubiquitous mobile ultrabroadband. Such figures dwarf the rates that are usually considered in pure 5G ultrareliable low latency communication scenarios [2]. The key features of 6G networks include mobility support of up to 1000 km/h (compared to 500 km/h in 5G), control-plane latency of less than 1 ms (compared to 10 ms in 5G), traffic capacity of tup to 1–10 Gb/s/m^2 (compared to 10 Mb/s/m^2 in 5G), 3-D localization premion of 1 cm (compared to 2-D precision of 10 cm in 5G), uniform 3-D user experience of up to 10 Gb/s (compared to 2-D experience of 50 Mb/s in 5G) [3]. Most importantly, 6G is expected to be able to implement the necessary technologies that will materialize a fully fledged tactile internet (TI) [4].

The vision of the TI includes the support of haptic information (i.e., touch, actuation, motion, vibration, and surface texture) real-time transmission over the Internet [5]. This concept is central to the realization of the future XR applications which cannot be supported by existing network infrastructures. For instance, existing centralized architectures do not sufficiently meet the QoE requirements mentioned above, as well as the inherent need for mobility of the XR applications. To this end, more distributed network architectures based on edge computing need to be investigated with the intention of bringing the XR applications closer to the

Manuscript received 8 September 2021; revised 28 March 2022; accepted 28 October 2022. Date of publication 15 November 2022; date of current version 6 February 2023. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme through the CHARITY Project under Grant 101016509. *(Corresponding author: Tarik Taleb.)*

Tarik Taleb is with the Faculty of Information Technology and Electrical Engineer, University of Oulu, 90570 Oulu, Finland (e-mail: tarik.taleb@oulu.fi).

Abderrahmane Boudi is with the Communications and Networking Department, ICTFicial Oy, 02130 Espoo, Finland, and also with the Laboratoire de la Communication dans les Systèmes Informatiques, École Nationale Supérieure d'Informatique, Algiers 16309, Algeria (e-mail: a_boudi@esi.dz).

Luis Rosa and Luis Cordeiro are with the Research and Development Department, OneSource, 3030-384 Coimbra, Portugal (e-mail: luis.rosa@onesource.pt; luis.cordeiro@onesource.pt).

Theodoros Theodoropoulos and Konstantinos Tserpes are with the Department of Informatics and Telematics, Harokopio University, 176 76 Kallithea, Greece (e-mail: ttheod@hua.gr; tserpes@hua.gr).

Patrizio Dazzi is with the Department of Computer Science, University of Pisa, 56126 Pisa, Italy (e-mail: patrizio.dazzi@unipi.it).

Antonis I. Protopsaltis is with the Department of Electrical and Computer Engineering, University of Western Macedonia, 50100 Kozani, Greece, and also with the Department Research and Development, ORamaVR S.A., 70013 Heraklion, Greece (e-mail: aprotopsaltis@uowm.gr).

Richard Li is with the Network Technologies Lab, Futurewei, Santa Clara, CA 95050 USA (e-mail: rli@futurewei.com).

Digital Object Identifier 10.1109/JIOT.2022.3222103

[1]https://www.bbc.com/news/technology-51968302

end users [6]. Moreover, it is essential to redesign future wireless access networks to enhance various aspects of the physical and medium access control (MAC) layers. It is also equally important to explore to the maximum the emerging network technologies, such as software-defined networking (SDN), network function virtualization (NFV), and network coding, in order to meet the strict reliability and latency requirements of XR applications [7], [8].

A number of ongoing initiatives, especially in the direction of ultralow latency and increased bandwidth, aim to accelerate the 6G vision implementation. Such an initiative is Open-RAN (O-RAN) which is promoting the idea of splitting the radio access network (RAN) into various parts based on the functionality, and enabling the network behavior to be dependent to the QoE requirement of the processed application. This disaggregation allows each RAN radio unit to deliver Quality of Service (QoS) guarantees independently, bringing several advancements, such as low latency and also network slicing. The key enabler in O-RAN is the support of cognitive-based technologies [i.e., artificial intelligence (AI)/ machine learning (ML)] as the means for deploying, optimizing, and operating the mobile networks, through the independent automated operational network functions.

Another equally important initiative is the New IP [9], [10], [11] which is proposed as a mitigation measure for the complex interconnection and extreme demands of a rapidly increasing number of physical and virtual objects over the Internet, which is hindered by the design of the existing IP protocol. The New IP integrates a contract to each New IP packet that is processed by the network and routers, allowing for high-precision communications, user-network interface, in-band signaling, telemetry, and user-defined networking. This idea allows the abandoning of existing IP-based communication principles: throughput should be linearly proportional to bandwidth; latency should be linearly proportional to physical distance; and, packet loss should be an inverse function of buffer sizes.

This article proposes a platform architecture that puts together the key enablers to support future XR applications and to cope with the relevant challenges, considering the shortcomings of existing technologies and the ongoing innovations in various fields. In particular, this article contributes to the state-of-art in the following ways.

1) It defines a platform architecture that leverages existing standards to support XR.
2) It showcases how Open RAN and New IP can be used to cope with the expected limitations of 5G networks, along with a wide deployment of XR services.
3) It demonstrates the feasibility of the proposed architecture by mapping its functionalities to existing tools and open sources.

To deliver these innovations, this article is structured in the following fashion. Section II presents the work that is currently proposed in satisfying the requirements for ultralow latency and ultrahigh bandwidth, as well as the need to support user mobility. Section III presents the proposed XR platform architecture. Sections IV and V focus on the ongoing initiatives, that are more relevant to the delivery of the expected QoE requirements for future XR services, namely, OpenRAN and New IP. To demonstrate the feasibility of the proposed architecture, and to validate the concept, Section VI introduces potential tools and open sources. To further showcase the feasibility of the proposed architecture, Section VII introduces two sets of experimentation: the first implements a proof of concept to show how the infrastructure would be constructed to deploy cloud-native XR applications, and the second evaluates a closed-loop mechanism that drives the infrastructure toward a state of high energy and cost efficiency while maintaining QoS. Finally, concluding remarks and plans for future work are presented in Section VIII.

II. Related Work

The main challenges that the desired architecture is meant to address are the need for ultralow latency/ultrahigh bandwidth and user mobility. As such, we investigate the related work focusing on these two aspects. This analysis will provide us with the necessary information to decide on the technologies that our architecture can be built upon. However, in order to preserve a complete view of a modern architecture, we extend our research toward the orchestration of network resources, especially in multidomain orchestration scenarios.

A. Ultralow Latency

The emergence of IoT-based applications has increased the pressure on academics and practitioners alike to develop technologies able to provide the highly desired low-latency computing and communications services. The literature presents a number of such technologies attacking the problem from diverse perspectives.

One such perspective is the RAN. In order to facilitate ultralow latency applications, Fog-RAN (F-RAN) [12] was introduced. The cornerstone of F-RAN is the utilization of equipment present at the RAN in order to connect IoT devices with the cellular network. This equipment involves various user devices, which are considered structural blocks of the F-RAN, and are being referred to as F-RAN nodes. Instead of exclusively utilizing cloud resources, F-RAN employs fronthaul wireless communications and collaborative computing of multiple F-RAN nodes near the users, in order to achieve ultralow latency.

The Cloud-RAN (C-RAN) [13] is another architectural approach that aims at leveraging the RAN in order to achieve low latency. C-RAN heavily relies on remote radio heads (RRHs) randomly located over the coverage area. By utilizing this architectural paradigm, it is possible to facilitate functionalities, such as content caching, in order to achieve ultralow latency. Instead of retrieving the requested content from the core network, F-RAN transmits it from a nearby RRHs. This implementation of content caching alleviates the fronthaul traffic that would otherwise create a potential bottleneck.

Another novel perspective, specifically designed to cater to latency-related QoS requirements of XR applications, is introduced in [14]. This architectural framework is based on optimally conducting task offloading in order to facilitate delay-sensitive traffic. It consists of the user, the edge, and the cloud

layer. In the case of collaborative processing, the tasks to be offloaded are selected based on a number of parameters, such as the cloud server transmission delay, and the computational delay at the user, edge, and cloud layers, respectively.

Some other architectural approaches focus on reducing queuing delay in order to facilitate latency-sensitive applications. For instance, the well-established DiffServ [15] provides expedited forwarding for some packets at the expense of others. In order to solve this issue, ultralow queueing Latency, ultralow Loss, and Scalable throughput (L4S) [16] was introduced. L4S aspires to relieve Internet services of the necessity to facilitate functionalities, such as traffic policing, and contracts that prioritize specific packets over others. While active queue management (AQM) can be utilized to improve the performance of all traffic flows, there is a limit to the extent that queueing delay can be reduced by exclusively altering the network. In the context of queueing delay, the degradation of the performance usually occurs when a rather large capacity-seeking flow is present at the bottleneck link along with other types of traffic. Furthermore, the use of congestion control mechanisms, that are implemented by standard TCP, introduces certain constraints [17]. The replacement of standard TCP protocols, that are currently being utilized, with scalable alternatives, able to exploit AQM to its full extent, should be able to provide better overall performance.

B. User Mobility

User mobility modeling is essential for service migration at runtime. This is even more relevant now, with the advent of IoT devices being connected to the infrastructure and users being dependent on specialized end devices such as head-mounted displays (HMDs). The challenge is twofold: on one hand, the need for protocols that will make the networking feasible as the service is "following" the client, and on the other hand, the need for more computational approaches that will enable the actual migration of the service.

In the category of the networking protocols, a noteworthy approach is MobilityFirst [18], an ICN network architecture that defines a flat and globally unique identifier to each network object independently of its network address. It then uses a service to map the unique identifier to the group of network addresses, allowing prompt handoff, as the service is moving following the client. The drawback of this approach is that, in the case of IoT devices, it still uses fixed-length addresses resulting in high power consumptions.

To this end, we also considered the work in [19] which studies and compares three LPWAN standards that take the mobility of the things, or devices, into consideration, namely, LoRaWAN, DASH7, and NB-IoT. These technologies are designed to offer a set of features, including wide-area and massive scale connectivity [20] for low power, low cost, and low data rate devices. This latter characteristic makes those standards inappropriate for XR applications.

In the front of physical migration approaches, we can identify three types of service relocation.

1) Handoff process, including "break-before-make" and "make-before-break," refers to the transfer of ongoing connection sessions following the user mobility. In this type of applications, the services are instantiated at the central clouds while the intermediate connections are transferred at the anchor nodes. Unfortunately, running applications at the central clouds have a negative impact on end-to-end delay and bandwidth, which is not suitable for high-interactive applications.
2) Service application migration across different edges. This type of service relocation requires data serialization to enable the user context mobility across different replicates deployed at the edge-nodes following user mobility. However, this technique has three main drawbacks.
   a) Application dependability that requires a dedicated service relocation for each service and application, which breaks the concept of network modularity.
   b) Data serialization that may require a higher computation time that has a negative impact on the time of service relocation.
   c) The deployment of replicates at different edges leads to resource under-utilization and over-provisioning.
3) Finally, system-level migration requires the migration of the whole microservice (i.e., container) across edges following user mobility. The main drawbacks of this technique are the migration of unnecessary data, such as operating system, and technological dependability. In fact, microservice migration across different multi-technological domains (e.g., running different operating system distributions) still needs extra works.

Checkpoint/Restore in Userspace (CRIU[2]) has been widely used for enabling system-level migration across different edges. Many studies have been proposed to optimize the service relocation which could be cost-related optimization or time related. In cost-related optimization solutions, where a migration decision has to be taken based on the incurred cost to avoid costly migrations while ensuring the system performance (e.g., QoE), a tradeoff between the performance and incurred costs was studied in order to reduce the cost while ensuring QoE. Taleb et al. [21] proposed an MDP-based optimization to capture the tradeoff between the migration cost and the user experience.

A tradeoff between E2E delay and the migration cost was a strategy followed by a few proposals, including mixed-integer linear programming (MILP) [22] and Lyapunov-based optimization model [23]. However, many solutions were proposed in order to avoid frequent migration costs, based on the user mobility, a prediction of future nodes where a service will be migrated can reduce even better the cost, either to reduce migration and transmission costs [24], the number of migrations [25] or network utilization [26]. In time-related optimization solutions, these solutions focus on reducing the downtime or the migration time. However, they all build on the premise of a fixed, largely available network, often bearing unrealistic properties in terms of latency and bandwidth.

[2]https://criu.org/Main

An alternative would be to predict the user movement or the general load at the edge nodes and proactively migrate the services and data. In edge computing, this is indeed a rather popular research area, e.g., [27], [28], [29], and [30]. These solutions also suffer from a lack of realistic conditions under which they perform, especially in the aspect of the availability of adequate network resources. Prediction mechanisms are trained over data sets that incorporate the assumption that network bandwidth is unrealistically high.

C. Orchestration

What makes the technologies examined up to this point such exceptional tools in regards to cloud-based orchestration is the fact that they have evolved in a way which allows their distinct operations to be aligned with each other. The aggregation of these technologies results in the emergence of highly adaptive cloud-based frameworks which are able to operate optimally despite the challenges they are presented with. These challenges derive from the need to facilitate various rather demanding services such as XR applications. Applications, such as these are often associated with strict latency-related and bandwidth-related QoS requirements. These requirements have to be implemented in an environment which is comprised of numerous, heterogeneous network assets. Nowadays, cloud-based frameworks have to facilitate a large number of computational and network resources. Furthermore, these resources are often part of different domains and/or located at entirely different regions. This has led to a drastic increase in the complexity which is associated with the orchestration of cloud-based resources. The orchestration of complex systems such as these requires the use of automation technologies, since the notion that these systems can be managed via human-centered intervention alone can no longer be sustained. These technologies have gradually formed an ecosystem which allows them to operate collaboratively and matured into the backbone of many cloud-based frameworks. The following analysis is focusing on highlighting the relevant orchestration technologies that will enable the support of nextgen XR services.

It is of paramount importance for network services (NSs) to be able to dynamically scale up or down in order to meet the QoS Requirements. In 2012 the NFV [31] paradigm was introduced by the ETSI standardization body. The cornerstone of the NFV paradigm is its inherent ability to decouple the software implementation of network functions from the resources they utilize. This decoupling enables the formation of Virtualized Network Functions which virtualize entire classes of network functions into building blocks which can form connections in order to create specific NSs. These NSs are able to dynamically scale up or down in accordance to the utilization of the various network resources. In order to provide highly reliable and scalable services, it is essential for the network to be able to instantiate, monitor, and repair the various NFV instances. This process is known as virtual network function (VNF) orchestration. In most cases, NFV management and orchestration (MANO) [32] is the component that acts as the orchestrator of the entire NFV system. NFV MANO is the core element in the management of NFV architecture. Its responsibilities include the orchestration of the NFV infrastructure and the life cycle management of the NSs. The orchestration process can be centralized, distributed, or hybrid. The initial perception of NFV was that it should be exclusively implemented in data centers. Yet it has soon become apparent that in order to fully exploit the advantages provided by NFV, it is of paramount importance for a service provider to be able to freely facilitate NFV in all possible locations.

Each Virtualized Network Function consists of one or more virtual machines (VMs) or containers. During the initial conceptual stages of the NFV network architecture, only VMs were considered to be viable options with regards to implementing this particular paradigm. While it is possible to support containers in operating systems, such as Linux and Windows, currently a large number of VNFs do not support these specific operating systems. Nowadays, the concurrent utilization of both VMs and containers is widely considered to be the optimal solution. With regards to managing VMs, Tacker is an Openstack[3] project, which is in charge of providing VNF Manager and VNF Orchestrator functionalities, in a manner which is aligned with OpenSource MANO. On the other hand, managing containers requires the use of tools such as Kubernetes.[4] Regarding the new virtualization technologies such as the ability to facilitate containerized VNFs and container infrastructure management, in November 2020, ETSI has published its first document containing specifications that enable containerized VNFs to be managed in an NFV framework [33]. The utilization of containerized VNFs provides numerous advantages, such as better service agility and performance. Furthermore, containerized VNFs present auto-scaling capabilities and can achieve service elasticity in runtime, due to their light-weight resource usage [34]. These specifications describe the new functions required for the MANO of containers. Both these virtual infrastructure managers (VIMs) are implemented in the MANO layer.

Cloud-native network functions (CNFs) are a successor to the Virtualized Network Functions. CNFs are containerized microservices that communicate with each-other via the use of standardized RESTful application programming interfaces (APIs). As telecom networks gradually integrated VNFs, it soon became vital for them to resort to cloud-native approaches which significantly shorten the time required in order to conduct various essential operations. Cloud-native approaches which utilize container-based network functions provide agility in the launch and upgrade of services. To enable a cloud-native approach, network functions are decomposed into microservices hosted within different containers. The CNFs are then able to scale automatically and communicate with each other via well-defined APIs. Furthermore, since it is needed to update only specific microservices at a time, the overall upgrade time is reduced. The Continuous Integration/Continuous Delivery deployment model is supported since network functions are decomposed into smaller

[3]https://www.openstack.org/
[4]https://kubernetes.io/

chunks. On top of that, through service discovery and orchestration, systems that utilize CNFs tend to be less prone to being affected by node failure.

Over the last years, various novel architectural paradigms that enable the orchestration of cloud-based services and applications have been introduced. There have been many variations of these architectural paradigms in order to cover the rather wide range of requirements that need to be met. Depending on the number of network domains that are involved in the orchestration process, it is possible to make a distinction between single-domain and multidomain orchestration. Each network slice consists of heterogeneous network resources which are combined in order to create virtual networks over a common infrastructure. In order to facilitate NSs that span over multiple domains, it is essential for network slices to be properly managed [35].

SDN [36] is capable of centrally managing network slices. SDN is a network management technology that enables the creation of dynamic network configurations [37]. These dynamic network configurations are deployed in order to keep up with the QoS requirements. This architecture is based on the disassociation of the control from the data plane. By treating these two planes as distinct entities, centralized control over the network's assets is established. In the context of delay-sensitive traffic, the QoS requirements are formulated in accordance with the latency requirements. The dynamic allocation of resources is based on the latency requirements and the priority imposed by the specified class of each traffic flow.

The gradual introduction of 5G and network slicing technologies have given birth to the necessity to be able to facilitate fully automated and E2E service management which might span over multiple distinct domains. The ETSI Zero touch network and Service Management (ZSM) [38] framework was introduced in order to solve this issue. The ZSM framework includes an E2E Service Management Domain which is in charge of E2E orchestration across different domains, E2E closed-loop management, and E2E analytics. Beyond the implementation of E2E deployments, the ZSM framework heavily focuses on establishing automation via closed-loop processes which assist network optimization. The two most notable paradigms of closed control loops are the observe, orient, decide, act (OODA) and the MAPE-K (Monitor, Analyze, Plan, and Execute). Experiential network intelligence (ENI) [39] framework utilizes closed control loops such as these in order to keep up with the QoS requirements. More specifically ENI is able to assist or direct network management systems based on network status and service level agreements. The ENI entity is responsible for providing recommendations or commands to an assisted system (AS), in order to establish intelligent network management. The ENI entity communicates with the ASs via an API broker that is able to perform the required translations. Three classes of ASs have been identified based on the degree that the AI is incorporated in the MANO processes of the network. The AS that belong to the first class do not utilize any form of AI. The second class consists of ASs that utilize AI but not as part of their operational control loop. The third class consists of ASs that incorporate AI technologies in their operational control loop in order to receive recommendations or commands.

D. Related Projects

Up to this point, there have been several notable projects, which utilize various aspects of the technologies mentioned so far. The ANASTACIA[5] project aims to develop a holistic security framework for IoT infrastructures. Leveraging new monitoring methodologies and tools, it is able to formulate autonomous decisions in order to provide dynamic security. The ANASTACIA platform is implemented by utilizing SDN controllers, NFV orchestration platforms, and IoT controllers. The use of these technologies enables ANASTACIA to provide an IoT infrastructure whereby the data streams of IoT devices can be monitored, processed, and routed in a dynamic manner, thus ensuring security throughout the platform. MiCADOscale is a multifunctional, cloud-agnostic orchestration, and autoscaling framework which supports Kubernetes deployments and is a byproduct of the COLA[6] project. MiCADOscale supports autoscaling functionalities at both VM and container level.

NSPIRE-5Gplus[7] aims to design a zero-touch, end-to-end smart network and service security management framework. INSPIRE-5Gplus is able to provide protection when managing 5G network infrastructure across multiple domains. INSPIRE-5Gplus is aligned with the key principles of ETSI ZSM reference architecture. MonB5G's[8] purpose is to implement a framework which facilitates the provisioning, deployment, and lifecycle management of numerous network slices. Furthermore, it utilizes the monitor–analyze–plan–execute (MAPE) paradigm and distributed closed feedback loops, supported by AI-driven operations, in order to ensure a certain degree of autonomic network operation. The ACCORDION project [40] aims to orchestrate the compute and network continuum formed between edge and public clouds in an intelligent manner. The derived deployment decisions shall be taken based on privacy, security, cost and time criteria.

III. Architecture Description

In this section, the general architecture of an XR platform is given. We first present an overview of the architecture components. We then detail each one, separately.

A. Overview

Fig. 1 depicts the general overview of the architecture of the proposed XR platform. The XR platform follows both the NFV and ZSM frameworks in order to create self-managed E2E network slices. It is composed of three planes: 1) the deployment plane; 2) the domain-specific monitoring and reaction plane; and 3) the E2E conducting plane. The deployment plane consists of the infrastructure where the XR services are

[5]http://www.anastacia-h2020.eu/
[6]https://project-cola.eu/
[7]https://www.inspire-5gplus.eu/
[8]https://www.monb5g.eu/

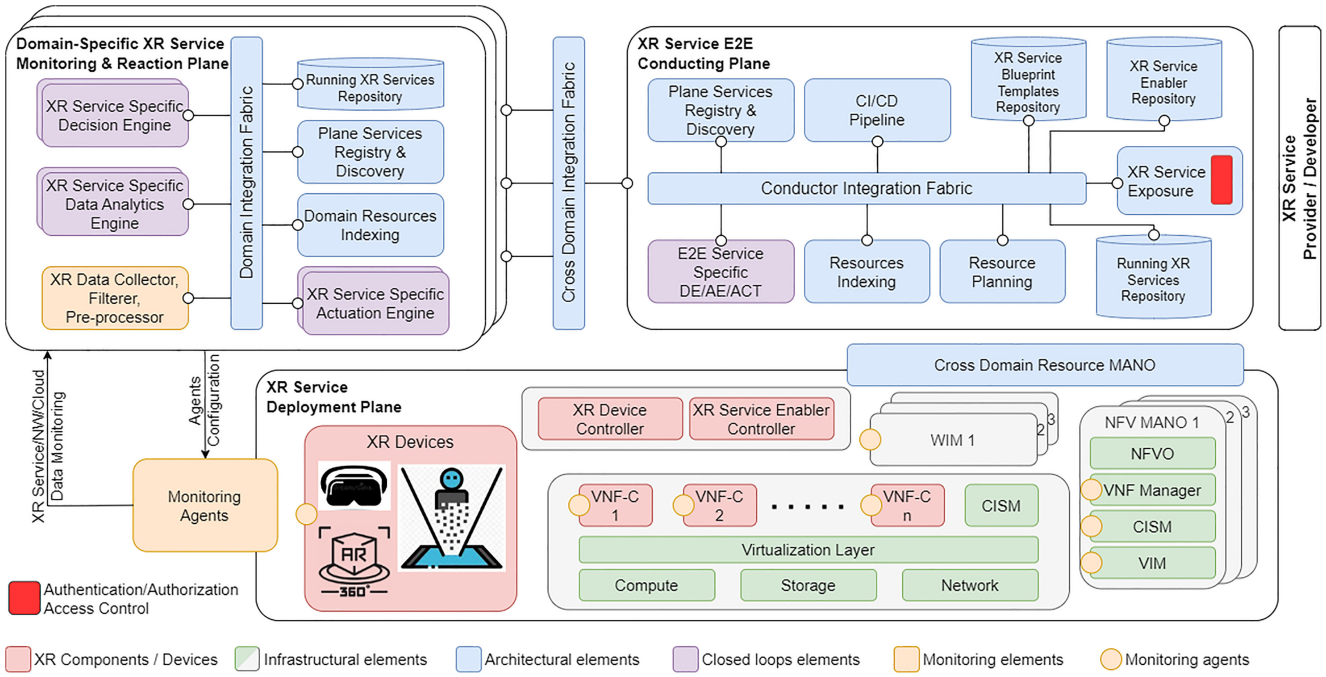


Fig. 1. High-level architecture of the proposed XR service provisioning platform.

actually running. It thus hosts the different VNFs that are composing the different XR services. The main responsibility of this plane is to manage the computational, network, and storage resources of the infrastructure. The domain-specific monitoring and reaction plane is responsible for monitoring the service inside a technological or administrative domain. This domain-level monitoring helps in detecting issues and mitigating them without having to resort to the E2E conducting plane. These local detection and mitigation mechanisms would accordingly lessen the burden exerted on the E2E conducting plane. The E2E conducting plane is responsible for creating the different subslices inside each domain and for monitoring the E2E KPIs of the XR services. It is also responsible for shifting the services between the different domains when necessary.

B. Deployment Plane

The deployment plane hosts the XR services. It is composed of different technological and administrative domains. Indeed, such domains can belong to different entities which may result, for instance, in different charging schemes and also different management APIs. These domains can be also different in the nature of the underlying technology, such as the RAN, edge and cloud domains. Furthermore, it shall be noted that an XR service can have different components (VNFs) running on different domains.

In order to manage the different domains, multiple NFV MANO instances, one for each domain, exist inside the deployment plane. Each NFV MANO can be decomposed into an NFV Orchestrator, a VNF manager, and a VIM; as per the ETSI NFV architecture. The NFV Orchestrator is responsible for the onboarding of NSs and VNFs and for performing the lifecycle management of NSs. It is also responsible for the validation and authorization of changes to the resources allocated to the VNFs. The VNF manager is responsible for the lifecycle management of one or a group of VNFs. Finally, the VIM is responsible for managing the infrastructure resources. Specifically, it manages the computational, network and storage resources.

The existence of different domains necessitates networks that connect/stitch all these domains. The component that is responsible for managing these networks is called the WIM, WAN—Wide Area Network—Infrastructure Manager. The WIM is a special case of a VIM. While the latter manages all of computational, storage, and networking resources, the former is specialized in managing networks. Its main objective is to connect/stitch the different VNFs within a single domain or across several technological and/or administrative domains. The deployment plane can accommodate several WIMs, used to "stitch" the different NSs that are deployed in different domains.

The virtualization layer offers a unified view of the computational, storage, and network resources. This unified view helps to aggregate and seamlessly run VNFs on top of the infrastructure. An NS can be composed of multiple VNFs, able to run either on VMs or on containers. Up until recently, both VMs and containers could not be run on the same infrastructure. Recently, ETSI defined the container infrastructure service management (CISM) that enables MANO infrastructure to support containerized workloads, and thus, VMs and containers can coexist on the same infrastructure. The main responsibility of CISM is to manage the infrastructure's resources and to perform the lifecycle management of the containers running on top of the container cluster. ETSI has proposed different architectures on how the container cluster

can be implemented [41]. For instance, the containers can be run on bare metal, in VMs, or they can be distributed between the two.

Monitoring the running VNFs is of utmost importance. It is the foundation upon which the functionalities of the domain-specific monitoring and reaction plane as well as the functionalities of the E2E conducting plane highly depend. Indeed, in order to be able to predict service-level agreements (SLAs) violation, and to promptly react to and mitigate such events, a thorough monitoring of the infrastructure and the running software is mandatory. The monitored data can differ in location and in nature. The monitored data can be service level information that can be gathered from VNFs. It can be infrastructure level, such as computational, storage, and sand network data. It can be also data from the end users, such as the QoE. Thus, besides the VNFs, a myriad of monitoring agents are deployed across the infrastructure and at different levels.

C. Domain-Specific Monitoring and Reaction Plane

The domain-specific monitoring and reaction plane is responsible for managing only one domain. Its main responsibilities are: 1) keep track of the resources consumption and of the XR services running in the domain; 2) process the monitored data; and 3) perform local analytics, make decisions and carry out the actuation which are specific to each running XR service.

This plane keeps track of the domain resource usage and lists the domain capabilities and the domain running services. The advantages of recording this kind of information are manifold. For instance, a "service registry and discovery" entity would allow different tenants to reuse and share the same service. Similarly, prediction mechanisms can use the evolution of resource utilization to predict eventual SLA violations and/or service degradations.

One of the responsibilities of this plane is the collection, filtering, and preprocessing of the monitored data. Data collection is done by agents located within the infrastructure, beside the VNFs. There are mainly two types of data collection agents, namely, the ones using the push method and the ones using the pull method. The former type resides beside the monitored entity and send the collected data to a server (i.e., a corresponding service or micro-service at the domain-specific monitoring and reaction plane), whilst the latter type sends the collected data only when instructed/inquired by the relevant service. Generally, both of these methods coexist in the same infrastructure. Filtering monitored data helps reducing the data size to be processed, by filtering out duplicates and unnecessary data. The level of filtering can be dynamically adjusted according to the needs of the monitored XR application. The preprocessing can have multiple forms. Its main purpose is to help other entities to seamlessly ingest the monitored data. For instance, it can be in the form of feature extraction algorithms that help reduce the feature number and the dimension of the data which, in turn, reduces the needed bandwidth for transfering the monitored data. It can be also in the form of converting the data into a specific format. Preprocessing can also consist of a distributed ML algorithm whereby the first few layers of the neural come network are calculated close to the data source [42].

Following the ZSM framework, this plane implements a local automation loop [43]. This loop consists of the monitoring entity described above, that is shared among all XR services, an analytics engine, a decision engine, and an actuation engine, that are dedicated to each XR service. The analytics engine consists of algorithms that ingest the monitored data and then produce insights or alerts. For instance, such algorithms can predict the state of the service, detect misbehaving services and attacks, and identify optimizations that can greatly enhance the QoS. These algorithms are mostly designed using ML algorithms. The decision engine uses the insights gained from the analytics engine in order to derive its decisions. It can receive alerts from the analytics engine, as it can directly ingest low-level monitored data. Its main purpose is to make sure that the running XR services keep meeting their SLAs. It can carry out decisions such as service recomposition, service migration, or even slight VNF reconfigurations. Finally, the actuation engine's role is to decide how to implement the decisions that were made by the decision engine. Mainly, it is responsible for translating decisions into actions, and executing the produced actions within the relevant entities.

D. E2E Conducting Plane

The E2E conducting plane is responsible for managing the overall XR framework. It is the entry point for XR providers/developers from which they launch their services. It is also responsible for the lifecycle management of XR services.

This plane keeps track of all the running XR services. It records, near real time, information about the resource consumption of all domains. It also holds information about domain capabilities and generic XR service blueprints. Thus, it is the main place where XR service planning is conducted. Indeed, upon receiving a request from an XR provider/developer, it translates the request into a blueprint, selects the set of domains where the service will be split upon, and finally carries out the negotiation with the respective domains.

Similar to the automation loop in the domain-specific monitoring and reaction plane, this plane also contains a loop. There is a loop for each XR service and it is responsible for E2E level service recomposition. The E2E analytics engine takes inputs from analytics engines of all domains where the XR service is running. Correlations between monitored data, produced from different domains, are used to produce alerts and state predictions of the XR service. The robustness of the E2E analytics engine algorithms is an important issue since this engine is responsible for monitoring E2E KPIs.

Even in cases whereby E2E KPIs are monitored by user equipment (UE) for reliable detection of service degradation, the E2E analytics engine is responsible for finding the reason behind such service degradations. The role of the E2E decision engine is to decide on the course of action that needs

to be taken in order to keep the XR service up and running. This means that the E2E decision engine can perform service recomposition per domain as it can perform an E2E-level service recomposition. Such decisions may consist in VNFs reconfiguration, scale-up and scale-down operations, VNFs migration, or even domains reselection. There are multiple reasons why service recompositions may become required [44]. For instance, they may be due to security concerns, to avoid a future service degradation or mitigate an existing one, or to optimize the resources utilization by the XR service. Finally, the E2E actuation engine is responsible for implementing the decisions made by the E2E decision engine.

During the last few years, the DevOps concept is becoming highly popular, as there is a growing number of companies that are embracing that concept. DevOps brings down the separation between the development team and the production team, which results in a smaller time to market for new features and products. One of the main pillars of DevOps is automation, which consists in automating the build, the deployment, and the monitoring of an application. Therefore, a CI/CD pipeline is needed in order to automate the build and deployment of XR services. This pipeline is used to integrate the service provided by an XR developer with the services provided by other developers or by the XR platform itself. Once the new version of the service is thoroughly tested, it can be deployed. For instance, the pipeline can perform a rolling update, where the newest version of the service is gradually deployed, as it can expand testing by performing a canary deployment, where only a fraction of the XR traffic is routed to the newest version of the application.

E. Integration Fabric

The integration fabric enables the interoperation and communication between the different functions. Through the integration fabric, the different functions play both roles of service consumer and service producer. It allows registration and discovery of services, which means that services should be able to register and be added into a catalog. Services can discover each other by searching the catalog for specific capabilities. It also allows the invocation of services, either by a direct request to a specific service or by a request to a class of services (Service Mesh concept). The integration fabric also offers dedicated communication channels between the different services. All of these features are protected by an authentication and authorization service.

The integration fabric inside the domain-specific monitoring and reaction plane and the E2E conducting plane helps the components interplane communication. It allows having default secure communication channels between the different components. The cross-domain integration fabric helps the communications between the domains and the communication with the E2E conducting plane.

F. Use Cases

In what follows, we shall show how XR services can leverage the XR platform to ensure the best performances. XR services are and will be used in many industries; entertainment and communication will be deeply impacted. One of the XR use cases will be the organization of remote live concerts where the musicians are depicted as holograms. In such a configuration, a band can perform live for a crowd that is split into several venues across different cities or even countries. Each venue can have a different type of holography devices. Also, the holography devices can be set up in concert halls, amphitheatres, stadiums, and even parks. The band members can perform from separate locations, while all different streams are sent to the different venues where they are synchronized. This use case has very stringent latency and bandwidth requirements, as the former should be extremely low, to ensure a good sound quality, and the latter can be in the realm of many Gbps (and even Tbps) for each hologram.

In order to implement the above scenario within the XR platform, the cooperation of multiple actors should take place. The musicians are considered content providers, while the crowds are content consumers. Both of these can be considered as end users. An XR provider is responsible for installing the infrastructure in the end-users places, providing the software that gathers the input, synchronizes the streams, and converts the output to the right format, in the case of different holography rendering technologies. The XR provider needs multiple network and service providers in order to run the needed software and to connect all the actors together. Thus, the XR provider would need an XR platform in order to deploy and maintain the smooth running of the service.

G. Procedures for XR Services Deployment and Management

This section presents the procedures of the XR platform that help in the deployment and maintenance of XR services.

In what follows, we consider the use case of a live XR concert using holography, where two musicians, present in two different cities, perform a live show that is projected in three venues, a music hall, a stadium, and a park. Given the ultralow latency requirements for music and the extremely high bandwidth requirement for holography, two network links would connect each musician to the cloud. Due to the fact that the synchronization of music happens in the cloud, links with deterministic latency should be used to carry the sound from the musicians to the cloud. For the holography links, some preprocessing can take place in the edge, close to the musicians, and the rest of the processing takes place in the cloud where it is also synchronized along with the audio before streamed to the crowds. Fortunately, due to human perception, the synchronization between audio and video is not very restrictive [45]. Finally, the resulting stream is sent to the three venues, where it is decoded, adapted to the holography devices and projected. Given the nature of the venues, the required resources for each venue may differ. Indeed, the stadium and the concert hall can have a wired connection with processing power that belongs to the XR provider acting as an edge, while the park venue would use wireless technology coupled with edge providers.

1) Launch of XR Service: The instantiation of an XR service follows the procedure illustrated in Fig. 2. An XR provider can be contacted by an end user in order to launch

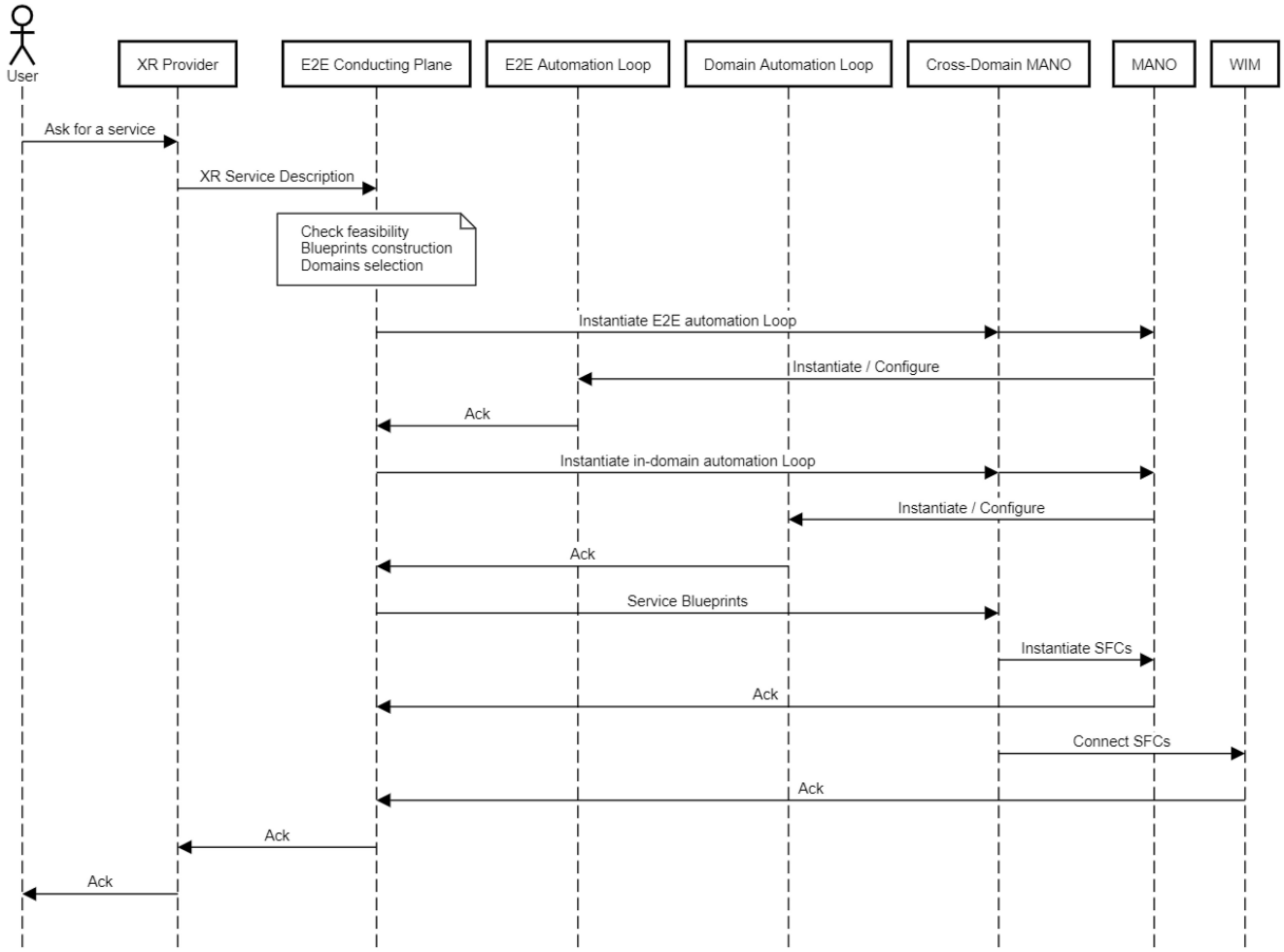


Fig. 2. Launch of a new XR service.

an XR service. The XR provider sends a request to the XR platform detailing the service to be launched. The XR platform or the E2E conducting plane checks if the service can be launched in the current state of the platform. This feasibility check makes sure that the platform has enough resources to accommodate the new service. Using a Blueprint Template Repository and an Enabler Repository, the request of the XR provider is translated into one or multiple detailed blueprints characterizing the XR service to be deployed. The XR platform checks the feasibility of the blueprints and selects one of them to be deployed. Once a detailed blueprint is selected, the domains that will host the XR service are selected. To launch services inside the domains, the E2E conducting plane uses an entity dubbed cross-domain resource MANO that offers a unified interface to all administrative and technological domains composing the XR platform. This interface first instantiates the required services for the E2E automation loop, and then initiates the automation loops of the new XR service within each domain and connects them to the E2E automation loop. For the MS, this can consist in setting up a monitoring system (e.g., Prometheus) that is specific for the XR service. For the AE and DE, it can consist in connecting them to all the relevant monitoring systems and also publishing their capabilities so they can be used by other services. The automation loops can be considered as a platform that can hold the algorithms for AE and DE. These algorithms can be dynamically added and removed at runtime. For instance, in the AE platform, it is possible to have many ML models for different purposes whereby they can be replaced by newer versions when needed. While in DEs, it is possible to have different ML models running in parallel voting on what would be the best decision, some of these algorithms may be proposed by the XR platform, XR providers, or even by third parties. Finally, the E2E conducting plane launches and configures the service function chains (SFCs) inside each domain, and uses the WIM to stitch the different SFCs across the different domains. The VNFs composing the SFCs should have monitoring agents that report to the domain-specific automation loop. Once the E2E XR service is up and running, the E2E conducting plane informs the XR provider that the XR service is ready, and ultimately the end user is accordingly notified.

2) Modification of XR Service: Fig. 3 depicts the procedures to modify slices in order to preserve the good functioning of an XR service. When an analytics engine notices or predicts a significant degradation in QoS/QoE in the near future, it sends alerts to the decision engine of its own domain. If the domain-specific decision engine can mitigate the issue inside the domain, it sends its decision to the respective

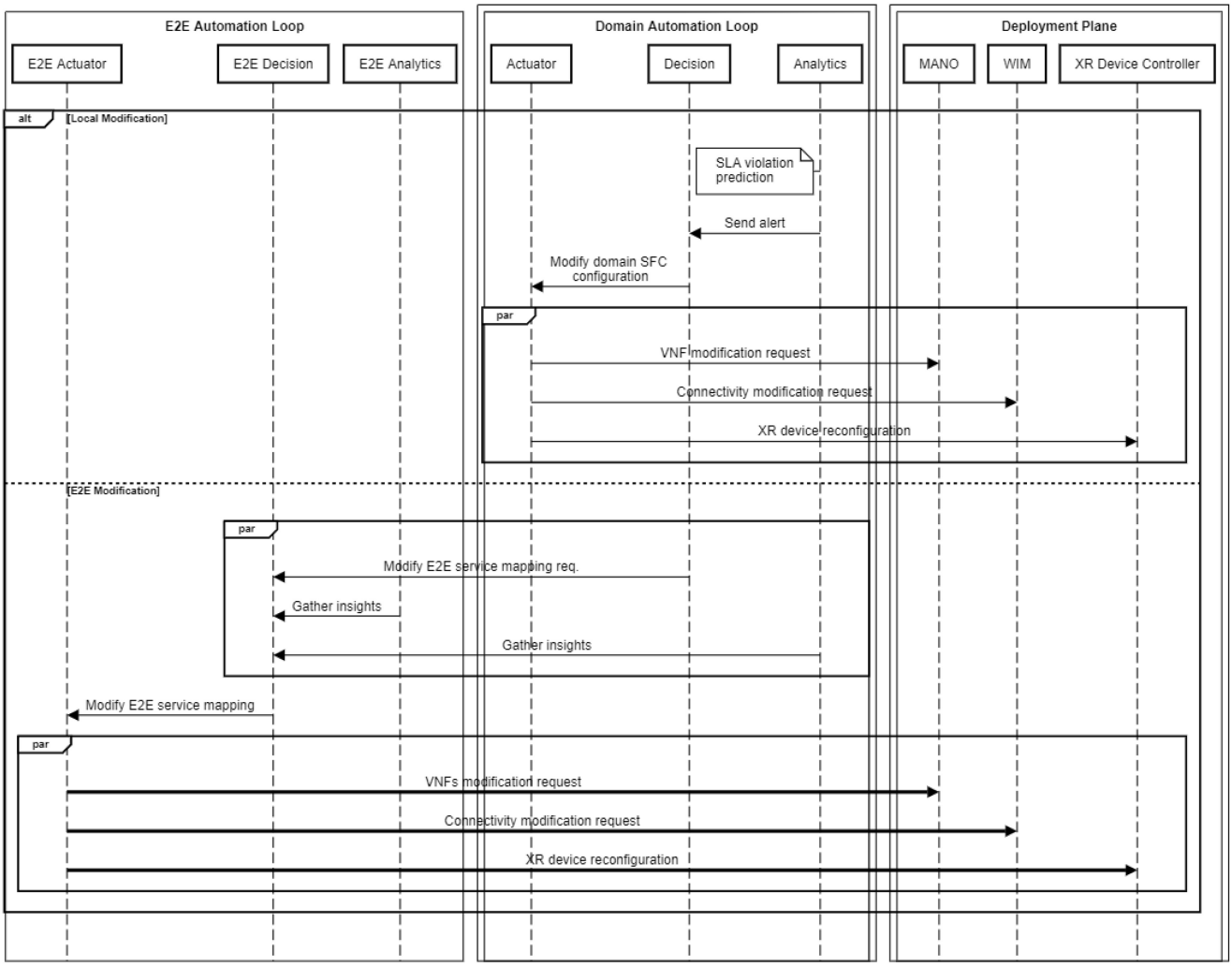


Fig. 3. Multidomain XR service reconfiguration.

actuator that will implement the decision within the boundaries of the domain. If the domain-specific decision engine cannot resolve the issue at the domain level, it sends a modification request to the E2E decision engine. The E2E decision engine may also receive alerts directly from the E2E analytics engine. After receiving an alert or a modification request, the E2E decision engine can decide to gather more insights and search for a new service recomposition that will keep the XR service healthy. Once a feasible configuration is found, it sends it to the E2E actuator. The latter translates the decision into actions and sends them to the relevant domains and/or XR device controllers.

IV. OpenRAN for XR Services

OpenRAN is an initiative led by most big actors of the telecom industry toward providing an open RAN solution. It consists in defining open and interoperable interfaces, virtualizing the RAN infrastructures and enabling RAN intelligence by leveraging AI and ML techniques [46]. Providing open interfaces and driving their standardization helps opening the RAN from a single-vendor closed environment to open multivendor deployment, enabling in the same time vendors, operators, and third parties to deploy innovative solutions and services as RAN applications. RAN virtualization would maximize the use of common off-the-shelf hardware and would allow to elastically deploy RAN service on the cloud, while AI/ML permits the optimization of the (virtual) RAN in real or near-real time.

The main idea behind OpenRAN is to drive the RAN to become multivendor and to open it to vendors, network operators, and third parties. This would permit to incorporate more intelligence at the RAN level. Such openness permits many small actors to build AI/ML solutions that can greatly improve the performance of the RAN, it would permit also service providers to customize RAN behavior to better suit their service needs. In short, OpenRAN can help reduce the network CAPEX and OPEX, it can improve the efficiency and performance of the network, and due to the enhanced agility of its architecture, it can quickly integrate new capabilities.

OpenRAN architecture consists in a Service Management and Orchestration (SMO) framework that contains the nonreal-time radio intelligent controller (Non-RT RIC), a Near-RT RIC, and all the interfaces between these and RAN components, such as remote unit (RU), centralized unit (CU), distributed unit (DU), and gNB base stations in the case of

5G deployments for instance. The SMO manages the cloud resources and the network functions whether being physical network functions (PNFs) or virtual/cloudified network functions (VNFs). The Non-RT RIC enables intelligent RAN optimizations in nonreal time, i.e., it leverages the services of SMO in order to provide a policy-based guidance leveraging AI/ML techniques. Near-RT RIC is considered as one of the RAN functions. Guided by the guidance of Non-RT RIC, Near-RT RIC also leverages ML models to offer a near-real-time optimization of the RAN.

OpenRAN is expected to greatly improve the QoE of XR services. Leveraging AI/ML techniques, it can achieve intelligent and proactive traffic steering capabilities which can offer some performance guarantees in the face of changing radio conditions. Also, it can help shifting from the current semi-static QoS framework toward service specific QoE prediction. This helps ensure proactive network optimization where the radio resources can be allocated in advance just before QoE degradation happens. OpenRAN allows dynamic configuration of the RAN resources, which, for instance, is changed according to the type of services requested by the users or by the status of the network. For instance, during congestion times, when the RAN would not be able to satisfy the requirements of all users, while by splitting the resources among all users would lead to poor QoE for them all, it may be better to reconfigure the network to satisfy the requirements of some prioritized users. The openness of the interfaces can help XR service providers to improve their services by implementing mitigation strategies at the service level according to the information reported by the RAN. For instance, such mitigation strategies consist in changing the resolution, frame per second (fps), or the used codecs [47], [48].

The proposed architecture natively supports the OpenRAN approach. Indeed, the RAN domain can be regarded as an instance of a "Domain-Specific XR Service Monitoring and Reaction Plane" where the SMO would be forming the closed-loop that drives the domain. In this particular instance, the Non-RT RIC would be encompassing all of AE, DE, and ACT. The openness of OpenRAN would permit the XR platform to dynamically deploy new policies to be followed by the Non-RT RIC or even replace it by another algorithm if deemed appropriate. This would allow the RAN to be continuously and dynamically configured to always support the currently deployed XR services.

V. New IP for XR Services

During the last decade, there has been a dramatic change in the very fabric of the applications produced. Concepts, such as Holography and XR are no longer considered novelties but actual application features that need to be implemented in a manner that guarantees that the various QoS requirements are met. The fact that it is of paramount importance for these applications to be able to operate in real time in order to provide an immersive experience to the end user makes them extremely latency–sensitive. On top of that, holography-based applications, in particular, require by nature huge amounts of bandwidth to be allocated. In other words, it is imperative to introduce network mechanisms that are able to provide: 1) guaranteed low end-to-end latency and 2) optimal utilization of the available bandwidth.

Up to this point, standard TCP and UDP have been considered the de facto transport layer protocols. Unfortunately, they do not seem to be able to keep up with the ever-changing landscape of cloud-based XR applications. Both of these protocols bear two distinct shortcomings that jeopardize the network's ability to keep up with the QoS requirements in regards to XR applications. The first one is their inability to guarantee a low upper bound of end-to-end latency [49], [50]. The second one is the fact that both of them, in their original form are not designed to optimally utilize the available network assets. The New IP initiative was introduced by the ITU-T Network 2030 Focus group. The New IP initiative is the study of various technologies that have been identified as of vital importance for the next evolution of the Internet. These technologies aim to provide advanced flexibility and deterministic services in the already established network paradigms. In order to do so, it is essential to reexamine the modus operandi of certain aspects of the Internet data plane, the protocols involved, and the subsequent architecture. The backbone of the New IP initiative is the New IP datagram format. This particular datagram format incorporates offsets that correspond to advanced functionalities that are crucial to XR applications. One of them is referred to as the New IP Payload. The purpose of the New IP Payload is to establish additional context for each payload. This type of context serves as a form of representation of the significance of each piece of information within each payload. That way, information within payloads can be rearranged in a manner that corresponds to their significance.

The inclusion of the New IP Contract facilitates a plethora of services, as well as their operational and administrative control at a level of packet granularity. Contrary to traditional QoS, Contracts are assurances that are implemented at packet level. Contracts are implemented via various specified Actions. Via the use of Contract Clauses, it is possible to specify under which conditions an Action is performed. As an example, let use a generic operation that selectively removes information from a payload each time there is not enough available bandwidth. Generally speaking, holography-based applications are notorious for being bandwidth intensive. Thanks to the New IP Payload, we know the relative position of the bytes of information that are less significant and thus their removal would not greatly affect the overall service. Thus, via the implementation of the New IP, it is possible to choose which parts of the actual Hologram to deliver/display, in case there is not available bandwidth for the entire structure. Two additional Actions that cater to the latency-related needs of XR applications are described in [11]. The first one is the Bounded Latency Action that instructs the router to deliver a packet within a specified time horizon. The second one is the Coordinate Action that allows multiuser applications to adjust the timing of their corresponding packet deliveries.

The significance of these Actions becomes rather apparent when contemplating two distinct requirements that the majority of XR applications share. The first one is the low end-to-end latency. In addition to that, some XR traffic flows

are temporally correlated to other ones. This fact introduces the necessity to facilitate synchronization among specific traffic flows. Due to these issues, the Deterministic Networking Working Group [51] was established. Although it is clearly stated that the Deterministic Networking Working Group is not involved with the modification of transport protocols, like the variations mentioned above, the study of technologies that operate alongside them is well within its scope of operations. The cornerstone of Deterministic Networking is the creation of deterministic data paths that are able to guarantee bounds of latency, loss, and jitter. The data paths are formulated in the context of layer 2 bridged and layer 3 routed segments. The New IP initiative focuses on large layer 3 networks and more specifically on developing methods of flow identification and packet forwarding over layer 3.

By classifying data flows, it is possible to establish specific data paths for the time-sensitive traffic flows. The reservation of specific network assets for each latency-sensitive service provides network-layer certainty of information transmission. Furthermore, by classifying data flows, one may simultaneously facilitate time-sensitive and best-effort services. This takes place by distributing the available transmission mediums between DetNet and non-DetNet flows in a fair manner. DetNet-enabled devices contain ports; each of which is equipped with a specific number of queues that are utilized by DiffServ and Best-Effort traffic. Each DetNet-enabled device in the network can be configured to utilize per-class or per-flow queuing [52].

The IEEE standard named IEEE 802.1Qch [53] (Cyclic Queuing and Forwarding) introduced the concept of cyclic operations in regards to coordinating queue and dequeue functionalities. The utilization of CQF relies on dividing time into intervals. Two queues are utilized for each class in order to perform enqueue and dequeue functionalities in separate time intervals. That means a traffic flow that arrives in interval x shall be put in one queue and shall be transmitted via the other queue during the $x+1$ interval. Then, the resulting frame shall arrive to a specified switch during the same time interval. This sets a harsh bound that dictates that the propagation latency has to be less than the selected time cycle. As a result, CQF is not suitable for large-scale networks due to the inherent difficulty of properly choosing a suitable time cycle. In [54], Cycle Specified Queuing and Forwarding is proposed in order to solve this issue. CSQF is rather similar to CQF with the exception of utilizing an explicit description of the various transmission cycles at every DetNet node present in the path spanning from source to destination.

Furthermore, the New IP initiative entails another key concept that needs to be taken into consideration in order to establish efficient forms of networking that are able to support next-gen XR applications. This concept is Multipath Routing. Multipath Routing is the simultaneous management and utilization of multiple paths in order to transmit streams of data flows. The implementation of this concept offers certain benefits, such as fault tolerance and increased bandwidth. Each stream is assigned a separate path. By doing so, multiple transmission queues are created, thus ensuring better utilization of the available bandwidth. In case the number of streams exceeds the number of available paths, some of them are chosen to share the available paths. On top of better transmission performance, Multipath Routing introduces advanced fault tolerance by assigning an alternative path to the stream, should the established one fail.

One way of implementing Multipath Routing is by utilizing routing strategies that operate in combination with the existing protocols. Equal-Cost Multipath Routing is the most notable Multipath Routing strategy. It is implemented by distributing traffic among various equal-cost paths by utilizing hash functions. One significant drawback of using this routing strategy is that it does not take into consideration the changes that are bound to occur in the status of the network status. This inability to keep up with the dynamic nature of the network leads to suboptimal load balancing. In [55], the Internet Engineering Task Force introduces Multipath TCP which is a set of extensions to standard TCP. These extensions enable transport connections to take place by utilizing multiple paths simultaneously. However, all the solutions explored to this point fail to provide a low upper bound of end-to-end latency. In [56], latency-controlled end-to-end aggregation protocol (LEAP) is introduced. LEAP is a multipath transport layer protocol that provides probabilistic end-to-end performance guarantees thanks to path multiplexing and interflow coding. The utilization of packet-level encoding enables the information of each data flow to be carried through all the available paths. The information is then retrieved at the destination. In [57], a rather similar approach is explored in the context of UDP for video streaming.

VI. Tools and Open Sources

Recently, we have witnessed an explosion of tools and open-source components focused on service, network, and infrastructure orchestration. ETSI-MANO aligned implementations, such as Open-Source MANO (OSM), ONAP, OpenBaton, or OPNFV, have been under active development in the last few years and are increasingly mature (and complex). Driven by the telecommunication sector, such MANO solutions leverage the advances of the virtualization and containerization technology to create a more cost-efficient and flexible approach for the deployment and management of VNFs and NSs on top of commodity hardware.

For instance, OSM, one of the most popular community-driven solutions, is an ETSI-hosted project used to model and orchestrate NSs with the support for different VIMs. In the proposed XR platform, these MANO tools can fit the XR Service Deployment Plane where distinct domains (and implementations), both technological and administrative, might coexist. Moreover, this multidomain vision requires an integrated cross-domain resource component, such as an actual ETSI MANO instance with multidomain support or an additional tool, such as Apache Libcloud or Terraform. The cross-domain resource component allows abstracting the different infrastructure providers APIs and provides the required interoperability across distinct underlying environments. For instance, Apache Libcloud allows interacting with multiple popular cloud providers using a unified API, whilst Terraform

can be used to reassemble a single workflow to efficiently manage infrastructure and specify different component blueprints (i.e., compute instances, storage, and network) within distinct service providers. Additional tools such as the Openslice can also have an important role into the onboarding and management of NSs and VNFs.

On the other hand, the implementation and management of a virtualization infrastructure is a great challenge. It requires a comprehensive approach to stitch together the ephemeral and distributed large number of microservices. Multidomain use cases require such microservices to be deployed and located across multiple infrastructure providers. In the same way, the wide spread of diverse edge/cloud environments impose different challenges pertaining to the automation and optimization of the overall orchestration process, the needed observability over the infrastructure (both north-south and east-west network traffic), and the security and privacy enforcement of the XR services across these hybrid edge/cloud environments.

In that regard, Kubernetes have become a de facto platform to support the orchestration of microservices within a Cloud-Native environment. Kubernetes, a cloud-native computing foundation (CNCF) graduated project and supported by all the major cloud providers, is today a popular choice for the deployment, automation, and management of container-based services and applications. Indeed, many distinct Kubernetes distributions (and platforms built on the top of Kubernetes) exist like OpenShift, AWS Elastic Kubernetes Service, Google Kubernetes Engine, Azure Kubernetes Service, or even Kubernetes-based platforms designed with lightweight environments in mind, such as Rancher, K3S, Microk8s, or KubeEdge. The later ones are more focused on resource-constrained deployments, such as local or edge domains. Thus, in the proposed XR platform, Kubernetes can be explored to support the container-based workloads of the envisioned next-generation applications. Additionally, tools such as KubeVirt can be also considered to address the inevitable transitioning from classical VM-based workloads through a common platform.

The myriad of integration approaches and technology options generates a side effect which results in highly complex and heterogeneous environments, especially when considering edge/cloud environments, as mentioned before. Within the proposed XR platform, this means the possibility to provide not only the requiring stitching between the provisioned XR services but also an efficient integration and communication among all the components of all planes and domains, including the Domain-Specific XR Service Monitoring and Reaction and the XR Service E2E Conducting planes.

For Cloud-Native environments, one feasible path is to leverage the concept of Service Mesh along with different integration fabrics (i.e., the domain and cross-domain integration fabrics and the Conductor Integration Fabric). The underlying Service Mesh concept relies on the usage of network proxies (the so-called sidecars), together with each service (or using different arrangements such as one proxy per node). Then, those network proxies allow to observe, control and implement security features across all the network traffic between components. While this provides a more unified approach to manage network communications within a Cloud-Native environment, it also brings additional latency (i.e., the network traffic needs to go through the proxies) and complexity (i.e., they also need orchestration). Nevertheless, the Service Mesh and the integration fabrics concepts might provide a more flexible communication strategy to support the network communications among all the XR platform elements.

Numerous Service Mesh implementations have emerged in the past years, such as Istio, NS Mesh, Linkerd, Consul, Traefik Mesh, Open Service Mesh, or GlooMesh. Most of these Service Mesh implementations rely on the Envoy sidecar implementation to realize the Service Mesh concept. Still, some of them address different use cases like multicluster management or support different network protocols. For instance, Istio, one of the most widely used Service Mesh implementations, works at layers 4–7. Amongst others, Istio provides service discovery, traffic management capabilities, traffic encryption between services, observability over the communications, and built-in access control mechanisms. Moreover, Istio and Envoy can also delegate the access control to external policy enforcement tools such as the Open Policy Agent. Rather than network focused, such an approach might turn into a more comprehensive approach for applying different policies across distinct components of the XR platform by leveraging a common policy-driven strategy and syntax. On the other hand, NS Mesh, another Service Mesh implementation, working at layers 2 and 3, is a more recent attempt to support an additional range of use cases and network protocols.

Moreover, given the ever-growing Service Mesh ecosystem, a standard Service Mesh interface was proposed to unify the consumption of Service Meshes APIs across different implementations. Such a standard interface plays a relevant role to allow multidomain setups, as discussed before. In the same way, specific tools, such as the GlooMesh, also address the problem of integrating multiple and different domains through a single management interface on top of already existing distinct Service Mesh environments. On the other hand, given the additional hop in the network communications, the Service Mesh concept implies a performance penalty in the already strict requirements of an XR service. Therefore, aside from the functionalities, the efficiency of different Service Mesh implementations needs to be considered for the full realization of the proposed XR platform.

The monitoring and reaction plane and the notion of closed automated loops are critical characteristics that shall be part of the XR platform. These loops, as discussed before, rely on the ability to collect and process different kinds of data as input for the analytics logic. Such data collection, which shall occur near real time, is essential to understand the services provided by the XR platform, the network communications, and the infrastructure (e.g., edge/cloud resources) where the various components run. The Service Mesh concept can also support such monitoring by providing service and network-related insights to the monitoring and reaction plane. Additional monitoring-related tools, such as Prometheus, can be used to gather insights from the different components in a more comprehensive way. Prometheus is a widely used monitoring solution for collecting and storing metrics from third-party

systems, supported by a growing number of plugins. In the proposed XR platform, Prometheus can be leveraged and integrated with the Service Mesh sidecars up to the integration fabric components. Other solutions, such as the ElasticSearch stack, more focused on logs can also fit the overall monitoring approach.

Likewise, for the feasibility of the integration fabric concept (which comprises the messaging bus), multiple tools and open-source components, such as Apache Kafka, RabbitMQ, or ActiveMQ, amongst many others, can be leveraged. For instance, Apache Kafka, one of the most widely used messaging systems, has the notion of a distributed set of messaging brokers organized within elastic clusters. Kafka brokers are used to storing and serving messages. Those messages, organized by topics and partitions, can then be partitioned and replicated, respectively, ensuring different service levels of performance, durability, or fault tolerance, and that is according to the needs of each use case. In the proposed XR platform, Kafka has a decisive role in providing asynchronous bus communication channels to interconnect the elements within each domain and all the XR platform domains. In this way, Kafka messages can be used to model all the communication functionalities (e.g., service registration, discovery, and monitoring messages) and facilitate the integration with external systems. As a reference example, OSM follows a similar approach, whereby Kafka has the role of messaging bus for OSM components.

VII. Performance Evaluation

Whilst the main intention beneath this article is to demonstrate the feasibility of the proposed architecture in the real world and that is by showcasing: 1) how compliant it is to ongoing relevant standards and 2) what open sources and tools that can be used for each component of the architecture, in this section, we will carry out two sets of experimentation to partially demonstrate that the architecture is effectively implementable. In the first experiment, we present some low-level implementation details about the proof of concept showing how the infrastructure would be constructed. We particularly address the cross-domain orchestration and monitoring issues. In the second experiment, we propose a smart mechanism that minimizes resource usage while maximizing the QoS. This mechanism is realized in a closed-loop fashion with all of: 1) the monitoring agents; 2) the analytical engine (i.e., prediction mechanism) that consumes monitored data; and 3) a decision engine that decides when to ask for new resources.

A. Proof of Concept—Application Deployment Time and Resource Usage Monitoring

Continuous monitoring can be useful to minimize the response time to incidents and to guarantee that the applications and the infrastructure behave as predicted. Namely, tracking cluster resources, such as memory, CPU, storage, and bandwidth, facilitates the process of managing cloud-native environments. As discussed before, through specific monitoring agents, those monitoring capabilities, are an integral part of closed-loops processes. With that in mind, hereunder, we conduct an experiment to demonstrate the monitoring capabilities on a Cloud Native deployment of the proposed architecture.

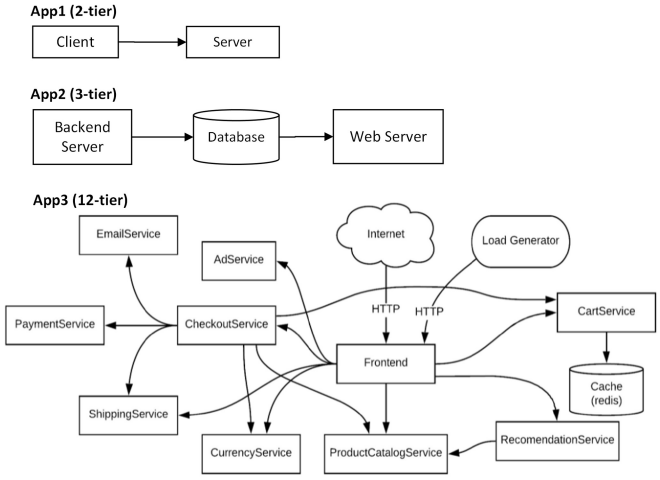


Fig. 4. Topology of applications, adapted from [58].

Considering a scenario whereby an XR application is deployed on multiple domains (i.e., having its microservices deployed on different Kubernetes clusters) through the Cross Domain Resource MANO, the Rancher tool was used in the experiment taking into account its multicloud provider support. Rancher enables the creation and orchestration of multiple Kubernetes clusters, through a cluster agent that is installed in all Kubernetes nodes. By having the support for different cloud providers, Rancher can intermediate and facilitate the orchestration of different domains. In the conducted experiment, Rancher is used to set up a two-node Kubernetes cluster that is used to deploy and monitor distinct micro-service-based applications. Additionally, a Prometheus and a Grafana installation are performed. This provides a simple and efficient way to visualize several natively supported clusters and pod-specific metrics. By leveraging this monitoring approach, and in particular the Prometheus tool, additional XR-specific instrumentation would be achieved by having additional libraries and Prometheus Exporters to expose virtually all kinds of XR-related metrics

In order to thoroughly evaluate monitoring capabilities, three different (i.e., topology-wise and purpose-wise) applications were deployed. A 2-tier application fulfils the objective of generating effective and realistic network traffic and resource usage, with the use of *iperf3* and *stress-ng*. A 3-tier application serves the purpose of providing a simple-yet-realistic standard architecture as a starting point for demonstration purposes. A 12-tier application consists of a Web-based e-commerce application to showcase a complex and realistic application. These different applications were chosen as reference scenarios of how next-generation XR applications, composed of numerous microservices and different topologies as shown in Fig. 4, can be effectively monitored on a cloud-native environment and their orchestration supporting a new wave of prediction, scheduling, and intelligent orchestration mechanisms.

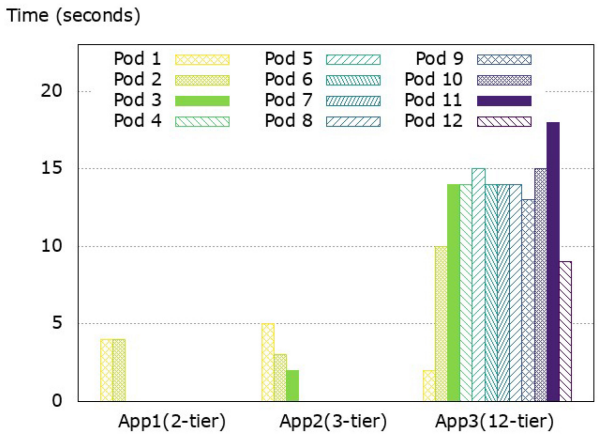


Fig. 5. Average pod deployment time of each application.

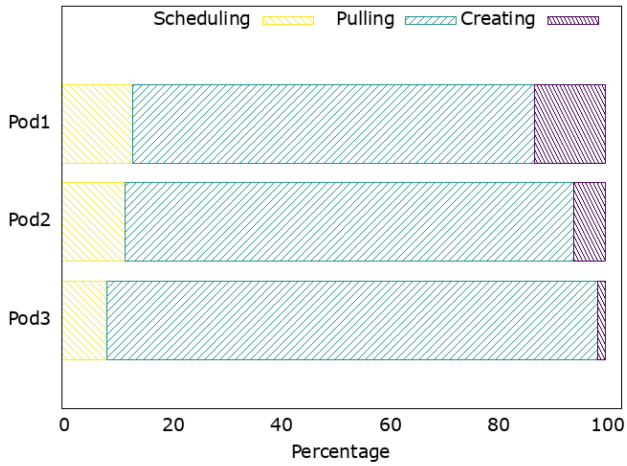


Fig. 6. Stages included in deployment time (3-tier application case).

Availability is a critical factor when providing XR services in cloud-native environments, since downtime (e.g., due to service migrations) can negatively impact overall user experience. To this extent, it is of the utmost importance to comprise a mechanism for measuring application deployment time (i.e., the time it takes from creation to proper functioning), in order to properly assess and regulate performance. Also, we highlight the fact that this deployment time, we commonly mention, does not account for service availability, i.e., we measure the time until the pod is running, not until the service itself is accessible and fully operational (e.g., some databases need time to migrate and web servers need time to initialize).

In order to evaluate the deployment time of the aforementioned applications, a component that calculates the deployment time based on events registered from Kubernetes was developed, as out-of-the-box metrics reported by Kubernetes (i.e., kube-state-metrics) neither provide such a mechanism nor account for time spent pulling a container image. Fig. 5 shows these time periods per pod and per application.

The observed values represent the average of five tests. The deployment time of an application as a whole is obtained by taking the maximum deployment time of its pods (i.e., the pod that took the longest time to be deployed). Values registered in Fig. 5 can be explained by the increasing complexity of each application. Indeed, the more tiers, the longer it takes to deploy an application. Additionally, in these tests, the images were not pulled, since this process was already performed in the first tests, and thus the images were present locally.

The first application has only two pods and does not show relevant differences for deployment time between the two pods. The second one has three pods, which depend on each other, and this explains the differences between each pod's deployment times. The last one contemplates the same reasoning for the differences in pod deployment time, since it has 11 pods, with dependencies among them. Such dependencies between services in cloud-native applications often exist and might have an impact on numerous operations (e.g., service migration and scaling). Microservice-based XR applications are not expected to be different. Indeed, they are expected to have complex topologies and numerous dynamic constraints. Thus, their management in (near) real time is a fundamental aspect of the envisioned orchestration.

In order to understand the discrepancy in deployment times across different services and applications, it is important to dissect the various stages of the deployment process. This process includes scheduling the pods, pulling the container(s) image(s), and finally creating and starting it. Fig. 6 represents the three services from application 2 (3-tier) and their deployment times. Even though the pulling stage depends on the size of the image, it is the stage that takes the longest. In order to counteract the image pull time, pods can be configured in such a way that the pull is only performed if the image is not present locally (cached) on the node bound to the pod. In this case, the first time the deployment was performed, the images were not cached and therefore were pulled, whereas subsequent deployments of the same pods did not need to pull the container(s) image(s), thus realizing a short deployment time

Nevertheless, it is important to debate how such an approach reflects on fast-paced and rapidly developing environments, as it raises a multitude of questions regarding node configuration and cluster management. Maintaining local images across nodes becomes ever more difficult when dealing with multinode, highly complex cloud-native environments. Prioritizing deployment time at the expense of complexity is a matter that demands its proper evaluation, and understanding its impact is crucial in order to properly adapt cloud-native environments to specific use cases. To get around this problem, one of the possibilities is the use of smart caching techniques. The use of these techniques during orchestration enables the prefetch of the required images of XR services and places them in the nodes or in close vicinity of the nodes.

In addition to measuring and analyzing deployment time, it is also important to evaluate and analyze mechanisms that allow for the visualization of resources (e.g., CPU and memory usage) for predictive behavior of both the application and infrastructure and thus take proactive actions to ensure the effectiveness and efficiency of the platform. In this sense, Grafana provides out-of-the-box dashboards to visualize such metric, as Figs. 7 and 8 portray.

The graphics show the CPU usage and the memory usage of the deployed 12-tier application. This application has different pods, each with different functions, which explains why different containers use the memory differently, depending on

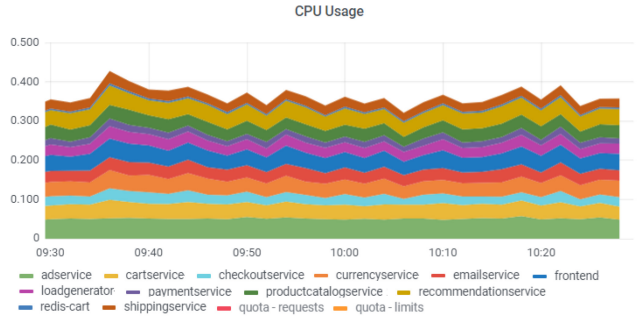


Fig. 7. CPU usage graph in case of the 12-tier application.

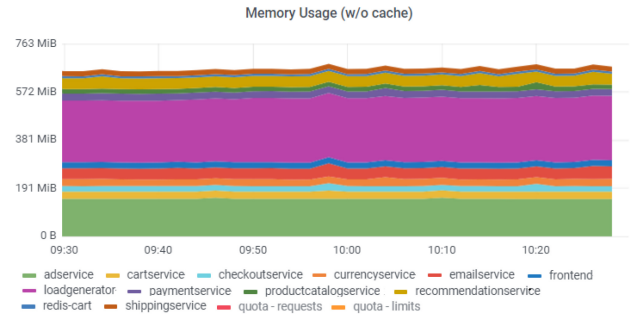


Fig. 8. Memory usage graph in case of the 12-tier application.

their function. Likewise, a multiuser XR application might behave differently according to the environment, the number of instances, users or even different settings of each user. Monitoring the resource usage at the pod level (or at the application level) as part of the closed-loop mechanims, is useful to detect early deviations that might indicate unhealthy situations or be used to predict individual service behaviors.

B. Closed-Loop Example

In this experiment, we demonstrate the operation of the domain automation loops by introducing a proactive autoscaler in the "XR Service Specific Decision Engine" component interface of the architecture. The key concept is to showcase the benefits of incorporating AI in the domain automation loops, when considering requirements that are characteristic of XR applications, such as latency and throughput.

Off-the-shelf autoscalers for Kubernetes-like nodes are operating mainly based on compute resource utilization metrics. As such, they are detached from the application-specific characteristics that are more directly linked to the quality provisioning levels in comparison to, e.g., CPU and memory utilization. XR application workflows are such an example, in which scaling strategies should also consider parameters that are related to the actual workload of the application. Furthermore, reactive autoscaling suffers from the inherent problem of the runtime deployment overhead. Depending on the technology, scaling operations (in-out or up-down) may introduce an overhead in time or costs that is suboptimal. This is particularly relevant in the XR-application workflows, where the application components are provided as large VMs (sometime in the order of dozens of GBs). On the other hand, proactive scaling mechanisms formulate decisions based on predictions that are created by a dedicated predictive mechanism. Given that each type of computational node requires a different time to complete its scaling operations, it is quite useful to perform multistep prediction to have access to a wider range of information regarding the expected state of the resources.

Toward that end, we employed the proposed architecture that enabled the out-of-the box integration of a model for proactive autoscaling that considers a richer state space than mere runtime compute utilization. The model is based on a novel multistep Deep-Learning prediction mechanism [59] that was originally used to predict network traffic. We conducted an experiment to evaluate this mechanism against a typical reactive autoscaler. The results have shown that the architecture facilitated the smooth integration of the model as a plug-in, maintaining its generic mechanisms for monitoring, analysis, and actuation.

The experiment is comprised of the following steps.

1) A reference implementation of the platform architecture domain automation loops is created using the CloudSim Plus[9] environment. The simulation itself is based on typical requirements of an XR application. The workloads produced closely resemble the ones that are associated with XR applications. Furthermore, information regarding these workloads is extracted from dedicated load-balancer entities that are in charge of properly distributing the workloads among the available computational nodes.
2) A proactive autoscaling model is trained based on the characteristics of the XR application. The model is integrated in the reference implementation to instantiate the logic of the XR Service Specific Data Analytics and Decision Engines. The performance of that system in terms of latency, throughput, and computational cost is then compared against a baseline reactive autoscaler. In both cases, the action of scaling up or down is conducted by the XR Service Specific Actuation Engine.

To evaluate a proactive autoscaler as part of a closed-control loop, described in the proposed architecture, we conducted a large-scale simulation using the CloudSim Plus environment. The duration of the simulation was one week and it included almost two million tasks that were produced and sent to the available computational nodes to be processed. The task production rate was based on multiple Gaussian probability distributions in order to simulate various typical periodic patterns in task production, as well as numerous sudden bursts in task production that are statistically likely to happen. The task distribution among the available computational nodes was conducted using a standard Round Robbin algorithmic approach. The simulation is based on two distinct time cycles. The first one is the production of tasks that take place once every second and the second one is the measurement of the tasks that were sent to each computational node during the last minute. The decision to scale up or down is being made once per minute independently for each computational node based on the measurement that corresponds to its number of tasks. During our experiments there were five computational nodes that were always operational and fifteen more that could be allocated if the scaling mechanism decided to do so. It was decided that the time that is required for the deployment of

[9]https://cloudsimplus.org/

TABLE I
EVALUATION OF STANDARD REACTIVE AND INTELLIGENT PROACTIVE SCALING MECHANISMS

	Standard Reactive	Intelligent Proactive
Avg Overall Latency (s)	3.141	0.609
Avg Latency (s)	5.181	0.712
Avg Throughput (tasks)	9489	15219
Avg Resource Usage (VMs)	9.71	10.31

a new computational node should be set to 5 min in order to be similar to the one that is required by VMs. After extensive research on various use cases, it was concluded that each computational node could handle at most an average of 50 tasks per minute, without showing any signs of deterioration in its performance. Thus, an upper-bound threshold of 40 tasks per minute was established. During the experimental evaluation, two scenarios were explored. During the first one, a standard reactive scaling mechanism was used. The reactive scaling mechanism is designed to allocate additional computational resources every time one of the working computational nodes receives more than 40 tasks per minute. On the other hand, in order to avoid over-provisioning of resources, each time a computational node receives less than ten tasks, this specific computational node is deallocated. A computational node can only be decommissioned, after all the tasks that were sent to it, have been fully processed. The second scenario is based on the use of an intelligent proactive scaling mechanism. The intelligent proactive mechanism operates in a similar manner with the exception that the scaling decisions are being made based on the predictions that are being produced by our Deep Learning-based prediction mechanism. Each computational node has a dedicated prediction mechanism that receives as input the last six measurements of tasks and produces a multistep prediction that corresponds to the next six 1-min time steps. At this case, we are only interested in the last time-step of the prediction since it is vital to gain knowledge regarding the number of tasks that shall be produced in a time horizon that surpasses the 5-min mark that is required to deploy a new computational node.

When comparing the two experimental scenarios, the results shown in Table I are obtained. The results are as follows.

1) The overall average latency that corresponds to each task was 3.141-s during the reactive scenario and 0.609 s when using the intelligent proactive scaling architecture. That shows an improvement of about 515%.
2) The average latency of each task during the 200 time steps when the most violent spikes in task production took place was 0.712 s for the proactive scenario and 5.181 s for the reactive one. Once again, that shows an impressive improvement of 728%. Furthermore, these results show that during times of extreme increase in task production, the ability of the intelligent proactive scaling mechanism to guarantee a comparatively low latency is reduced by about 15%, while the ability of the reactive scaling mechanism drops by a significant 165%.
3) In terms of throughput during the ten-time steps when the most violent spikes in task production took place, the intelligent proactive scaling mechanism and the reactive scaling mechanism managed to complete 15219 tasks and 9489 tasks, respectively. That means that the incorporation of AI managed to improve the throughput metric as well by a margin of about 160%.
4) All these significant improvements in the performance come at a cost. Thankfully, the incorporation of AI managed to keep this cost to a minimum. During the simulation, the proactive scaling mechanism utilized 6% more computational resources. More specifically, the average allocation of computational nodes during the entire simulation was 9.71 VMs for the reactive scaling mechanism and 10.31 VMs for the proactive scaling mechanism.

When taking all these factors into consideration, it becomes apparent that the incorporation of AI methodologies in closed-control loops, such as the ones explored within the context of this article can provide substantial benefits in terms of reduced latency and increased throughput. Both these metrics are extremely important in the context of XR applications.

VIII. CONCLUSION AND FUTURE WORKS

There are clear signals that in the near future, XR applications will challenge the computing and communication infrastructures requiring an unprecedented level of QoE, definitely beyond the one that can be achieved with nowadays technologies. This work proposes an architecture that puts together the key enablers to support the challenges for the support of future XR applications, considering the shortcomings of existing technologies and the ongoing innovations. The design of the proposed XR platform follows the approach at the basis of the ZSM framework to create self-managed E2E network slices. Such a platform is organized around three planes: 1) the deployment plane (i.e., focused on the management of the compute, network, and storage resources); 2) the domain-specific monitoring and reaction plane (i.e., responsible for monitoring the service inside a technological or administrative domain); and 3) the E2E conducting plane (i.e., working at the level of services, thus aimed at creating subslices in each domain and monitoring the E2E KPIs of the XR services or shifting the services between the different domains, when necessary).

The proposed architecture is described and contextualized on XR applications by presenting specific use-case scenarios dealing with service launch and runtime management. This article also demonstrated the feasibility of the proposed architecture by mapping the envisioned functionality to existing tools and open sources. Furthermore, this article demonstrated that the architecture was effectively implementable through two sets of experiments; the first showing the deployment times of different XR applications on a multidomain cloud-native environment along with the necessary monitoring capabilities, and the second evaluating, based on simulations, a proactive resource autoscaler mechanism as part of an E2E closed loop of the envisioned architecture.

In the future, the plan is to validate the overall proposed architecture by setting up a test bed, designed, and implemented based on a selection of the technologies mentioned above. This will provide the opportunity to validate and evaluate the approach and, possibly, to improve it on the basis of the results achieved.

REFERENCES

[1] K. Boos, D. Chu, and E. Cuervo, "Flashback: Immersive virtual reality on mobile devices via rendering memoization," in *Proc. 14th Annu. Int. Conf. Mobile Syst., Appl., Services*, New York, NY, USA, 2016, pp. 291–304.

[2] C. Chaccour, M. N. Soorki, W. Saad, M. Bennis, and P. Popovski, "Can terahertz provide high-rate reliable low latency communications for wireless VR?," 2021, *arXiv:2005.00536*.

[3] V. Raghavan and J. Li, "Evolution of physical-layer communications research in the post-5G era," *IEEE Access*, vol. 7, pp. 10392–10401, 2019.

[4] W. Khalid, H. Yu, R. Ali, and R. Ullah, "Advanced physical-layer technologies for beyond 5G wireless communication networks," *Sensors*, vol. 21, no. 9, p. 3197, 2021.

[5] S. K. Sharma, I. Woungang, A. Anpalagan, and S. Chatzinotas, "Toward tactile Internet in beyond 5G era: Recent advances, current issues, and future directions," *IEEE Access*, vol. 8, pp. 56948–56991, 2020.

[6] M. Maier, M. Chowdhury, B. P. Rimal, and D. P. Van, "The tactile Internet: Vision, recent progress, and open challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 138–145, May 2016.

[7] G. Fettweis et al., *The Tactile Internet-ITU-T Technology Watch Report*, Int. Telecom. Union, Geneva, Switzerland, 2014.

[8] Z. S. Bojkovic, B. M. Bakmaz, and M. R. Bakmaz, "Vision and enabling technologies of tactile Internet realization," in *Proc. 13th Int. Conf. Adv. Technol., Syst. Services Telecommun. (TELSIKS)*, 2017, pp. 113–118.

[9] R. Li, "Towards a new Internet for the year 2030 and beyond," in *Proc. 3rd Annu. ITU IMT-2020/5G Workshop Demo Day*, 2018, pp. 1–21.

[10] Z. Chen, C. Wang, G. Li, Z. Lou, S. Jiang, and A. Galis, "New IP framework and protocol for future applications," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2020, pp. 1–5.

[11] R. Li et al., "New IP: A data packet framework to evolve the Internet," in *Proc. IEEE HPRS Workshop New IP*, May 2020, pp. 1–28.

[12] Y.-Y. Shih, W.-H. Chung, A.-C. Pang, T.-C. Chiu, and H.-Y. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE Netw.*, vol. 31, no. 1, pp. 52–58, Jan./Feb. 2017.

[13] A. Checko et al., "Cloud RAN for mobile networks—A technology overview," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 405–426, 1st Quart., 2015.

[14] R. Abhishek, "A collaborative edge-cloud framework for XR applications," Internet-Draft draft-abhishek-coin-xr-edge-cloud-00, Internet Eng. Task Force, Fremont, CA, USA, Apr. 2021.

[15] W. Courtney et al., "An expedited forwarding PHB (per-hop behavior)," IETF, RFC 3246, Mar. 2002.

[16] B. Briscoe, K. D. Schepper, and M. Bagnulo, "Low latency, low loss, scalable throughput (L4S) Internet service: Architecture," Internet-Draft draft-ietf-tsvwg-l4s-arch-03, Internet Eng. Task Force, Fremont, CA, USA, Oct. 2018.

[17] S. Floyd, "Highspeed TCP for large congestion windows," IETF, RFC 3649, 2003.

[18] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, 'Mobilityfirst: A robust and trustworthy mobility-centric architecture for the future Internet," *SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 16, no. 3, pp. 2–13, Dec. 2012.

[19] W. Ayoub, A. E. Samhat, F. Nouvel, M. Mroue, and J.-C. Prévotet, "Internet of Mobile Things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs standards and supported mobility," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1561–1581, 2nd Quart., 2018.

[20] M. Mohammadkarimi, M. A. Raza, and O. A. Dobre, "Signature-based nonorthogonal massive multiple access for future wireless networks: Uplink massive connectivity for machine-type communications," *IEEE Veh. Technol. Mag.*, vol. 13, no. 4, pp. 40–50, Dec. 2018.

[21] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 369–382, Apr.–Jun. 2019.

[22] X. Sun and N. Ansari, "Primal: Profit maximization avatar placement for mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2016, pp. 1–6.

[23] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, Sep. 2015.

[24] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "On enabling 5G automotive systems using follow me edge-cloud concept," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5302–5316, Jun. 2018.

[25] D. Gonçalves, K. Velasquez, M. Curado, L. Bittencourt, and E. Madeira, "Proactive virtual machine migration in fog environments," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2018, pp. 742–745.

[26] B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran, "MIGCEP: Operator migration for mobility driven distributed complex event processing," in *Proc. 7th ACM Int. Conf. Distrib. Event-Based Syst.*, New York, NY, USA, 2013, pp. 183–194.

[27] J. Violos, E. Psomakelis, K. Tserpes, F. Aisopos, and T. Varvarigou, "Leveraging user mobility and mobile app services behavior for optimal edge resource utilization," in *Proc. Int. Conf. Omni-Layer Intell. Syst.*, New York, NY, USA, 2019, pp. 7–12.

[28] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. IEEE 27th Annu. Int. Symp. Personal, Indoor, Mobile Radio Commun. (PIMRC)*, 2016, pp. 1–6.

[29] A. S. Gomes et al., "Edge caching with mobility prediction in virtualized LTE mobile networks," *Future Gener. Comput. Syst.*, vol. 70, pp. 148–162, May 2017.

[30] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 134–144, May 2019.

[31] N. Paper, "Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action," in *Proc. SDN OpenFlow World Congr.*, 2012, p. 73.

[32] M. Ersue, "ETSI NFV management and orchestration—An overview," presentation at the IETF, 2013.

[33] "Network functions virtualisation (NFV) release 4; Management and orchestration; requirements for service interfaces and object model for OS container management and orchestration specification," ETSI, Sophia Antipolis, France, document GS NFV-IFA 040, V4.1.1, Nov. 2020.

[34] S. Natarajan, A. Ghanwani, D. Krishnaswamy, R. Krishnan, P. Willis, and A. Chaudhary, "An analysis of container-based platforms for NFV," IETF draft slides-94-nfvrg-10-1, IETF, Fremont, CA, USA, Apr. 2016.

[35] T. Taleb, I. Afolabi, K Samdanis, and F. Z. Yousaf, "On multi-domain network slicing orchestration architecture & federated resource control," *IEEE Netw.*, vol. 33, no. 5, pp. 242–252, Sep. 2019.

[36] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, 1st Quart., 2015.

[37] M. Karakus and A. Durresi, "Quality of service (QoS) in software defined networking (SDN): A survey," *J. Netw. Comput. Appl.*, vol. 80, pp. 200–218, Feb. 2017.

[38] C. Benzaid and T. Taleb, "AI-driven zero touch network and service management in 5G and beyond: Challenges and research directions," *IEEE Netw.*, vol. 34, no. 2, pp. 186–194, Mar./Apr. 2020.

[39] Y. Wang et al., "Network management and orchestration using artificial intelligence: Overview of ETSI ENI," *IEEE Commun. Stand. Mag.*, vol. 2, no. 4, pp. 58–65, Dec. 2018.

[40] I. Korontanis et al., "Inter-operability and orchestration in heterogeneous cloud/edge resources: The accordion vision," in *Proc. 1st Workshop Flexible Resource Appl. Manage. Edge*, 2020, pp. 9–14.

[41] "Network functions virtualisation (NFV) release 3; architecture; report on the enhancements of the NFV architecture towards 'Cloud-native' and 'PaaS,"' ETSI, Sophia Antipolis, France, document GR NFV-IFA 029, V3.3.1, Nov. 2019.

[42] M. Bagaa, T. Taleb, J. Riekki, and J. Song, "Collaborative cross system AI: Toward 5G system and beyond," *IEEE Netw.*, vol. 35, no. 4, pp. 286–294, Jul./Aug. 2021.

[43] C. Benzaid, T. Taleb, C. T. Phan, C. Tselios, and G. Tsolis, "Distributed AI-based security for massive numbers of network slices in 5G & beyond mobile systems," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit)*, Jun. 2021, pp. 401–406.

[44] R. A. Addad, T. Taleb, H. Flinck, M. Bagaa, and D. L. C. Dutra, "Network slice mobility in next generation mobile systems: Challenges and potential solutions," *IEEE Netw.*, vol. 34, no. 1, pp. 84–93, Jan./Feb. 2020.

[45] "Technical specification group services and system aspects; virtual reality (VR) media services over 3GPP, version 16.0.0," 3GPP, Sophia Antipolis, France, Rep. 26.918, Dec. 2018.

[46] "O-RAN use cases and deployment scenarios," ORAN Alliance, Alfter, Germany, White Paper, 2020.

[47] Z. Nadir, T. Taleb, H. Flinck, O. Bouachir, and M. Bagaa, "Immersive services over 5G and beyond mobile systems," *IEEE Netw. Mag.*, vol. 35, no. 6, pp. 299–306, Nov./Dec. 2021.

[48] T. Taleb, Z. Nadir, H. Flinck, and J. Song, "Extremely-interactive and low latency services in 5G and beyond mobile systems," *IEEE Commun. Stand. Mag.*, vol. 5, no. 2, pp. 114–119, Jun. 2021.

[49] J. Prados-Garzon and T. Taleb, "Asynchronous time-sensitive networking for 5G backhauling," *IEEE Netw.*, vol. 35, no. 2, pp. 144–151, Mar./Apr. 2021.

[50] J. Prados-Garzon, T. Taleb, and M. Bagaa, "Optimization of flow allocation in asynchronous deterministic 5G transport networks by leveraging data analytics," *IEEE Trans. Mobile Comput.*, early access, Jul. 26, 2021, doi: 10.1109/TMC.2021.3099979.
[51] N. Finn, P. Thubert, B. Varga, and J. Farkas, "Deterministic networking architecture," Internet-Draft draft-ietf-detnet-architecture-03, IETF, Fremont, CA, USA, 2017.
[52] A. Nait Abbou, T. Taleb, and J. Song, "A software-defined queuing framework for QoS provisioning in 5G and beyond mobile systems," *IEEE Netw.*, vol. 35, no. 2, pp. 168–173, Mar./Apr. 2021.
[53] *IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks*, IEEE Standard 802.1, 2018.
[54] M. Chen, X. Geng, and Z. Li, "Segment routing (SR) based bounded latency," Internet-Draft draft-chen-detnet-sr-based-bounded-latency-01, Internet Eng. Task Force, Fremont, CA, USA, May 2019.
[55] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," IETF, RFC 6182, 2011.
[56] F. Chiariotti, S. Kucera, A. Zanella, and H. Claussen, "Leap: A latency control protocol for multi-path data delivery with pre-defined QoS guarantees," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2018, pp. 166–171.
[57] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.
[58] GoogleCloudPlatform. "Github repository—googlecloudplatform/microservices-demo." 2021. [Online]. Available: https://github.com/GoogleCloudPlatform/microservices-demo
[59] T. Theodoropoulos, A.-C. Maroudis, J. Violos, and K. Tserpes, "An encoder-decoder deep learning approach for multistep service traffic prediction," in *Proc. IEEE 7th Int. Conf. Big Data Comput. Service Appl. (BigDataService)*, 2021, pp. 33–40.

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 Professor with the Center of Wireless Communications, The University of Oulu, Oulu, Finland. He is the Founder and the Director of MOSA!C Lab, Espoo, Finland. From October 2014 and December 2021, he was a Professor with the School of Electrical Engineering, Aalto University, Espoo. Prior to that, he was working as a Senior Researcher and 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 October 2005 till March 2006, he worked as a Research Fellow with the Intelligent Cosmos Research Institute, Sendai. He has been also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group 2. His research interests lie in the field of telco cloud, network softwarization and network slicing, AI-based software-defined security, immersive communications, mobile multimedia streaming, and next-generation mobile networking.

Prof. Taleb is the recipient of the 2021 IEEE ComSoc Wireless Communications Technical Committee Recognition Award in December 2021, the 2017 IEEE ComSoc Communications Software Technical Achievement Award in December 2017 for his outstanding contributions to network softwarization. He is also the (co-) recipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize in May 2017, and many other awards from Japan. Some of his research work have been also awarded best paper awards at prestigious IEEE-flagged conferences. He served as the General Chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference (WCNC'19) held in Marrakesh, Morocco. He was the Guest Editor in Chief of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Series on Network Softwarization and Enablers. He served on the editorial board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, *IEEE Wireless Communications Magazine*, IEEE JOURNAL ON INTERNET OF THINGS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, and a number of Wiley journals. Till December 2016, he served as the Chair of the Wireless Communications Technical Committee, the largest in IEEE ComSoC. He served on the IEEE Communications Society Standardization Program Development Board.

Abderrahmane Boudi received the M.Sc. and Ph.D. degrees from the Higher National School of Computer Science, Algiers, Algeria, in 2013 and 2020, respectively.

He is currently an Assistant Professor with the École Nationale Supérieure d'Informatique, Algiers, and also a Researcher with ICTFICIAL Oy, Espoo, Finland, working on several projects on XR, AI, and digital twinning. Prior to this, he was a Research Assistant with MOSA!C Lab/AALTO, Espoo, in 2017 and 2020. His research interests include computing in the edge/cloud continuum, intelligent feedback control systems, SDN, and 5G and beyond networks.

Luis Rosa received the Ph.D. degree in information science and technology from the University of Coimbra, Coimbra, Portugal, in 2021.

He has been involved as a researcher in various European research projects. He has more than ten publications in journals, conferences, and book chapters on those topics. His research interests include edge/cloud computing, networks, security, and critical infrastructure protection.

Luis Cordeiro received the M.Sc. degree in communications and telematics from the University of Coimbra, Coimbra, Portugal, in 2007.

He is a CTO with OneSource, Lisbon, Portugal. He has been actively involved in more than ten European research projects since 2004, mostly in the fields of networking, cloud, and security. He has several publications in journals, book chapters, and conferences. He has an extensive background in the areas of data communications, security, infrastructure management, and virtualization.

Theodoros Theodoropoulos received the Eng. Diploma degree from the School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece, in 2022. 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 two 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 and edge computing.

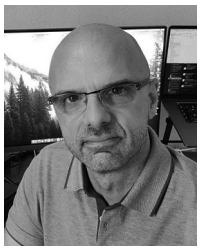
Konstantinos Tserpes received the Engineering Diploma degree from the Department of Computer Engineering and Informatics, School of Engineering, University of Patras, Patras, Greece, in 2003, and the M.Sc. degree in information systems and the Ph.D. degree in software engineering from the School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece, in 2005 and 2008, respectively.

He is an Associate Professor with the Department of Telematics and Informatics, Harokopio University of Athens, Kallithea, Greece. He has been involved in several EU- and Nationally-funded projects tackling research challenges in application domains, such as multimedia, e-governance, post-production, finance, and e-health. He has authored more than 100 publications in international scientific conferences and journals and has been cited multiple times by the work of his peers. His research interests revolve around intelligent methods to resolve resource allocation issues. Together with his team, they have applied those findings along with their software engineering skills in designing and implementing computing platforms enabling classes of novel applications with extraordinary requirements, including ultralow latency, big data stream processing, resource-constrained execution, real-time performance, resilient, and adaptive operation.

Patrizio Dazzi received the B.Sc. degree in computer science and the M.Sc. degree in computer technologies (computer systems and networks) degree from the University of Pisa, Pisa, Italy, in 2003 and 2004, respectively, and the Ph.D. degree in computer science and engineering from the IMT School for Advanced Studies Lucca, Lucca, Italy, in 2008.

He is a Researcher with a strong interest in high-performance distributed systems, particularly in models and algorithms for the decentralized management of computations and data. He is currently a Senior Researcher with the University of Pisa, Pisa, Italy, he is the Co-Founder and the Co-Leader of Pervasive AI Laboratory, a joint initiative of the University of Pisa and the National Research Council of Italy. Previously, he spent 17 years with the National Research Council, Rome, Italy, where he was a permanent researcher belonging to the High-Performance Computing Laboratory. He is active in the community of large distributed systems, he has been an organizer of conferences and workshops, and he has given speeches, and promoted and edited journal special issues. He has been the Scientific Coordinator of the EU-South Korea H2020 BASMATI project, he served as a Project Coordinator for the EU H2020 ACCORDION Project and he is currently serving as an Innovation Coordinator for the EU H2020 TEACHING Project. He has coauthored about 100 papers in international journals and conferences on topics related to High-Performance Distributed Systems. He has coauthored the book *Cloud Broker and Cloudlet for Workflow Scheduling* (Springer). All the activities conducted in the context of these projects are (or were) focused on the efficient management of large distributed computing infrastructures, leveraging either traditional optimization techniques as well as machine learning-based ones.

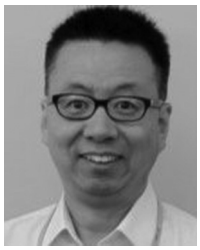
Dr. Dazzi served and serves as a program committee member in many conferences in the field of large, distributed systems, including IEEE ICDCS, IEEE CLOUD, ACM SoCC, EuroPar, ACM SIGKDD, ACM WSDM, and IEEE HPCS. He is a member of the Executive Committee of the IEEE Technical Committee on Cloud Computing. He has a long-lasting experience as a speaker in international venues.



Antonis I. Protopsaltis received the B.Sc. (Hons.) degree in computer science and the M.Sc. degree (Hons.) in computer science (software engineering) from Concordia University, Montreal, QC, Canada, in 1994 and 1996, respectively, and the Ph.D. degree in computer science from the University of Ioannina, Ioannina, Greece, in 2010.

He is a Computer Scientist specializing in computer graphics and computer-aided design (CAD). He is a Special Teaching Fellow of Computer Graphics with the Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani, Greece, where he is consulting the Gane Development Student Group. He has conducted research on virtual and augmented reality, and geometric modeling methods for reverse engineering of scanned objects using curves, surfaces, and discrete models, based on geometric constraints of cross-sections. He has actively been involved in several Greek or European funded research projects in these areas. His research interests include the areas of computer graphics, virtual/augmented/extended reality (VR/AR/XR), geometric modeling of 3-D objects, reverse engineering of 3-D objects, CAD, and scientific visualization.

Dr. Protopsaltis is a reviewer for international scientific journals of high impact. In his professional career, he has been employed as a consultant and a lead graphics programmer in the industrial sector.



Richard Li received the Ph.D. degree in mathematics with concentration in computer science and artificial intelligence from the Instituto Superior Tecnico, Universidade de Lisboa, Lisbon, Portugal, in 1993.

He is the Chief Scientist and the Vice President of Network Technologies at Futurewei, Santa Clara, CA, USA.

Dr. Li served as the Chairman of ITU-T FG Network 2030 from 2018 to 2020, and as the Vice Chairman of the Europe ETSI ISG Next-Generation Protocols (NGP) from 2016 to 2019. He has also served as the Co-Chair of steering committees and technical program committees of some academic and industrial conferences. He is extremely passionate about advancing ICT infrastructure technologies and solving problems in their entirety, thus creating a bigger and long-term impact on the networking industry. During his career, he spearheaded network technology innovation and development in routing and MPLS, mobile backhaul, metro and core networks, data center, cloud, and virtualization. He currently leads a team of scientists and engineers to develop technologies for next-generation network architectures, protocols, algorithms, and systems in the support of emerging and forward-looking applications and industry verticals in the context of New IP, Network 2030, and 5G/B5G/6G.