Multimedia Service Orchestration in Multi-tier Edge Computing Environments

Inaugural dissertation
of the Faculty of Science,
University of Bern

presented by
Hugo Leonardo Melo dos Santos
from Belém, Brazil

Supervisor of the doctoral thesis:
Professor Dr. Torsten Ingo Braun
Institut für Informatik, Universität Bern
Professor Dr. Eduardo Coelho Cerqueira
Institute of Technology, Federal University of Pará

Multimedia Service Orchestration in Multi-tier Edge Computing Environments

Inaugural dissertation
of the Faculty of Science,
University of Bern

presented by
Hugo Leonardo Melo dos Santos
from Belém, Brazil

Supervisor of the doctoral thesis:
Professor Dr. Torsten Ingo Braun
Institut für Informatik, Universität Bern
Professor Dr. Eduardo Coelho Cerqueira
Institute of Technology, Federal University of Pará

Accepted by the Faculty of Science.

Bern, May 2023

The Dean:
Prof. Dr. Marco Herwegh

FEDERAL UNIVERSITY OF PARÁ
INSTITUTE OF TECHNOLOGY
GRADUATE PROGRAM IN ELECTRICAL ENGINEERING

HUGO LEONARDO MELO DOS SANTOS

MULTIMEDIA SERVICE ORCHESTRATION IN MULTI-TIER EDGE COMPUTING ENVIRONMENTS

Belém

2023

HUGO LEONARDO MELO DOS SANTOS

MULTIMEDIA SERVICE ORCHESTRATION IN MULTI-TIER EDGE COMPUTING ENVIRONMENTS

Thesis presented to the Institute of Technology at Federal University of Pará in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering under joint-supervision agreement between Federal University of Pará and University of Bern.

Supervisor/Orientador: Prof. Dr. Eduardo Coelho Cerqueira
Co-supervisor/Coorientador: Prof. Dr. Torsten Ingo Braun

Belém

2023

HUGO LEONARDO MELO DOS SANTOS

MULTIMEDIA SERVICE ORCHESTRATION IN MULTI-TIER EDGE COMPUTING ENVIRONMENTS

Thesis presented to the Institute of Technology at Federal University of Pará in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering under joint-supervision agreement between Federal University of Pará and University of Bern.

Examining Committee

Prof. Dr. Eduardo Coelho Cerqueira
(Federal University of Pará - Supervisor)

Prof. Dr. Torsten Ingo Braun
(University of Bern - Co-supervisor)

Prof. Dr. Denis Lima do Rosario
(Federal University of Pará - Internal Member)

Prof. Dr. Marcos César da Rocha Seruffo
(Federal University of Pará - Internal Member)

Prof. Dr. Antonio Alfredo Ferreira Loureiro
(Federal University of Minas Gerais - External Member)

The minutes of the defense, which include the signatures of the members of the Examining Committee, are archived by the Federal University of Pará

I devote this work first to my family, fiancée and friends, especially my parents, Emanuel and Luiza, who supported me during my whole life and led to the outcome of this PhD.

Acknowledgement

I dedicate my sincere thanks to:

– To Professor Dr. Eduardo Cerqueira and Dr. Torsten Braun for the opportunity, guidance, encouragement, patience, knowledge and confidence placed in me.

– To Professor Dr. Denis Rosário, for the dedication and effort offered to make this thesis possible.

– The team of the Brazilian and Swiss universities, Lucas Bastos, Wellington Lobato, Pedro Cumino, Fábio Rocha, Felipe Rocha, Joahannes Costa, Diego Oliveira, Alisson Medeiros, Dimitris Xenakis, Negar Emami, Eric Samikwa, Jakob Schaerer, Jesutofunmi Ajayi, Mostafa Karimzadeh, Eirini Kalogeiton, Daniela S. Schroth, Priska Grunder, especially by the colleagues Iago Medeiros, Bruno Martins, Carlos Rocha, Lucas Pacheco, and Derian Sobral by the aid and talks in diverse moments.

"Invention is the most important product of man's creative brain. The ultimate purpose is the complete mastery of mind over the material world, the harnessing of human nature to human needs"

Nikola Tesla

Abstract

Multimedia Service Orchestration in Multi-tier Edge Computing Environments

Supervisor: Prof. Dr. Eduardo Coelho Cerqueira
Co-supervisor: Prof. Dr. Torsten Ingo Braun

High Definition Video on Demand (VoD) and immersive multimedia services, such as Virtual Reality (VR) and Augmented Reality (AR), have been attracting thousands of new users every day for digital spaces. Static and mobile users usually watch multimedia services during entertainment, retailing, gaming, training, traveling, advertising, and commuting activities. However, multimedia services have stringent requirements in terms of Quality of Service (QoS) and Quality of Experience (QoE) support for static and/or mobile users in heterogeneous cloud-based ecosystems. Thus, new efficient approaches for orchestrating networking and computing resources and distributing multimedia flows in multi-tier mobile edge computing environments are required. The orchestration schemes must be developed to choose suitable edge nodes to execute multimedia-based services while improving the usage of networking and computing resources and providing QoE/QoS support for static and/or mobile users. This thesis contributes to efficiently meeting VoD, VR, and AR services needs in multi-tier edge computing environments. The first contribution is a service orchestrator to define an appropriate edge node to stream VoD, considering QoE, QoS, and monetary costs. The second contribution is a mobile-aware orchestrator that cooperates with edge nodes on the ground space and selects and places flying edge nodes to improve multimedia flow distribution with QoS support in congestion or failure periods. The third contribution is a mobility-aware Service Function Chaining (SFC) orchestrator for static and mobile scenarios. The proposed scheme aims to map, instantiate, and re-instantiate Service Function (SF) into multiple edge nodes considering networking and computing parameters. The proposed solutions were widely compared to related works on different scenarios. The results show that the proposed approaches outperform the state-of-the-art schemes.

Contents

List of abbreviations

5G	Fifth Generation technology standard for cellular networks
ABR	Adaptive Bitrate
AHP	Analytic Hierarchy Process
AP	Access Point
AR	Augmented Reality
AWS	Amazon Web Services
BC	Betweenness Centrality
BE	Background virtual Environment
BS	Base Station
BBU	BaseBand Unit
CPU	Central Process Unit
D2D	Device-to-device
DASH	Dynamic Adaptive Streaming over HTTP
DRL	Deep Reinforcement Learning
E2E	end-to-end
ECS	Container Orchestration Service
FEC	Flying Edge Computing
FI	Foreground Interactions

FIBRE Future Internet Brazilian Environment for Experimentation

HD High Definition

HMD Head Mounted Display

HMG Head Mounted Glasses

HTTP Hypertext Transfer Protocol

IA Interaction

ISP Internet Service Provider

MEC Multi-access Edge Computing

MOS Mean Opinion Score

MPD Media Presentation Descriptor

MUAR Multi-User Augmented Reality

MUVR Multi-User Virtual Reality

NS-3 Network Simulator 3

PPC Partial Parallel Chaining

PDR Packet Delivery Ratio

PSO Particle Swarm Optimization

QoE Quality of Experience

QoS Quality of Service

RAN Radio Access Network

RTT Rount-Trip Time

SDN Software Defined Networking

SF Service Function

SFC Service Function Chaining

SINR Signal-to-Interference-plus-Noise Ratio

SLA Service Level Agreement

SPR Service Provisioning Rate

SSIM Structural Similarity Index Measure

TCP Transport Control Protocol

TOPSIS Technique for Order of Preference by Similarity to Ideal Solution

UAV Unmanned Aerial Vehicles

UHD Ultra High Definition

VO Virtual Object

VoD Video on Demand

VR Virtual Reality

XML Extensible Markup Language

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Overview

The Fifth Generation technology standard for cellular networks (5G) and Beyond 5G will enable a wide range of High Definition (HD) videos and immersive multimedia services, which might attract thousands of new customers every day [1, 2]. For instance, the worldwide revenue of US$94.9 billion of Video on Demand (VoD) market is expected to grow US$157.9 billion in 2027 [3]. Customers of VoD services can decide when and where to watch movies or TV shows with Ultra High Definition (UHD) with screen resolutions of 2K, 4K, and 8K. Furthermore, immersive multimedia services with Virtual Reality (VR) and Augmented Reality (AR) expect 2.6 million users by 2027 [4]. Among different sorts of business, the main immersive multimedia services enhance operations of retail, manufacturing, games and entertainment, training, tourism and travel, healthcare, and advertising & marketing [5, 6]. In this way, users' consumption behaviour happens with static users during daily activities at work, school, or home and for mobile users while commuting or moving around smart cities [7, 8, 9].

Users often consume multimedia content on mobile devices, which provide innovative features to display high-quality video content while shrinking the device size [10]. In this way, mobile devices improve ergonomic aspects to extend a more comfortable experience with lightweight, slim, and wireless connectivity. Specifically, VoD exhibition occurs on smartphones and laptops, but immersive multimedia services occur on Head Mounted Displays (HMDs). These HMDs may have local or cloud-based remote processing to allow users to manipulate Virtual Objects (VOs) and receive computer-generated feedback in terms of visual effects and sounds [11, 12]. VR videos make people feel completely immersed in an alternate and digital reality without any relation to the actual reality [13]. Alternatively, AR manipulates real elements with overlaid digital information, creating a

new layer of perception and a complemented reality. The upcoming features of immersive multimedia services create a novel experience with Multi-user Virtual Reality (Multi-User Virtual Reality (MUVR)) [14] and Multi-User Augmented Reality (MUAR) [15], allowing actions of other users to change surrounding elements of the common alternate reality [16].

In general, VoD needs a smooth playback of bitrates reaching 6 Mbps [17, 18]. However, network changes have a negative impact on the Quality of Experience (QoE) in terms of delay, stall, rebuffering, playback start time, and bitrate variability [19, 20, 21]. Immersive multimedia services have even more stringent Quality of Service (QoS) requirements in terms of bandwidth and latency. For instance, MUVR reaches 25 Mbps with 6 ms one-way trip latency [22], and MUAR reaches 100 Mbps with 12 ms maximum latency [23]. Failing to achieve the immersive multimedia service requirements makes users feel motion sickness if the service's remote processing response time exceeds 13 ms [24]. In this way, the current networking and computational infrastructures must support new elements to provide multimedia streaming and immersive flows to fixed and mobile users with low latency while improving the usage of networking and computational resources [25].

VoD and immersive multimedia services require high quality connectivity support to nodes providing computational resources close to the users to achieve QoE/QoS requirements. In this way, geographically distributed multi-tier edge computing nodes bring computational resources to the network's periphery (*e.g.,* neighbourhood, city-wide, or metropolitan scales) [25, 26, 27, 28]. In this way, multi-tier edge nodes must collaboratively work with central cloud servers by deploying hierarchically organized Multi-access Edge Computing (MEC) tiers at the network edge, providing powerful computational resources into small datacenters to attend to local demands [29, 30, 31]. Specifically, hierarchically distributed MEC nodes consist of three tiers, where lower tiers have higher connectivity to users but limited resources than upper tiers. The *Regional Tier* comprises wide coverage nodes of Radio Access Network (RAN) embedded with edge nodes deployed at Internet Service Providers (ISPs), or regional scale at BaseBand Units (BBUs). The *Local Tier* comprises local coverage nodes with medium and short-range wireless communication deployed at Base Stations (BSs) and WiFi Access Points (APs). The *Mobile Tier* comprises local coverage and mobile nodes, such flying Unmanned Aerial Vehicless (UAVs) embedded with wireless communication, powerful processing capability, and limited energy supply, forming a Flying Edge Computing (FEC) [32, 33]. Hence, each tier provides different capabilities in terms of processing performance, location proximity, deployment cost, and edge node mobility on each layer.

Figure 1 depicts a multi-tier edge computing environment with the static and mobile edge nodes embedded with processing and cache units and interconnected by optical and wireless links. Each tier has constraints, as the *Regional Tier* has a top-level, providing the widest coverage area and the cheapest monetary cost to maintain. The *Local Tier* has a mid-level and covers small areas with mid-term monetary costs. The *Mobile Tier* has low-level, covers small areas with high-end monetary costs, has the most limited

processing capacity, and is powered by a battery supply. All tiers work collaboratively to provide remote processing and low-latency services. The low-level tiers may efficiently share their workloads with higher-level tiers with greater processing capabilities without violating stringent service requirements.

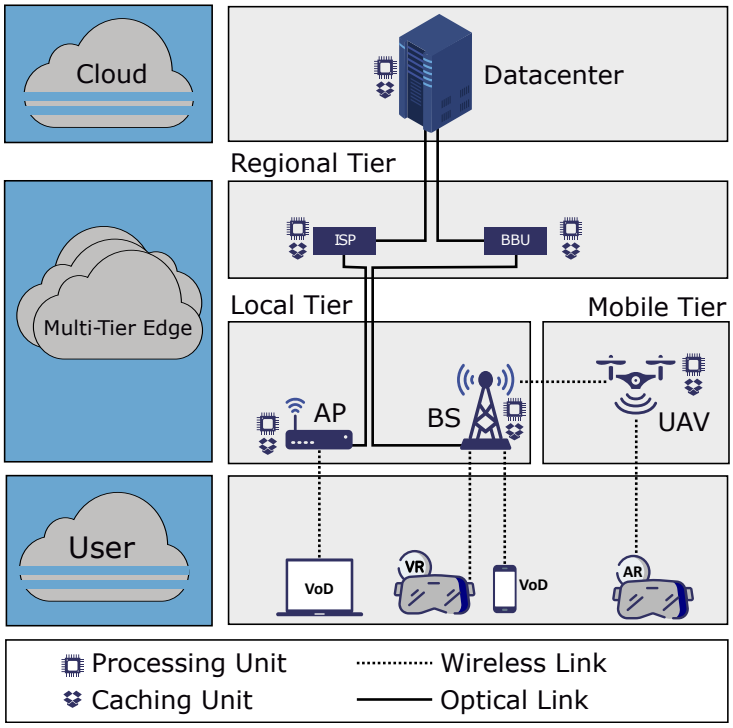

Figure 1: Multi-tier edge computing environment.
Source: author's own.

Remote service processing and caching at multi-tier edge computing traditionally has a monolithic design of all functions on the same node, requiring an efficient usage of network resources due to the extra overhead and load [34]. A monolithic service is a type of software architecture in which all application components, such as the user interface, business logic, and data storage, are tightly coupled and integrated into a single codebase. Specifically, a VoD service includes a monolithic service composed of an entire set of microservices, such as caching, transcoding, and streaming, where a video streaming session needs high and stable data rates in the transmission links to avoid poor QoE in terms of low video quality and stalling [19]. On the other hand, decomposing the monolithic service into specialized elements allows distributed microservice deployment as a Service Function (SF) to enhance QoS/QoE by choosing what, when, and where the SF instantiation may happen. Service Function Chaining (SFC) tends to improve the usage of processing resources but needs to transmit more raw data, requiring an efficient orchestration of computing elements to avoid network overload events [23]. For instance, a MUVR service is composed of different microservices, namely, i) the synchronization microservice to centralize and synchronize MUVR players' actions; ii) the object viewing microservice partially reused among players to render foreground and common background objects iii) VR the coding layer to transform VOs into a video-like format playable at the VR device [35, 22].

In this context, cloud and multi-tier edge computing work collaboratively. The cloud performs service orchestration to control and manage edge nodes to meet specific multimedia service requirements efficiently [27, 36, 37]. For instance, an efficient content orchestrator scheme must periodically monitor network and user requirements to select the most suitable node in the multi-tier edge computing environment with fixed and mobile edge nodes for a user to access the multimedia content hosted in a monolithic or SFC fashion. Hence, service orchestration schemes aim to provide QoS/QoE assurance and efficient usage of network resources, *i.e.,* minimize latency and achieve higher throughput while improving the use of computational resources [38, 39].

1.2 Motivation

A multi-tier edge computing environment enables the execution of monolithic VoD services at each tier of the network edge, where an efficient service orchestration scheme seeks user-oriented decision-making to cause a positive impact on QoE, QoS, and specific monetary cost for VoD service distribution by offering service instantiation closer to users. However, the service orchestration scheme must take into account QoS/QoE metrics (*i.e.,* delay, stall, rebuffering, playback average, and bitrate variability) to efficiently select edge nodes to provide monolithic services at each tier of the edge computing environment [40]. In addition, the service orchestration scheme may be ineffective in QoE aspects as soon as the importance degree of each metric does not correspond with the degradation. Therefore, how, where, and when to efficiently orchestrate the usage of edge nodes for monolithic VoD services in a multi-tier edge computing scenario is not a trivial task [41, 38, 39].

Another important topic in distributed edge computing is using FEC services [42] for providing networking and computing services to static and mobile users on the ground space. UAVs will be part of last-mile 5G and beyond networks and support FEC elements. UAVs will densely populate aerial space in smart cities, providing networking and computing services to ground users. FEC is composed of one or a set of UAVs, can cover large areas, and assure the quick deployment of services in 5G and beyond systems [43]. FEC will cooperate or even replace edge computing services on the ground network in case of congestion/failure/disaster events. FEC diverges from traditional MEC with static edge node by its capabilities for dynamically positioning a set of UAVs to provide edge services in the sky to ground users [44]. In this sense, FEC provides *Mobile Tier* nodes with the potential to provide connectivity [45, 46] and edge computing services to designated areas from the sky to meet temporary and unexpected demand with QoS assurance [47]. The benefits of *Mobile Tier* nodes are highly impacted by the service orchestrator, which must find the location of poorly-served mobile users from *Regional Tier* and *Local Tier*, as well as select and place suitable *Mobile Tier* nodes to best support users of monolithic multimedia services on the ground. Hence, an efficient service orchestrator in a scenario with the assistance of *Mobile Tier* nodes must select the most suitable *Mobile Tier* nodes to provide networking and computing for multimedia services

as close as possible to the poorly-served mobile users, considering the trade-off between minimal QoS requirements and *Mobile Tier* energy issues [44]. Besides, the *Mobile Tier* node needs to periodically adjust its position on varying network conditions due to changes on the ground space, including the mobility of users. However, developing an efficient mobile-aware orchestrator to select and place energy-aware *Mobile Tier* nodes to provide different QoS level support for mobile users on the ground is still an open issue [48].

AR and VR services are accelerating adoption in digital societies. New approaches must be proposed to allow the distribution of immersive content to single and multiple users with QoS support [2, 49]. Immersive flows can be decomposed into elements with specialized ordered Service Functions (SFs) to form a Service Function Chain (SFC). Thus, an efficient SFC orchestration scheme is needed to improve computing and networking resource usage at network edges while assuring low latency for static and mobile users. Specialized SF provides significant flexibility to instantiate fragmented services and improve multi-tier edge computing efficiency with SF chaining. Moreover, part of the SFC has non-negligible similarities among users, providing a further step on computational resource utilization efficiency [35]. Depending on the service nature, the SFC may be decomposed into parallel SF in case of complementary SFs and need dynamic reinstantiation caused by mobility events of users [23, 50, 26, 34]. Furthermore, mobility events may lead to sub-optimal SFC instantiation routes as soon as mobile users move far away from where the service request started, adding an extra delay or even violating service requirements. In this way, the service orchestration scheme for SFC must consider multiple metrics during the decision-making process to reduce latency, maintain stringent requirements for smoothly running services, and improve the overall computational resources utilization. Hence, how, where, and when to instantiate and re-instantiate the set of SFs and to establish a route of a given set of parallelizable, ordered, and location-based SFs for SFC based services is not a trivial task, especially with mobile users, and it is still an open issue [12].

1.3 Goals and Contributions

Motivated by the limitations of the state-of-the-art approaches in improving the usage of computational and networking resources for VoD and immersive multimedia services in multi-tier edge computing environments, this thesis proposes service orchestration schemes for multi-tier edge computing environments organized into multiple tiers to provide efficient dissemination of VoD and immersive multimedia services to static and mobile users. Specifically, we propose a service orchestration scheme for monolithic VoD services, which takes QoE and QoS metrics into account to choose suitable nodes to execute services in the cloud, *Regional Tier*, and *Local Tier*. We also design a service orchestration scheme for immersive multimedia monolithic services with the assistance of *Mobile Tier* nodes, which takes into account energy constraints and distance of *Mobile Tier* nodes to the poorly-served mobile users to choose suitable nodes in *Local Tier* and *Mobile Tier* to execute networking and computing services. Finally, we introduce a ser-

vice orchestration scheme for immersive multimedia SFC in *Regional Tier* and *Local Tier* nodes that takes its decision based on information about ordered and parallel SFs and limited computational resources from multi-tier edge nodes and networking elements. In this context, the goals of this thesis can be achieved by answering three research questions regarding service orchestration in a multi-tier edge computing environment, as follows:

1. **How to orchestrate VoD services for mobile users in multi-tier edge computing environments with QoE support?**

   We propose a multi-tier service orchestrator scheme called Fog4Video, which chooses an appropriate edge node to cache and stream VoD flows with low monetary cost and high QoE assurance. Fog4Video can be executed in the central cloud, *Regional Tier*, and *Local Tier*, where each tier has different processing and networking capabilities and monetary costs to provide VoD services for users. The proposed scheme considers multiple metrics (*i.e.,* available bandwidth, delay, and monetary cost, besides the QoE metrics for VoD, namely the number of stalls and stalls duration) to deploy VoD services in specific edge nodes and for mobile users. In addition, the decision-making process acknowledges periodical reports of QoE from the users to assess the video streaming from each edge node. The QoE values serve as inputs for a real-time method to compute each criterion's influence factor to define the edge node's QoE improvement potential. A detailed description of Fog4Video [51] is presented in Chapter 3.

2. **How to orchestrate monolithic multimedia services for mobile users in flying multi-tier edge computing environments with QoS support?**

   We propose an energy and mobility-aware service orchestration scheme called FLYED, which selects and orchestrates resources in *Local Tier* and *Mobile Tier* to provide networking and computing services to mobile users on the ground with QoS support. The proposed scheme cooperates or even replaces monolithic services hosted in the *Local Tier* on the ground network by providing remote rendering and object recognition functions to immersive multimedia services while reducing latency and increasing Service Provisioning Rate (SPR) and Packet Delivery Ratio (PDR). A detailed description of FLYED [52] is presented in Chapter 4.

3. **How to orchestrate SFC for immersive multimedia services in mobile multi-tier edge computing environments with QoS support?**

   We propose an online and mobility-aware service orchestration scheme for immersive multimedia services called MSF. The proposed approach improves the service acceptance ratio and the usage of computational and networking resources from *Regional Tier* and *Local Tier*. In addition, the proposed scheme minimizes the delay of the ordered and parallel SFs chains in real-time and reinstantiate services into optimal network routes in multi-tier edge nodes. A detailed description of MSF [53] is presented in Chapter 5.

1.4 Thesis Organization

The remaining of the thesis is structured as follows: Chapter 2 describes the related works highlighting the service orchestration schemes' limitations and advantages of literature. Chapter 3 introduces a service orchestration scheme in multi-tier edge computing environments for VoD services and its evaluation. Chapter 4 introduces a service orchestration scheme in multi-tier flying edge computing environments for monolithic services and its evaluation. Chapter 5 introduces an SFC orchestration scheme in multi-tier edge computing environments for MUVR and MUAR services and its evaluation. Chapter 6 concludes this thesis and defines future research directions. Figure 2 summarizes the context of research questions regarding service orchestration in multi-tier edge computing environments.

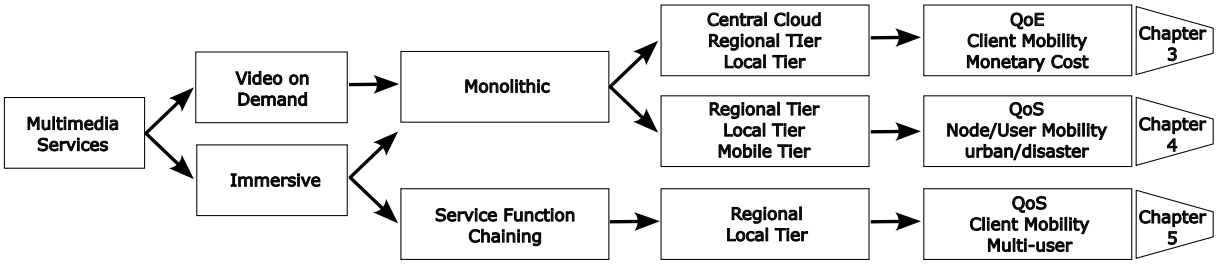

Figure 2: Thesis Summary.
Source: author's own.

CHAPTER 2

Related Works

This Chapter introduces related works about the service orchestration proposals in multi-tier edge computing environments, highlighting the benefits and limitations of each one. The Chapter divides service orchestration schemes as follows: *(i)* multi-tier edge computing service orchestration for VoD; *(ii)* multi-tier flying edge computing service orchestration for monolithic services;*(iii)* multi-tier edge computing service orchestration for SFC environments.

2.1 Video on Demand Service Orchestration in Multi-tier Edge Computing Environments

A multimedia service like VoD typically relies on a monolithic design to deliver the content to static and mobile users. Specifically, the monolithic service deployment assumes the instantiation of the complete set of functions, such as caching, transcoding, encoding, and streaming functions, into a microservice deployed in a given cloud-based tier. In this sense, finding the best node in a multi-tier edge environment is important to instantiate the multimedia service based on a service orchestration, which delivers the Adaptive Bitrate (ABR) video content for each user request. Hence, a multimedia service orchestration scheme is essential to avoid potential playback abandonment of users with poor QoE by streaming video from edge nodes with proper connectivity and resources. In this section, we introduce recent research on VoD service orchestration in multi-tier edge computing environments addressing the instantiation of monolithic VoD service based on different decision-making parameters. This section also identifies gaps in edge computing service orchestration schemes.

Yang et al. [38] studied the orchestration of a *Local Tier* to assist predictive

ABR video streaming for on-road driving scenarios based on Deep Reinforcement Learning (DRL). The DRL enables a proactive orchestration scheme to maximize the long-term QoE only in terms of bitrate. However, DRL does not consider key parameters in its decision-making process, including monetary costs of cloud resources, and does not analyze QoE metrics (*e.g.,* stalling).

Mehrabi et al. [39] enabled jointly 360° video rendering of a navigable viewport larger than the field of view of the HMD from a *Local Tier*. The HMD embedded caching and encoding functions for rendering part of a video. The *Local Tier* assisted rendering part of the video with an orchestration scheme, optimizing the tradeoff between average video bitrate and delay. They leveraged the interpolation between the user's viewport size and delay, converted the problem to an integer nonlinear programming model, and then designed an online algorithm. However, the proposed scheme does not provide a dynamic instantiation of edge nodes, nor does it integrate multiple tiers, monetary costs, mobile nodes, and the most perceptive QoE metric of stalling.

Khattak et al. [54] orchestrated multi-tier edge computing resources to allow the distribution of low-latency applications. The authors considered *Local Tier* in conjunction with a *Mobile Tier*. They used *Mobile Tier* by enabling ubiquitous vehicles for location-aware service instantiation to provide QoS support for mobile users. They presented a multi-tier edge computing scheme for multimedia services in mobile scenarios. They used cache size to evaluate the impact on energy consumption, delay, and cache hit ratio. However, the proposed scheme lacks in terms of dynamic instantiation of edge nodes, integration of multiple tiers, use of monetary costs, and support to QoE.

Rosário et al. [18] relied on multi-tier edge computing to facilitate the migration of multimedia services demanding QoE guarantees. The authors used a service orchestration scheme to migrate service data from one node to another based on request patterns and downtime issues of migration. However, the proposed scheme does not consider monetary cost in its decision-making process and does not orchestrate multiple nodes at the same tier.

Chiang et al. [40] relied on *Regional Tier* and *Local Tier* to collaborative cache popular videos. The authors proposed a social-aware proactive cache strategy embedded with social network user interactions and video streaming. They only used average bitrate as a QoE metric. The proposed QoE-driven video adaptation scheme is based on a bitrate parameter to dynamically transcode cached videos into appropriate resolution at network edges for each request. However, they lack in terms of dynamic instantiation of edge nodes, integration of multiple tiers, use of monetary costs, and support to QoE.

Pacheco et al. [55] proposed a multi-tier edge computing environment to provide multimedia services with QoS support. The authors proposed a proactive service orchestration scheme based on a mobility predictor to instantiate and migrate services to decide when the service migration becomes necessary. The service orchestration scheme also considers service requirements and monetary costs during decision-making. However, the proposed scheme does not discuss and analyze the proposal's impact on the user QoE.

Table 1 summarizes the main characteristics of previous service orchestration schemes in terms of decision-making criteria, dynamic instantiation, monetary cost, and architecture. Based on our state-of-the-art analysis, we conclude that VoD services deployed in multi-tier edge computing can improve the QoE of multimedia applications for static and mobile users. In this context, it is important to consider QoE, QoS, and monetary aspects during decision-making to better deliver multimedia content based on user needs and network conditions. It is important to highlight that QoE measures users' perceived visual quality level, while QoS measures the service's computational resources and time constraints. Monetary costs are the costs of services/resources in a cloud-based node during a certain period. In summary, all related service orchestration schemes considered QoS aspects, but only a few considered bitrate as a QoE metric and neglected video stalling [38, 39, 18, 40]. The service orchestration must follow a dynamic instantiation approach to improve networking and computing resources used during the mobility/transfer of multimedia content from one edge node to another during handover events. Few works proposed a dynamic instantiation scheme in distributed edge scenarios [38, 18, 40, 55]. Finally, this thesis assumes the existence of a multi-tier edge architecture with *Regional Tier*, *Local Tier*, and *Mobile Tier* together as a unique architecture. Only Pacheco et al. [55] considered the monetary costs of cloud resources in the decision-making process. A few existing works discussed multi-tier edge computing schemes [54, 18, 40, 55]. The proposed Fog4video scheme improved the delivery of VoD flows by taking multiple QoS and QoE parameters, a dynamic instantiation approach, and monetary costs of cloud resources into a multi-tier edge computing architecture.

Table 1: Summary of Existing Proposals for Service Orchestration for VoD Services in a Multi-tier Scenario

Work	Criteria	Dynamic Instantiation	Monetary Cost	Architecture
Yang et. al. [38]	QoE*/QoS	✓		single-tier
Mehrabi et. al. [39]	QoE*/QoS			single-tier
Khattak et. al. [54]	QoS			multi-tier
Rosário et. al. [18]	QoE/QoS	✓		multi-tier
Chiang et. al. [40]	QoE*/QoS	✓		multi-tier
Pacheco et al. [55]	QoS	✓	✓	multi-tier
Fog4Video [51]	QoE/QoS	✓	✓	multi-tier

2.2 Monolithic Service Orchestration in a Multi-tier Flying Edge Computing Environment

The next decade will provide high computing and networking 5G/6G applications to mobile users in smart cities with QoS support, including immersive services such as AR. 5G systems will allow remote service processing from the central cloud and multi-tier edge computing elements to meet future mobile services' computation-intensive and

delay-sensitive requirements [23]. However, mobile users in specific city locations can experience poor QoS due to networking/computing overload events or even failures. In this context, *Mobile Tier* in multi-tier edge computing environments has a great potential to provide computing and networking capabilities from the sky to ground users with QoS support [44].

Mobile Tier diverges from *Regional Tier* and *Local Tier* by its capabilities for dynamically positioning a set of UAVs as *Mobile Tier* nodes in 3D spaces to provide edge services from the aerial space [44]. *Mobile Tier* node can be equipped with communication equipment and computing services to provide connectivity and different levels of QoS support in the sky to ground users, minimizing the response time and providing high throughput to applications of static and mobile users [44]. *Mobile Tier* nodes will seamlessly integrate 5G/6G systems with *Mobile Tier* assistance with QoS assurance [47]. This section presents the existing proposals about *Mobile Tier* nodes for service providers in smart cities. It also identifies gaps in orchestration schemes in a multi-tier flying edge computing environment.

Tang et al. [56] considered the UAV-based Internet of Medical Things scheme for real-time health monitoring. The authors suggested processing and analyzing delay-sensitive data to improve the survival rate for cardiovascular and cerebrovascular diseases. In this sense, they proposed UAVs acting as *Mobile Tier* nodes to perform the data analysis. They proposed a service orchestration scheme to instantiate services and place *Mobile Tier* nodes considering the influence of energy consumption and the context of the area. The proposed scheme used Particle Swarm Optimization (PSO) based algorithm to optimize placement in the serving area of the medical devices and reduced the number of UAVs while maintaining QoS for applications. However, the proposed approach considers only static medical devices and lacks support for mobile users on the ground space.

Tan et al. [57] introduced cache-enabled *Mobile Tier* nodes to instantiate AR services in a smart city scenario and proposed a service orchestration scheme to minimize the energy consumption of *Mobile Tier*. The authors sequentially established two non-convex programming problems and optimized delay and energy performance during AR data uploading and downloading. However, the proposed scheme does not consider mobile environments on the ground space.

Pandey et al. [47] used *Mobile Tier* nodes to give computing support to static users on the ground space. In this sense, multiple *Mobile Tier* nodes had instantiated services to static ground users using applications demanding high throughput and low latency. The proposed service orchestration scheme used an incentive mechanism to encourage *Mobile Tier* nodes to trade their idling computational resources for the cooperative application execution. However, the proposal does not support mobile users on the ground space.

Tang et al. [58] proposed a service orchestration scheme by instantiating services in *Mobile Tier* using terrestrial vehicles left behind and *Mobile Tier* nodes into a disaster area. During a disaster event, the service orchestration offloads services from ground users.

The proposed service orchestration scheme used K-means to cluster users and services by functionality. The authors tackled energy consumption and network delay in a particular scenario where even vehicles may be unavailable in the disaster area and do not support user mobility.

Li et al. [59] proposed an energy-aware service orchestration scheme by integrating non-orthogonal multiple access to improve energy efficiency using a power allocation and subchannel assignment method. The authors evaluated the energy efficiency of instantiated caching services using in *Mobile Tier* nodes. However, they do not tackle *Mobile Tier* node placement and assume an optimized position of edge nodes on the ground and aerial spaces.

Table 2 summarizes the related works for service orchestration schemes in multi-tier flying edge computing environments regarding decision-making criteria, architecture, mobility-, and energy awareness. By analyzing the existing proposals, we can see that *Mobile Tier* nodes in 5G networks can significantly improve the QoS support of mobile users on the ground space due to temporary failures, network congestion, disaster, or processing overload events. We can also see some common trends, *e.g.,* some works relied on service orchestration schemes for offloading ground user services to *Mobile Tier* nodes [56, 47], and other proposals used caching services at *Mobile Tier* nodes [59, 57]. Specifically, it is important to consider QoS, mobility, and energy issues for orchestrating edge nodes on the ground and aerial spaces during decision-making. Based on the state-of-the-art analysis, a service orchestration scheme must provide mobility and QoS support in terms of PDR and delay aspects to improve connectivity to mobile users on the ground. To the best of our knowledge, only FLYED considered all these important aspects in a mobile multi-tier edge computing environment.

Table 2: Summary of Existing Proposals for Service Orchestration for Multimedia Services with Mobile-tier Nodes

Works	QoS	Architecture	Location	Mobility	Energy
Tang et al. [56]		single-tier	urban		✓
Tan et al. [57]	delay	single-tier	urban		✓
Pandey et al. [47]		single-tier	urban		
Tang et al. [58]	delay	single-tier	disaster		
Li et al. [59]		single-tier	urban	✓	✓
FLYED	PDR, delay	multi-tier	urban/disaster	✓	✓

2.3 Service Function Chaining Orchestration in Multi-tier Edge Computing Environments

MUVR and MUAR supports multiple mobile users interacting with new digital dimensions. Such clients view VOs in immersive multimedia services displayed by com-

puting and battery-constrained HMD for VR and Head Mounted Glasses (HMG) for AR [22, 16]. For instance, users may use MUAR services anywhere, even when commuting on trains, buses, or carpooling. In this sense, MUAR services might be the next level of everyday life, such as VO signs to find a specific product in a shopping market or synchronize VOs for a touristic visit or group training services. In this way, multiple co-located mobile clients interact in real time with a common set of VOs. In a MUAR service, mobile users with HMGs establish a session with a multi-tier edge computing and consume computing-intensive services, such as rendering unique and common VOs [26]. Hence, multiple and synchronized mobile users must handle the same views in an interactive space and react to each others' actions with low latency [16].

Multi-tier edge computing environments have constraints and can not provide a low response time for mobile users without mobility support (*i.e.,* up to 5 ms as required for MUAR) [12]. MUAR services can be decomposed into a set of ordered SF, such as frame acquisition, pre-processing, object detection, object recognition, location-based VO caching, and other functions, for instantiation at multi-tier edge computing [16], where SFC reduces latency and improve the usage of computing, storage, and networking resources. In this way, an SFC session can be organized in a set of parallelizable, ordered, and location-based SFs to improve the efficiency of computational resources utilization in terms of processing, storage, and networking. For instance, it is important to parallelize SF chains to reduce the delay since two or more SFs are executed in parallel whose result needs to be merged later into the service chain. Hence, SF chaining avoids redundant processing and transmission of a significant volume of data while reducing delay. In this section, we introduce recent proposals on multi-tier edge computing service orchestration schemes addressing the instantiation of SFC services for MUVR and MUAR. Finally, we identify gaps in edge computing service orchestration schemes for MUVR and MUAR applications.

Santos et al. [60] suggested remote rendering of immersive multimedia services. However, remote nodes must surpass the stringent end-to-end delay requirement, which cannot exceed 20 ms to avoid motion sickness. In this sense, the authors proposed a mixed-integer linear programming formulation for efficient service orchestration in multi-tier edge computing environments. The model considered *Regional Tier*, *Local Tier*, and central cloud nodes. In addition, the service orchestration scheme aimed to satisfy the delay requirements for multimedia applications. However, the proposed scheme does not support SFCs-based MUVR and MUAR applications with QoS support.

Akhtar et al. [23] proposed resource orchestration and management for remote rendering immersive multimedia services in *Local Tier*. The authors investigated the problem of optimal instantiation of SFCs and the network traffic steering using wired and wireless millimeter-wave (mmWave) links. The proposed scheme provided a comprehensive "microscopic" binary integer program and a heuristic. However, the proposed solution does not consider the handovers of users or chances in the networking/computing conditions, where such events require the reinstantiation SFs.

Wang et al. [61] proposed distributed edge computing SFC services and in-

troduced an online SFC orchestration scheme at the network edges to minimize service latency by jointly considering the effect of user mobility, edge capacity, and service migration. The proposed SFC approach focuses on single-user and linear SFC scenarios. It also provided a service migration feature unsuitable for remote rendering of MUAR services because of the long system downtime.

Wang et al. [62] presented a distributed edge computing approach to provide fast and stable provision of video streaming and AR services to minimize latency and increase computing efficiency. The proposed SFC orchestrator minimized the overall scheduling latency using a deep reinforcement learning-based algorithm. However, the proposal focuses only on static and single-user scenarios. Thus, as expected in smart cities, it is unsuitable for dynamic mobile environments.

Lin et al. [63] proposed Partial Parallel Chaining (PPC) scheme to overcome the instantiation of parallel SFCs at the same edge nodes. They claim that parallel instantiation in different nodes introduces a non-negligible cost to duplicate and merge packets. They designed a scheme to identify optimal SF parallelism and minimize latency. However, their technique applies parallel SFC of network functions focused on single and static users.

Medeiros et al. [64] studied a latency- and energy-aware SFC orchestration scheme splitting the SFs of VR services in single edge node. The proposed solution was modelled as an optimization problem to minimize latency and energy consumption of HMGs. However, the proposed SFC orchestrator focuses on single-user and linear SFC scenarios.

Table 3 summarizes the existing works for SFC orchestration in multi-tier edge computing environments for MUAR and MUVR services regarding decision-making criteria, mobility, parallel, multi-user, and multi-tier environments. Using the SFC orchestrators in edge nodes can significantly improve the QoS support of mobile users for MUAR services because they can reduce the latency of sub-optimal routes, network congestion, and/or processing overload events. We can also identify some common trends, *e.g.,* some works relied on SFC orchestration for offloading HMG to edge nodes [23, 60, 64] and others used parallel SFC to provide service to multiple users [63]. However, a set of works only consider static users and do not provide online service instantiation with mobility support [60, 23, 62, 63, 65]. The proposal [63, 65] considered parallel SFC to improve computational resources utilization to multiple static users. Several works take into consideration point-to-point SFCs [60, 23, 61, 62, 64]. Based on the state-of-the-art analysis, a SFC orchestration scheme must consider mobility, SFC parallelization, multiple users, and QoS support to improve the efficiency of multi-tier edge computing resources. To the best of our knowledge, MSF considered all important aspects into a mobile multi-tier edge computing environment. Summary of Existing Proposals for Service Orchestration for Multimedia Services with Mobile-tier Nodes

Table 3: Summary of Existing Proposals for SFC Orchestration in Edge Computing environments

Work	QoS	Mobility	Parallel	multi-user	architecture
J. Santos et al. [60]	✓				multi-tier
Akhtar et al. [23]	✓				single-tier
Wang et al. [61]	✓	✓			multi-tier
Wang et al. [62]	✓				single-tier
Lin et al. [63]	✓		✓	✓	single-tier
Medeiros et al. [64]	✓	✓			single-tier
MSF [53]	✓	✓	✓	✓	multi-tier

CHAPTER 3

Video on Demand Service Orchestration in Multi-tier Edge Computing Environment with Quality of Experience Support

The objective of this Chapter is to address research question: *1) How to orchestrate VoD services for mobile users in multi-tier edge computing environments with QoE support?*

This Chapter introduces the Fog4Video orchestrator proposal [51][1]. Fog4Video selects an appropriate edge node to stream VoD streaming with QoE support in a multi-tier edge computing environment considering the network and edge node conditions and user's requirements during the decision-making process. Fog4Video performs service orchestration in a hierarchical cloud-based infrastructure, where multi-tier edge nodes could cache the content and also provide VoD services, improving the QoE of VoD services. The hierarchical design stands for each tier's performance, location, and cost deployment. Fog4Video classifies the connectivity and resources of each available edge node into a multi-criteria rank, where it considers the Analytic Hierarchy Process (AHP) method to assign different degrees of importance for each criterion to provide better QoE for each user. In the following, Section 3.1 presents the proposed multi-tier edge architecture. Section 3.2 describes the Fog4Video orchestration scheme. Section 3.3 shows performance evaluation results collected in a simulation environment. Finally, Section 3.4 describes the final remarks of this Chapter.

[1]Partially reproduced in this chapter – Copyright © 2020 Elsevier.

3.1 Scenario and Architecture Overview

Multimedia service providers are disrupting the TV and video industry, where big multimedia players, such as YouTube, Amazon, and Netflix, are replacing the current broadcasters [66]. In this sense, the market players are attracting thousands of new customers every day, where they are expecting to access VoD services with high QoE while optimizing their needs in terms of image definition and playback smoothness [19]. Therefore, VoD providers require new schemes to deliver high-quality video streaming in dynamic and adaptive 5G networks during usage peaks, for instance, by using Hypertext Transfer Protocol (HTTP) streaming [17]. Many providers lower the video quality to reduce network traffic for smoother playback. However, users abandon the multimedia services due to stalls, rebufferings, playback average, and bitrate variability [19, 20].

In the last few years, caching services and edge computing have gained much attention in both industry and academia as new schemes to improve the QoE delivery to mobile users [36, 67]. Edge computing extends central cloud computing services with low latency and additional computing, storage, and networking resources closer to the user. Edge computing offers VoD services cached closer to the user, providing the transmission requirements properly in terms of delay and bandwidth [68]. In this sense, a multi-tier edge computing environment allows the execution of VoD services in different tiers to match the topology and distributed workload properties of VoD applications while assuring QoE requirements [69, 70]. In this context, new services must be created to offer VoD services to mobile users while optimizing the usage of heterogeneous network resources. For instance, VoD providers can run part of VoD services in the central cloud in a more cost-effective fashion but with higher values of latency for static users with low QoS requirements. However, VoD providers can migrate part of the VoD content to an edge node granting low latency with a high monetary cost for mobile users.

Figure 3 shows the modules and components of the proposed multi-tier edge architecture, which relies on two types of nodes: centralized cloud and distributed edge computing nodes. The nodes work collaboratively to provide VoD services with QoS/QoE support from mobile users. Edge nodes can be deployed anywhere in a network, organized in tiers between the mobile devices (at the bottom) and the central cloud (at the top) [18, 29, 71], as it can be seen in Figure 3. The *Client Module* can be mobile users consuming a VoD flow. The *Multi-tier Edge Module* can be any device in the RAN, *e.g.,* BS, or AP providing multimedia services to a few dozens or hundreds of mobile devices. A replica of a tier can take place in the network, such as BBU or ISP, to adjust the distribution of VoD flows according to the different network conditions. Besides, mobile users could become an edge node to relay the video content via Device-to-device (D2D) wireless communication for mobile devices with high and similar traffic demands. On top of such multi-tier architecture, there is the *Cloud Module*. The *Cloud Module* is responsible for providing VoD services in a centralized fashion keeping the entire database in a central cloud datacenter [72].

The *Client Module* consists of a *Client Agent* that manages communication among

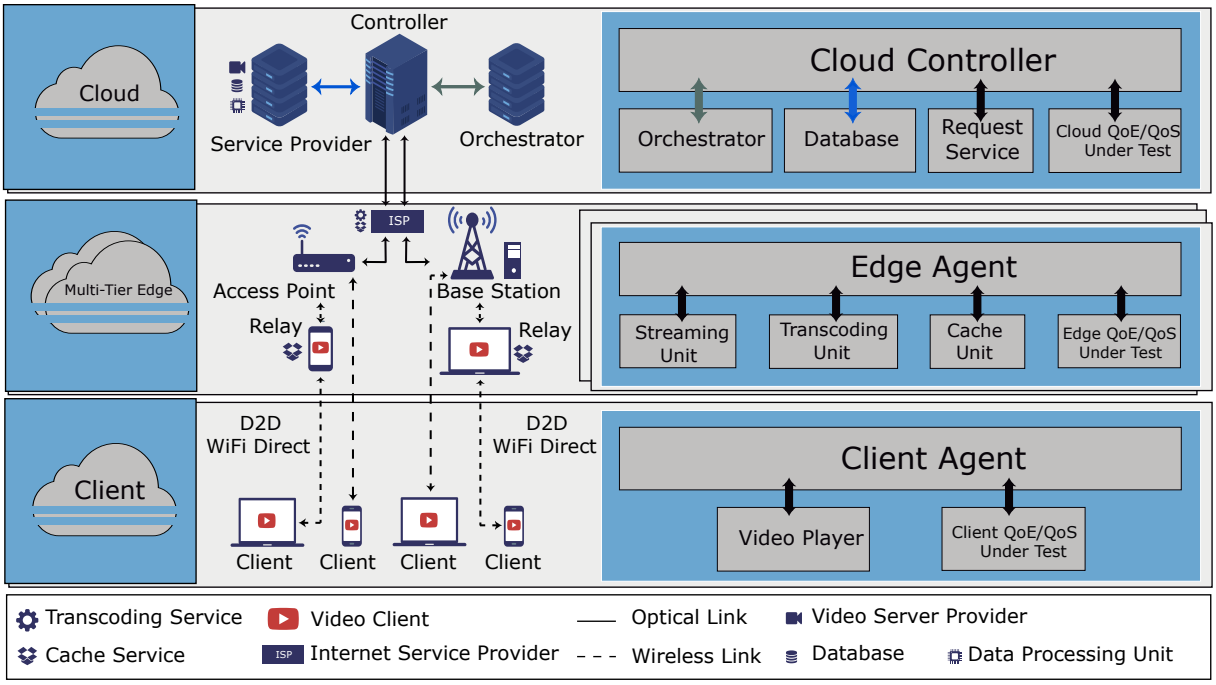

Figure 3: Multi-tier Edge Architecture for VoD Services.
Source: [51]

Video Player and *Client QoE/QoS Under Test*. The *Client Agent* plays the role of an interface between *Cloud* and *Client Modules*, synchronizing control and data flow in both directions. Moreover, this module manages the migration of VoD services to the edge node. The *Video Player* component downloads video content on the mobile device screen. While displaying video content, the *Client QoE/QoS Under Test* collects and reports VoD and QoE and QoS measurements, such as playback start time, stall duration, Mean Opinion Score (MOS), throughput, Rount-Trip Time (RTT), packet loss, and others, to understand QoE related to VoD.

The *Multi-tier Edge Module* is composed of *Regional Tier* and *Local Tier*. It includes an *Edge Agent* connecting the *Streaming Unit*, *Transcoding Unit*, *Cache Unit*, and *Edge QoE/QoS Under Test* components. The *Edge Agent* also plays the role of an interface between *Cloud* and *Multi-Tier Edge Modules*, synchronizes the control and data flow exchange in both directions, and manages/provides communication among internal modules. *Transcoding Unit* is implemented at network edges to adapt the video codec, bitrate, or resolution according to the network conditions, device capabilities, or QoE characteristics [73]. However, it can run only on a tier with sufficient resources since it requires more processing, data exchange, and memory capabilities. The *Cache Unit* stores redundant copies of given video content close to the users. The *Streaming Unit* streams the content from *Cache Unit* for each *Video Player* request based on *Orchestrator* decisions. The *Edge QoE/QoS Under Test* collects QoE and QoS measurements and replays the demands made by the *Orchestrator*.

The *Cloud Module* consists of a *Cloud Controller* connecting the *Orchestrator*, *Video Database*, *Request Service*, and *Cloud QoE/QoS Under Test*. The *Orchestrator*

coupled to the *Controller* steers decision-making on management and operation tasks based on the QoE/QoS reports from the users. For example, the *Orchestrator* decides about the use of edge nodes, proceeds to migrate services, evaluates available resources, and considers a specific orchestrator scheme to define where, what, and when a user must download the video from a different *Streaming Unit*. It holds input from the *Cloud QoE/QoS Under Test*, VoD requirements, and high-level management information, such as network-wide policies or Service Level Agreement (SLA). The *Video Database* stores the VoD content for traditional Dynamic Adaptive Streaming over HTTP (DASH) in the *Service Provider*, while the *Request Service* coupled to the *Controller* distributes the content to the *Video Player* at the user device. Moreover, it controls from which tier the *Video Player* must download the content, including when the download shifts from one edge node to another. The *Cloud QoE/QoS Under Test* built-in the *Controller* collects QoE and QoS measurements and replays the demands made by the *Orchestrator*. The *Cloud Controller* manages communication among internal modules, synchronizes the control and data flow exchange, and sends decisions taken by the *Orchestrator* to edge nodes. These decisions generate control flows to *Edge and Client Agents*. Hence, the controller of each tier can start the VoD service procedures to optimize the QoE of delivered videos.

In this architecture, the *Video Player* requests a video from the *Request service* at the cloud. Next, *Request Service* sends back the Media Presentation Descriptor (MPD), which is an Extensible Markup Language (XML) file that contains information about the available video chunks, as it happens in DASH-based environments. It also includes other metadata users must choose between the available video chunks. In a VoD scenario, the video is divided into multiple chunks, and each chunk can be requested with a different bitrate representation to avoid buffer underflow, preventing stalling in varying network conditions. In this sense, the *video Player* requests the next chunk with an appropriate bitrate based on the available transmission resources. In other words, the bitrate increases with sharpened network conditions and decreases in case of buffer underflow.

The *Client QoE/QoS Under Test* periodically sends QoE feedback, *i.e.,* the number of stalls and stalls duration values, to the *Orchestrator* unit. Stalls are interruptions of the video playback. Fog4Video supports an *Orchestrator* scheme that considers information about network and edge node conditions and user requirements to perform its real-time content orchestration decision-making process, *i.e.,* it chooses an appropriate *Streaming Unit* from a given tier for the user to download the video. The *Orchestrator* can also decide to keep downloading from the current edge node *Streaming Unit* as decided previously. Therefore, the Fog4Video orchestrator performs decision-making and executes a load balance scheme among edge nodes, meeting the user needs better, *i.e.,* avoiding *Streaming Unit* overload while improving VoD distribution.

3.2 Fog4Video Orchestration Scheme

Fog4Video supports two main phases, namely, Analysis and Decision-making & Execution. The Analysis phase collects important metrics for the decision-making process, *i.e.,* available bandwidth, delay, number of stalls, stall duration, and cost to deploy VoD services in a given tier. Afterwards, this information is evaluated in a Decision-making & Execution phase, determining the best *Streaming Unit* to download the video. Finally, the *Cloud Controller* sends all decisions taken by the *Orchestrator* to edge nodes and users.

3.2.1 Analysis Phase

Cloud QoE/QoS Under Test, *Edge QoE/QoS Under Test*, and *Client QoE/QoS Under Test* collect information from cloud, edge nodes, and users, to understand their needs and conditions to make the best decision. Specifically, Fog4Video receives the QoS characteristics, *i.e.,* available bandwidth and delay, collected by *Cloud QoE/QoS Under Test* and *Edge QoE/QoS Under Test*, since these values impact on the QoE quality level. The VoD service uses Transport Control Protocol (TCP) stream, which is highly affected by high latency in best-effort Internet [74]. The video adaptation algorithm is based on TCP and it depends on quick feedback given by the users [75, 76]. Therefore, the delay is essential to provide a more accurate response to the characteristics of the next chunk to be sent. The *Orchestrator* gives preference to idle and more cost-efficient edge nodes before collecting QoS reports from the users.

This phase also considers QoE metrics, *i.e.,* number of stall events, and stall duration, collected by *Client QoE/QoS Under Test*. Specifically, *Video Player* buffers store downloaded chunks before playing out and stops the video playback. As soon as the buffer level is empty, the video playback cannot continue, since there is insufficient data available in the buffer [19]. The interruption lasts until the fulfillment of a complete chunk in the buffer. These interruptions are stall events, and their duration is called stall duration. These two well-known QoE objective metrics have the most crucial factor in QoE since they directly impact the continuity of the VoD session [19]. For instance, users who experience more interruptions in the video tend to watch the video for a shorter duration and are likely to be dissatisfied in the case of four or more interruptions [77, 78]. Furthermore, viewers prefer a single but long stall event instead of several short stall events. Hence, the number of interruptions and their duration affect VoD QoE [79].

Fog4Video also considers $cost_k$ to process data in a given edge node k, which depends on the amount of Central Process Unit (CPU) time the *Streaming Unit* or *Transcoding Unit* use CPU for processing P_k and the monetary cost per hour H_k. In general, the scale deployment of top-level tiers tends to offer resources in a cost-effective manner per processing unit compared to low-level tiers. The value $cost_k$ to stream the VoD service in an edge node is computed as follows.

$$cost_k = P_k \times H_k \tag{3.1}$$

Eq. 3.2 computes the overall cost C_k to process a video chunk to deploy VoD services in a given edge node. It depends on $cost_k$ to process the bitrate representation r of a given video v and binary variable $\alpha^r_{v,k}$. A *true* value of $\alpha^r_{v,k}$ stands for the transcoding of a chunk b^r_v in an edge node k.

$$C_k = cost_k \times \sum_{0}^{k} \sum_{0}^{v} \sum_{0}^{r} b^r_v \alpha^r_{v,k} \tag{3.2}$$

Fog4Video checks the resource availability of the edge nodes, *i.e.,* the number of available computation resources to adapt video content accordingly to the video bitrate requested by the users from a specific edge node. In this way, the orchestrator acquires a global view to understand network and edge conditions as a way to choose the appropriate nodes. In the end, Fog4Video effectively takes a decision to improve the QoE and QoS of VoD services, while maintaining low monetary costs.

3.2.2 Decision-making & Execution Phase

At this phase, the Fog4Video orchestrator scheme at the central cloud is responsible for selecting the best *Streaming Unit* of a given edge node from which the user should download the video. In the first step, the scheme creates a list L of candidate edge nodes by checking the resource availability to compute content adaptation of each edge node to deploy the VoD service, as shown in Algorithm 1. Fog4Video receives a chunk request in a bitrate $b^{r,t}_v$ at a given time t. Then, Fog4Video checks the resource demand A_k on time $t-1$ based on $b^{r,t-1}_v$ and $\alpha^{r,t-1}_{v,k}$ for all edge nodes and videos in line 4. Fog4Video evaluates if the resource availability T_k can support the current resource demand in A_k in line 5 for insertion of the edge node k in L in line 6.

Algorithm 1: Computing edge node candidate list

Input: $b^{r,t}_v, b^{r,t-1}_v, \alpha^{r,t-1}_{v,k}$
Output: L
Data: V, K, T_k

```
1 A_k = 0
2 foreach k ∈ K do
3     foreach v ∈ V do
4         A_k = A_k + b_v^{r,t-1} × α_{v,k}^{r,t-1}
5         if T_k ≥ A_k + b_v^{r,t} then
6             Push request b_v^{r,t} for candidate edge node k into list L
```

From the list of candidate edge nodes, we consider network, edge node, and user metrics for decision-making, which have different degrees of importance in the decision-

making process. In this context, Fog4Video considers AHP [80] to compute the influence factor for each parameter. Specifically, AHP is a multi-criteria decision-making process capable of balancing inputs with different degrees of importance. AHP combines qualitative and quantitative elements for the analysis, allowing the system to find an ideal solution considering several metrics in the decision-making process. AHP recognizes a pairwise comparison between the numerical values of each parameter and their relative degrees of importance to adjust their weights at runtime. As a result, a higher weight means higher importance for the corresponding criterion. The pairs must not contradict each other, *e.g.,* if the metric i is twice more important compared to metric j, then j has 1/2 importance than i. We consider seven importance levels to compare each pair of parameters, indicating how essential one parameter is compared to others, as shown in Table 4.

Table 4: Pairwise Importance Levels

$m_{i,j}$	Degrees of Importance
1	i is as important as j
2	i is slightly important than j
3	i is more important than j
4	i is much more important than j
1/2	i is slightly less important than j
1/3	i is less important than j
1/4	i is much less important than j

We consider a comparison matrix $A = M_{n\,x\,n}$ with lines and columns representing the metrics considered for decision-making to represent all pairwise comparisons. Variable n denotes the number of elements compared, as shown in Eq. 3.3. Each $m_{i,j}$ value in the matrix means how important the $i-th$ element is compared to the $j-th$ element. The degree of importance of each level depends on subjective judgment related to abandonment rates of VoD due to poor QoE. We set the values to achieve improvements in terms of QoE while also considering others metrics unless stated otherwise.

$$A = M_{n\,x\,n} = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,n} \end{pmatrix} \tag{3.3}$$

Fog4Video uses the following QoE metrics: number of stalls as F, stall duration as E, delay as D, and cost for deploying VoD as C_k. The comparison matrix M indicates which parameters have higher priority than others, as shown in Eq. 3.4. For instance, in the first line, the number of stalls F metric is twice more significant than stall duration E in the second line and three times more important than delay D in the third line. It is essential to highlight that if one criterion is considered twice more relevant than another, the other is 1/2 as important as the first. Note that the matrix's main diagonal must

always contain the value 1 as we compare a metric with itself.

$$M = \begin{array}{c} \\ F \\ E \\ D \\ C_k \end{array} \begin{array}{c} \begin{array}{cccc} F & E & D & C_k \end{array} \\ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1/2 & 1 & 3/2 & 3 \\ 1/3 & 2/3 & 1 & 2 \\ 1/4 & 1/3 & 1/2 & 1 \end{pmatrix} \end{array} \tag{3.4}$$

AHP measures the influence factor $I_{i,k}$ assigning pairwise comparisons with the data on each edge node k. The influence factor is given by the sum of the multiplication of the current value of a metric $P_{i,k}$, *i.e.,* F, E, D, C_k, with the relative importance of the other metrics, as shown in Eq. 3.5. For example, if the values in P_k are $F = 1$, $E = 2$, $F = 15$ and $C_k = 1$, the influence factor of the delay metric would $15 \times (1 \times (1/3) + 2 \times (2/3) + 15 \times 1 + 1 \times 2)$, based on the third line of Eq. 3.4.

$$I_{i,k} = P_{i,k} \times \sum_{j=1}^{n} m_{i,j} \tag{3.5}$$

The influence factor of each metric serves as input for the score S_k of the current conditions in each edge node k, which is given in Eq. 3.6.

$$S_k = \sum_{i=1}^{n} I_i \tag{3.6}$$

As each video may have different QoE requirements, we consider a weight matrix W to give different priorities for each video stream in V. In this sense, each column in W is the weight given by a video, as each type of video has different characteristics and needs specific management. In this case, weights were assigned to each video type, as shown in Eq. 3.7.

$$W = \begin{pmatrix} w_1 & w_2 & \cdots & w_v \end{pmatrix} \tag{3.7}$$

The decision matrix DM considers the combination of each video weight in W and score S_k of each edge node, based on Eq. 3.8

$$DM = \begin{pmatrix} w_1 * S_1 & w_1 * S_2 & \cdots & w_1 * S_k \\ \vdots & \vdots & \ddots & \vdots \\ w_v * S_1 & w_v * S_2 & \cdots & w_v * S_k \end{pmatrix} \tag{3.8}$$

The matrix DM parameters have a significant variation, making a low-accuracy analysis for the decision. To decrease the discrepancy between the values of DM, we perform a normalization in every parameter of DM using the arithmetic average $\overline{DM_k}$ of the values of column k. The calculation gets the difference between a given tier and the average of all tiers, parameter by parameter, as shown in Eq. 3.9. We have the normalized matrix $\eta_{v,k}$ with the exact dimensions of DM.

$$\eta_{v,k} = DM_{v,k} - \overline{DM_k} \tag{3.9}$$

Afterwards, we measure the Euclidean distance ξ between the attributes of the edge node chosen in $t-1$ compared to the current conditions of the other edge nodes within the overlapping regions $\eta_{v-1,k-1}$, based on Eq. 3.10. Considering the ξ value, we select the edge node with the highest value.

$$\xi = \sqrt{\sum_{k=1}^{K} \left(\eta_{v,k} - \eta_{v-1,k-1}\right)^2} \tag{3.10}$$

Based on the higher value of ξ, Fog4Video acknowledges the potential of an edge node to stream video content for a user when it meets QoE requirements and monetary cost. In this sense, Fog4Video informs its decision via the *Cloud Controller* to edge nodes and users, detailing which *Streaming Unit* the user must request the given chunk.

Each edge node embeds a set of modules and components for the multi-tier edge computing to assist the orchestrator. Fog4Video defines one phase for analysis and another for decision-making & execution to support a real-time dynamic orchestrator. For the last phase, an AHP method balances the multi-criteria inputs and executes the decision-making.

3.3 Fog4Video Evaluation

This section describes the evaluation methodology, including scenario description, simulation parameters, and metrics used to evaluate different VoD service orchestration schemes. We define the scenario and simulation parameters in Section 3.3.1. We discuss the results and findings in Section 3.3.2.

3.3.1 Scenario Description and Methodology

We implemented Fog4Video by using in the Network Simulator 3 (NS-3) on version 29 following the scenario presented in Figure 4. For the wired infrastructure, we

considered the partial topology of the Future Internet Brazilian Environment for Experimentation (FIBRE) testbed [18] to configure delays of the long-haul communication between edge nodes and the central cloud. We distributed the multi-tier edge nodes organized into tiers 1, 2, and 3 connected to the central cloud. Tier 3 is a *Local Tier* with a Wi-Fi AP in a local datacenter with resource availability of 10 Mbps for transcoding. Tiers 1 and 2 are *Regional Tiers* with a resource availability of 20 Mbps for transcoding, respectively. In the cloud, there is a powerful datacenter with a resource availability of 100 Mbps for transcoding. Figure 4 shows the delay of long-haul communication between cloud and tiers. For the wireless infrastructure, we consider Wi-Fi 802.11n APs, channel bonding of 40 MHz in the center of a square area of 50 m^2 providing access to 40 randomly distributed mobile devices.

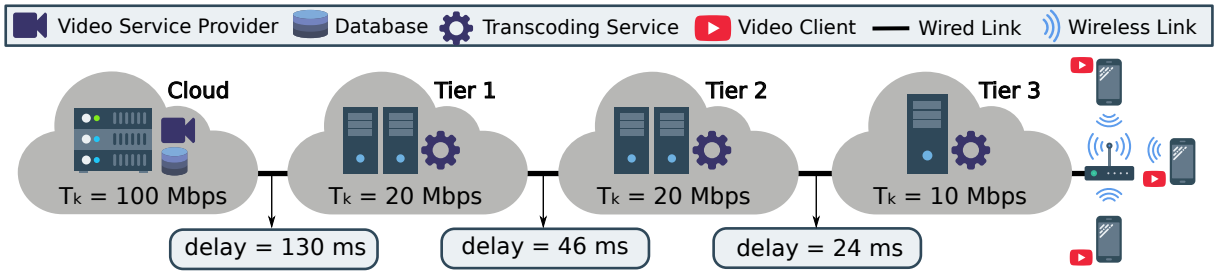

Figure 4: Multi-tier Edge Scenario.
Source: [51]

The mobile devices followed a linear continuous video request rate of 10 requests per second. By default, the video streaming initiates from the cloud node while the *Orchestrator* becomes aware of the QoE/QoS metrics of each user, such as stalling and delay. The adaptation algorithm is rate-based, where the bitrate starts from the lowest value and, for smoothing between each quality level, switches one level at a time following a conservative bitrate switching profile. The *Video Player* buffers at least 2000 ms, which is the size of a chunk. Therefore, as soon as the buffer does not have content to render (*i.e.,* stalling event), it has to re-buffer a complete chunk to play out the video again. We considered the Big Buck Bunny, Sunflower version video downloaded from the video library [81]. The *Video Player* at the user requests a video at a given time. Precisely, we use a High Definition video with a duration of 600 seconds, configured with 30 frames per second, and encoded into eight commonly used bitrates of 400, 650, 1000, 1500, 2250, 3400, 4700, and 6000 kbps [18], as shown in Table 5.

Table 5: Video Resolution and Bitrate Configurations

Resolution	180	360	360	540	540	720	1080	1080
bitrate (Mbps)	400	650	1000	1500	2250	3400	4700	6000

QoE metrics overcome the limitation of QoS metrics to capture aspects of VoD related to the human perception [82]. In this way, we apply well-known QoE metrics for VoD services, namely bitrate, bitrate switch events, number, and duration of stalls [19]. Due to the conservative behavior of the adaptation algorithm, we consider the first 20

chunks as the *initial* bitrate since the first chunk always starts with the lowest bitrate, the last 20 chunks for the *final* bitrate, and *average* bitrate of all the chunks. Depending on the orchestrator, an edge node, chosen with the best QoE and cost improvement potential, can reply to most of the requests. We consider the Jain Fairness index F to express the concentration of requests [83] to measure requests fairness between edge nodes for each scheme. The index calculation is denoted in Eq. 3.11, where x_i means the number of requests in edge node F_k.

$$F_k = \frac{[\sum_{i=1}^{k}(x_i)]^2}{k\sum_{i=1}^{k}(x_i)^2} \tag{3.11}$$

We also evaluate the cost C_k to deploy VoD services in a given edge node k, which is computed based on Eq. 3.2. The cost C_k for a given edge node k depends on the amount of time the *Streaming Unit* or *Transcoding Unit* uses resources for processing a chunk, causing a monetary cost M_k per hour of usage. In this way, we computed the monetary cost of CPU time per hour based on Amazon Web Services (AWS) Cost of Ownership Calculator[2]. The CPU time proportionally decreases when renting a higher number of CPU cores in the same AWS region. We considered the deployment by using Amazon Container Orchestration Service (ECS) into four regions and three setups and the monetary cost of each CPU core per hour, as well as memory and storage, are shown in Table 6.

Table 6: Monetary cost of CPU core time per hour based on Amazon ECS Setup

Node	CPU Cores	Memory	Storage	Cost
Cloud	40	160 GB	10 TB	$0,07272
Tier 1	8	32 GB	2 TB	$0,22896
Tier 2	8	32 GB	2 TB	$0,22896
Tier 3	4	16 GB	1 TB	$0,42408

We conducted 33 simulations for each of the three different edge service orchestrator schemes, namely, Random, Greedy, and Fog4Video. Then, we analyzed their impact on delivering VoD content with QoE support and provide a 95% confidence interval. All schemes leverage the resource availability of the edge nodes. The Random strategy chooses the edge nodes with equal chances among all of them. The Greedy scheme selects the edge node owning the smallest delay. Fog4Video evaluates the collected metrics to choose the best serving edge node.

3.3.2 Simulation Results

Figure 5 shows the number of users per tier for each individual chunk, *i.e.,* tier 3, tier 2, tier 1, and cloud. This analysis provides information about the edge node selection

[2]https://www.awstcocalculator.com/

behavior of each orchestrator and from each tier where the chunks were requested along the video playback. By analyzing the results, it is possible to conclude that all of the users start requesting video from the cloud. Afterwards, each scheme selects the *Streaming Unit* for the user to download the video in different ways. For instance, the Random strategy selects the tiers with a 25% probability since it is one tier between four candidates. On the other hand, the Greedy scheme prefers to select nodes with lower latency but picks the remaining tiers due to the resource availability in each tier.

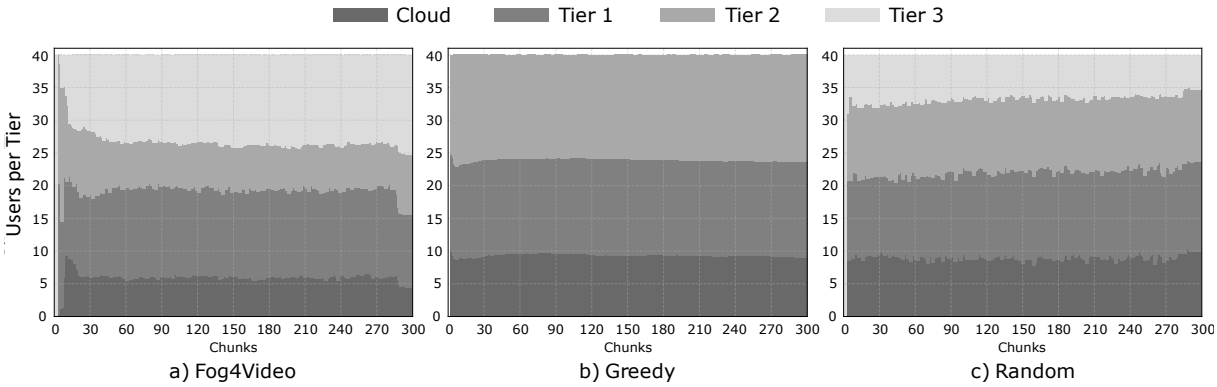

Figure 5: Number of users requesting video chunks from each tier.
Source: [51]

Finally, Fog4Video selects the appropriate *Streaming Unit* based on network, edge node, user information, and cost, leading to higher use of tier 2 and the cloud. Tier 2 and tier 1 have the same monetary costs, but the delay is better in tier 2, leading a more significant share. The use of tier 3 resources grows gradually before the 15th chunk because the bitrates of all users are small at the beginning. When the bitrate grows, the users move to other tiers capable of adapting the content as requested. In the last 15 chunks, the download of some users finishes, and other users start to request from more cost-effective tiers.

Figure 6 depicts the costs for the VoD service deployment C_k when the system is configured with different orchestrator schemes. By analyzing the costs, it is possible to conclude that Fog4Video reduces the cost by up to 24.04% and 16.32% compared to Greedy and Random, respectively. Fog4Video provides lower costs because it selects closer and more expensive tiers only when poor QoE is detected, despite the other schemes. On the other hand, Greedy and Random strategies do not follow this approach. In the last two cases, Greedy decides for the nodes with the lowest delays representing a closer distance between edge nodes and users. However, this decision incurs higher costs. The Random strategy has lower costs because of the lower number of users requesting from the more expensive tier.

Figure 7 shows the bitrate *initial* and *final*, as well as the *average* bitrate received by the user downloading the video via different orchestrator schemes, *i.e.,* Fog4Video, Greedy, and Random. The video starts with a lower bitrate, *i.e.,* 400 kbps, regardless of the scheme. We can also see that Fog4Video delivered the *final* bitrate up to 19.3% and 27.9% higher than Greedy and Random, respectively. The higher bitrate occurs be-

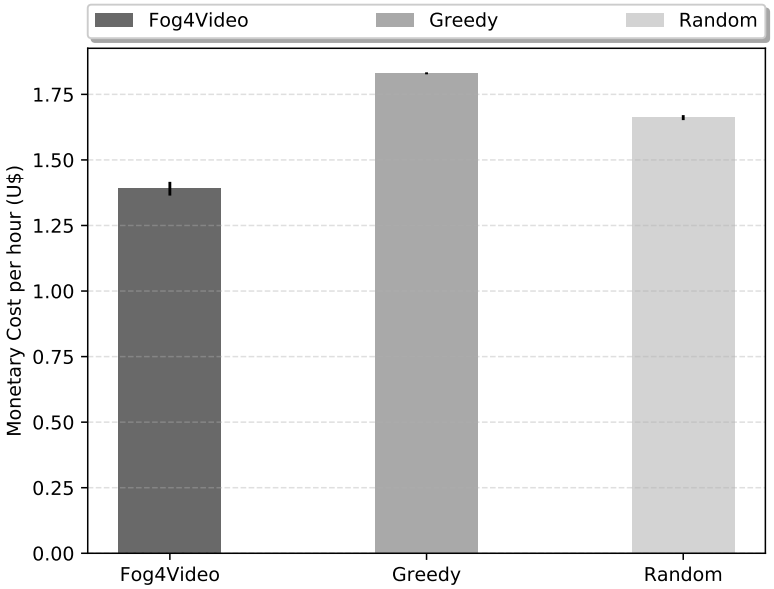

Figure 6: Cost to Deploy VoD Services.
Source: [51]

cause Fog4Video chooses the best *Streaming Unit* based on metrics like delay combined with QoE. The lower delay allows the adaptation algorithm to better predict the network conditions between the user and edge node with more frequent and updated information. Moreover, Fog4Video provides more fairness between the users, giving a better opportunity to users with worse QoE support, as shown in Figure 8. The fairness allows the users to have more room to increase their bitrate and thus to have a better bitrate than the users using Greedy and Random schemes. However, the AHP method needs around 15 chunks to evaluate the performance of each tier resulting in a lower *initial* bitrate for Fog4Video. Finally, the *average* bitrate delivered by Fog4Video is 30.91% and 35.05% higher than provided by Greedy and Random schemes, respectively. The average is higher because Fog4Video has a short period to adapt and converge. In this case, the users can request better bitrates earlier, when compared to those users using Greedy and Random, giving them a better overall bitrate.

Figure 8 shows Jain's Fairness Index for the user's distribution on edge nodes, which is computed by Eq. 3.11. The index shows the concentration of requests when the system is configured with different schemes. The Random scheme is the fairest because the probability to choose a tier is equal between all of them. However, this performance does not result in better QoE or cost-effective results by the scheme. In this sense, Fog4Video offers the best trade-off between application performance and fairness, achieving a high score on the fairness index while cost-effectively improving the quality level of VoD services. Fog4Video decides to allocate requests to edge nodes with more potential to enhance QoE, and the fairness stands because of the usage of cheaper edge nodes. Moreover, the Greedy scheme has a worse performance because it concentrates the requests to the closest nodes even though it does not necessarily reflect in a better distribution of VoD services.

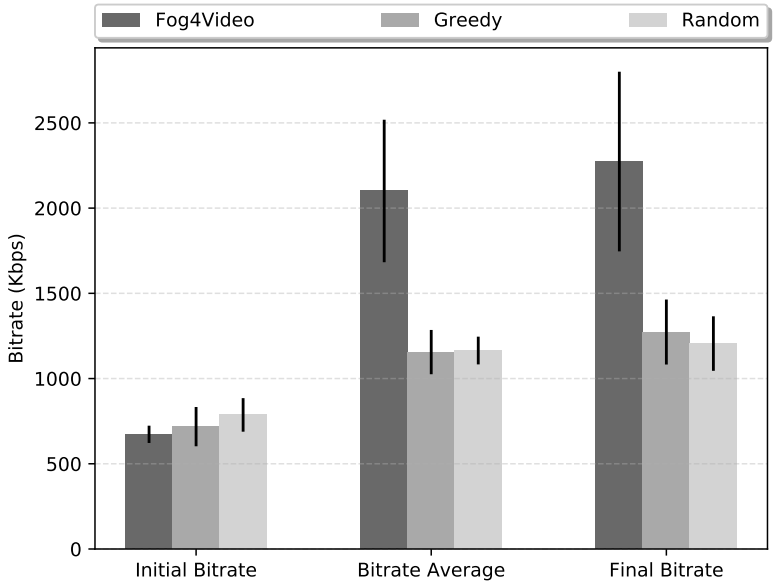

Figure 7: Impact of Orchestrator on Cost and Initial, Final, and Average Bitrate.
Source: [51]

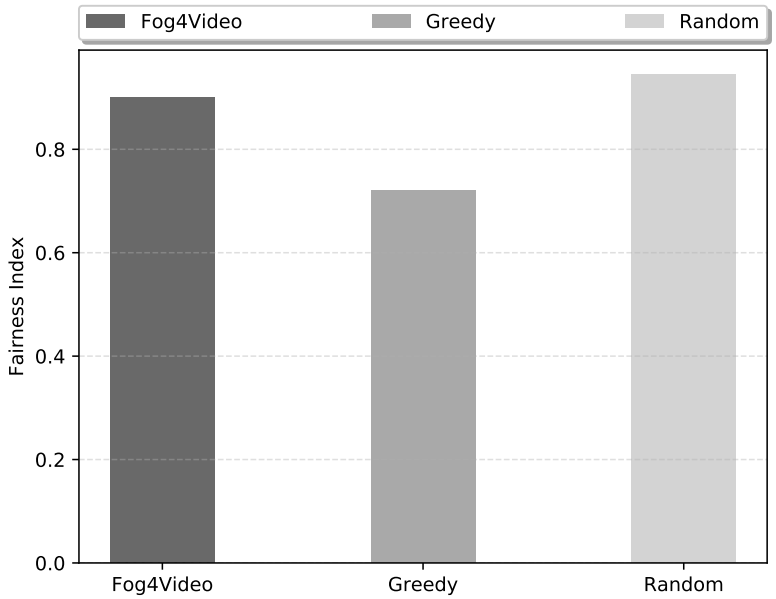

Figure 8: Fairness Index.
Source: [51]

Figure 9 shows the number of stalls per user for videos delivered by each service orchestrator scheme. Fog4Video reduces the number of stall events by 71.44% and 71.18% compared to Greedy and Random schemes, respectively. Number of stalls metrics have a significant influence on the quality of VoD services, where high values could result in the viewer most likely leaving the video service. The interruption is a direct consequence of buffer starvation at the player, which is caused by poor network conditions, *i.e.,* long delay, between the user and the *Streaming Unit*. By analyzing the results, we can see that Fog4Video delivered videos for 40 users with less than one stall per user. For example,

around 16 users experienced a single stall during the *Video Player*, which lasted about 0.68 seconds. The reduced number of stalls happens because Fog4Video proactively selects the best *Streaming Unit* based on network, edge node, and user information. The VoD metrics played an essential role in identifying how an edge node can potentially improve the user's satisfaction. On the other hand, the Greedy and Random schemes selected the *Streaming Unit* without considering such metrics, which do not avoid overloaded *Streaming Unit* for video delivery.

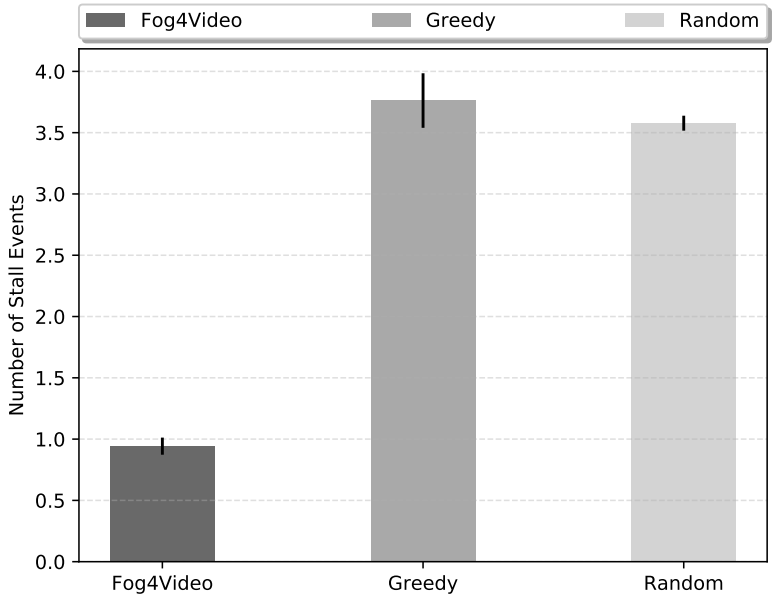

Figure 9: Impact of Orchestrator on the Number of Stalls per user.
Source: [51]

Figure 10 shows the duration of stalls per user for videos delivered by each service orchestrator scheme. Fog4Video reduces the duration of stall events by 72.45% and 65.23% compared to Greedy and Random schemes, respectively. The stall duration has a significant influence on the quality of VoD services, where high values could result in the viewer most likely leaving the video service due to long wait. The interruption is a direct consequence of buffer starvation at the player, which is caused by poor network conditions, *i.e.,* long delay, between the user and the *Streaming Unit*. By analyzing the results, we can see that Fog4Video delivered videos for 40 users with stall duration up to 1.2 seconds per user. The reduced stall duration happens because Fog4Video proactively selects the best *Streaming Unit* based on network, edge node, and user information. The VoD metrics played an essential role in identifying how an edge node can potentially improve the user's satisfaction. On the other hand, the Greedy and Random schemes selected the *Streaming Unit* without considering such metrics, which do not avoid overloaded *Streaming Unit* for video delivery.

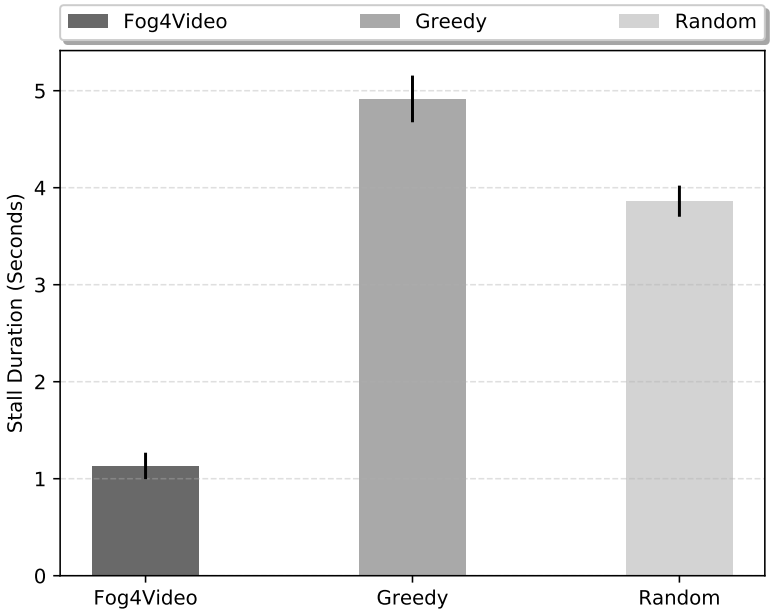

Figure 10: Impact of Orchestrator on the Duration of Stalls per user.
Source: [51]

3.4 Chapter Conclusion

In this Chapter, we introduced the Fog4video, which chooses an appropriate edge node to stream VoD streaming with QoE support in a multi-tier edge computing environment considering the network, edge node, and user's information for decision-making. Fog4Video orchestrates VoD services in a hierarchical network infrastructure improving the QoE. The hierarchy is modeled considering each tier's performance, location, and cost deployment. Fog4Video classifies each available edge node's networking and computing resources with the AHP method to assign different degrees of importance for each parameter to deliver VoD with a better quality level for the users.

From our evaluation analysis, we identified that Fog4Video delivered videos with up to 30% QoE improvement compared to other orchestrator schemes. When the system is configured with Fog4Video, the number of stall events reduces by up to 70%, and the stall duration is reduced by up to 65%. These results are an essential achievement of Fog4Video since stall duration, and stall events significantly minimize the most detrimental factors that affect user perception. Fog4Video also improved the average bitrate by up to 35% and reduced monetary cost by up to 24% compared to other orchestrator schemes.

CHAPTER 4

Monolithic Service Orchestration in Multi-tier Flying Edge Computing Environments with Quality of Service Support

The objective of this Chapter is to address research question: *2) How to orchestrate monolithic multimedia services for mobile users in flying multi-tier edge computing environments with QoS support?*

We introduce the FLYED service orchestrator proposal to provide monolithic multimedia services for mobile users in flying multi-tier edge computing environments with QoS support. FLYED chooses an appropriate *Mobile Tier* node (to cooperate or even replace *Local Tier* nodes on the ground network) to assist monolithic services with QoS support in a multi-tier edge computing environment considering the network, edge node and service requirements in its decision-making process. FLYED performs service orchestration in a hierarchical network infrastructure, where multi-tier edge nodes provide computing and networking resources to immersive multimedia and VoD services. The hierarchical design stands for mobility, energy constraints, processing capacity, and distance of node placement on each layer. FLYED classifies each available edge node's networking and computing resources into a multi-criteria rank, where it considers the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method to assign different degrees of importance for each parameter to improve the QoS for groups of poorly-served users. Section 4.1 introduce the proposed multi-tier edge architecture. Section 4.2 shows performance evaluation results in a simulation environment. Section 4.4 describes the final remarks of the Chapter.

4.1 Scenario Overview and System Model

5G/6G will support a diversity of new applications and will provide remote service processing outside HMD/HMG devices but in multi-tier edge computing environments to meet the computation-intensive and delay-sensitive requirements of future 5G/6G mobile applications [23]. Specifically, multi-tier edge computing provides computational resources to different services and other functions in *Local Tier*, avoiding uploading data to the central cloud [39]. However, user or application requirements might lead to high traffic demand in a particular region for a given period. Thus a high concentration of users or many user requests leads to more ubiquitous mobile communication in hard-to-reach and computationally demanding scenarios [84, 85].

In this context, UAVs with communication equipment and computing capabilities could act as *Mobile Tier* to assure connectivity and different QoS levels in the sky to mobile users on the ground space, minimizing the response time and providing high throughput to a set of applications, as shown in Figure 11. *Mobile Tier* differs from *Local Tier* by its capabilities for dynamically positioning a set of UAVs. *Mobile Tier* cooperates with *Local Tier* and provides connectivity and computational resources to designated areas to meet temporary and unexpected demand for QoS assurance [47].

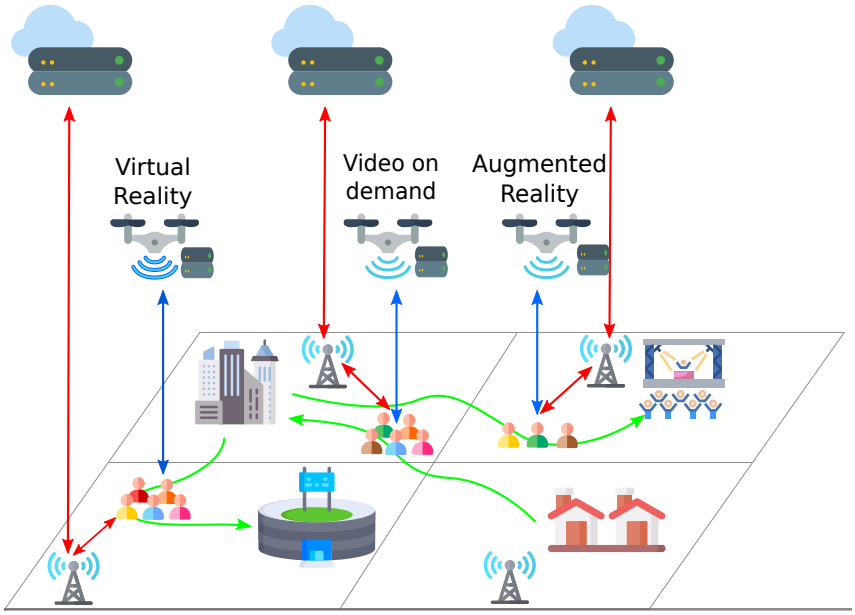

Figure 11: Multi-tier edge computing environment for mobile edge nodes and users. Source: author's own.

We consider a multi-tier edge computing, where *Local Tier* is in RAN devices, *i.e.,* macro cells or small cells such as expected in 5G networks [86] and *Mobile Tier* is in UAVs. In this scenario, a given mobile user $u \in U$ consumes a service denoted as $k \in K$ with different QoS requirements, which are modeled in terms of service priority l_k, delay threshold d_k, and throughput b_k. For example, an AR application has higher priority and demands a higher throughput to meet QoS levels for the users than other more simple

services such as VoD services. We also consider that each service k has a certain amount of bits of input size of s_k, and the number of service instructions per bit as z_k.

A set of *Mobile Tier* nodes V flying in the aerial space and equipped with communication and computation resources can be triggered to provide connectivity, computing, and storage services from the sky to ground mobile users. *Mobile Tier* nodes $v \in V$ and users $u \in U$ are mobile elements $i \in I = V \cup U$ and report their location $g_{i,t}$ at a given timestamp t, defined as a 4-tuple of geographical coordinates (x, y, z, t) in a 3D space, *i.e.,* Cartesian coordinate, and altitude above the ground. In addition, each mobile element moves with a given speed s_i ranging between a minimum $s_{i,min}$ and a maximum $s_{i,max}$ speed limit on a given trajectory (θ_i). The trajectory θ_i can be characterized by successive time-stamped locations $g_{i,t}$ obtained from geographical measurements, which is denoted as $\theta_i = \{g_{i,0}, g_{i,1}, ..., g_{i,t}\ \}$.

Eq. 4.1 calculates the service processing time in *Mobile Tier* node $d_{p,v,k}$ considering the service input size s_k, service CPU instructions per bit z_k, the offloaded percentage of a service to a *Mobile Tier* node $p_{v,k}$, and the service's number of CPU cycles per bit $p_{c,v,k}$ allocated in a *Mobile Tier* node [87].

$$d_{p,v,k} = \frac{p_{v,k} * s_k * z_k}{p_{c,v,k}} \tag{4.1}$$

Eq. 4.2 calculates the *Mobile Tier* node service response delay $d_{v,k}$ considering the delay to upload $d_{ul,v,k}$ and download data $d_{dl,v,k}$, and the service processing time $d_{p,v,k}$.

$$d_{v,k} = d_{ul,v,k} + d_{dl,v,k} + d_{p,v,k} \tag{4.2}$$

Eq. 4.3 calculates the amount of energy required for service processing in a given *Mobile Tier* node v based on *Mobile Tier* node capacitance value κ_v, the service's number of CPU cycles per bit $p_{c,v,k}$ allocated in a *Mobile Tier* node v, and the service processing time $d_{p,v,k}$. We rearrange the processing energy as a function of the percentage of remote service processing in a *Mobile Tier* node $p_{v,k}$, service input size s_k, service number of instructions per bit z_k.

$$e_{p,v} = \kappa_v * (p_{c,v,k})^3 * d_{p,v,k} = \kappa_v * p_{v,k} * s_k * z_k * (p_{c,v,k})^2 \tag{4.3}$$

Eq. 4.4 calculates the total energy consumption (e_v) of a given UAV v based on the required energy to fly $e_{fly,v}$ during a time, to upload service data $e_{ul,v}$, and to download service data $e_{dl,v}$. The total energy consumption of a *Mobile Tier* node to execute a service must not exceed the minimum energy required for the *Mobile Tier* node to reach the recharging base (e_{rb}).

$$e_v = e_{fly,v} + e_{ul,v} + e_{p,v} + e_{dl,v} \tag{4.4}$$

FLYED assumes the existence of a Software Defined Networking (SDN) controller [88] responsible for monitoring the network and multi-tier edge computing conditions of each service on the ground space. The controller triggers FLYED upon detecting poor QoS in a certain area of a smart city and informs the minimal service requirements of ongoing services, as shown in Figure 11. The controller groups all mobile users u in a cluster of poor QoS area as soon as their QoS parameters are below the minimum threshold for delay d_k, PDR loss pdr_k, or throughput b_k. The clustering formation maps a *Mobile Tier* node to poorly-served users considering an incremental clustering method based on K-means to find the ideal number of clusters to cover the area. Specifically, each cluster of poorly-served mobile users has a centroid. The clustering method checks whether the distance from a given user to the cluster's centroid is within the communication range of the *Mobile Tier* node. The centroids' positions of the clusters provide a reference distance to the *Mobile Tier* node reach. Each group of poorly-served users enters an ordered queue considering the priority level of the group l_k and the number of users of the cluster.

Upon detecting poorly-served mobile users, FLYED is triggered and will start the orchestration process based on collaborative work with *Local Tier* (ground space) and *Mobile Tier* (aerial space) nodes. FLYED chooses the suitable *Mobile Tier* nodes to best support users on the ground considering the trade-off between minimal QoS requirements and *Mobile Tier* energy issues. In addition, FLYED periodically adjusts the positions of *Mobile Tier* nodes to the best position in the multi-tier edge environments to offload multimedia services using a multi-criteria technique.

4.2 Multi-tier Flying Edge Computing Service Orchestration Scheme

FLYED distinguishes two phases, namely, *Mobile Tier* Selection and *Mobile Tier* Placement. In the *Mobile Tier* Selection phase, it determines the appropriate edge node to assist the poorly-served users' group based on the groups' average priority level and size, the proximity of the *Mobile Tier* node to the groups' centroid, and the remaining energy supply of the *Mobile Tier* node. Afterwards, this information provides the target position of *Mobile Tier* node, which determines the trajectory that maximizes the group's QoS.

4.2.1 *Mobile Tier* Selection Phase

The Selection phase triggers the *Mobile Tier* node to provide networking and computing services for mobile users with poor QoS experience. First, FLYED orders the clusters of poorly-served users based on service priority levels l_k. For each cluster, FLYED avoids cases where the UAV takes too long to move to the cluster's centroid or does not have enough energy to process the service during the entire service duration given by Eq. 4.4. Afterwards, FLYED uses TOPSIS [89] to find a *Mobile Tier* node to support each

cluster based on the distance D between a *Mobile Tier* node and the cluster centroid and battery (energy) percentage B of a *Mobile Tier* node.

TOPSIS combines quantitatively and qualitatively the input parameters by ranking the solution based on weights and impacts, where weight means the quantitative factor of criteria importance. TOPSIS decides based on a hierarchical structure in a matrix of pairwise combinations between the numerical values of each parameter and their relative degrees of importance. The pairwise comparison denotes the relevance of one parameter compared to the other, *e.g.,* if the parameter i is twice more important than parameter j, then j has 1/2 importance than i.

Based on an exploratory data analysis, we consider the decision matrix A, where lines and columns represent the input parameters used for decision-making, *i.e., D* and B, as shown in Eq. 4.5. We consider three importance levels (1/3, 1, and 3) of pairwise comparison. It is important to observe that the distance parameter D is three times the importance of the battery percentage B. Consequently, the battery percentage B is 1/3 times the importance of the distance D. Another factor is the diagonal of the matrix, which is always 1 because it compares the parameter with itself.

$$A = \begin{array}{c} \\ D \\ B \end{array} \begin{array}{c} \begin{array}{cc} D & B \end{array} \\ \begin{pmatrix} 1 & 3 \\ 1/3 & 1 \end{pmatrix} \end{array} \tag{4.5}$$

We set parameters weight set w_1, w_z for the distance and battery parameters by varying the weights in a hyper-parameter search space that provides a more successful SPR. As a result, we obtain the weight set as $0.27, 0.73$, which means that distance and battery have weights of 0.27 and 0.73, respectively.

Consider the input data matrix M, where rows refer to the input data for each *Mobile Tier* node v and columns indicate the criteria $z \in Z$. $Z = \{1, 2\}$, where $z = 1$ and $z = 2$ represents criteria D and B, respectively, as shown in Eq. 4.5. TOPSIS evaluates each row of matrix M with the decision matrix A in Eq. 4.6 regarding the criteria Z.

$$M_{v\,x\,z} = \begin{pmatrix} m_1 & \cdots & m_{1,z} \\ \cdots & \ddots & \cdots \\ m_{v,1} & \cdots & m_{v,z} \end{pmatrix} \tag{4.6}$$

The criteria B and D have values with a different range, and thus we normalize each value within the range [0,1] based on Eq. 4.7, constructing a normalized matrix M'.

$$m'_{v,z} = \frac{m_{v,z}}{\sqrt{\sum_{j=1}^{\eta(V)} m_{j,z}^2}} \quad v = 1, 2, ...\eta(V), z = 1, 2 \tag{4.7}$$

We weigh the parameters by forming a quadratic weight matrix W from weight set values. We form a line as $[w_1 \ ... \ w_z]$ and replicate each line until we can multiply matrix M' and W with dimension $v\,x\,z$ and $z\,x\,z$, respectively. After multiplying, we obtain the weight matrix $H_{v,z}$ as shown in matrix 4.8

$$H_{v\,x\,z} = \begin{pmatrix} h_1 & \cdots & m_{1,z} \\ \cdots & \ddots & \cdots \\ h_{v,1} & \cdots & h_{v,z} \end{pmatrix} \tag{4.8}$$

Afterwards, we need to find the parameter's best bst_z and worst wst_z values (*i.e.,* D and B) in the weighted matrix H. We compute the Euclidian distance E_{v,bst_z} between the row elements of matrix H related to *Mobile Tier* node v and parameter's best value bst_z in Eq. 4.9. In a similar manner, we compute the Euclidian distance E_{v,wst_z} between the row elements of matrix H related to *Mobile Tier* node v and parameter's worst value wst_z in Eq. 4.10

$$E_{v,bst_z} = \sqrt{\sum_{z=1}^{\eta(Z)} \left(h_{v,z} - h_{bst_z}\right)^2} \quad z = 1, 2, ..., \eta(Z) \tag{4.9}$$

$$E_{v,wst_z} = \sqrt{\sum_{z=1}^{\eta(Z)} \left(h_{v,z} - h_{wst_z}\right)^2} \quad z = 1, 2, ..., \eta(Z) \tag{4.10}$$

Finally, the closest Euclidian distance to the parameter's best value and the farthest Euclidian distance to the parameter's worst values denote the relative closeness ξ_v of a *Mobile Tier* node to the ideal solution. The TOPSIS score assumes the highest ξ_v to select the *Mobile Tier* node supporting a group of users based on Eq. 4.11. The next phase happens after FLYED has repeated this operation for each group from the highest to lowest group service priority average, discarding previously chosen or low-energy *Mobile Tier* nodes.

$$\xi_v = \frac{E_{v,wst_z}}{E_{v,bst_z} - E_{v,wst_z}} \tag{4.11}$$

4.2.2 *Mobile Tier* Placement Phase

The TOPSIS results will be used by the SDN controller, where it will quickly trigger the selected *Mobile Tier* nodes to fly for the desired area. The *Mobile Tier* node

leaves the cluster's centroid and moves to a position where more demanding users are. The *Mobile Tier* placement phase uses a PSO algorithm to define the best positions of *Mobile Tier* nodes during their QoS support missions. PSO is a fast algorithm that uses sets of possible *Mobile Tier* node locations called particle sets. The algorithm tries to recursively find the optimal position as an optimization problem involving local and global best fit for each of its *Mobile Tier* nodes [90]. Specifically, PSO considers an objective function, a search space, and a stop criteria. Each PSO element has its local fit, and the best local fit is the global best-fit position. PSO uses constants like inertia ι, the local fit value/exploration level φ_1 and neighborhood/global best-fit value/exploitation level φ_2.

We determine the current geographical position of a *Mobile Tier* node g_v with its coordinate along the map area that requests the *Mobile Tier* node service. Equation 4.12 presents the *Mobile Tier* node standard x, y, and z coordinates. The *Mobile Tier* node and the mobile user are aware of their positions.

$$g_v = [x_v, y_v, z_v] \tag{4.12}$$

The PSO algorithm determines the *Mobile Tier* node's future position g'_v by randomly spreading particles $g_{par} \in P$ into a 3D map and a movement speed m_{par} to the particles to find solutions in multiple directions, as shown in Eq. 4.13.

$$g'_{par} = g_{par} + \overrightarrow{m_{par}} \tag{4.13}$$

The updated particle movement m'_{par} depends on three main components, as shown in Eq. 4.14, namely: the inertial element ι; the personal best value influence φ_1; the global best value influence φ_2; two random numbers r_1 and r_2 from interval $[0, 1]$; the personal best position g_p of particle par; and the global best positions of all particles g_P.

$$m'_{par} = \iota * \overrightarrow{m_{par}} + \varphi_1 * r_1 \left(g_p - g_{par}\right) + \varphi_2 * r_2 \left(g_P - g_{par}\right) \tag{4.14}$$

The objective function calculates a 3-dimensional weighted centroid based on the gap between the position of the particle g_{par}, the near future position of the users g'_{par}, and the priority level l_k. Therefore, the algorithm minimizes the objective function, as shown in Eq. 4.15.

$$min \sum_{par=1}^{\eta(P)} l_k * \sqrt{\sum_{a=1}^{3} (g_{par,a} - g'_{par,a})^2} \tag{4.15}$$

Finally, groups of mobile users on the ground space will continue consuming services with QoS support from last-mile *Mobile Tier* nodes. One group lasts until users finish consuming the service or when users spread and dismantle the previous clustering

round. After the group dismantles, the *Mobile Tier* node can recharge in case of low energy based on Eq. 4.4 or return to the aerial space, where it can be selected for a new QoS support mission.

4.2.3 FLYED Algorithm

Algorithm 2 presents the FLYED operations, where we consider a set of users U, a set of *Mobile Tier* nodes V, and a set of poorly-served users G. The algorithm forms each group of poorly served users $g \in G$ calling the function Clustering at line 1. The cluster formation initializes with the minimum number of *Mobile Tier* nodes $v \in V$ equal to 1 at line 17. The function initializes the distance from users of the group to the group's centroid as infinity at line 18 and enters in a while until this distance from all users $v \in V$ of the group are in the communication range of the *Mobile Tier* nodes at line 19. A k-means method receives the maximum number of clusters and the set of users U and returns users' group $g \in G$ at line 20. The function updates the distance from users to their group's centroid at line 21. The search for the ideal number of groups within the communication range of the *Mobile Tier* node $v.com\_range$ breaks the loop if the number of clusters match the number of *Mobile Tier* nodes at lines 23-25. The function returns the set of groups G at line 27.

After finding the set of groups G, the algorithm organizes the groups in descending order based on the service priority average at line 2. In this order, the TOPSIS method finds the suitable *Mobile Tier* node to assist the given group at line 4 and calls the Placement function at line 4. The function receives the given group g, the suitable *Mobile Tier* node m, and the *distance* to the cluster's centroid at line 7. The controller sets the given *Mobile Tier* nodes m based on its policies at line 8 and finds the servicing position calling PSO method at line 9. The *Mobile Tier* node m starts host, process, and answer the services when it reaches the servicing position at lines 10-14. Finally, the *Mobile Tier* node broadcasts itself as idle if the service terminates at line 15.

4.3 Evaluation

This section describes the evaluation methodology, including scenario description, simulation parameters, and metrics used to evaluate different monolithic service orchestration schemes. We define the scenario and simulation parameters in Section 4.3.1. We discuss the proposal's results and findings in Section 4.3.2.

4.3.1 Scenario Description and Methodology

Simulation experiments were carried out by using the Python3 Scikit-learn library with the Network Simulator NS-3.36. We consider the following metrics to evaluate the performance of service orchestration schemes in a scenario with mobile edge nodes and

Algorithm 2: FLYED Algorithm.

```
   Input: U, V
 1 G = Clustering()
 2 for g.servicePriority ε order[highest, lowest] do
 3 |   for Vε TOPSIS[battery, distance] do
 4 |   |   Placement(g, v)
 5 |   end
 6 end
 7 Placement (g, v, distance):
 8 |   Controller associates v to assist g in a distance
 9 |   v.position = PSO(v, g)
10 |   while v at v.position do
11 |   |   v hosts service
12 |   |   v processes service
13 |   |   v answers service
14 |   end
15 |   v broadcasts itself as idle
16 Clustering (U):
17 |   n_clusters = 1
18 |   ∀ u.centroid_distance = ∞
19 |   while ∀ u.centroid_distance ∈ U ≥ v.com_range do
20 |   |   G = k_means(n_clusters, U)
21 |   |   forall u update(u.centroid_distance)
22 |   |   n_cluster++
23 |   |   if n_cluster = η(V) then
24 |   |   |   break
25 |   |   end
26 |   end
27 |   return G
```

users: SPR, PDR, jitter, delay, and energy consumption. SPR measures the percentage of users perfectly meeting their service requirements. PDR means the ratio between the total number of received packets divided by the total number of delivered packets. The jitter is the average packet delay variation. A smaller jitter value is also a desirable metric. The delay measures the average time of service in three phases, uploading from the user to the edge node, processing at the edge node, and downloading to the user, as stated by Eq. 4.2. The energy consumption $\in [0\%, 100\%]$ means the total energy consumed on average by each *Mobile Tier* node during the mission, as calculated in Eq. 4.4. Small energy consumption is preferable since UAVs may complete their mission and become ready for more actions without returning to the base.

We conducted 33 simulations to provide results with a 95% confidence interval. The scenarios support MEC nodes installed on 4 macro cells and 9 small cells positioned to entirely cover an area of 6 km^2 with 100, 200, and 300 mobile users. We limited the number of *Mobile Tier* nodes to 4, 8, and 12 with 250m maximum communication range, power 23 dBm, altitude 20m, LTE-A, and Hybrid Buildings propagation loss for transmission parameters. The *Mobile Tier* nodes comprise a unit processing 60W Nvidia Jetson AGX Orin with 64GB RAM and 275 TOPS, and a battery of 350 kJ. All tiers support the same multimedia services, including caching, encoding, and transcoding multimedia services. The simulation lasts 300s, and mobile users randomly request one of the services during the first 10 seconds of the simulation and follow the mobility trace of the city of Cologne, Germany. Table 7 summarizes the simulation parameters.

Table 7: Simulation Parameters of Monolithic Service Orchestration Schemes.

Parameters	Values
Scenario Area	6 km^2 (3000m X 2000m)
UAV Speed	13 m/s
Number of UAVs	5, 8 or 12
Number of users	100/200/300
UAVs Transmission Power	23 dBm
UAVs Height	20 m
UAVs Type of Transmission	ITU's Line-of-Sight (LOS)
UAVs Maximum LOS	250 m
Propagation Loss Model	Hybrid Buildings
PHY / MAC	LTE-A
Simulation Time	300 s
UAV Battery (kJ)	350

The performance of FLYED is compared to a scenario with only one *Reginal Tier* node and several *Local Tier* nodes and with existing orchestration schemes, namely, Tang et al. [58], and Pandey et al. [47] implemented by us in the NS-3. Specifically, FLYED performs its orchestration procedures based on service priority, *Mobile Tier* node energy, and distance to the user. Tang et al. [58] orchestrate services of poorly-served users by using K-means to group users into clusters covering unlimited areas. The cluster's centroids are the target location of the *Mobile Tier* node. It also orchestrates services

using a PSO technique unaware of mobility to assign *Mobile Tier* nodes, but we adapted the proposed scheme to scenarios with mobile users. Pandey et al. [47] consider an incentive game mechanism to assign *Mobile Tier* nodes to each node individually without concerns about QoS, mobility, and energy consumption. [47] focuses on the minimal distance between the nodes.

Table 8 summarizes the services used in the experiments, as well as, their priorities and QoS (minimal delay and throughput) requirements. Mobile users request a service following a Poisson distribution. The controller triggers service orchestration schemes when a mobile user's minimal delay or loss needs cannot be assured by *Local Tier* nodes on the ground space. On average, one *Mobile Tier* node can simultaneously provide computing and networking services for up to 8 mobile users. We consider three distinct service types [23], *i.e.,* AR, VR, and VoD. AR has the highest throughput, lowest delay, and priority level $l_k = 3$. VR has the second highest throughput, similar delay with AR, and priority level $l_k = 2$. VoD has the least throughput, highest delay tolerance, and a priority level $l_k = 1$.

Table 8: Services Requisitions.

Data Type	Priority	Delay	Throughput
AR	3	13ms	97 Mbps
VR	2	13ms	25 Mbps
VoD	1	250ms	10 Mbps

4.3.2 Simulation Results

Figure 12 shows the SPR time series for 200 mobile users and 12 *Mobile Tier* nodes. The green area shows the percentage of well-served users of the *Local Tier*. The colors blue, red, and pink show the percentage of well-served users of the *Mobile Tier* by the compared works and the white color in the remaining area to reach 100% denotes the percentage of poorly-served users. FLYED achieves the highest SPR compared to Pandey and Tang. FLYED finds, places, and manages *Mobile Tier* nodes efficiently, achieving up to 8% better SPR than Pandey and Tang. In summary, FLYED achieved good QoS up to 200 users, but other works only supported 100 users because of their poor QoS detection and *Mobile Tier* placement approaches. *Local Tier* fastly needs *Mobile Tier* assistance, and *Mobile Tier* schemes take a few seconds to enable the selected *Mobile Tier* node to reach the effective place of assistance. FLYED finds the best compromise to stay close to users with the highest service priorities and outperforms the existing works for all services and scenarios.

Figure 13 shows the impact on PDR of orchestrating the *Local Tier* alone and the tier combined with FLYED, Pandey, and Tang with 12 *Mobile Tier* nodes. FLYED achieves the highest PDR of users needing *Mobile Tier* assistance because it can select an appropriate number of *Mobile Tier* nodes operating simultaneously. In addition, FLYED creates groups composed of poorly-served users considering the group's mobility and the

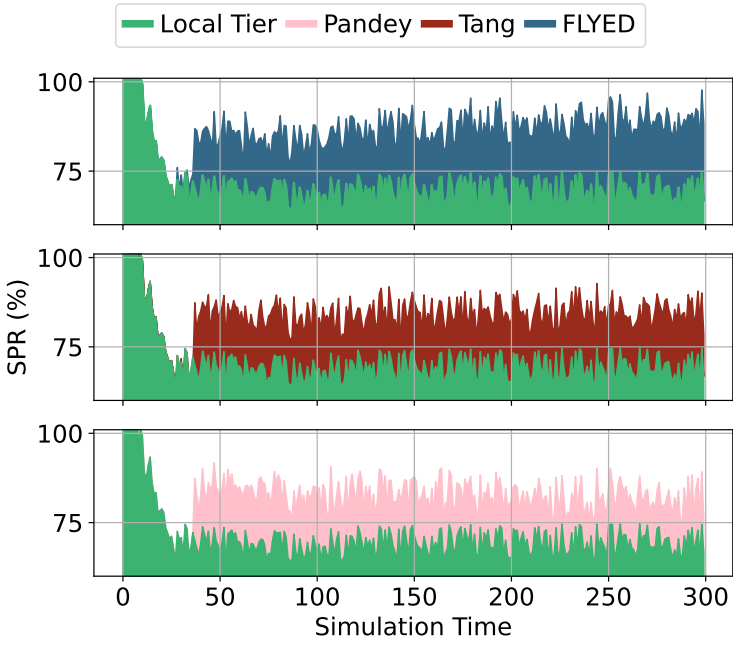

Figure 12: SPR for 200 users.
Source: author's own.

distance from *Mobile Tier* node to users shorter than the communication range. Creating groups covering an inappropriately large area is more likely to cause interference between the static BSs and the *Mobile Tier* nodes. The PDR shows the importance of placing *Mobile Tier* nodes far from static BSs to improve PDR and, consequently, the effective throughput because more packets can reach the destination during a time window.

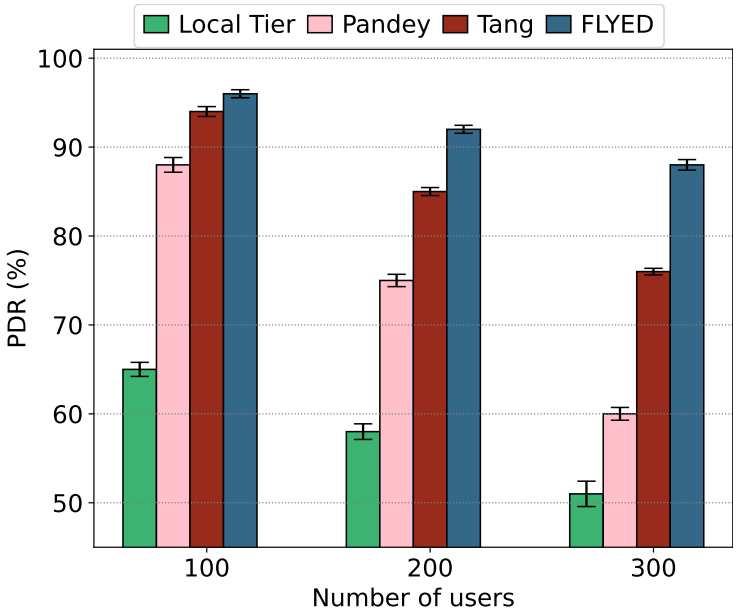

Figure 13: PDR for a Scenario with 12 *Mobile Tier* nodes.
Source: author's own.

Figure 14 shows the impact on PDR of orchestrating the *Local Tier* alone and the tier configured with FLYED, Pandey, and Tang with 8 *Mobile Tier* nodes. Reducing

the number of *Mobile Tier* to 8 nodes leads to a PDR decrease by up to 5% on average compared to 12 *Mobile Tier*. The reduction happens because fewer edge nodes must assist a similar service load in all scenarios. The *Mobile Tier* nodes need to move farther distances, take longer to reach the desired placement position, and users remain poorly served for longer. Additionally, the standard deviation shows that service assistance is slightly more unstable.

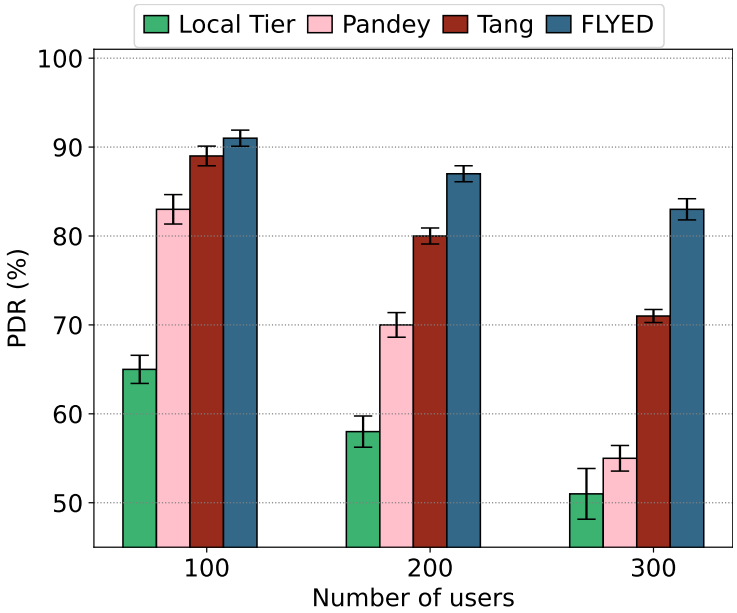

Figure 14: PDR for a Scenario with 8 *Mobile Tier* nodes.
Source: author's own.

Figure 15 shows the impact on PDR of orchestrating the *Local Tier* alone and the tier configured with FLYED, Pandey, and Tang with 4 *Mobile Tier* nodes. The impact of 4 *Mobile Tier* nodes on PDR increases proportionally to the number of nodes. In this way, the PDR reduces by up to 8.5% on average compared to 12 *Mobile Tier* nodes because of the higher networking and processing overloads. The *Mobile Tier* nodes lack in providing enough computing resources for many users. The service orchestrator likely chooses all *Mobile Tier* nodes if available.

Figure 16 shows the impact on the delay of orchestrating the *Local Tier* alone and the tier combined with FLYED and related works with 12 *Mobile Tier* nodes. All scenarios but the one with 300 users had a delay below 13 ms. FLYED provided less interference among BSs and had fewer retransmissions, saving more resources blocks of the LTE time division scheme. In this way, the network remains uncongested longer and supports more users with high QoS at the same time. FLYED selects *Mobile Tier* to offload network traffic from congested BSs and coordinates the handover process to the *Mobile Tier* node. The related works depend on Signal-to-Interference-plus-Noise Ratio (SINR) of the antenna to connect in the network. They often move a *Mobile Tier* to the service position, but the user device interprets a better coverage from the static BS. This process wastes *Mobile Tier* resources and leads to poor *Mobile Tier* performance.

Figure 17 shows the impact on the delay of orchestrating the *Local Tier* alone

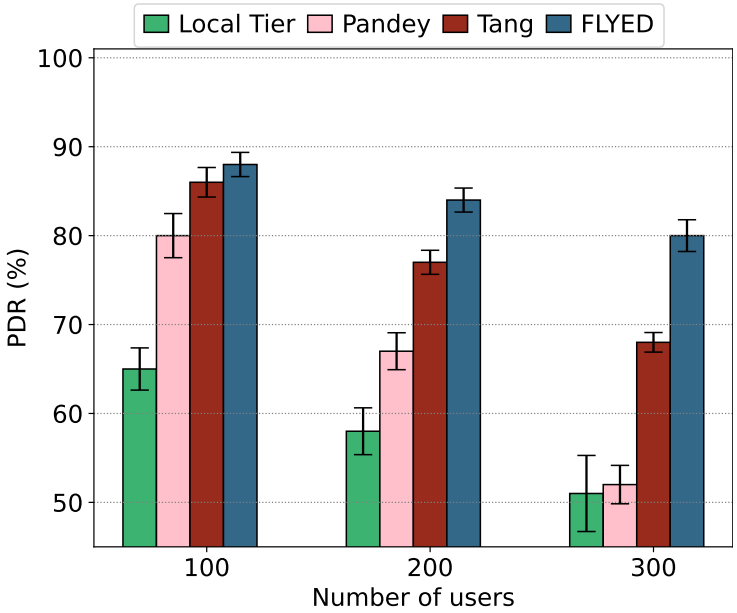

Figure 15: PDR for a Scenario with 4 *Mobile Tier* nodes.
Source: author's own.

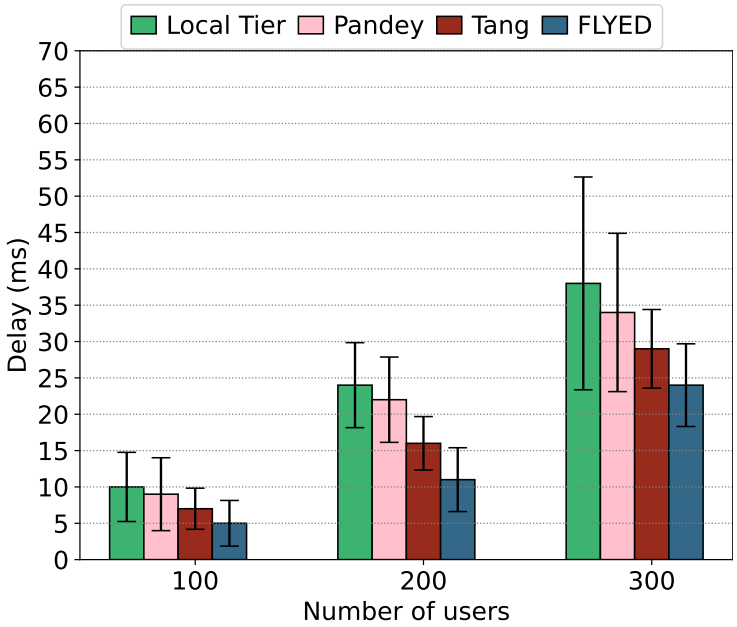

Figure 16: Delay for a Scenario with 12 *Mobile Tier* nodes.
Source: author's own.

and the tier combined with FLYED and related works with 8 *Mobile Tier* nodes. FLYED provides the lowest delay on average for all scenarios with 8 *Mobile Tier* nodes. However, only the case with 200 users has a delay above 15 ms on average, and most AR and VR users are poorly served. In this way, users have more frequent periods of poor service in compared works than in FLYED. The delay is up to 4.5 ms higher than the scenario with 12 *Mobile Tier* nodes. In this case, only the scenario with 100 users could meet the delay requirements on average. Additionally, the compared works have more standard deviation reflecting more service instability.

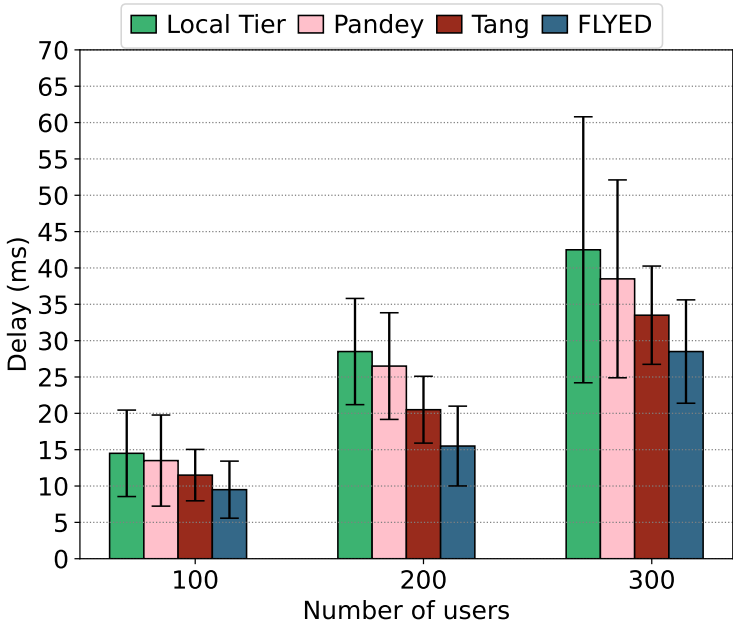

Figure 17: Delay for a Scenario with 8 *Mobile Tier* nodes.
Source: author's own.

Figure 18 shows the impact on the delay of orchestrating the *Local Tier* alone and the tier combined with FLYED and related works with 4 *Mobile Tier* nodes. FLYED provides the lowest delay on average for all scenarios with 4 *Mobile Tier* nodes. However, all cases with 100, 200, and 300 users have a delay above 15 ms on average, and AR and VR requirements can not be met. In this way, users have more frequent periods of poor service than FLYED. The delay reaches up to 9.3 ms higher than the scenario with 12 *Mobile Tier* nodes. Only the scenario with 100 users could meet the delay requirements on average for FLYED. Additionally, the compared works have more standard deviation reflecting more service instability.

Figure 19 shows how the PDR improves when the *Mobile Tier* contains 4, 8, and 12 nodes of a randomly selected group of poorly served users needing assistance on the 200 users scenario. *Local Tier* supports 100 users with good QoS but remaining 100 more users needs *Mobile Tier* assistance. The service orchestration schemes activate a *Mobile Tier* node after 7 seconds of simulation, and the suitable *Mobile Tier* node takes 12 seconds to reach the service position. Close to reaching the position, the *Mobile Tier* node starts to provide the service and improve the PDR metric. FLYED provides better PDR because the *Mobile Tier* node stays closer to the users with the highest service priorities, and the same *Mobile Tier* node follows the users during the entire simulation. A slightly better performance for 12 nodes is achieved because each *Mobile Tier* needs to support fewer users.

The energy consumption on each *Mobile Tier* node depends on 4 factors. Regarding distance, the energy consumption depends on the number of poorly served users and the maximum number of *Mobile Tier* nodes. Regarding processing load, the energy consumption depends on the number of poorly served users and the maximum number of *Mobile Tier* nodes to divide the total load. In our scenario, the processing unit takes

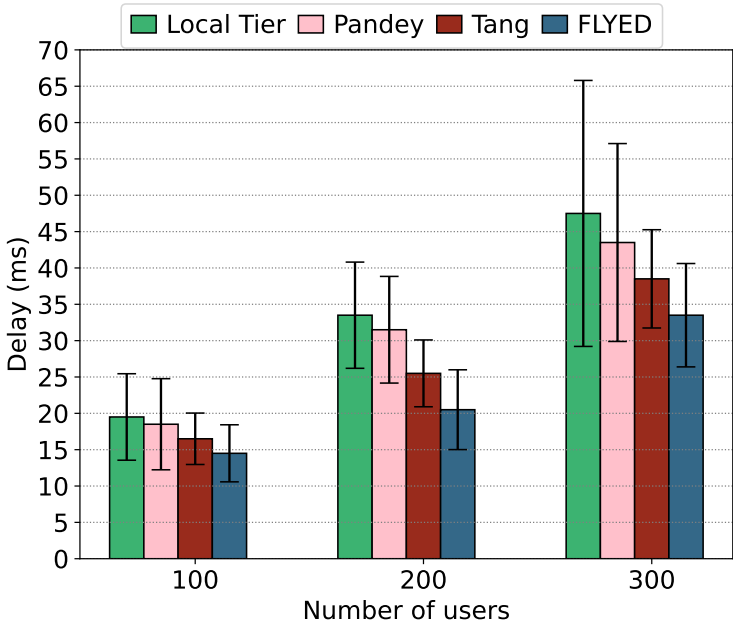

Figure 18: Delay for a Scenario with 4 *Mobile Tier* nodes.
Source: author's own.

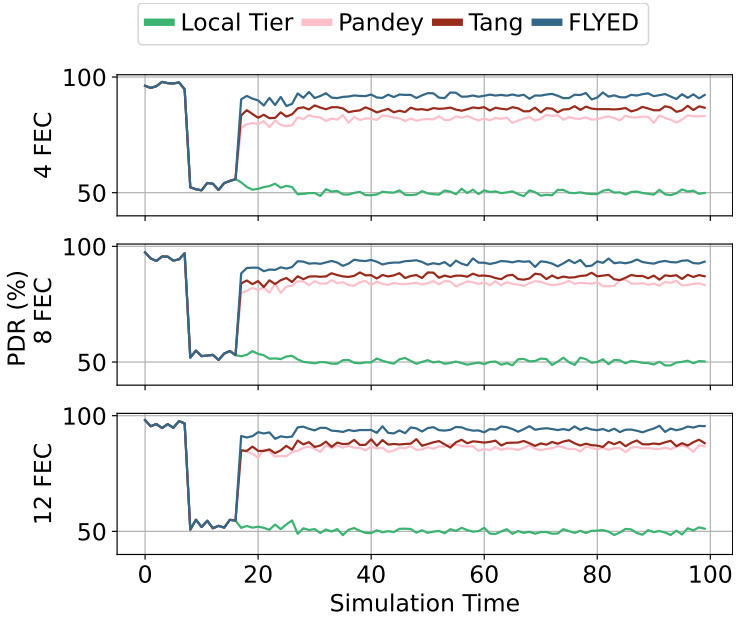

Figure 19: Delay over the time for a scenario with 200 mobile users.
Source: author's own.

considerable energy to process AR and VR services based on the bits processed. The energy consumption is up to four times higher than the energy to fly.

Figure 20 shows the average impact on energy consumption on each *Mobile Tier* node for the scenario with 12 nodes. FLYED chooses the poorly served users based on the placement distance and remaining energy. FLYED energy consumption increases by up to 6.2% and 9.4% on average, comparing the scenario with 100 users to 200 and 300 users, respectively. Additionally, FLYED has reduced the energy consumption by up to 9.1% compared to related works in the same conditions. The scenario with 12 *Mobile Tier*

tiers has the highest computational capacity, and the average consumption per *Mobile Tier* node is the lowest.

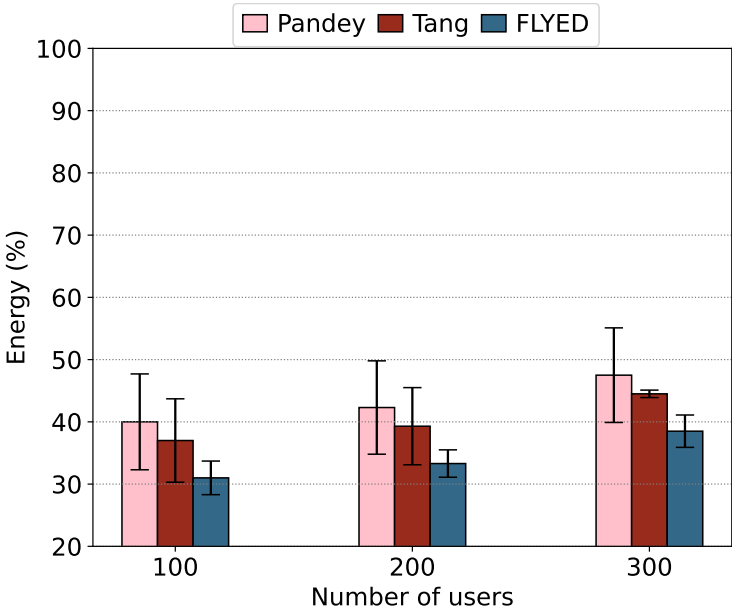

Figure 20: Energy consumption for a scenario with 12 *Mobile Tier* nodes. Source: author's own.

Figure 21 shows the average impact on energy consumption on each *Mobile Tier* node for the scenario with 8 nodes. FLYED had the lowest energy consumption compared to related works. FLYED energy consumption increases by up to 5.2% and 10.3% on average, comparing the scenario with 100 users to 200 and 300 users, respectively. Additionally, FLYED has reduced energy consumption by up to 8.7% compared to related works in the same conditions. The scenario with 8 *Mobile Tier* tiers has the mid-term computational capacity. The average consumption per *Mobile Tier* node has a higher processing load than the scenario with 12 *Mobile Tier* nodes. Specially, the average processing load is closer to the scenario with 4 *Mobile Tier* nodes than 12 *Mobile Tier* nodes.

Figure 22 shows the average impact on energy consumption on each *Mobile Tier* node for the scenario with 4 nodes. FLYED energy consumption increases by up to 11.5% and 19.3% on average, comparing the scenario with 100 users to 200 and 300 users, respectively. Additionally, FLYED has reduced energy consumption by up to 9.3% compared to related works in the same conditions. The average consumption per *Mobile Tier* node is the highest, leading to frequent usage of the full processing capacity of the *Mobile Tier* nodes. Especially, the energy consumed during the simulation was up to 92% for Pandey and 81.7 % for FLYED in the scenario with 300 users. The scenario with 4 *Mobile Tier* tiers has the lowest computational capacity, and, on average, each node consumes more energy.

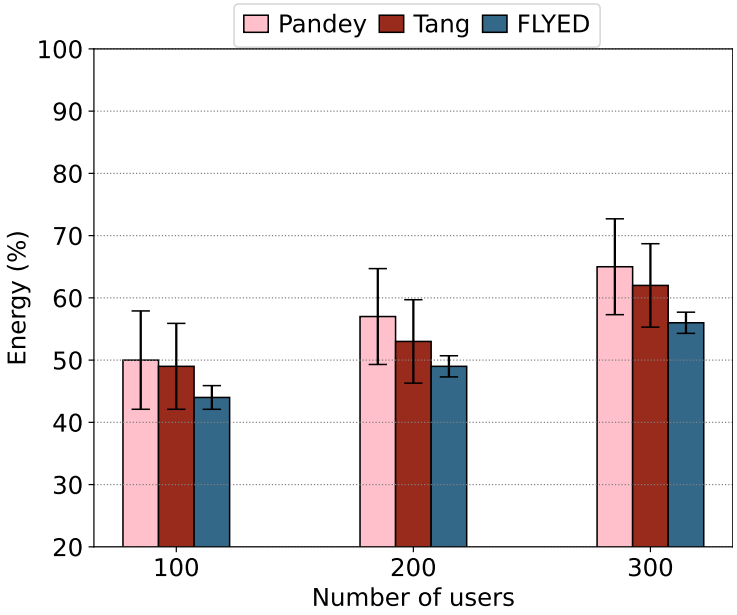

Figure 21: Energy consumption for a scenario with 8 *Mobile Tier* nodes. Source: author's own.

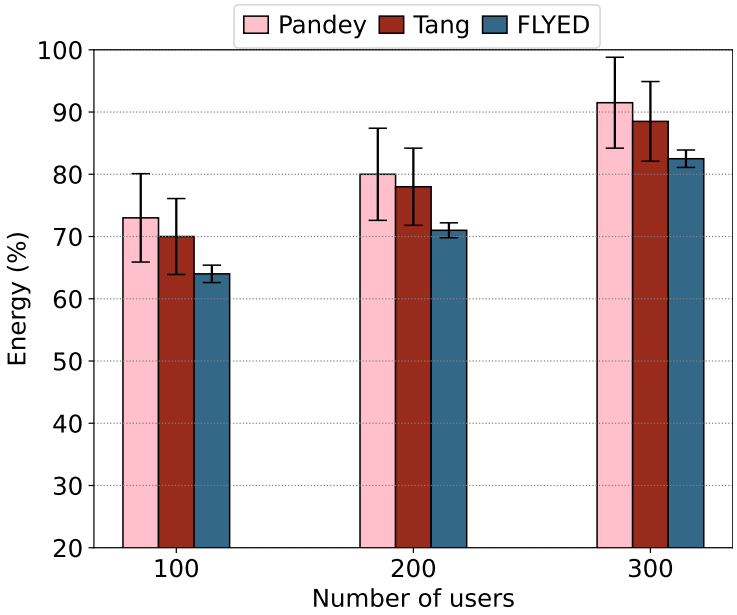

Figure 22: Energy consumption for a scenario with 4 *Mobile Tier* nodes. Source: author's own.

4.4 Chapter Conclusion

This Chapter introduced the FLYED orchestrator, which chooses an appropriate *Mobile Tier* node to cooperate or even replace *Local Tier* nodes on the ground network to assist monolithic services with QoE support in a multi-tier edge computing environment considering the network, edge node, and service requirements for its decision-making process. FLYED orchestrates service in a hierarchical network infrastructure, where multi-tier edge nodes provide computing and networking resources to immersive multimedia and

VoD services while improving the QoS. The hierarchical design classifies mobility, energy constraints, processing capacity, and distance of node placement on each layer. FLYED classifies the connectivity and resources of each available edge node based on a TOPSIS method to assign different degrees of importance for each criterion to provide better service for groups of poorly-served users.

Simulation results showed that FLYED improved SPR by up to 8%, and decreased PDR, delay, and energy by up to 28%, 40%, and 10%, respectively, when it is compared to Pandey and Tang. In general, FLYED had better results compared to Pandey and Tang proposals. The impact of the number of users and *Mobile Tier* nodes on FLYED can be summarized as follows: the impact of the number of users on PDR was up to 28%; the impact of the number of users on delay was up to 13.9 ms; the impact of the number of users on average energy consumption was up to 19%; the impact of the number of *Mobile Tier* nodes on PDR was up to 8%; the impact of the number of *Mobile Tier* nodes on delay was up to 10.4%; the impact of the number of *Mobile Tier* nodes on average energy consumption was up to 44.1%.

CHAPTER 5

Service Function Chaining Orchestration in Mobile Multi-tier Edge Computing Environments with Quality of Service Support

The objective of this Chapter is to address the research question: *3) How to orchestrate SFC for immersive multimedia services in mobile multi-tier edge computing environments with QoS support?*

This Chapter introduces MSF orchestrator to choose appropriate *Regional Tier* and *Local Tier* nodes to host SF chained services with QoS and mobility support [65] [1]. MSF considers network, edge node, and service requirements during its decision-making process. MSF performs its SFC orchestration procedures based on a hierarchical network infrastructure, where multi-tier edge nodes provide computing and networking resources to mobile users, improving the QoS for immersive multimedia services. The hierarchical design stands for each tier's processing, networking, and storage capacities. MSF classifies networking and computing resources of each available edge node into a multi-criteria rank, where each considers a heuristic based on dynamic programming to orchestrate computing and networking resources. MSF aims to minimize the delay while meeting computational resources requirements, multiple destinations SFC requests, and provides mobility support with dynamic instantiation of immersive multimedia services.

The remainder of this Chapter is organized as follows: Section 5.1 introduces the system model and scenarios together with a presentation of MUVR and MUAR services. Section 5.2 describes the MSF orchestration scheme. Section 5.4 discusses the final

[1] Partially reproduced in this chapter –

remarks of the Chapter.

5.1 Scenario Overview and System Model

MUVR and MUAR connects several users at the same time, interacting with the same VOs, requiring synchronized interaction in a short response time [91, 22]. MUVR relies on HMD to display fully covering the user's eyes and tracking their head movement, allowing them to look around a virtual environment as if they were there but in static scenario. On the other hand, MUAR enables interaction of multiple users with VOs in the same virtual world, which relies on HMG devices embedded with a semi-transparent display allowing a user to see the real world and sensors to track the user's surroundings and overlay computer-generated images, videos, or animations onto the real world even when they move [16].

MUVR and MUAR SFC can rely on an SF chaining orchestration scheme to monitor sessions, instantiate SFs in a given edge node, establish routes for a given set of parallelizable, ordered, and location-based SFs. Figure 23 shows a mobile user requesting an SF chain in a multi-tier edge computing environment with two tiers, *Regional Tier* (noted as RT in 23) and *Local Tier* (noted as LT in Figure 23). The SFC starts receiving the feedback data from mobile users and synchronizes users in SF_1 located in the source node, where this SF can be instantiated anywhere starting from the *Regional Tier* node or the closest *Local Tier* node. The SFC has four linear chained SFs, where SF_1 and SF_2 are instantiated in the *Regional Tier* RT_1 and SF_3 and SF_4 are instantiated two hops away from SF_2 in the *Local Tier* node LT_2. The SF chain bifurcates into two parallel SFs, where SF_5 is instantiated in node LT_3 and SF_6 in node LT_4 The parallel SFs converge to SF_7 linked to SF_8 located in the destination node, where both are instantiated in LT_5. Finally, the data flows addressed to the mobile user are organized into a session with common VOs. Each user in the set is associated with its edge node, namely the destination node. The SFC orchestration looks to instantiate the SF chain to the destination nodes of nearby interacting users.

In terms of a multi-tier edge computing environment, *Regional Tier* and *Local Tier* nodes can be modeled on an undirected graph $G = (V, E)$ for a set of edge nodes V and links E to interconnect edge nodes. For each pair of edge nodes v and $v' \in E$, we denote link latency as $d_{vv'}$, total bandwidth capacity as $b^c_{vv'}$, free bandwidth capacity as $b^f_{vv'}$, and used bandwidth capacity as $b^u_{vv'}$, where $b^c_{vv'} = b^f_{vv'} + b^u_{vv'}$. For a given edge node $v \in V$, we denote the total CPU capacity as p^c_v, CPU capacity free as p^f_v, and CPU capacity used as p^u_v, where $p^c_v = p^f_v + p^u_v$. We represent the total high-speed storage capacity as m^c_v, free storage capacity as m^f_v, and used storage capacity as m^u_v of node $v \in V$, where $m^c_s = m^f_s + m^u_s$.

We consider a set of users U consuming an SFC of a session $s \in S$ with a one-way delay constraint u^d. In this sense, we denote user location l for a session s as $u^s_l \in U$. Each session s contains a tuple of information: user id, user position, and SF sets. We model

the SFC on a directed graph $G^a = (V^a, E^a)$. We decompose a SFC into two parallel flows from SF_1 to SF_8 through SF_5 and SF_6 into two finite sets of ordered and linear SFs, each set as $SF = sf_1, ..., sf_j$.

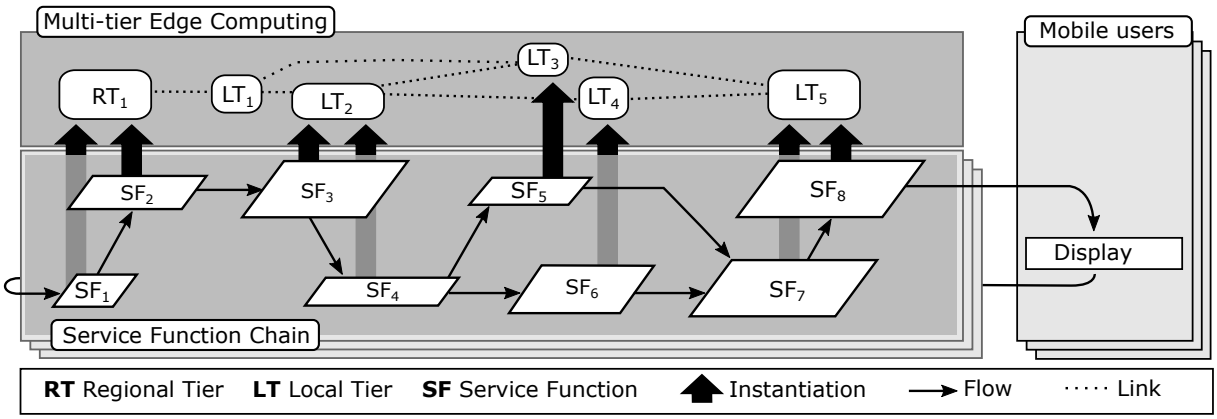

Figure 23: SFC orchestration in multi-tier edge computing.
Source: author's own.

We define two instantiation queues to store new and ongoing SFC session information. The first is a priority queue Q_m for re-instantiation of ongoing sessions of mobile users, and the second is a regular instantiation queue Q_i for a new session. Both queues receive a tuple t composed of $W(v, sf_j)$ for each SFC. We denote $W(v, sf_j)$ as a route for a given user to consume SFC from a set of SF, *i.e.,* the ordered edge nodes that provide each sf_j. We denote the Boolean matrix $N(v, sf_j)$ with true values to track the connectivity of edge nodes v capable of hosting SF sf_j between two ordered SFs. We denote $D(v, sf_j)$ as the cumulative latency of the chosen route of edge nodes of the entire SFC. We denote matrix $O(v, sf_j)$ as the temporary state of an edge node topology with remaining computational resources after instantiating an SF sf_j at edge node v.

We aim to minimize the delay on matrix $D(v, sf_j)$, where the Boolean decision variable α_{v,sf_j} denotes the instantiation of sf_j in edge node v. We consider the link bandwidth, processing, storage, and delay threshold constraints in Equations 5a, 5b, 5c, and 5d, respectively.

$$Min \sum_{j=1}^{J} \sum_{v=1}^{V} D(v, sf_j) * \alpha_{v,sf_j} \tag{5.1}$$

Subject to:

$$b_{vv'} \geq sf^{b}_{j-1,j}, \quad \forall v \in V \tag{5.1a}$$

$$p^{f}_{v} \geq sf^{p}_{j}, \quad \forall v \in V, \forall j \in J \tag{5.1b}$$

$$m^{f}_{v} \geq sf^{m}_{j}, \quad \forall v \in V, \forall j \in J \tag{5.1c}$$

$$\sum_{j=1}^{J} \sum_{v=1}^{V} D(v, sf_j) * \alpha_{v,sf_j} \leq u^{d} \tag{5.1d}$$

Each SF has specific processing, caching, and bandwidth requirements. MUVR and MUAR services have an order of execution and can be organized into an SF chain. The SFC needs an orchestrator to map and instantiate SFs at network edges. The SFC orchestration scheme must consider specific order and partially parallelize the SF execution. More details about MUVR and MUAR are described in the following sections.

5.1.1 MUVR Service in Multi-tier Edge Computing

MUVR consists of several processing components to display and handle objects into a HMD virtual space. In interactive MUVR, different users typically have identical perspectives of the same objects. For instance, highly dynamic MUVR scenarios show more than 30% similarity when users are near the same points of interest and exceed a redundancy ratio of 50% in the pixel values for consecutive frames [13]. MUVR plays an essential role in allowing user interaction between themselves [92]. Therefore, MUVR streaming to multiple and synchronized users needs to handle the same views and react to each other actions with low latency. Central cloud computing servers have response time higher than 20 ms [25], which is not suitable for MUVR. In this way, MUVR moves parts of its execution to multi-tier edge computing, consisting of several highly connected datacenters on the network edge to provide users with closer but limited computing resources. MUVR is a computing and networking-intensive service that can rapidly consume all edge resources.

A MUVR service can work as a chain, where existing elements perform specialized functions, such as syncing users, rendering, proactive rendering, encoding, and transcoding [23]. Such specialized elements on the MUVR chain could avoid redundant processing and transmission of many panoramic frames at the network edge. In this context, edge computing and networking resources may quickly be overloaded, mainly because an MUVR view is panoramic and has 60 frames per second, where each frame could contain 2GB of raw data. Each specialized element in order of MUVR can be allocated into SF containers to form a SFC request. An SFC request is a set of SFs for a given set of users. For instance, an SF can serve multiple users for frame similarities reaching up to 90% [35]. In this way, handling MUVR frames can decrease computing and networking resource utilization while improving the overall QoS and accepting more MUVR users.

Edge nodes work collaboratively with multi-tier edge nodes to provide MUVR services with minimized delay and resource guarantees. Specially, the *Synchronation* module considers the *Interaction (IA)* SF to centralize MUVR users and to synchronize their actions. The *Object Viewing* module holds the *Foreground Interactions (FI)* and *Background virtual Environment (BE)* SFs to identify interactive or background MUVR VOs. The *Coding* module transforms objects into a video-like format to play on the HMD. Figure 24 shows the MUVR SF chain.

SFC is ideal for deploying MUVR SFs in multi-tier edge computing architectures because SFC reuses computational resources among multiple users, which is not possible in a monolithic approach. MUVR partially prevents redundant object processing and improves service scalability. For instance, rapid instantiation of idle SFs containers placed into edge nodes provides fast MUVR service initialization without migrating SFS containers after choosing the suitable edge nodes. Therefore, each node has previously stored MUVR SFs. In the *Synchronization* module shown at the bottom of Figure 24, the MUVR service synchronizes users, moves around VOs simultaneously for all of them, and allows multi-user interactions. Each SFC request has its IA SF that connects the users of that session, and the instantiation tends to move from the high-level tier to the low-level tier in the centrality of the users' location.

In the *Object View* module shown at the center of Figure 24, the FI element in red in the right of the *Object Viewing* module in Figure 24 has an SF dealing with VOs that a user manipulates closely in arms range. Therefore, FI is the unique perspective view of an object because frame prediction accuracy and similarity between several users are low, and each user has a unique SF for this purpose [35]. In parallel with FI element, the BE element has several parallelized SFs and requires more computational and bandwidth resources because of a high number of VOs. The content going through BE is fairly predictable following the user‘s movement. Hence, objects in the BE can be pre-rendered and prefetched from the node during the session to satisfy the stringent per-frame rendering latency requirements [91].

Users with a common view perspective of BE objects have in-frame similarity probability and frame predictive rendering accuracy in anticipating rendering partially. We divide the BE element into four parallel SFs shown in *Object Viewing* module in Figure 24. *(i)* an SF for common objects identically viewed view among users with similarity probability $P(h) > 0.9$ and predictive frame rendering with probability $P(f) < 0.17$ [35]. *(ii)* an SF for common objects identically viewed among users but without predictive frame rendering with probability $1 - P(f)$ and frame similarity probability $P(h)$. The two previously mentioned SFs are green in the middle left of the *Object Viewing* module in Figure 24. *(iii)* an SF for uncommon objects uniquely viewed by a single user with probability $1 - P(h)$ and predictive frame rendering $P(f)$. *(iv)* an SF for uncommon objects uniquely viewed by a single user with probability $1 - P(h)$ and without predictive frame rendering with probability $1 - P(f)$. The two previously mentioned SFs are red in the center of the *Object Viewing* module in Figure 24. SFs *(i)* and *(ii)* share computing resources between the users in sessions with a significant number of VOs. Nevertheless,

SFs *(iii)* and *(iv)* can not share resources because the VOs are different.

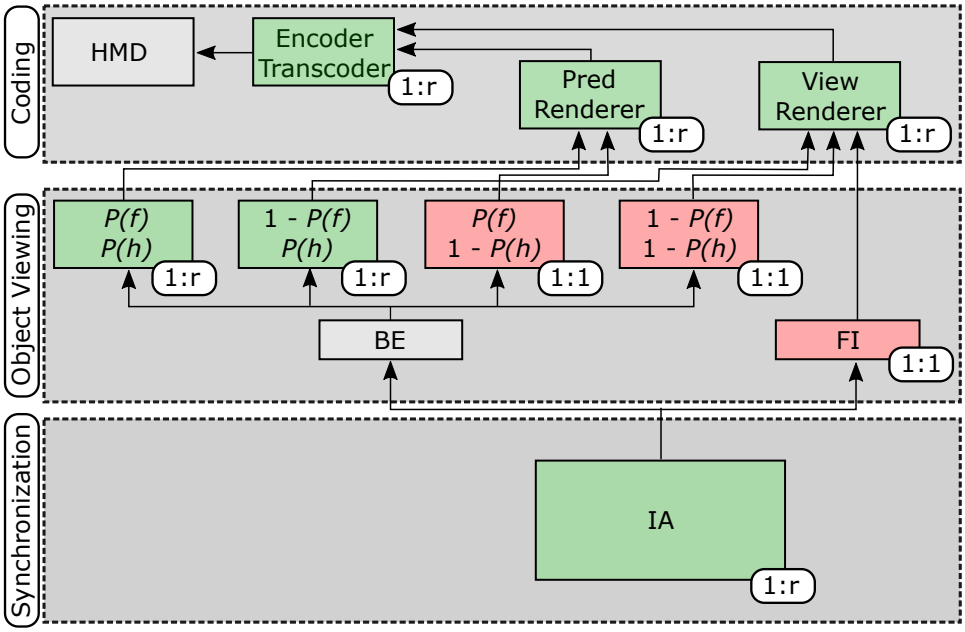

Figure 24: MUVR SF Chain.
Source: [65]

In the *Coding* module shown at the top of Figure 24, the view renderer SF renders the streams of the actual view from the *Object View* module. Similarly, predictive frames have a rendering process in the pred-renderer SF. The next SF encodes and transcodes the content in a video-like format and streams everything to display on the HMD.

5.1.2 MUAR Service in Multi-tier Edge Computing

In a MUAR service, mobile users with HMGs request a session for remote processing of computing-intensive tasks, such as rendering unique and common VOs [26]. Thus, multiple and synchronized mobile users must handle the same views in an interactive space and react to each others' actions with low latency [16]. However, edge computing services alone cannot provide low response time for mobile users (*i.e.,* up to 5-6 ms as required for MUAR [12]). MUAR services can be decomposed into a set of ordered SFs, such as frame acquisition, pre-processing, object detection, object recognition, location-based VO caching, and other functions, for instantiation at edge nodes [16]. In this way, a SF chaining session can be organized in a set of parallelizable, ordered, and location-based SFs to reduce latency and improve the efficiency of computational resources utilization in terms of processing, storage, and networking [93]. For instance, it is important to parallelize SF chaining to reduce the latency since two or more SFs are executed in parallel whose result needs to be merged later into the service chain [94]. Hence, SF chaining avoids redundant processing and transmission of a significant volume of data while reducing latency.

A MUAR SFC scheme assumes the containerized image of SFs in the multi-tier edge nodes for fast service initialization, avoiding SF container migration while re-

instantiating. The edge nodes have high-speed links interconnecting them and their computational resources. A MUAR SFC scheme follows a specific sequence to get a video frame, to extract features of the image, to recognize the objects into a single SFC flow for unique and another flow for common view, to overlay VOs into the video frame with rendering, to transform several frames into a video format, and to display the content to the mobile user. Each session has its SFs instantiated in the *Regional Tier* or *Local Tier* nodes, where an *IA* SF works as the node of MUAR service. This SF receives uploaded data from the AR application at the HMG/mobile users and synchronizes the transformations of VOs for all users, allowing multi-user interaction and aligning the virtual 3D coordinate system by establishing reference points with the real world with GPS data and buildings.

In a MUAR service, mobile users with HMG request a service to the *SFC Request Service* in the *Regional Tier* using any wireless communication network (*i.e.,* 5G and WiFi). The SFC request is composed of information about the user, MUAR service addresses, a list of SFs, life-cycle duration, and the current user location. Figure 25 shows the modules contained in the *Regional Tier* that perform SFC orchestration. The *SFC Request Service* module interacts with mobile users and notifies the orchestration scheme supported by the *Orchestrator* module at the *Regional Tier*. The *Orchestrator* groups nearby users to use computational resources of MUAR services, builds a MUAR SFC session with common SFs, and dynamically chooses a suitable edge node. The *SF Instantiator* module makes the routing decision to establish a route between a user and a given set of parallelizable, ordered, and location-based SFs deployed at the edge nodes, considering computational and network constraints, latency, and mobile user's position. The *Orchestrator* module triggers the *Controller* to implement the MUAR SFC route and notifies the *Resource Manager* module to update its states. The *Resource Manager* monitors and keeps the information about each edge node (i.e., available network and computing resources). It also triggers the *Orchestrator* when the bandwidth, computing, or both resources at the network edges are insufficient to accommodate new requests. The *Mobility Manager* module detects and points out where the mobile users will move [95] and also notifies the *Orchestrator* module to update the location-based SF *Matching/Caching*. In this sense, the *Orchestrator* module provides a dynamic instantiation proceeding in response to *Mobility Manager* module and *Resource Manager* module information, *i.e.,* the *Orchestrator* module adapts routes and instances SFs according to users' mobility, as well as computing/networking resources.

The center of Figure 25 shows that MUAR SFC is composed of common (in blue in Figure 25) and uncommon SF(in red in Figure 25). Common SFs have shareable data and processing of VOs for mobile users, *i.e.,* the service node into *IA* SF, *Recognition* SF, *Caching/Matching* SF, and *View Renderer* SF. Uncommon SFs have only specific user information, *i.e., Feature Extraction* SF, *Unique* SF, and *Encoder/Transcoder* SF. For instance, a *Detection* SF highlights objects used as a reference for positioning VOs in the scene. The *Feature Extraction* SF pre-processes objects to forward as input to the *Recognition* SF. The SFC forks the flow to a *Unique SF* flow for exclusive VOs view for a single user, such as more complex ones with interactive features on the foreground and the

other flows to identify from which original image VOs belong to by using further additional information stored in *Caching/Matching* SF for commonly viewed VOs in the interactive space. The SFC flows converge in the *View Renderer* SF, which renders the streams of the actual view and the *Encoder/Transcoder* SF formats VOs for video streaming on the HMG.

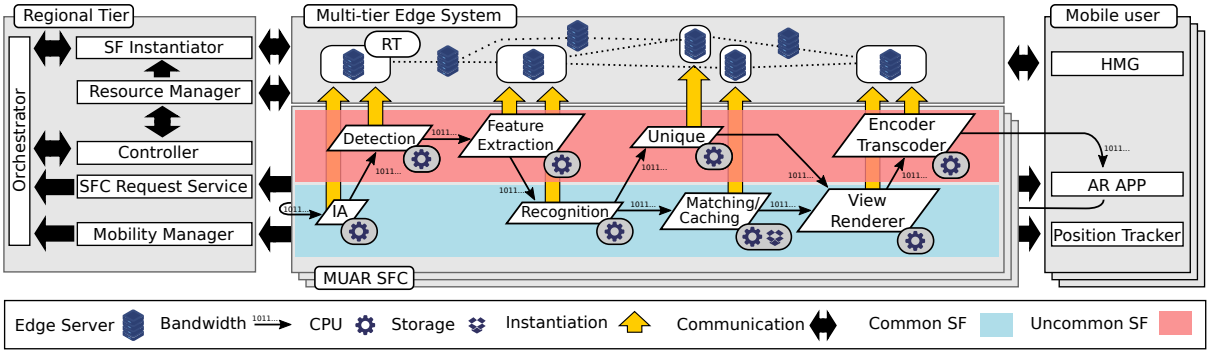

Figure 25: Multi-tier Edge Architecture for MUAR SFC.
Source: author's own.

Figure 25 shows an SFC orchestration system for MUAR services in a multi-tier edge computing environment. The SFC orchestrator instantiates SFs anywhere in the edge nodes and dynamically reinstantiate SFs based on the new destination nodes where mobile users associate while moving around. The multi-tier edge environment contains the *Regional Tier* node (RT node in Figure 25) and others *Local Tier* nodes. The MUAR SFC request starts receiving the feedback data from mobile users and synchronizes users in *IA* SF located in the source node, where this SF can be instantiated anywhere starting from the *Regional Tier* node or the closest *Local Tier* node. The SFC has four linear chained SFs, where *IA* and *Detection* SFs are instantiated in the *Regional Tier RT* and *Feature Extraction* and *Recognition* are instantiated two hops away from the previous SF. The SF chain bifurcates into two parallel SFs, where *Unique* and *Matching/Caching* SFs are instantiated in parallel into different edge nodes. The parallel SFs converge flow to the *View Renderer* SF and forward data to the destination node hosting the *Encoder/Transcoder* SF. Finally, the SFC flow is delivered to the HMG of the mobile user. The mobile users interacting with the same VOs are organized into sessions. Users from different sessions but nearby are associated with location-based VOs cached in the *Matching/Caching* SF based on the position tracker information. Each user in the set is associated with its edge node, namely the destination node. The SFC orchestration looks to instantiate the SF chains to the destination nodes of nearby interacting users. The HMGs close the loop of uploading and downloading data from the MUAR SFC service by decoding, and displaying the SFC output.

5.2 Mobility-aware Service Function Chaining Orchestration Scheme

This section presents the Mobility-aware Service Function chaining orchestration scheme (MSF). The scheme uses a heuristic to provide a suitable, quick, and practical response for MUAR services. MSF divides a complex problem into multiple overlapping and independent subproblems. Once a subproblem has been solved, the overlapped part of the solution is reused for the next subproblem saving time to recalculate the available resources. Hence, MSF becomes faster by computing partial solutions only once. MSF instantiates the shortest route for SFCs with proper computing p_u^c, storage m_u^c, and bandwidth b_u^c resources. MSF temporally reserves resources during the route calculation of each SF, avoiding extra computing time of the *Controller* module to update its state and speed up the instantiation and implementation procedures.

The SFC orchestration scheme has two algorithms. Algorithm 3 illustrates the *Orchestrator* algorithm, which receives and maintains all the SFC information of mobile users and their locations to trigger an SFC re-instantiation procedure.

Algorithm 4 maps each unique SF meeting computational demand into an edge node, orderly instantiates each SF, and minimizes latency by using edge nodes and making its decision based on bandwidth, CPU, and caching resource availabilities, as well as connectivity resources. Algorithm 3 receives new user requests and mobility events for the mobile user location l for a session s ($u_l^s \in U$) to insert into the corresponding queue at lines 1-8. For each new request, the algorithm builds a MUAR SFC session s by grouping nearby users u_l^s to interact with themselves. Afterwards, the algorithm gets the user location for a session u_l^s into the queue of new SF instantiation Q_i. The algorithm adapts the location-based SF and allocates the u_l^s data into the queue of re-instantiating SFCs Q_m for each new mobility event. Afterwards, Algorithm 1 calls Algorithm 4 to obtain the tuple t for each SFC for the priority queue Q_m at lines 10-11. If the route has been found, the controller implements the SFC route. In case of failure (*i.e.,* when the session is not accepted), the algorithm releases SFCs of other mobile users from Q_m and cancels that session at lines 12-16. The same proceeding occurs for the secondary queue of new users Q_i at lines 18-26. In this sense, all users remain interacting from the start to the end of a given session.

We combine the multi-tier edge graph with edge and vertices costs. Edges are described in terms of bandwidth and latency, and vertices are described in terms of CPU and storage. We obtain the shortest route using the Dijkstra-based algorithm combined with the resource constraints of each SF and guarantee the SFs order by performing a backtracking route calculation from users to the source reference point in the *Regional Tier*. Thus, the orchestration scheme brings the synchronization *IA* SF closer to mobile users, reducing the latency.

Algorithm 4 instantiates the ordered SFs into a set of interconnected nodes, where the transmitted data passing in the nodes forms a route. The algorithm initializes the

Algorithm 3: MSF Orchestrator

```
1  foreach u_l^s ∈ U do
2      if User u_l^s is new then
3          Group user u_l^s based on location l
4          Build session s for location l if non existent
5          Q_i.push(u_l^s)
6      if User u_l^s is triggered by Mobility Manager then
7          Modify location-based SF of u_l^s
8          Q_m.push(u_l^s)
9  do
10     try t ← SF_Instantiator (Q_m.peek())
11         Controller implements t
12     catch instantiationFailure(s)
13         if Q_m.peek() ∈ s then
14             Q_m.pop()
15             Deinstantiate users of session s
16             Cancel session s
17 while Q_m not empty
18 do
19     try t ← SF_Instantiator (Q_i.peek())
20         Controller implements t
21     catch instantiationFailure(s)
22         if Q_i.peek() ∈ s then
23             Q_i.pop()
24             Deinstantiate users of session s
25             Cancel session s
26 while Q_i not empty
```

connectivity matrix N with *false* values, sets cumulative latency matrix D to infinity values, and the tuple t as *false* at lines 1-3. The algorithm iterates over each ordered SF sf_j of mobile user location u_l^s and defines the location l if sf_j is location-based at lines 4-6. The algorithm iterates over each edge node $v \in V$ at line 6. Then, it selects a suitable edge node v if it has processing p_v^f and caching m_v^f capabilities to node sf_j at line 8. The algorithm checks all edge nodes, including itself, for connectivity with the previous SF sf_{j-1} and sets *true* if it exists at lines 9-10. If a delay value $d_{vv'}$ exists and the required bandwidth from the previous SF to the current SF is lower than the available bandwidth of the path between edge node v and v' are *true*, a variable delay sums the latency of the partial path temporarily at lines 11-12. Given the lowest delay at line 13, we set the solution in matrices N, D, W, and O saves in the tuple t at lines 14-18. If the end-to-end (E2E) latency of a given route W from the *IA* SF at an edge node v_{sf_1} and the destination SF at an edge node $v_{sf_{dst}}$ is lower than SFC request latency u^d threshold at line 19, the tuple t is returned to Algorithm 3 at line 20. Otherwise, SF instantiation

informs a failure at line 21.

Algorithm 4: SF_instantiator

Input: u_l^s
Output: t
1 $N(v, sf_j) = False$;
2 $D(v, sf_j) = +\infty$;
3 $t \leftarrow False$;
4 **foreach** $sf_j \in u_l^s$ **do**
5 **if** sf_j *is location-based* **then**
6 Mark location l in sf_j
7 **foreach** $v \in V$ **do**
8 **if** $p_v^f > sf_j^p$ ***and*** $m_v^f > sf_j^m$ **then**
9 **foreach** $v' \in V$ **do**
10 **if** $N(v', sf_{j-1}) = True$ **then**
11 **if** $\exists d_{vv'}$ ***and*** $b_{vv'} > sf_{j-1,j}^b$ **then**
12 $delay = D(v', sf_{j-1}) + d_{vv'}$;
13 **if** $delay \leqslant D(v, sf)$ **then**
14 $N(v, sf_j) = True$;
15 $D(v, sf_j) = delay$;
16 $W(v, sf_j) = W(v', sf_{j-1}) \cup d_{vv'}$;
17 $O(v, sf_j) = O(v', sf_{j-1})$;
18 $t \leftarrow tuple(N, D, W, O)$;
19 **if** $get\_delay(t.W(v_{sf_1}, v_{sf_{dst}})) \leqslant u^d$ **then**
20 **return** t;
21 instantiationFailure(s);

Given successful MSF instantiation, we have the path information on matrix $W(v_{sf_{src}}, sf_{dst})$, where W contains a set of paths interconnecting the ordered SFs. For instance, $W(v_{sf_{src}}, v_{sf_{dst}}) = [[v_{src}], [v_{sf_{src}}, ..., v_{sf_{k-1}}], [v_{sf_{k-1}}, ..., v_{sf_k}], [v_{sf_k}, ..., v_{sf_{dst}}]]$ and each element of W is a set of nodes, where the first element nodes the previous SF $v_{sf_{k-1}}$, and the last element nodes SF v_{sf_k}. The nodes between $v_{sf_{k-1}}$ and v_{sf_k} are on the path from one to another. Therefore, route information is easily extracted from matrix W.

The computation complexity of the proposed solution is $O(k \cdot V^2)$. The heuristic takes k iterations to converge, with k as the length of the SFC request. The heuristic iterates at all edge nodes $v \in V$ with each other to map each SFs. In each iteration, the heuristic checks the connectivity V iterations.

5.3 Evaluation

This section describes the evaluation methodology, including scenario description, simulation parameters, and metrics for evaluating different SFC orchestration schemes.

We define the scenario and simulation parameters of MUVR service and discuss the results and findings in a static scenario in Section 5.3.1. We define the scenario and simulation parameters of MUAR and discuss the results and findings in the mobile scenario in Section 5.3.2.

5.3.1 Impact of SFC Orchestration for MUVR with Static Users

In this section, we introduce the scenario description of SFC orchestration in multi-tier edge computing for static MUVR users, compare MSF to three related works and highlight the findings.

5.3.1.1 Scenario Description and Simulation Methodology

We implemented an SFC orchestration scheme for MUVR services using a well-known network package on python3 called NetworkX to simulate network resources. We added the remaining SFC resource requests, such as node capacity and storage capacities and the SFC orchestrator. We generated random network topologies based on an Erdos-Renyi graph with 25 nodes and internode link probability set on 20% with random delays between 1 and 2 milliseconds [64].

The MUVR considers the interaction up to 4 MUVR users, where they partially share frames of MUVR service based on prediction and similarity probabilities. The similarity $P(h)$ and prediction $P(f)$ probabilities range from 65% to 90% and 17% to 25% intervals, respectively [22]. The similarity is given by the Structural Similarity Index Measure (SSIM), a method for predicting the perceived quality of pictures or video frames. We consider the number of computing resources used to detect and map similar regions of MUVR tiles proportional to the uniform similarity probability distribution [96]. For instance, 65% of frame similarity enables processing a frame region for all users at once, saving redundant processing and waste of computing resources. A MUVR stream does not need extra computing resource, from one to four users [22].

Figure 26 shows the *Object Viewing* module of a MUVR service. In a session, users share up to 90% of the BE traffic into two common SFs of predictable and non-predictable (green boxes in Figure 26). The remaining traffic of BE goes to two uncommon SFs of predictable and non-predictable (red boxes at the center of Figure 26). The uncommon BE SFs and the FI SF adds proportionally to the number of users. In this way, scenarios with 1, 2, 3, and 4 users add 5, 8, 11, 14 only for the *Object Viewing* module.

Figure 27 shows the *Coding* module of MUVR of two parallel SFs, one for predictive rendering *Pre Renderer* SF and one for the actual view rendering *View Renderer* SF. In a session, the output of predictable BE SFs goes *Pred Renderer* SF and the out of non-predictable BE SFs and FI SF goes to the *View Renderer* SF. The two rendering SFs forward data to the *Encoder/Transcoder* SF. In this way, three additional shareable SFs (green boxes in Figure 27) of *Coding* and one more (*IA* SF) from the *Synchronization*

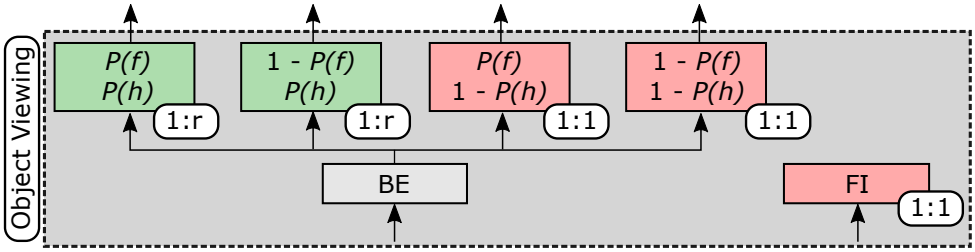

Figure 26: Object Viewing module in MUVR SF Chain.
Source: author's own.

module makes the SF number go to 9, 12, 15, and 18 SFs.

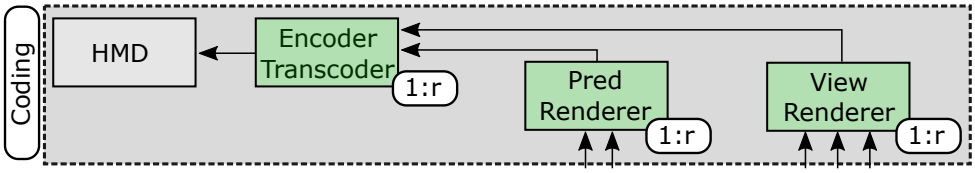

Figure 27: Coding module in MUVR SF Chain.
Source: author's own.

The monolithic approach has 2, 3, 4, and 5 MUVR elements for 1, 2, 3, and 4 users, respectively, because they need an element with the same function of *IA* SF as the main server. For instance, one MUVR SFC request of 4 users contains 18 SFs towards the destination nodes of each user. More users playing together means that more computing resources can be saved. However, the only element shared in the monolithic case is the *IA* SF and all elements are in the same edge node, which tends to overload a single edge node.

We consider 300 SFC requests with an arrival rate of 0.1Hz, random uniform distribution for a lifetime up to 120 s, and request latency of 6 ms. The SFC request is rejected if latency and resource requirements are not met, causing an impact on the service acceptance ratio. The MUVR streams are divided into BE for far away and FI for close objects, respectively. Each user needs 266 kbps for FI and 25 Mbps for BE, bit BE is partially shared among other users. We consider similar edge nodes consuming 20% and 15% of each one's processing and storage capacity, respectively, for each user of storage capacity. We repeated the simulation 33 times to achieve a confidence interval of 95%. Table 9 summarizes simulation parameters.

We compare MSF with the k-shortest paths algorithm that finds a number of k shortest paths between destination and source, where k was set as 8. However, k-shortest combines computing resources with the shortest path by picking between nodes with the most CPU and then storage resources. Furthermore, we compare MSF with a Betweenness Centrality (BC) based algorithm, which measures the centrality of a graph based on the shortest paths. The node with the highest number of shortest paths is the anchor for the middle SF, and then the algorithm recursively deploys the remaining SFs from this anchor node on the way to the source and destinations. Another approach compared is the monolithic deployment of MUVR.

Table 9: Simulation Parameters for MUVR

Parameter	Value
Nodes	25
Edge nodes	0.3×Nodes
Latency of inter-edge links	uniform(1, 2) ms
Link Bandwidth capacity	1Gbps
Capacity CPU / Storage	100% / 100%
Incoming / SFC request rate	300 / 0.1 Hz
Request lifetime	uniform(120) s
Request latency	6 ms
FI / BE bandwidth	266 Kbps / 25Mbps
Request CPU / Storage usage	20% / 15% of a node
SSIM probability	uniform(0.65, 0.9)
Prediction probability	uniform(0.17, 0.25)
Request number of users	up to 4
SF Number of Simulations	33

We consider six metrics to evaluate MSF with other algorithms and monolithic deployment. The algorithms are k-shortest, and BC path adapted to deal with CPU and storage resources constraints. The metrics are: *(i)* SFC request acceptance ratio is the number of accepted SFC requests divided by the total number of SFC requests. *(ii)* bandwidth utilization is the ratio between link usage and all network links. *(iii)* CPU utilization is the ratio of CPU usage of all accepted SFC requests and the sum of CPU resources of edge nodes. *(iv)* storage utilization is the ratio of storage usage for all MUVR SFC requests and the sum of all edge nodes. *(v)* end-to-end latencies given by the path from source to the destinations of an SFC request. *(vi)* orchestration decision time.

5.3.1.2 Simulation Results

Figure 28 shows the acceptance ratio for 300 SFC requests in a multi-tier edge computing environment with 25 edge nodes. MSF is able to accept 10-20% more requests than BC and around 72% more than k-shortest. Compared to monolithic, MSF nearly accepts the double number of requests in the worst case, as shown in Figure 28. MSF achieves a better acceptance ratio because it finds the shortest available path, and k-shortest chooses only the shortest one. BC overloads the network in the centrality of the topology between *Regional Tier* node and the destination node.

Figure 29 shows the delay for 300 SFC requests in a multi-tier edge computing environment with 25 edge nodes. MSF has a similar delay to monolithic and less than 1 ms lower than k-shortest. BC has the highest delay because it places SFs in the middle path. MSF achieves low delay while accepting more SFC requests. It happens the compared works overload the same paths and edge nodes that are frequently chosen.

Figure 30 shows the CPU utilization for 300 SFC requests in a multi-tier edge computing environment with 25 edge nodes. MSF accepts more SFC requests but does

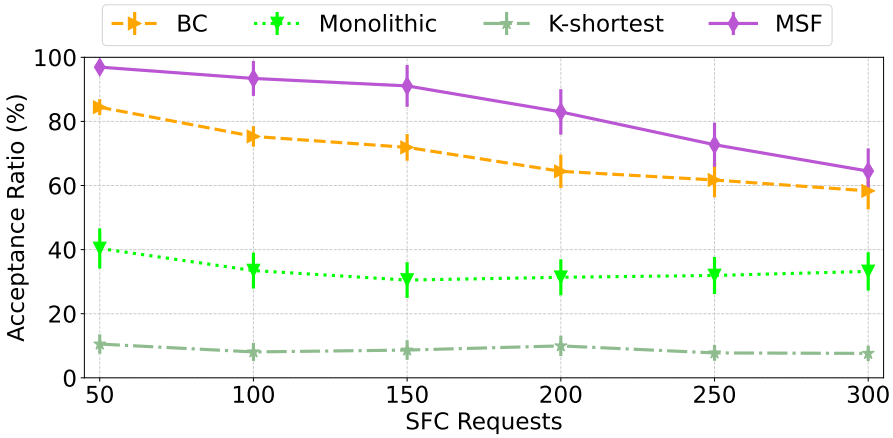

Figure 28: MUVR Acceptance ratio.
Source: [65]

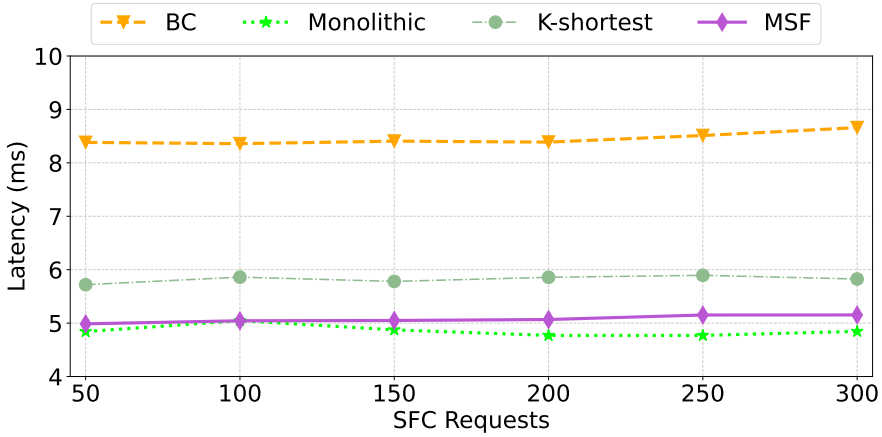

Figure 29: MUVR Delay.
Source: [65]

not have more CPU utilization than BC. Compared to the monolithic scheme, MSF used 28% fewer CPU resources. BC used similar resources, and k-shortest had the lowest acceptance ratio. CPU utilization is proportional to the acceptance ratio. MSF has more flexibility to find alternative and efficient paths, avoiding overloading frequently the most convenient links and nodes.

Figure 31 shows the storage utilization for 300 SFC requests in a multi-tier edge computing environment with 25 edge nodes. MSF accepts more SFC requests but does not have more CPU utilization than BC. Compared to the monolithic scheme, MSF used 20% fewer CPU resources. BC used a similar number of resources, and k-shortest had the lowest acceptance ratio. Storage utilization is proportional to CPU utilization mainly because CPU constraints are more intense than storage utilization.

Figure 32 shows the bandwidth utilization of 300 SFC requests in a multi-tier edge computing environment with 25 edge nodes. MSF had similar bandwidth utilization as the monolithic approach, considering almost twice the amount of SFC requests were accepted.

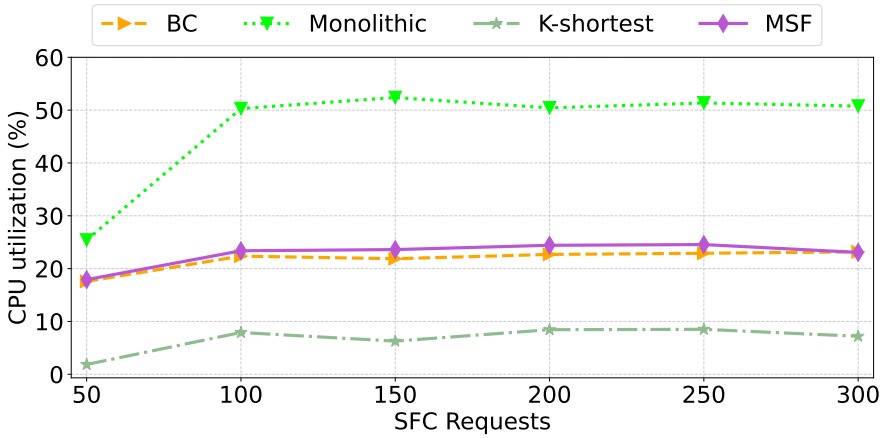

Figure 30: MUVR CPU utilization.
Source: [65]

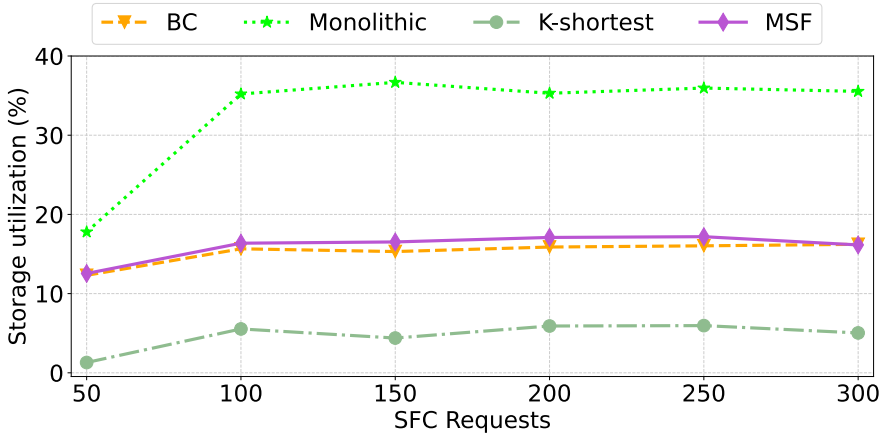

Figure 31: MUVR Storage utilization.
Source: [65]

MSF shows more efficient network usage compared to BC for accepting fewer SFC requests and requiring more computational resources. K-shortest had smaller utilization for having a significantly lower acceptance ratio. MSF instantiates the SFs enabling the usage of edge nodes closer but bringing the SF instantiation to far away nodes, avoiding overloading the edge nodes and links.

Figure 33 shows the decision time for 300 SFC requests in a multi-tier edge computing environment with 25 edge nodes. MSF took around 2 seconds to receive the incoming SFC requests and instantiate the SFCs into the edge nodes. BC had the closest acceptance ratio value, but its decision time took up to 300 seconds, which is unpractical for MUVR services. MSF took longer time than k-shortest and monolithic but provided significant benefits on the other metrics. MSF needs to check more instantiation options than monolithic because of the combination to instantiate the SFC, which is why higher acceptance ratio exists.

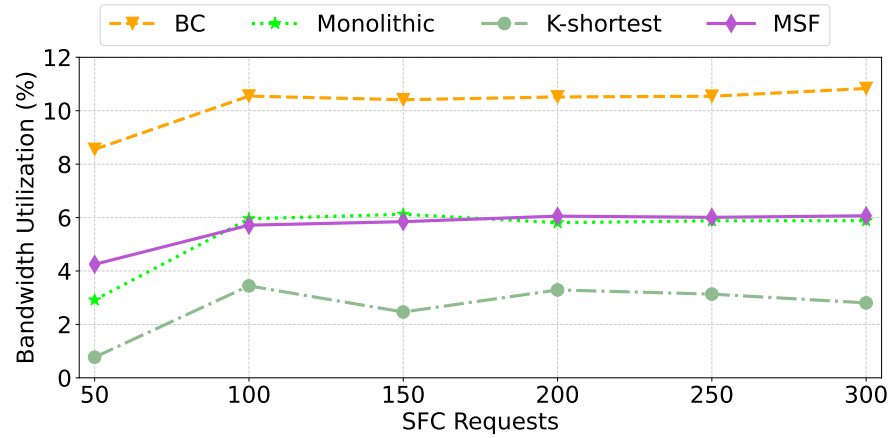

Figure 32: MUVR Bandwidth utilization.
Source: [65]

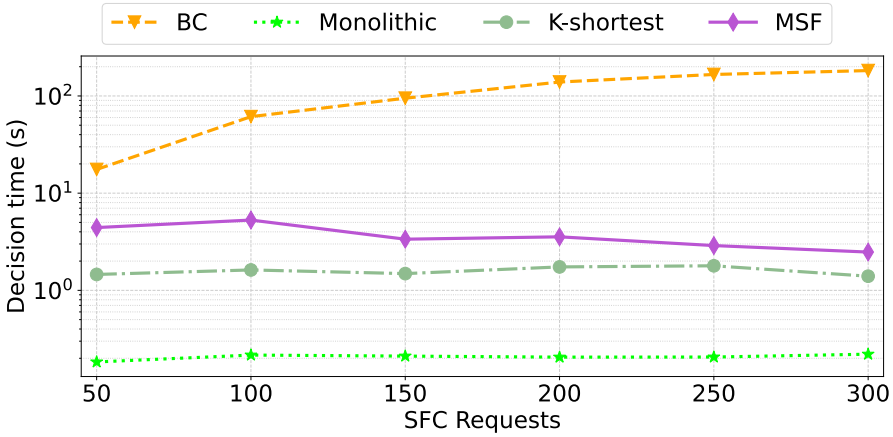

Figure 33: MUVR Decision time.
Source: [65]

5.3.2 Impact of SFC Orchestration for MUAR with Mobile Users

This section introduces the scenario description of SFC orchestration in multi-tier edge computing for mobile MUAR users, compares MSF to three related works, and highlights the findings.

5.3.2.1 Scenario Description and Simulation Methodology

We implemented MSF using NetworkX to simulate distributed edge resources in terms of latency, computing, and caching resources coordinated by the mobility-aware orchestrator. We used the edge network topology of Palo Alto City [23] with 36 edge nodes and inter-node latencies with a Poisson distribution of a mean of one millisecond, as shown in Figure 34. We chose 1/3 of the nodes as edge servers prioritizing the ones directly connected to more edge nodes, as shown in red in Figure 34. The distributed edge computing is connected to the Central Cloud controller at node 3. The links between

edge nodes are 1 Gbps. We assume that up to 50% of all mobile clients randomly move with uniformly distributed speeds from 0 to 45 km/h, as expected in urban cities. The mobile client is linked to the closest edge server operating as BS. A mobility event is triggered when the *Mobility Manager* detects a handover to a new access point or a new edge server [95]. Table 10 summarizes simulation parameters.

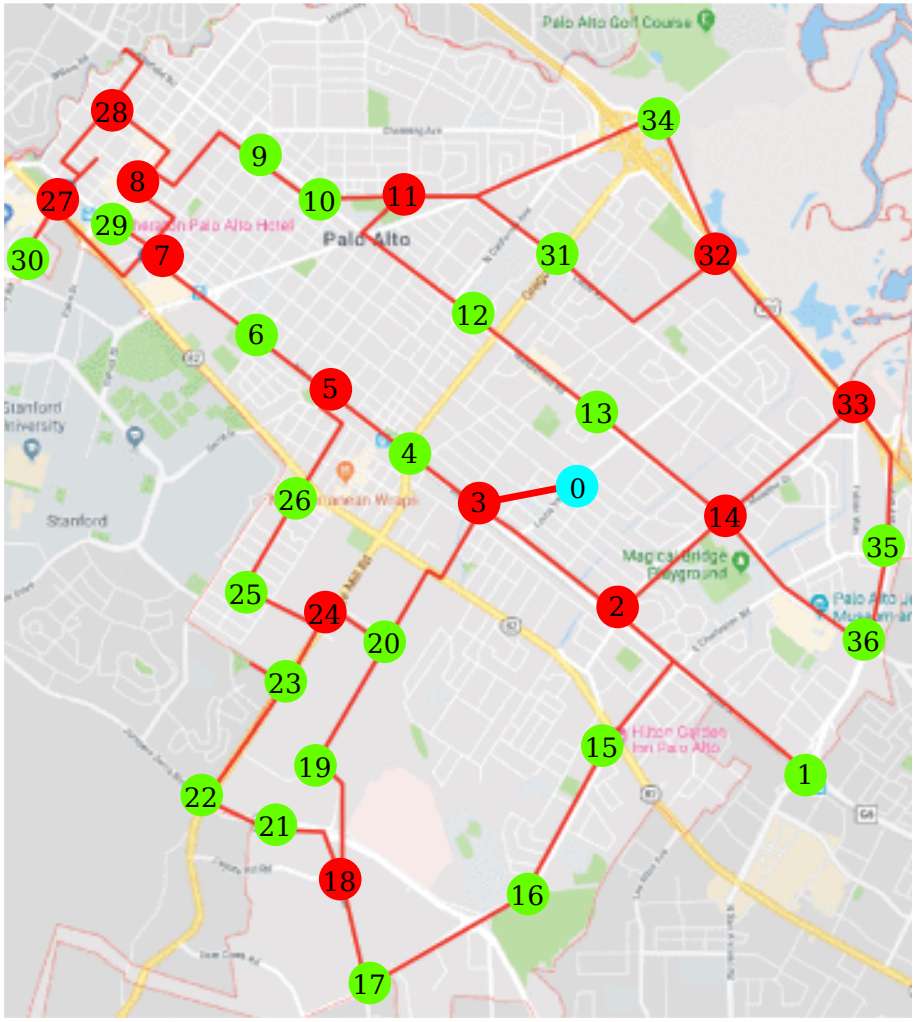

Figure 34: Multi-tier edge computing topology of Palo Alto City.
Source: author's own.

Figure 35 shows the MUAR SFC and the order of SFs. The AR app transmits a negligible bitrate for controls feedback to the *IA* SF and transmits 150 Mbps as the initial flow video from HMG device to the *Detection* SF [23]. We considered a video upload coded with 60 frames per second [97]. The output of *Detection* SF has a 400x400 pixels image on RGB with an average of 0.48 MB per frame [98]. The *Feature Extraction SF* from each frame contains from 4 to 12 VO units with 25 KB for each VO and the bitrate varies in the interval [100, 300]. The *Recognition* SF bifurcates to identify cached or non-cached VOs. We consider a conservative cache hit ratio distribution of 33%. In this way, a frame with 12 VOs has 4 cached VOs in the *Matching/Caching* SF. The *Matching/Caching* SF is the only SF that requires high-speed storage of rendered VOs and needs only to transmit these VOs to the next SF. The *Unique* SF determines the rendering of the remaining 8

non-cached VOs and joins the flow with the *Matching/Caching* SF to the next SF. The parallel SFs transmit the VOs for rendering in the *View Renderer* SF but it renders only 66% of 12 VOs, since the others were cached rendered, and saves redundant processing of VOs. VOs's size follows a normal distribution from 0 to 50 MB [16]. On average, the minimum/maximum caching size of VOs in the scene is 120/360 MB considering only 4/12 VOs in the frame, and a conservative cache hit ratio distribution of 33% [16]. The *View Renderer* SF them overlays VOs in the initial flow video and sends to the next SF. The *Encoder/Transcoder* SF compresses the rendered output and transmits 97Mbps video to the HMG [23].

Each user demands a maximum of 25% of an edge node's CPU and cache capacities. The CPU demand of each SF is a function of the amount of data processed, where every 10 bits takes one CPU cycle [16]. MUAR considers the interaction of 4 users in small-sized groups for training, sightseeing, and gaming. The number of users determines the session size and the number of SFs. Each MUAR user has common and uncommon SFs for identical and different VO view, respectively. *IA* SF, *recognition* SF, *Matching/Caching* SF and *View Renderer* Sf are common(in the blue box in Figure 35). *Detection* SF, *Feature Extraction* SF, *Unique* SF, and *Encoder/Transcoder* SF are uncommon, and each user needs one SF of each by himself. In this way, SFs with the highest processing and caching requirements can be shared among users, saving computational resources. For instance, a session of size 1 has 8 SFs, 4 uncommon SFs, and 4 common SFs. A session of size 4 has 20 SFs, 4 commons, and 4×4 uncommon SFs. The bitrate of common and uncommon SFC flows depends on the number of VOs in the scene C_s and the cache hit ratio C_{hr}. The bitrate of common flow is $150/0.23/0.14/C_{hr} * C_s/97$ Mbps, and the bitrate of uncommon is $150/0.23/0.14/(1 - C_{hr}) * C_s/97$ Mbps.

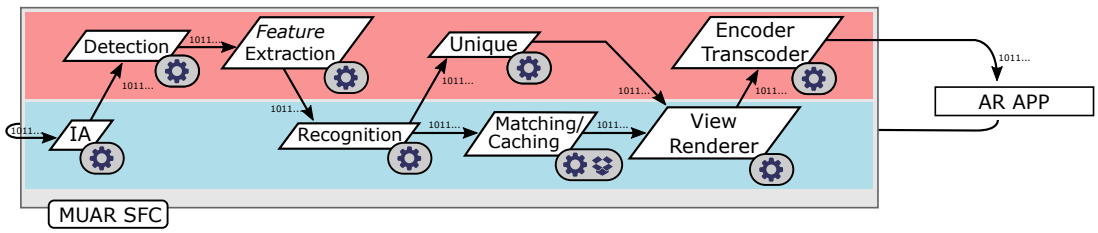

Figure 35: MUAR SFC.
Source: author's own.

We consider 50 SFC requests with an incoming arrival rate of a Poisson distribution with a mean of 15 s, lifetime random uniform distribution up to 120 s, and request latency of 6 ms. The MUAR session request acceptance happens in case of meeting latency and resource requirements. We divide MUAR streams into common and uncommon SFs. The delivery bandwidth to HMGs is 97 Mbps, partially shared among other users. We consider each user consumes 25% of a node capacity for the entire SF chain but can fragment into multiple nodes. We consider each *Matching/Caching* SF location consuming 25% of the storage capacity of a node capacity. We repeated the simulation experiments 33 times to achieve a confidence interval of 95% for each MUAR orchestration scheme.

The performance of MSF was analyzed together with existing baseline approaches,

where all of them support the same Mobility Manager scheme based on Ngo et al [95]. PPC [63] reduces the latency by deciding if SFs for single users will be executed in parallel in the same or different edge nodes. MSF Mobility-unaware (MSF-MU) enhanced with location-based SFC orchestration [99] keeps the existing SFC instantiation but attaches a new route from the final SF toward the mobile client. Finally, MSF has a mobility-aware SFC instantiation that continually adapts the entire route of edge nodes.

Table 10: Simulation Parameters for MUAR

Parameter	Value
Nodes	36 [23]
Edge servers nodes	0.3×Nodes [23]
Mobile user speed	0 - 60 km/h
Latency of inter-edge links	uniform(1, 2) ms
Link Bandwidth capacity	1Gbps
User Resource Load Request	25% CPU / 25% Cache
Incoming session rate	Poisson($\alpha = 20$)
Session Request lifetime	Poisson($\alpha = 120$) sec
Session Request latency	5 ms [23]
Session Request size	up to 4
Cache Hit Ratio(C_{hr})	$zipf(1/3)$ [16]
Cache size (C_s)	120 Mb [16]
CPU cycles/Mbit	$10e^6$ [16]
Inputs Unique SFCs (Mbits)	$150/0.23/0.14/(1 - C_{hr}) * C_s/97$ [16]
Inputs Cache SFCs (Mbits)	$150/0.23/0.14/C_{hr} * C_s/97$ [16]
Number of Simulations	33

We consider the following metrics to evaluate MUAR-based approaches, namely *(i)* Session acceptance ratio is the number of accepted sessions divided by the total number of sessions; *(ii)* Latency is given by latency from the source to mobile users. *(iii)* CPU utilization is the ratio of CPU usage of all accepted SFC requests and the CPU resources sum up of all edge servers; *(iv)* bandwidth utilization is the ratio between link usage of all accepted SFC requests and the bandwidth resources sum up of all edge links.

5.3.2.2 Simulation Results

Figure 36 shows the acceptance ratio for scenarios with speeds of 0 (static), 20, 40, and 60 km/h. MSF achieves similar performance to MSF-MU in static scenarios. When the speed is of 20 km/h and 60 km/h, MSF improves the MUAR session acceptance ratio by up to 5% and 13% compared to MSF-MU and 8% and 15% compared to PPC, respectively. MSF consistently achieves 8% or higher acceptance ratio compared to related works at all speeds. MSF-MU and PPC reject more sessions because they do not adapt SFCs well after handovers. They only adjust the routes from the chain's last SF to the mobile users, causing an additional latency and increasing the session blocking probability compared to MSF as the speed increases. The acceptance ratio stabilizes for speeds

higher than 40km/h for MSF, but other schemes have continuous and increasing service degradation.

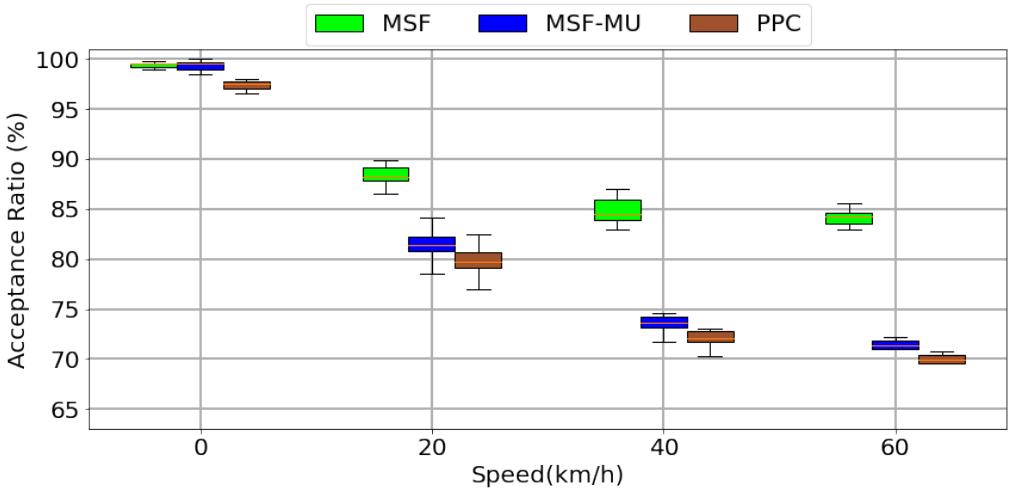

Figure 36: MUAR Acceptance Ratio.
Source: author's own.

Figure 37 shows the latency for scenarios with speeds of 0 (static), 20, 40, and 60 km/h. On average, when the speed of mobile users is 20 km/h and 60 km/h, MSF achieves lower latency up to 10% and 27% compared to MSF-MU and 24% and 41% compared to PPC, respectively. MSF uses links more efficiently, and more edge servers can deliver more sessions with QoS support to mobile users. MSF has stable latency with increasing speeds, but other schemes must connect to more distant edge servers, which increases the latency. The latency of MSF-MU and PPC increases with mobility events because of their static instantiation process of SFCs.

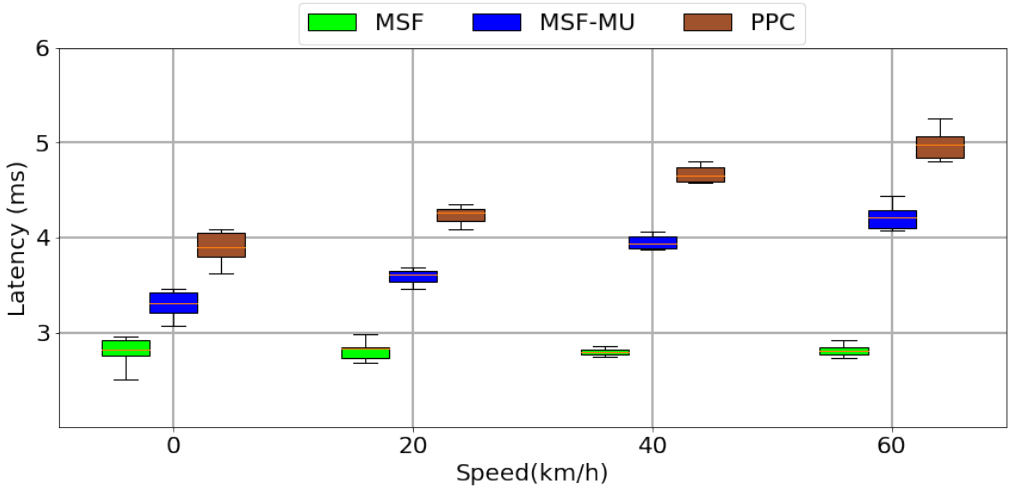

Figure 37: MUAR Latency.
Source: author's own.

Figure 38 shows the bandwidth utilization for scenarios with speeds of 0 (static), 20, 40, and 60 km/h. On average, when the speed of mobile users is of 20 km/h and

60 km/h, MSF reduces the bandwidth utilization up to 1% and 1.8% compared to MSF-MU and 1.8% and 3% compared to PPC, respectively. MSF mitigates the usage of links because the instantiation brings the SFs closer to users in a more efficient way when compared to related works. The speed increases the usage of links because the SFs are instantiated at network edges far away from where the session was originally started. Users connecting to different edge servers force the MUAR system to transmit the same content through more links toward the users' location. MSF uses bandwidth resources most efficiently, and more edge servers can deliver MUAR sessions with QoS support to mobile clients.

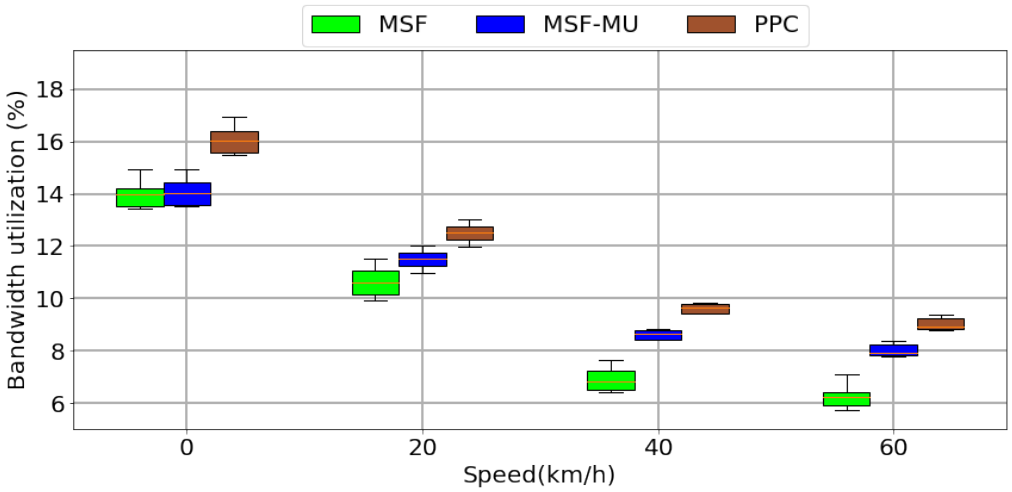

Figure 38: MUAR Bandwidth utilization.
Source: author's own.

Figure 39 shows the CPU utilization of all edge servers for scenarios with speeds of 0 (static), 20, 40, and 60 km/h. MSF achieves higher CPU utilization by up to 2% compared to related works because more MUAR sessions were accepted and orchestrated by MSF. The CPU utilization decreases with higher speeds proportionally with the acceptance ratio. The CPU utilization depends on the available bandwidth of the links to reach the edge servers and the end-to-end latency of each network edge hop along the SF chain path. For instance, in a scenario where the speed of mobile users is of 40 km/h, MSF consumed 1.8% more CPU resources and, consequently, accepted 11% more sessions than MSF-MU. MSF-MU and PPC use more bandwidth resources, leading to fewer edge server options to instantiate SFs and accept the MUAR session requests.

In summary, mobility scenarios add an extra challenge for keeping MUAR services with low latency. Handovers also increase the usage of computational resources and the number of blocking sessions. The arrival of more SFC requests at the network edges gradually saturates the resources of areas and a mobile user might more likely move to these areas, augmenting the chances of service disruption during the SFC life-cycle. Thus, a SFC orchestration scheme for mobile and dynamic smart multi-user environments must be developed taking mobility, SFC parallelization, and QoS support into consideration to improve the efficiency of distributed edge computing resources while providing a better

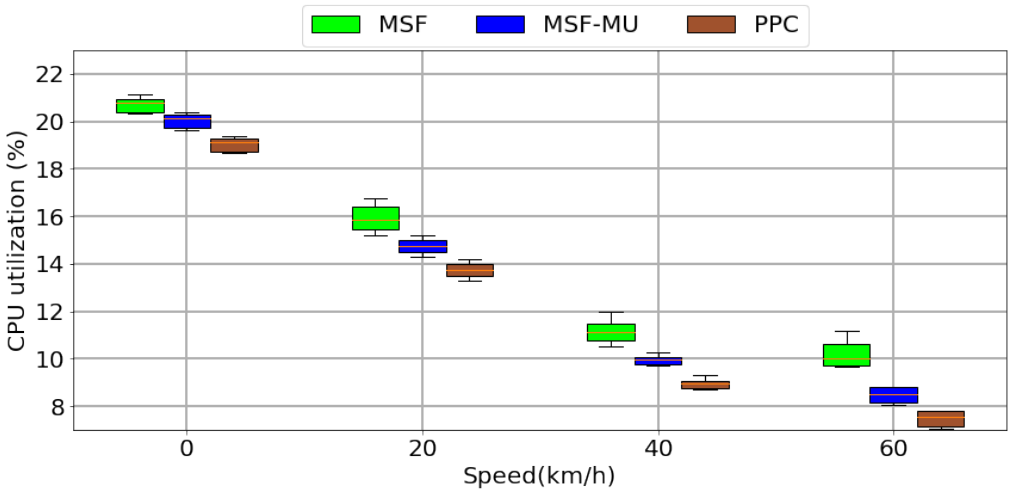

Figure 39: MUAR CPU utilization.
Source: author's own.

experience for MUAR users.

5.4 Chapter Conclusion

In this Chapter, we introduced a service orchestration of ordered and parallel SFC for MUVR and MUAR services in a multi-tier edge computing environment. We proposed MSF as a mobility-aware heuristic to orchestrate computing and networking resources. MSF minimized delay while meeting CPU, storage, bandwidth, and multiple destinations SFC requests for low latency services. The orchestration is achieved by considering a mobile-aware online SFC optimization algorithm to re-instantiating parallelizable server client based SFCs into optimal routes accordingly to the new position of mobile users and edge services, latency threshold, CPU, bandwidth, and storage resources.

The results show that MSF demonstrated significant benefits compared to related works and with the monolithic approach mainly regarding the service acceptance ratio. MSF achieved up to 15% higher acceptance ratio while delivering the MUAR SFC with up to 41% better latency in scenarios with mobile users randomly moving. MSF provides higher resource scalability in a multi-tier edge computing environment than other SFC orchestration schemes and monolithic deployment.

CHAPTER 6

Conclusion

This chapter summarizes the content of each of the previous chapters, highlights the contributions of this thesis, and shows the publication related to this thesis.

6.1 Summary

Chapter 1 introduced a set of low latency multimedia streaming problems and how service orchestration schemes could be used to overcome those issues. In summary, the main issues for handling efficient service orchestration schemes are the following: *(i)* temporary and poor connectivity of monolithic services; *(ii)* mobility support for SFC of immersive multimedia users. Motivated by those issues, Chapter 1 introduces the research questions listed as *(i)* How to orchestrate VoD services for mobile users in multi-tier edge computing environments with QoE support? *(ii)* How to orchestrate monolithic multimedia services for mobile users in flying multi-tier edge computing environments with QoS support? *(iii)* How to orchestrate SFC for immersive multimedia services in mobile multi-tier edge computing environments with QoS support?

Chapter 2 presented existing works related to service orchestration schemes for different scenarios, highlighting the benefits and limitations of each one. The chapter divides service orchestration in *(i)* multi-tier static edge computing environment for VoD services; *(ii)* flying multi-tier mobile edge computing environments for monolithic multimedia services;*(iii)* multi-tier static edge computing environments for SFC-based services.

Chapter 3 introduced Fog4Video [51]. Fog4Video is a service orchestrator scheme for improving VoD services with QoE support in a multi-tier static edge computing environment. Fog4video selects an appropriate edge node to cache, transcode, and stream VoD flows based on a set of parameters, including available transcoding bandwidth, delay,

number of stalls, stalling duration, and monetary cost to deploy VoD services in a given static edge computing tier or in a central cloud server. Fog4video used QoE and delay reported by the users connected to edge nodes to feed its AHP multi-criteria decision-making scheme.

Chapter 4 introduced FLYED [52]. FLYED is a service orchestrator scheme for immersive multimedia services with QoS support in a multi-tier mobile edge computing environment. FLYED selected and placed UAV nodes to assist poorly-served users due to poor connectivity considering distance and energy. FLYED used PDR and delay reported by the users connected to the MEC and FEC nodes to trigger the orchestration process. The decision-making process is based on the TOPSIS method to select edge nodes and a PSO algorithm to place FEC nodes on the aerial space while keeping the distribution of immersive multimedia services with QoS support.

Chapter 5 introduced MSF for static and mobile environments [53, 65]. MSF is an orchestration scheme for ordered and parallel SFC for MUVR and MUAR in multi-tier edge computing scenarios. MSF proposed a mobility-aware service orchestration scheme based on a heuristic divided into resource mapping, SF instantiation, and SFC reinstantiation procedures. Simulation results aimed to show that MSF can improve the usage of networking and computational resources for MUVR and MUAR service.

6.2 Future Works

This section briefly describes potential future works and research opportunities for the solutions presented in this thesis.

- Monolithic service migration: an orchestrator can be developed to efficiently migrate cached content between edge nodes. The orchestrator can prefetch of most popular videos, and their representations would imply storage and transmission costs between caches and links of the edge nodes. These costs could guide the selection of the best edge nodes to cache more representations of the same video and avoid frequent transcoding. Besides saving transcoding costs, fewer redundant transmissions in the backhaul could provide better QoS and QoE support for video distributions.

- SFC orchestration in multi-tier flying edge environments: A mobility-aware orchestrator can be deployed to coordinate *Mobile Tier* nodes to provide chained immersive services for improving the distribution of content and hosting services for more mobile users. *Mobile Tier* nodes can be configured to execute SFs and improve the usage of networking and computational resources while assuring QoS for immersive multimedia flows.

- Intelligent bitrate adjustment for SFC: Immersive multimedia services can be configured to adapt their visual quality by transmitting fewer data to maintain smooth playback and avoid delay violation due to a slow transmission link. Besides, the

acceptance ratio can increase by partially reducing bitrate of other SFCs and admitting more simultaneous service requests.

- More realistic service implementation: Future works can implement the proposed schemes in more realistic scenarios. The results could be enriched with data collected from real devices, including CPU, memory, networking, computational values.

- Resilience: It is important to increase the resilience of multi-tier mobile edge environments. Thus, resilience-aware orchestrator schemes should be configured to take smart decisions based on intelligent autonomous algorithms during failure events, including networking and computing failure events.

6.3 Publications related to the thesis

The service orchestration schemes presented in Chapters 3, 4, and 5 were published or are still under review as follows:

1. Chapter 3: **H. Santos**, D. Alencar, R. Meneguette, D. Rosário, J. Nobre, C. Both, E. Cerqueira, T. Braun, "A Multi-Tier Fog Content Orchestrator Mechanism with Quality of Experience Support". *Computer Networks*, 2020. [51]

2. Chapter 4: **H. Santos**, I. Medeiros, C. Rocha, D. Rosário, E. Cerqueira, T. Braun, "A mobility-aware flying edge computing service orchestration with quality of service support". *submitted*, 2023. [52]

3. Chapter 5: **H. Santos**, B. Martins, D. Rosário, E. Cerqueira, T. Braun, "Multi-criteria service function chaining orchestration for multi-user virtual reality services". *GLOBECOM*, 2022. [65]

4. Chapter 5: **H. Santos**, B. Martins, D. Rosário, E. Cerqueira, T. Braun, "Mobility-aware Service Function Chaining Orchestration for Multi-user Augmented Reality". *submitted*, 2023. [53]

6.4 Publications in collaboration:

The research of this thesis led to collaboration into several publications in important journals and conferences as shown in the following:

1. [100] - Z. Zhao, L. Pacheco, **H. Santos**, M. Liu, A. Di Maio, D. Rosário, E. Cerqueira, T. Braun, X. Cao, "Predictive UAV base station deployment and service offloading with distributed edge learning", *IEEE Transactions on Network and Service Management*, 2021.

2. [101] - P. Cumino, W. Lobato Junior, T. Tavares, **H. Santos**, D. Rosário, E. Cerqueira, L. A. Villas,, M. Gerla, "Cooperative uav scheme for enhancing video transmission and global network energy efficiency", *Sensors*, 2018.

References

[1] R. Shahzadi, M. Ali, H. Z. Khan, and M. Naeem, "Uav assisted 5g and beyond wireless networks: A survey," *Journal of Network and Computer Applications*, vol. 189, p. 103114, 2021.

[2] Global Data, "Mobile gaming - thematic research, research and markets," June 2021.

[3] Statista, "Digital media report - video-on-demand," March 2023.

[4] ——, "Ar & vr - worldwide," March 2023.

[5] ——, "Augmented reality (ar), virtual reality (vr), and mixed reality (mr) market size worldwide in 2021 and 2028," August 2022.

[6] Logitech. (2023, January.) Logitech g cloud gaming handheld. [Online]. Available: https://www.logitechg.com/en-us/products/cloud-gaming/cloud-handheld-gaming.940-000198.html

[7] S. Jiang, B. Moyle, R. Yung, L. Tao, and N. Scott, "Augmented reality and the enhancement of memorable tourism experiences at heritage sites," *Current Issues in Tourism*, vol. 26, no. 2, pp. 242–257, 2023.

[8] C. Papakostas, C. Troussas, A. Krouska, and C. Sgouropoulou, "Measuring user experience, usability and interactivity of a personalized mobile augmented reality training system," *Sensors*, vol. 21, no. 11, p. 3888, 2021.

[9] S. Engl and L. Nacke, "How mobile is mobile gaming? contextual influences on mobile player experience–a model proposition," *Mensch & Computer 2010 Entertainment Interfaces Track*, 2010.

[10] S. Park, S. Bokijonov, and Y. Choi, "Review of microsoft hololens applications over the past five years," *Applied sciences*, vol. 11, no. 16, p. 7259, 2021.

[11] Mordor Intelligence, "Mobile gaming market - growth, trends, covid-19 impact, and forecasts (2021 - 2026)," July 2021.

[12] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5g mobile edge computing: architectures, applications, and technical aspects," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021.

[13] V. Kelkkanen, M. Fiedler, and D. Lindero, "Synchronous remote rendering for vr," *International Journal of Computer Games Technology*, vol. 2021, 2021.

[14] C. C. W. Keung, J. I. Kim, and Q. M. Ong, "Developing a bim-based muvr treadmill system for architectural design review and collaboration," *Applied Sciences*, vol. 11, no. 15, p. 6881, 2021.

[15] K. Ahuja, M. Goel, and C. Harrison, "Bodyslam: Opportunistic user digitization in multi-user ar/vr experiences," in *Proceedings of the 2020 ACM Symposium on Spatial User Interaction*, 2020, pp. 1–8.

[16] Z. Huang and V. Friderikos, "Proactive edge cloud optimization for mobile augmented reality applications," in *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021, pp. 1–6.

[17] F. Araújo, D. Rosário, E. Cerqueira, and L. A. Villas, "A hybrid energy-aware video bitrate adaptation algorithm for mobile networks." in *15th IEEE/IFIP Wireless On-demand Network systems and Services Conference (WONS)*, 2019, pp. 146–153.

[18] D. Rosário, M. Schimuneck, J. Camargo, J. Nobre, C. Both, J. Rochol, and M. Gerla, "Service migration from cloud to multi-tier fog nodes for multimedia dissemination with qoe support," *Sensors*, vol. 18, no. 2, p. 329, 2018.

[19] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2015.

[20] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Computer Networks*, vol. 130, pp. 94–120, 2018.

[21] A. S. Gomes, B. Sousa, D. Palma, V. Fonseca, Z. Zhao, E. Monteiro, T. Braun, P. Simoes, and L. Cordeiro, "Edge Caching with Mobility Prediction in Virtualized LTE Mobile Networks," *Future Generation Computer Systems*, vol. 70, pp. 148–162, 2017.

[22] J. Meng, S. Paul, and Y. C. Hu, "Coterie: Exploiting frame similarity to enable high-quality multiplayer vr on commodity mobile devices," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 923–937.

[23] N. Akhtar, I. Matta, A. Raza, L. Goratti, T. Braun, and F. Esposito, "Managing chains of application functions over multi-technology edge networks," *IEEE Trans. on Network and Service Management*, vol. 18, no. 1, pp. 511–525, 2021.

[24] P. Caserman, M. Martinussen, and S. Göbel, "Effects of end-to-end latency on user experience and performance in immersive virtual reality applications," in *Entertainment Computing and Serious Games: First IFIP TC 14 Joint International Conference, ICEC-JCSG 2019, Arequipa, Peru, November 11–15, 2019, Proceedings 1*. Springer, 2019, pp. 57–69.

[25] F. Hu, Y. Deng, W. Saad, M. Bennis, and A. H. Aghvami, "Cellular-connected wireless virtual reality: Requirements, challenges, and solutions," *IEEE Communications Magazine*, 2020.

[26] X. Ran, C. Slocum, Y.-Z. Tsai, K. Apicharttrisorn, M. Gorlatova, and J. Chen, "Multi-user augmented reality with communication efficient and spatially consistent virtual objects," in *16th International Conference on emerging Networking EXperiments and Technologies*, 2020.

[27] Y. T. Woldeyohannes, A. Mohammadkhan, K. Ramakrishnan, and Y. Jiang, "Cluspr: Balancing multiple objectives at scale for nfv resource allocation," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1307–1321, 2018.

[28] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, T. Jiang, J. Crowcroft, and P. Hui, "Edge intelligence: Architectures, challenges, and applications," *arXiv preprint arXiv:2003.12172*, 2020.

[29] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[30] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, 2019.

[31] P. Cruz, N. Achir, and A. C. Viana, "On the edge of the deployment: A survey on multi-access edge computing," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–34, 2022.

[32] V. Nguyen, T. T. Khanh, P. Van Nam, N. T. Thu, C. S. Hong, and E.-N. Huh, "Towards flying mobile edge computing," in *2020 International Conference on Information Networking (ICOIN)*. IEEE, 2020, pp. 723–725.

[33] X. Huang, X. Yang, Q. Chen, and J. Zhang, "Task Offloading Optimization for UAV-assisted Fog-enabled Internet of Things Networks," *IEEE Internet of Things Journal*, vol. 4662, no. c, 2021.

[34] S. Zheng, Z. Ren, W. Cheng, and H. Zhang, "Minimizing the latency of embedding dependence-aware sfcs into mec network via graph theory," in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[35] M. Cui, S. Zhong, B. Li, X. Chen, and K. Huang, "Offloading autonomous driving services via edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 535–10 547, 2020.

[36] C. C. Byers, "Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled iot networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 14–20, 2017.

[37] L. Chunlin, M. Chuanli, C. Yi, and L. Youlong, "Optimal media service selection scheme for mobile users in mobile cloud," *Wireless Networks*, vol. 25, no. 6, pp. 3179–3192, 2019.

[38] W. Yang, X. Chi, L. Zhao, and Z. Xiong, "Qoe-based mec-assisted predictive adaptive video streaming for on-road driving scenarios," *IEEE Wireless Communications Letters*, vol. 10, no. 11, pp. 2552–2556, 2021.

[39] A. Mehrabi, M. Siekkinen, T. Kämäräinen, and A. yl Jski, "Multi-tier cloudvr: Leveraging edge computing in remote rendered virtual reality," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 2, pp. 1–24, 2021.

[40] Y. Chiang, C.-H. Hsu, and H.-Y. Wei, "Collaborative social-aware and qoe-driven video caching and adaptation in edge network," *IEEE Transactions on Multimedia*, vol. 23, pp. 4311–4325, 2020.

[41] G. Baranwal, R. Yadav, and D. P. Vidyarthi, "QoE Aware IoT Application Placement in Fog Computing Using Modified-TOPSIS," *Mobile Networks and Applications*, vol. 25, no. 5, pp. 1816–1832, 2020.

[42] X. He, R. Jin, and H. Dai, "Multi-hop task offloading with on-the-fly computation for multi-uav remote edge computing," *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 1332–1344, 2021.

[43] H. Wang, X. Wang, X. Lan, T. Su, and L. Wan, "Bsbl-based auxiliary vehicle position analysis in smart city using distributed mec and uav-deployed iot," *IEEE Internet of Things Journal*, vol. 10, no. 2, pp. 975–986, 2022.

[44] L. Pacheco, H. Oliveira, D. Rosário, Z. Zhao, E. Cerqueira, T. Braun, and P. Mendes, "Towards the future of edge computing in the sky: Outlook and future directions," in *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2021, pp. 220–227.

[45] E. Montero, C. Rocha, H. Oliveira, E. Cerqueira, P. Mendes, A. Santos, and D. Rosário, "Proactive radio-and qos-aware uav as bs deployment to improve cellular operations," *Computer Networks*, vol. 200, p. 108486, 2021.

[46] Z. Zhao, P. Cumino, C. Esposito, M. Xiao, D. Rosário, T. Braun, E. Cerqueira, and S. Sargento, "Smart unmanned aerial vehicles as base stations placement to improve the mobile network operations," *Computer communications*, vol. 181, pp. 45–57, 2022.

[47] S. R. Pandey, K. Kim, M. Alsenwi, Y. K. Tun, and C. S. Hong, "A crowd-enabled task execution approach in UAV networks towards fog computing," *Proceedings - 2021 IEEE International Conference on Big Data and Smart Computing, BigComp 2021*, pp. 246–251, 2021.

[48] H. Guo, X. Zhou, Y. Wang, and J. Liu, "Achieve load balancing in multi-uav edge computing iot networks: A dynamic entry and exit mechanism," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 725–18 736, 2022.

[49] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, 2019.

[50] N. Siasi, M. A. Jasim, A. Yayimli, and N. Ghani, "Service function chain survivability provisioning in fog networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1117–1128, 2021.

[51] H. Santos, D. Alencar, R. Meneguette, D. Rosário, J. Nobre, C. Both, E. Cerqueira, and T. Braun, "A multi-tier fog content orchestrator mechanism with quality of experience support," *Computer Networks*, p. 107288, 2020.

[52] H. Santos, I. Medeiros, C. Rocha, D. Rosario, E. Cerqueira, and T. Braun, "A mobility-aware flying edge computing service orchestration with quality of service support," *Submitted*, 2023.

[53] H. Santos, B. Martins, D. Rosario, E. Cerqueira, and T. Braun, "Mobility-aware service function chaining orchestration for multi-user augmented reality," *Submitted*, 2023.

[54] H. A. Khattak, S. U. Islam, I. U. Din, and M. Guizani, "Integrating fog computing with vanets: A consumer perspective," *IEEE Communications Standards Magazine*, vol. 3, no. 1, pp. 19–25, 2019.

[55] L. Pacheco, D. Rosário, E. Cerqueira, L. Villas, T. Braun, and A. A. Loureiro, "Distributed user-centric service migration for edge-enabled networks," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 618–622.

[56] C. Tang, C. Zhu, X. Wei, J. J. Rodrigues, M. Guizani, and W. Jia, "UAV Placement Optimization for Internet of Medical Things," *2020 International Wireless Communications and Mobile Computing, IWCMC 2020*, pp. 752–757, 2020.

[57] Z. Tan, H. Qu, J. Zhao, S. Zhou, and W. Wang, "UAV-aided edge/fog computing in smart IoT community for social augmented reality," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4872–4884, 2020.

[58] C. Tang, C. Zhu, X. Wei, H. Peng, and Y. Wang, "Integration of uav and fog-enabled vehicle: Application in post-disaster relief," *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, vol. 2019-Decem, pp. 548–555, 2019.

[59] Y. Li, H. Zhang, K. Long, S. Choi, and A. Nallanathan, "Resource Allocation for Optimizing Energy Efficiency in NOMA-based Fog UAV Wireless Networks," *IEEE Network*, vol. 34, no. 2, pp. 158–163, 2020.

[60] J. Santos, J. van der Hooft, M. T. Vega, T. Wauters, B. Volckaert, and F. De Turck, "Efficient orchestration of service chains in fog computing for immersive media," in *17th International Conference on Network and Service Management (CNSM)*. IEEE, 2021, pp. 139–145.

[61] L. Wang, M. Dolati, and M. Ghaderi, "Change: Delay-aware service function chain orchestration at the edge," in *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2021, pp. 19–28.

[62] T. Wang, J. Zu, G. Hu, and D. Peng, "Adaptive service function chain scheduling in mobile edge computing via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 164 922–164 935, 2020.

[63] I.-C. Lin, Y.-H. Yeh, and K. C.-J. Lin, "Toward optimal partial parallelization for service function chaining," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2033–2044, 2021.

[64] A. Medeiros, A. Di Maio, T. Braun, and A. Neto, "Service chaining graph: Latency- and energy-aware mobile vr deployment over mec infrastructures," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 6133–6138.

[65] H. Santos, D. Rosario, E. Cerqueira, and T. Braun, "Multi-criteria service function chaining orchestration for multi-user virtual reality services," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 6360–6365.

[66] S. Park and Y. Kwon. (2019) Research on the Relationship between the Growth of OTT Service Market and the Change in the Structure of the Pay-TV Market. [Online]. Available: https://ideas.repec.org/p/zbw/itse19/205203.html

[67] A. Zhang, Q. Li, Y. Chen, X. Ma, L. Zou, Y. Jiang, Z. Xu, and G.-M. Muntean, "Video super-resolution and caching—an edge-assisted adaptive video streaming solution," *IEEE Transactions on Broadcasting*, vol. 67, no. 4, pp. 799–812, 2021.

[68] A. S. Gomes, B. Sousa, D. Palma, V. Fonseca, Z. Zhao, E. Monteiro, T. Braun, P. Simoes, and L. Cordeiro, "Edge caching with mobility prediction in virtualized lte mobile networks," *Future Generation Computer Systems*, vol. 70, pp. 148–162, 2017.

[69] D. Alencar, C. Both, R. Antunes, H. Oliveira, E. Cerqueira, and D. Rosário, "Dynamic microservice allocation for virtual reality distribution with qoe support," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 729–740, 2021.

[70] J. Aguilar-Armijo, C. Timmerer, and H. Hellwagner, "Space: Segment prefetching and caching at the edge for adaptive video streaming," *IEEE Access*, vol. 11, pp. 21 783–21 798, 2023.

[71] B. Ahat, A. C. Baktır, N. Aras, İ. K. Altınel, A. Özgövde, and C. Ersoy, "Optimal server and service deployment for multi-tier edge cloud computing," *Computer Networks*, vol. 199, p. 108393, 2021.

[72] F. Wang and V. K. Lau, "Multi-level over-the-air aggregation of mobile edge computing over d2d wireless networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 10, pp. 8337–8353, 2022.

[73] H. A. Pedersen and S. Dey, "Enhancing mobile video capacity and quality using rate adaptation, ran caching and processing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 996–1010, 2015.

[74] P. Georgopoulos, M. Broadbent, A. Farshad, B. Plattner, and N. Race, "Using software defined networking to enhance the delivery of video-on-demand," *Computer Communications*, vol. 69, pp. 79–87, 2015.

[75] D. Yun and K. Chung, "Dynamic segment duration control for live streaming over HTTP," in *2016 International Conference on Information Networking (ICOIN)*. IEEE, 2016, pp. 206–210.

[76] Y. Shuai, M. Gorius, and T. Herfet, "Low-latency dynamic adaptive video streaming," in *2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*. IEEE, 2014, pp. 1–6.

[77] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial delay vs. interruptions: Between the devil and the deep blue sea," in *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*. IEEE, 2012, pp. 1–6.

[78] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing," in *Multimedia (ISM), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 494–499.

[79] Y. Qi and M. Dai, "The effect of frame freezing and frame skipping on video quality," in *2006 international conference on intelligent information hiding and multimedia*. IEEE, 2006, pp. 423–426.

[80] T. L. Saaty, "Analytic hierarchy process," *Encyclopedia of Biostatistics*, vol. 1, 2005.

[81] V. T. Library, "Download Guide: Big Buck Bunny, Sunflower version," accessed date: Aug 2019, [online] http://bbb3d. renderfarming.net/download.html.

[82] E. Liotou, D. Tsolkas, N. Passas, and L. Merakos, "Quality of experience management in mobile cellular networks: key issues and design challenges," *IEEE Communications Magazine*, vol. 53, no. 7, pp. 145–153, 2015.

[83] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, 1984.

[84] D. Mishra, A. M. Vegni, V. Loscrí, and E. Natalizio, "Drone networking in the 6g era: A technology overview," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 88–95, 2021.

[85] P. Yang, X. Cao, C. Yin, Z. Xiao, X. Xi, and D. Wu, "Proactive drone-cell deployment: Overload relief for a cellular network under flash crowd traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2877–2892, Oct 2017.

[86] A. Costa, L. Pacheco, D. Rosário, L. Villas, A. F. Loureiro, S. Sargento, and E. Cerqueira, "Skipping-based handover algorithm for video distribution over ultra-dense vanet," *Computer Networks*, p. 107252, 2020.

[87] J. Xue, Q. Hu, Y. An, and L. Wang, "Joint task offloading and resource allocation in vehicle-assisted multi-access edge computing," *Computer Communications*, vol. 177, no. 4, pp. 77–85, 2021.

[88] O. Kalinagac, G. Gür, and F. Alagöz, "Prioritization based task offloading in uav-assisted edge networks," *Sensors*, vol. 23, no. 5, p. 2375, 2023.

[89] Y.-J. Lai, T.-Y. Liu, and C.-L. Hwang, "TOPSIS for MODM," *European Journal of Operational Research*, vol. 76, no. 3, pp. 486–500, 1994.

[90] J. Sánchez-García, D. G. Reina, and S. L. Toral, "A distributed PSO-based exploration algorithm for a UAV network assisting a disaster scenario," *Future Generation Computer Systems*, vol. 90, pp. 129–148, 2019.

[91] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H.-S. Lee, "Furion: Engineering high-quality immersive virtual reality on today's mobile devices," *IEEE Transactions on Mobile Computing*, vol. 19, no. 7, pp. 1586–1602, 2019.

[92] J. Dahl, E. Marsh, C. Lewis, and F. C. Harris Jr, "Muvr: A multiuser virtual reality framework for unity," in *Proceedings of 31st International Conference*, vol. 88, 2022, pp. 61–70.

[93] A. Dhakal, X. Ran, Y. Wang, J. Chen, and K. Ramakrishnan, "Slam-share: visual simultaneous localization and mapping for real-time multi-user augmented reality," in *Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies*, 2022, pp. 293–306.

[94] J. Illing, P. Klinke, M. Pfingsthorn, and W. Heuten, "Less is more! support of parallel and time-critical assembly tasks with augmented reality," in *Proceedings of Mensch und Computer 2021*, 2021, pp. 215–226.

[95] M. V. Ngo, T. Luo, H. T. Hoang, and T. Q. Ouek, "Coordinated container migration and base station handover in mobile edge computing," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[96] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.

[97] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 756–764.

[98] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 3384–3391.

[99] H. Santos, D. Rosario, E. Cerqueira, and T. Braun, "Multi-criteria service function chaining orchestration for multi-user virtual reality services," in *IEEE Global Communications Conference*. IEEE, 2022.

[100] Z. Zhao, L. Pacheco, H. Santos, M. Liu, A. Di Maio, D. Rosário, E. Cerqueira, T. Braun, and X. Cao, "Predictive uav base station deployment and service offloading with distributed edge learning," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 3955–3972, 2021.

[101] P. Cumino, W. Lobato Junior, T. Tavares, H. Santos, D. Rosário, E. Cerqueira, L. A. Villas, and M. Gerla, "Cooperative uav scheme for enhancing video transmission and global network energy efficiency," *Sensors*, vol. 18, no. 12, p. 4155, 2018.

Declaration of consent

on the basis of Article 18 of the PromR Phil.-nat. 19

Name/First Name: Hugo Leonardo Melo dos Santos

Registration Number: 19-125-822

Study program: PhD of Science in Computer Science

Bachelor ☐ Master ☐ Dissertation ☑

Title of the thesis: Multimedia Service Orchestration in Multi-tier Edge Computing Environments

Supervisor: Prof. Dr. Torsten Braun and Prof. Dr. Eduardo Cerqueira

I declare herewith that this thesis is my own work and that I have not used any sources other than those stated. I have indicated the adoption of quotations as well as thoughts taken from other authors as such in the thesis. I am aware that the Senate pursuant to Article 36 paragraph 1 litera r of the University Act of September 5th, 1996 and Article 69 of the University Statute of June 7th, 2011 is authorized to revoke the doctoral degree awarded on the basis of this thesis.
For the purposes of evaluation and verification of compliance with the declaration of originality and the regulations governing plagiarism, I hereby grant the University of Bern the right to process my personal data and to perform the acts of use this requires, in particular, to reproduce the written thesis and to store it permanently in a database, and to use said database, or to make said database available, to enable comparison with theses submitted by others.

21.04.2023

Place/Date

Hugo Leonardo Melo Santos

Signature

Hugo Santos

huggosan@gmail.com | Belém, Brazil | 0000-0002-3189-0291 | Hugo Santos

Education

Federal University of Pará (UFPA) and Universität Bern *Belém, Brazil - Bern, Switzerland*

Joint-supervision PhD in Electrical Engineering and Computer Science 2018 - 2023

- Advisor: Prof. Dr Eduardo Cerqueira
- Joint-supervision: Prof Dr. Torsten Braun

Federal University of Pará (UFPA) *Belém*

MS in Electrical Engineering 2017 - 2018

- Advisor: Prof. Dr Eduardo Cerqueira

Federal University of Pará (UFPA) *Belém*

BS in Computer Engineering 2010 - 2016

- Advisor: Prof. Dr Eduardo Cerqueira

Technical Skills

Programming	C, C++, Python, Bash script, and Java
Professional Softwares	TensorFlow, SciPy, Scikit-learn, Jupyter-notebook
Development tools	Office, LaTeX, Inkscape, Git, VS Code, Overleaf, NS-3 and terminal
Languages	Portuguese (Native) and English (Advanced)

Publications during the PhD

[1] **H. Santos**, D. Alencar, R. Meneguette, D. Rosário, J. Nobre, C. Both, E. Cerqueira, T. Braun, "A Multi-Tier Fog Content Orchestrator Mechanism with Quality of Experience Support". *Computer Networks*, 2020.

[2] **H. Santos**, I. Medeiros, C. Rocha, D. Rosário, E. Cerqueira, T. Braun, "A mobility-aware flying edge computing service orchestration with quality of service support". *submitted*, 2023.

[3] **H. Santos**, B. Martins, D. Rosário, E. Cerqueira, T. Braun, "Multi-criteria service function chaining orchestration for multi-user virtual reality services". *GLOBECOM*, 2022.

[4] **H. Santos**, B. Martins, D. Rosário, E. Cerqueira, T. Braun, "Mobility-aware Service Function Chaining Orchestration for Multi-user Augmented Reality". *submitted*, 2023.

[5] L. Bastos, B. Martins, **H. Santos**, I. Medeiros, P. Eugênio, L. Marques, D. Rosário, E. Nogueira, E. Cerqueira, M. Kreutz, A. Neto, "Predictive Fraud Detection: An Intelligent Method for Internet of Smart Grid Things Systems". *submitted*, 2023.

[6] L. Marques, P. Eugênio, **H. Santos**, L. Bastos, H. Santos, D. Rosário, E. Nogueira, E. Cerqueira, M. Kreutz, A. Neto, "Analysis of Electrical Signals by Machine Learning for Classification of Individualized Electronics in the Internet of Smart Grid Things (IoSGT) architecture". *submitted*, 2023.

[7] **H. Santos**, P. Eugênio, L. Marques, H. Oliveira, D. Rosário, E. Nogueira, A. Neto, E. Cerqueira, "Internet of Smart Grid Things (IoSGT): Prototyping a Real Cloud-Edge Testbed". *Anais do XIV Simpósio Brasileiro de Computação Ubíqua e Pervasiva*, 2022.

[8] Z. Zhao, N. Emami, **H. Santos**, L. Pacheco, M. Karimzadeh, T. Braun, A. Braud, B. Radier, P. Tamagnan, "Reinforced-lstm trajectory prediction-driven dynamic service migration: A case study", *IEEE Transactions on Network Science, Engineering*, 2022.

[9] Z. Zhao, M. Karimzadeh, L. Pacheco, **H. Santos**, D. Rosário, T. Braun, E. Cerqueira, "Mobility management with transferable reinforcement learning trajectory prediction", *IEEE Transactions on Network and Service Management*, 2020.

[10] Z. Zhao, L. Pacheco, **H. Santos**, M. Liu, A. Di Maio, D. Rosário, E. Cerqueira, T. Braun, X. Cao, "Predictive UAV base station deployment and service offloading with distributed edge learning", *IEEE Transactions on Network and Service Management*, 2021.

[11] L. Pacheco, I. Medeiros, **H. Santos**, H. Oliveira, D. Rosário, E. Cerqueira, A. Neto, "A Handover Algorithm for Video Sharing over Vehicular Networks", *9th Latin-American Symposium on Dependable Computing (LADC)*, 2019.

[12] P. Bezerra, A. Melo, A. Douglas, **H. Santos**, D. Rosário, E. Cerqueira, "A collaborative routing protocol for video streaming with fog computing in vehicular ad hoc networks". *International Journal of Distributed Sensor Networks*, 2019.

[13] P. Cumino, W. Lobato Junior, T. Tavares, **H. Santos**, D. Rosário, E. Cerqueira, L. A. Villas,, M. Gerla, "Cooperative uav scheme for enhancing video transmission and global network energy efficiency", *Sensors*, 2018.