Universidade Estadual de Campinas
Instituto de Computação

Eduardo de Souza Gama

An Architecture for Adaptive Video Streaming: QoE Forecasting, Content Steering, and Scalable Resource Management through Edge-Cloud Computing

Uma Arquitetura para Streaming de Vídeo Adaptável: Previsão de QoE, Direcionamento de Conteúdo e Gerenciamento de Recursos Escalável por meio da Computação em Borda-Nuvem

CAMPINAS
2025

Eduardo de Souza Gama

An Architecture for Adaptive Video Streaming: QoE Forecasting, Content Steering, and Scalable Resource Management through Edge-Cloud Computing

Uma Arquitetura para Streaming de Vídeo Adaptável: Previsão de QoE, Direcionamento de Conteúdo e Gerenciamento de Recursos Escalável por meio da Computação em Borda-Nuvem

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Luiz Fernando Bittencourt
Co-supervisor/Coorientador: Prof. Dr. Roger Kreutz Immich

Este exemplar corresponde à versão final da Tese defendida por Eduardo de Souza Gama e orientada pelo Prof. Dr. Luiz Fernando Bittencourt.

CAMPINAS
2025

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

G14a

Gama, Eduardo de Souza, 1991-
An architecture for adaptive video streaming : QoE forecasting, content steering, and scalable resource management through edge-cloud computing / Eduardo de Souza Gama. – Campinas, SP : [s.n.], 2025.

Orientador: Luiz Fernando Bittencourt.
Coorientador: Roger Kreutz Immich.
Tese (doutorado) – Universidade Estadual de Campinas (UNICAMP), Instituto de Computação.

1. Redes de computadores. 2. Streaming de vídeo adaptável. 3. Computação de borda. 4. Computação em nuvem. 5. Qualidade de experiência (Redes de computadores). 6. Continuum de borda-nuvem. I. Bittencourt, Luiz Fernando, 1981-. II. Immich, Roger Kreutz. III. Universidade Estadual de Campinas (UNICAMP). Instituto de Computação. IV. Título.

Informações complementares

Título em outro idioma: Uma arquitetura para streaming de vídeo adaptável : previsão de QoE, direcionamento de conteúdo e gerenciamento de recursos escalável por meio da computação em borda-nuvem
Palavras-chave em inglês:
Computer networks
Adaptive video streaming
Edge computing
Cloud computing
Quality of experience (Computer networks)
Edge-cloud continuum
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Luiz Fernando Bittencourt [Orientador]
Kelvin Lopes Dias
Denis Lima do Rosário
Juliana Freitag Borin
Rodolfo Jardim de Azevedo
Data de defesa: 30-04-2025
Programa de Pós-Graduação: Ciência da Computação

Objetivos de Desenvolvimento Sustentável (ODS)
ODS: 9. Inovação e infraestrutura

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-2932-3765
- Currículo Lattes do autor: https://lattes.cnpq.br/6712266903078953

Dedicatory

To my family and friends.

Internet traffic is like
Dark Matter – mysterious
in both complexion and
impact on network
resources and App QoE.

(Lyn Cantor)

Acknowledgments

To God, for without faith, I would not have made it this far.

To my parents, Eduardo and Cláudia, for their unwavering belief in my abilities and unconditional love.

To my fiancee, Vitória Santos, for being a constant presence throughout this journey, offering me patience and the calm I needed during challenging times. You contributed to making this journey easier.

To my advisor and co-advisor, Prof. Dr. Luiz Fernando Bittencourt and Roger Kreutz Immich, for their dedication, commitment, and invaluable guidance in shaping my academic journey. I was able to learn a lot from you throughout this period.

To my colleagues and friends, I made inside and outside the university, thank you for the fun times I always had with you. The welcome I received from my first days in Campinas was essential for me to feel comfortable even though I was so far from home. Thank you very much.

To the faculty and staff of the UNICAMP Institute of Computing, who dedicated their time to participate in this research and contributed to my research in various ways.

Thanks to Ericsson S.A., and by the São Paulo Research Foundation (FAPESP) grant agreement no. 2021/00199-8, The Brazilian National Council for Scientific and Technological Development (CNPq) proc. 465446/2014-0, and National Education and Research Network (RNP) grant agreement no. 688941 for their financial support, which was essential to the development of this doctoral research.

And to all those who, in one way or another, contributed to or supported the realization of this goal—thank you.

Resumo

Os serviços de streaming de vídeo dominam uma boa parte do tráfego da Internet, impulsionados pelo crescimento no uso de dispositivos conectados e pela diversificação dos provedores de conteúdo. Esse cenário impõe desafios significativos aos operadores de rede, especialmente na garantia da Qualidade de Experiência (QoE) dos usuários, que já são complexas devido ao comportamento adaptativo dos mecanismos de Streaming Adaptativo Dinâmico sobre HTTP (DASH). Motivada por esse cenário, esta tese apresenta uma arquitetura baseada em aprendizado para streaming de vídeo adaptativo na rede Edge-Cloud, abordando desafios relacionados à QoE, escalabilidade e gerenciamento de recursos por meio da integração do Streaming Adaptável HTTP (HAS) com os Serviços de Redirecionamento de Conteúdo (CSS). O sistema proposto direciona dinamicamente as solicitações dos usuários para otimizar a entrega de vídeo de forma transparente para os clientes HAS.

Primeiramente, avaliamos uma arquitetura que aproveita ambientes multinível de Edge-Cloud, alocando dinamicamente serviços de vídeo entre múltiplos nós de borda e nuvem. Essa abordagem trata desafios como a seleção de nós de borda e a mobilidade dinâmica dos usuários, fatores que impactam diretamente a eficiência da rede e a satisfação do usuário. As avaliações experimentais destacam a necessidade de uma arquitetura capaz de se adaptar em tempo real às variações da demanda da rede, influenciando interrupções, estabilidade de taxa de bits e QoE em diversos cenários, incluindo ambientes com usuários móveis e condições de carga dinâmica.

Em seguida, um protótipo de uma arquitetura de streaming de vídeo adaptável que integra computação de borda e nuvem, CSS e técnicas de aprendizado de máquina (ML) para otimizar a entrega de vídeo foi desenvolvido. A arquitetura apresenta um Serviço de Planejamento capaz de implantar contêineres em nós de ponta para mitigar violações do Objetivo de Nível de Serviço (SLO). O uso de uma Rede Neural Recorrente (RNN) para prever potenciais violações de SLO permite ajustes preventivos na escalabilidade do serviço na ponta. O ambiente considera tecnologias de HAS por meio do player dash.js, direcionando dinamicamente as solicitações dos usuários para nós de ponta regionais. Assim, a arquitetura reduz a latência ao direcionar dinamicamente as solicitações dos usuários, já que essas solicitações são tratadas regionalmente, e garante uma QoE estável durante a reprodução de vídeo. Os resultados mostram melhorias na eliminação da configuração manual da rede, destacando o potencial da integração entre borda e nuvem para serviços de streaming de vídeo, redução do uso de cache e QoE sustentada em condições de rede dinâmicas. A arquitetura demonstra manutenção significativa da QoE, com redução de até 48% nas violações de SLO em comparação com uma abordagem reativa, evitando o provisionamento excessivo durante períodos de baixo tráfego.

Abstract

Video streaming services account for a significant portion of Internet traffic, driven by the growing adoption of connected devices and the increasing diversity of content providers. This landscape presents major challenges for network operators, particularly in ensuring Quality of Experience (QoE) metrics, which are already complex due to the adaptive behavior of Dynamic Adaptive Streaming over HTTP (DASH) mechanisms. Motivated by this, this thesis introduces a learning-based architecture for adaptive video streaming in the Edge-Cloud network, addressing challenges in QoE, scalability, and resource management by integrating HTTP Adaptive Streaming (HAS) with Content Steering Services (CSS). The proposed system dynamically directs user requests to optimize video delivery transparently for HAS clients.

Firstly, we evaluated an architecture that leverages multi-level Edge-Cloud environments, dynamically allocating video services across multiple edge and cloud nodes. This discussion addresses challenges such as edge node selection and dynamic user mobility, which directly impact network efficiency and user satisfaction. Experimental evaluations highlight the necessity for such environments to have an architecture capable of adapting to real-time variable network demands, affecting interruptions, bitrate stability, and QoE in various scenarios, including mobile user environments and dynamic load conditions.

Next, we developed a prototype of an adaptive video streaming architecture that integrates edge and cloud computing, CSS, and machine learning (ML) techniques to optimize video delivery. The architecture features a Planner Service capable of deploying containers at edge nodes to mitigate Service Level Objective (SLO) violations. Using a Recurrent Neural Network (RNN) to forecast potential SLO violations enables preemptive adjustments in the service scalability in the edge. The environment considers HTTP Adaptive Streaming (HAS) technologies via the dash.js player, dynamically steering user requests to regional edge nodes. Thus, the architecture reduces latency by dynamically steering user requests, as these requests are handled regionally, and ensures a stable QoE during video playback. The results show improvements in eliminating manual network configuration, highlighting the potential of edge-cloud integration for video streaming services, reduced cache usage, and sustained QoE under dynamic network conditions. The architecture demonstrates significant QoE maintenance, with up to 48% reduction in SLO violations compared to a reactive approach while avoiding over-provision during low traffic periods.

List of Figures

1.1 The distribution of downstream internet traffic across various content categories in 2023 [1]. 19

2.1 Overview of MPD DASH Model. 26
2.2 Overview of an End-to-End HTTP Adaptive Video Streaming pipeline, illustrating the process from multimedia source acquisition through encoding, segmentation, and Content Distribution servers, culminating in client-side playback. 27
2.3 A General Overview of the multi-tier network environment. 29
2.4 Overview of Content Steering Process Architecture: Directing user requests to the content delivery systems. 31
2.5 Containers vs Virtualization based on Virtual Machines [2]. 34

4.1 A General Overview of the multi-tier network environment on the left side, and the MIGRATE architecture on the right side (Adapted from [3]). . . . 45
4.2 Sequence Diagram of the video streaming communication. 47
4.3 Overview of the multi-tier network environment. 52
4.4 Bitrate switches, stalls, buffer size, the startup delay in seconds of a DASH player was requesting a video with ten bitrate levels varying from 50 to 4,500Kbps and from nodes in different tiers. 53
4.5 Users' Quality of Experience (QoE) average. Each bar group represents a different Planner decision: Full-Edge, Reactive, and Proactive. 54
4.6 Average QoE for each user group. Each mark represents the final user QoE for the following approaches: Cloud-only, Edge cache, and Mobility. 55
4.7 Average QoE for each user group. Each mark represents the final user QoE for the following approaches: Cloud-only, Edge cache, and Mobility. 57
4.8 Number of segments provisioned per node. 58

5.1 Workflow Architecture of a Content Steering-Enabled Adaptive Video System. 61
5.2 Workflow of the Content Steering Service (CSS) in the AVENUE Architecture, illustrating dynamic interactions between the user, edge cache, Cloud, and CSS for the content delivery provisioning. 65
5.3 Scenario a Content Steering-Enabled Adaptive Video System. 67
5.4 QoE Performance with CA, RA, and RR (y-axis) and the total number of users in the x-axis. 70
5.5 Cache Hit Performance with CA, RA, and RR (y-axis) and the total number of users in the x-axis. 71
5.6 Fairness Performance with CA, RA, and RR (y-axis) and the total number of users in the x-axis. 72

6.1 An Overview of the Proposed Architecture. Domain-Specific Service Monitoring is responsible for providing continuous monitoring resources. The Planner is responsible for realizing decision-making deployments at the edge. Then, Content Steering works as a load balancer. 76
6.2 The Domain-Specific Service Monitoring is responsible for providing continuous monitoring resources. 77
6.3 Users' QoE difference (ΔQ^u) between consecutive video segments varying user loads (20 to 60 users) and different Adaptive Bitrate (ABR) algorithms. 79
6.4 The Planner is responsible for realizing the decision-making deployments at the edge. 82
6.5 Scenario Evaluation Testbed for Adaptive Video Streaming System in Edge-Cloud Continuum. 85
6.6 Segment distribution across cloud and edge nodes for varying user counts (20–60). Each bar group shows provisioning under different Planner strategies: Full-Edge, Reactive, and Proactive. 89
6.7 Comparison of Service Level Objective (SLO) violations for different Planner algorithms: (a) Dynamic, (b) Throughput, and (c) Buffer, with varying user counts. 90

List of Tables

2.1 Comparison of Content Delivery Network (CDN) Switching Methods Based on Key Metrics . . . 32

3.1 Comparative Analysis of Related Works. . . . 39
3.2 Comparative Analysis of Related Works. . . . 41

4.1 Notation used in the proposed model 46

5.1 Execution time with standard deviation for Selection Algorithms. . . . 68
5.2 Review of the results considering the performance profiles: QoE, Cache hits, and Fairness load. Two scenarios analyses with 10 and 50 users. . . . 73

6.1 Summary of resource categories and their associated metrics used for system performance analysis. . . . 78
6.2 Table of notations and description. . . . 81
6.3 Average and Standard Deviation of Pull and Deploy Times 88

List of Acronyms

ABR : *Adaptive Bitrate*

AMust : *Adaptive Multimedia Streaming*

AP : *Access Point*

API : *Application Programming Interface*

AR : *Augmented Reality*

AVC : *Advanced Video Coding*

AV1 : *AOMedia Video 1*

CA : *Content Aware*

CAPEX : *Capital Expenditures*

CDN : *Content Delivery Network*

CSS : *Content Steering Service*

DASH : *Dynamic Adaptive Streaming over HTTP*

DinD : *Docker-in-Docker*

DNS : *Domain Name System*

DRM : *Digital Rights Management*

HAS : *HTTP Adaptive Streaming*

HD : *High Definition*

HDTV : *High Definition Television*

HTTP : *Hypertext Transfer Protocol*

HEVC : *High-Efficiency Video Coding*

HLS : *HTTP Live Streaming*

ILP : *Integer Linear Programming*

LLM : *Large Language Models*

LSTM : *Long Short-Term Memory*

LAC : *Location Area Code*

LRC : *Laboratório de Redes de Computadores*

LEO : *Low Earth Orbit*

LRU : *Least Recently Used*

MEC : *Multi-access Edge Computing*

ML : *Machine Learning*

MPEG : *Moving Picture Experts Group*

MPD : *Media Presentation Description*

NIST : *National Institute of Standards and Technology*

OCI : *Open Container Initiative*

OpenFog : *OpenFog Consortium*

OTT : *Over-the-Top*

PDMA : *Planner Decision Maker Algorithm*

QoE : *Quality of Experience*

QoS : *Quality of Service*

QUIC : *Quick UDP Internet Protocol*

RA : *Region-Aware*

RL : *Reinforcement Learning*

RNN : *Recurrent Neural Networks*

RR : *Round Robin*

RTMP : *Real-Time Messaging Protocol*

RTP : *Real-Time Transport Protocol*

SLA : *Service Level Agreement*

SLO : *Service Level Objective*

SD : *Standard Definition*

SSF : *Strongest Signal First*

TCP : *Transmission Control Protocol*

TTL : *Time-To-Live*

UDP : *User Datagram Protocol*

UHD : *Ultra-High Definition*

VoD : *Video on Demand*

VR : *Virtual Reality*

VVC : *Versatile Video Coding*

Contents

1 Introduction 18
1.1 Motivation . . . 18
1.2 Research Questions . . . 20
1.3 Main Contributions . . . 21
1.4 Thesis Structure . . . 22

2 Background 24
2.1 Basic Concepts of Adaptive Video Streaming . . . 24
2.2 Cloud and Edge Computing . . . 28
2.3 Content Steering at the Edge-Cloud Network . . . 30
2.4 Container management systems . . . 33
2.5 Quality of Experience . . . 34
2.6 Chapter Conclusions . . . 36

3 Related Work 37
3.1 Edge-Cloud Network-Based HAS Systems . . . 37
3.2 Service Allocation Management . . . 39
3.3 Chapter Conclusions . . . 41

4 Adaptive Video Streaming in Multi-tier Edge-Cloud Environment 43
4.1 MIGRATE . . . 44
4.1.1 Network and System Model . . . 45
4.2 Monitoring Component . . . 47
4.3 Performance Evaluation . . . 51
4.3.1 Impact of a Multi-tier Approach . . . 52
4.3.2 Analysis in Multi-tier Edge-Cloud Networks . . . 53
4.3.3 MIGRATE performance . . . 56
4.4 Chapter Conclusion . . . 59

5 An Architecture for Content Steering on Edge-Cloud Computing 60
5.1 Seamless Coordination in Content Steering Service . . . 61
5.2 Performance Evaluation . . . 66
5.2.1 Experimental Setup and Parameters . . . 66
5.2.2 Results and Discussion . . . 68
5.3 Chapter Conclusions . . . 72

6 Scalable Learning-Based Adaptive Video Streaming Architecture 74
6.1 Proposed Architecture Design Overview . . . 75
6.2 Measurement Methodology in Monitoring Structure . . . 77

6.3 Planner Service Implementation through Adaptive Video Streaming 80
6.3.1 Offline Component Operations 81
6.3.2 Online Operations 83
6.4 Performance Evaluation 85
6.4.1 Experimental Setup and Parameters 85
6.4.2 Results and Discussion 88
6.5 Chapter Conclusions 91

7 Conclusions and Future Works **92**
7.1 Conclusions 92
7.2 Future Works 93
7.2.1 Immersive Video Streaming 93
7.2.2 Sustainable HAS System 93
7.2.3 Learning HAS System 94
7.2.4 Emerging Protocols 94
7.2.5 Rights Management and Content Security 94
7.2.6 Satellites 95
7.3 Scientific Production 95

Bibliography **97**

Chapter 1

Introduction

Section 1.1 presents the motivation behind this thesis and states the core contributions. Section 1.2 outlines the fundamental research questions (RQ1–RQ3) essential for addressing the stated problem. Section 1.3 lists the contributions of this thesis and summarizes its main outcomes. Afterward, Section 1.4 provides an overview of the following chapters and their contents.

1.1 Motivation

Video streaming services account for the majority of Internet traffic [1, 4]. The increasing use of internet-connected devices and the emergence of content providers have fueled this demand. According to a report by Sandvine, video streaming accounts for nearly 65% of all Internet traffic, highlighting a significant and ongoing trend [1]. As shown in Figure 1.1, the distribution of downstream volume per subscriber across different content categories highlights the dominance of video streaming, with On-Demand streaming as the most significant contributor to Internet traffic across regions (Americas, Europe, and APAC). In the second position, live streaming stands out on Internet consumption, demonstrating the increasing reliance on bandwidth-intensive multimedia content applications. As a consequence, efficient delivery of adaptive video streaming has become an important area of research and development [5–7], as it not only improves users' QoE but also helps manage network congestion and reduce operational costs for service providers.

To meet the stringent requirements of video streaming services - such as high-quality communication channels and a constant, uninterrupted flow of information - major industry players like Microsoft, Apple, Adobe, and Netflix have adopted the HTTP Adaptive Streaming (HAS) paradigm [8, 9]. Given the similarities of the most HAS solutions, the Moving Picture Experts Group (MPEG) proposed a standard called Dynamic Adaptive Streaming over HTTP (DASH), and, similarly, Apple developed the HTTP Live Streaming (HLS) protocol, a proprietary streaming solution widely adopted in the video industry. Both DASH and HLS have become the backbone of video delivery, enabling more than 50% of video streams worldwide, making them the most utilized technologies by Over-the-Top (OTT) providers [1].

The HAS technologies procedure involves chunk-based streaming over HTTP. The

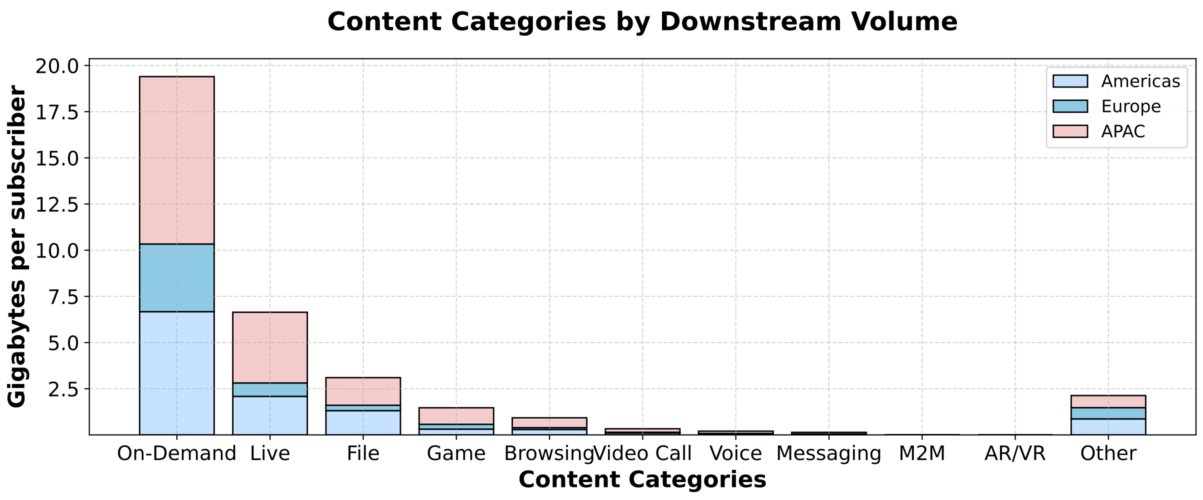


Figure 1.1: The distribution of downstream internet traffic across various content categories in 2023 [1].

video player can dynamically select the bitrate according to the perceived available bandwidth. This solution employs dynamic adaptation to variations in network conditions to provide a seamless (or at least smoother) streaming experience. The adoption of such technology is expected to grow due to the increasing demand for high-quality and high-resolution (e.g., 4K or 8K) video sequences, advancements in video codecs such as High-Efficiency Video Coding (HEVC) and Versatile Video Coding (VVC), and the increasing popularity of bandwidth-intensive immersive streaming experiences. As a result, an Adaptive Video Streaming Architecture presented in this thesis is developed to provide entirely transparent services to ensure that HAS clients realize their decision-making free from any confusion or risk.

In such systems, the Content Provider (CP) encodes each video segment into multiple formats and resolutions, such as Standard Definition (SD), High Definition (HD), and Ultra-High Definition (UHD). All versions of the video segments are transported and stored in the network core within CDNs. During playback, the user's player requests segments at a specific bitrate, as determined by the ABR algorithm. One or multiple in-network components with a broader view of the network can provide video players with fair network resources and assist them in making precise adaptation decisions. This will enable video usage with extremely high bandwidth. Video streaming systems include a decision point embedded within the CDN, responsible for managing CDN selection solutions [10–14]. Modern HAS technologies like DASH and HLS have standardized a session-aware Content Steering Service (CSS). During each session, users regularly communicate with the CSS to receive updated instructions on which CDN to retrieve video segments. For example, a user may initially fetch video segments at a given node A during a streaming session. Suppose the cache in node A becomes congested or experiences delays. In that case, the CSS can instruct the user to switch to cache B, which offers better playback performance. The Adaptive Video Streaming system must be able to switch between different CDNs seamlessly during a video session.

Moreover, Edge computing has become an architectural video delivery component, allocating multimedia content closer to end users and handling requests geographically near them rather than forwarding to distant servers. Each system component fulfills a specialized role in adaptive video streaming. In session-aware scenarios, implementing mechanisms that can dynamically direct users between the Edge and Cloud ensures seamless connectivity, significantly reducing latency and minimizing playback errors [15].

To ensure service reliability and user satisfaction, adaptive video streaming architectures must incorporate mechanisms that go beyond traditional performance metrics by explicitly considering QoE as a central Key Performance Indicator (KPI) in enforcing SLOs [16–18]. Unlike conventional metrics such as throughput or latency, QoE reflects the end-user's perceived quality, encompassing factors like video resolution, playback smoothness, startup delay, and rebuffering frequency. In this thesis, QoE is treated as a first-class metric for decision-making across the Edge-Cloud continuum. The integration of QoE into SLO ensures that adaptation logic, cache redirection, and scaling decisions prioritize user-centric outcomes. This is particularly relevant in the proposed architecture, where session-aware services like the CSS and Machine Learning (ML)-based forecasting mechanisms continuously monitor and optimize system behavior. By aligning system responses with QoE thresholds, an adequate architecture can maintain high user satisfaction even under volatile network conditions or variable content demand.

This thesis proposes and analyzes a hierarchical video streaming approach and designs an architecture using Edge-Cloud computing concepts. The thesis contribution introduces the CSS mechanism into adaptive video streaming systems, in which the user is steered to a node based on information received from the Controller and traffic demands. Furthermore, the thesis presents a mechanism that employs ML for forecasting time-series data, enabling advanced decision-making in the scaling of video streaming services onto edge nodes in regional networks.

1.2 Research Questions

Although many research works address VoD streaming services combined with edge/cloud computing, some aspects are rarely explored in current works [5, 19]. Typically, video streaming architectures aim to reduce traffic load and improve video delivery's Quality of Experience (QoE). However, such solutions often overlook the behavior of the video player used by the user, as well as aspects related to interoperability between the users, CDNs, and the decision-making mechanisms within the Edge-Cloud environments.

- HAS systems face challenges in delivering high-quality video experiences in dynamic Edge-Cloud environments, where resource availability and user demand fluctuate significantly in a shorter time window. To address these challenges, CSS facilitates the dynamic selection of the most suitable content sources considering the actual network services state. Consequently, the player can quickly adapt to changing environments and view seamless video playback while the architecture optimizes the Edge resource utilization.
  Research Question 1: How can HAS systems leverage CSS to improve their adaptability across dynamic Edge-Cloud environments?

- Maintaining users' QoE guarantees within the specified SLO is an important requirement for both Content Providers and Network Providers. However, the dynamic nature of Edge-Cloud environments, characterized by network fluctuation and variations in resource allocation, makes it challenging. To address this, using time-series data and ML techniques enables the system to take preemptive actions instead of reacting to problems after they arise. By analyzing these patterns, the system can autonomously scale up and down the services and redistribute workloads while maintaining the system's performance.
  Research Question 2: Can ML techniques accurately forecast future SLO violations using historical time-series data?

- Adopting ML for forecasting QoE also introduces complexity, as it demands reliable data collection pipelines, additional resource consumption, and careful feature selection. Addressing these requirements efficiently can ensure that the benefits of predictive modeling outweigh the associated costs. In adaptive video streaming systems, a proactive ML-driven strategy for anticipating QoE issues represents a transformative approach. By forecasting performance problems before they occur, this strategy outperforms reactive methods, which only address post-occurrence problems.
  Research Question 3: What is the impact of the proactive strategy (ML model) in the network performance compared to the reactive approach?

To address the abovementioned issues, this thesis designed a prototype video delivery architecture focusing on DASH technology. The architecture integrates recent CSS standards to support real-time decision-making processes. Where CDN selection is automated and continuously optimized to match network conditions and viewer demand.

1.3 Main Contributions

The main contribution of this thesis is the proposal and deployment of an architecture for HAS Systems, leveraging Edge-Cloud computing concepts to enhance efficiency, scalability, and users' QoE. The architecture adheres to DASH Industry Forum (DASH-IF) Working Group specifications for creating an independent, self-managed adaptive video streaming. This contribution consists of a shared database repository with relevant information and three main components: 1) Domain-Specific Service Monitoring, 2) Planner Service, and 3) CSS. Multimedia content services at CDN have been abstracted to maintain clarity and emphasize the core functional components. The CDN works in a stateless manner, while the proposed architecture works together to scale cache services at the edge in order to respect SLO guarantees.

The architecture integrates a methodology for supporting Recurrent Neural Networks (RNN)-based ML models, particularly Long Short-Term Memory (LSTM), to enhance adaptive video streaming. This methodology is divided into three key phases: data gathering, model training, and real-time execution. The data-gathering phase employs Domain-Specific Service Monitoring to collect diverse metrics from various sources across the Edge-Cloud continuum, including QoE feedback, infrastructure performance, and network statistics. The collected data undergoes preprocessing, such as feature selection, normalization, and dimensionality reduction, to ensure that only relevant data is used for training the models.

In the training phase, historical data is used to train LSTM models offline, allowing them to learn temporal dependencies and patterns in network behavior. The trained models, preprocessing function, and data window size configuration are deployed in the online phase. During the execution phase, the decision-maker algorithm leverages these models to predict QoE violations as SLO and proactively scale resources, such as deploying cache containers at edge nodes. This proactive strategy minimizes SLO violations and ensures certain QoE levels under fluctuating network conditions. By integrating continuous monitoring, predictive modeling, and resource management, the architecture effectively balances Edge and Cloud utilization while dynamically adapting to user demands.

1.4 Thesis Structure

This document is organized as follows:

- **Chapter 2** provides background definitions for concepts used throughout this thesis. It begins by exploring the principles of HAS paradigm, focusing on DASH protocol, detailing their architectures, functionalities, and roles in modern video delivery systems. Highlight the Edge-Cloud Network and its significance in addressing latency, scalability, and resource optimization challenges in video streaming. Additionally, it reviews key concepts such as CSS and QoE metrics, emphasizing their relevance to the thesis objectives.

- **Chapter 3** presents the literature review, covering the state-of-the-art in adaptive video streaming over the Edge-Cloud. Moreover, These insights underpin the contributions of the thesis, emphasizing the development of an architecture using CSS, an orchestration framework tailored to enhance the QoE, and its adaptive scalability in Edge-Cloud environments.

- **Chapter 4** presents an analysis and evaluation of an adaptive video streaming architecture in an Edge-Cloud environment. The chapter introduces MIGRATE, a multi-tiered architecture designed to optimize video content delivery by leveraging multi-layer resources at that edge to enhance user QoE. It outlines challenges like edge node selection impacting network efficiency and user satisfaction.

- **Chapter 5** delves into the design, implementation, and evaluation of the CSS within adaptive video streaming systems. It emphasizes seamless coordination among users, edge caches, cloud, and CSS to content delivery and improve different resource metrics. The chapter details the system's workflow, illustrating how real-time metrics and regional proximity guide the load balancing dynamic.

- **Chapter 6** presents an Adaptive Video Streaming architecture designed for scaling containers across the Edge-Cloud according to demand fluctuations. We highlight three main contributions: i) The proposed architecture integrates the CSS to select a suited endpoint during video sessions dynamically; ii) Through continuous real-time monitoring, the system ensures seamless, up-to-date information about video playback, network, and node changes. iii) A ML model is deployed capable of forecasting users QoE.

- **Chapter 7** concludes this thesis, shows an overview of the contributions, discusses the research questions, describes directions for future work, and presents the publications produced as a result of this thesis.

Chapter 2

Background

This chapter presents the foundational concepts and definitions explored in this thesis. Section 2.1 presents a vision that ranges from encoding video content into digital formats to packing video in HAS content, specifically in DASH content, for continuous delivery. Section 2.2 introduces the potentials of the Edge-Cloud Networks. Each component within these architectures ensures a seamless viewing experience for end users. In Section 2.3, we designed an architecture for Content Steering Services. Initially, a centralized steering server orchestrates operations, followed by an Edge-Cloud Continuum approach. The service's main objective is to show users' directness to a video streaming service in real time, balancing the load and improving scalability. Section 2.4 highlights the advantages of containerization for resource management and microservices deployment. Section 2.5 delves into the objective QoE metric used to evaluate and enhance user satisfaction. Finally, Section 2.6 concludes the chapter by summarizing its key insights to the thesis.

2.1 Basic Concepts of Adaptive Video Streaming

Although traditional video delivery methods have their merits, they face certain limitations when it comes to connection-oriented protocols like Real-Time Messaging Protocol (RTMP) [20] or Real-time Transport Protocol (RTP) [21] for connectionless protocols. Typically, the media server pushes the media to the client in these protocols. However, these methods require help scaling the infrastructure, depending on specific vendors, and having high maintenance costs necessitating complex and expensive servers. Over time, new technologies have emerged to overcome these limitations and enhance video streaming [5]. One of the most popular solutions is HAS, which offers several benefits over traditional methods. Adaptive streaming is a dynamic video delivery technique that adjusts the video playback quality in real time based on the user's network conditions, with the primary goal of providing seamless viewing playback by adapting to changes in available bandwidth. DASH [22] and Apple's HLS [23] are two of the most popular HAS solutions in use today. These methods, which handle over half of all video streams, are particularly prevalent in on-demand and live-streaming platforms. It is essential to understand the mechanisms and technologies discussed in HAS paradigm with a focus on DASH to understand the fundamental concepts related to adaptive video streaming. This section introduces key terms and definitions that underpin adaptive video delivery systems.

Video Segmentation and Codecs

Video is a sequence of frames displayed at a specific frame rate to create the perception of motion. In adaptive video streaming scenarios, videos are divided into segments or chunks, smaller portions of the original video. Each segment typically represents a few seconds of playback, commonly for 2–10 seconds.

Different coding formats can encode the video, such as Advanced Video Coding (AVC), HEVC, VVC, VP9, and AOMedia Video 1 (AV1). AVC, a widely adopted standard, offers broad compatibility but falls short in compression efficiency for high-quality content. HEVC, its successor, addresses these limitations with advanced techniques like larger block sizes and improved motion compensation, offering better efficiency for UHDTV and streaming, though licensing issues slow its adoption. VP9 and AV1, as open and royalty-free formats, provide comparable efficiency to HEVC, gaining traction in web browsers and online platforms despite limited hardware support. VVC, the latest standard, delivers superior compression for high-resolution and immersive content but challenges licensing and hardware adoption. Together, these codecs cater to evolving demands in video streaming and delivery.

These codecs generate multiple encoded representations of the video at different bitrates, attending the diverse network bandwidths. This process ensures a bitstream compatible with decoders on various client devices.

Media Presentation Description

The segments of each video are listed in a manifest file called Media Presentation Description (MPD), which further contains the available representations and provides details such as video duration, content availability, media types (e.g., H.264, H.265, etc.), resolutions, minimum and maximum bandwidths, alternative encoded multimedia components, locations of media segments, and other content characteristics.

In Figure 2.1, we illustrate a typical structure of a MPD file, it can give us an understanding of the DASH standard [24]. The MPD is organized into multiple Period elements, each corresponding to a distinct media content segment, enabling temporal segmentation for seamless playback. Within each Period, AdaptationSet elements group media streams based on codec, resolution, or bitrate characteristics, facilitating adaptability to varying network conditions and device capabilities. Each AdaptationSet comprises several Representation elements, which provide details of the available quality levels, defined by attributes such as bitrate (e.g., 100 Kbit, 200 Kbit). Representations include essential components such as the Init Segment for initialization and subsequent media Segment URLs, ensuring efficient content delivery. The diagram also introduces SubSet elements within a Period, which enable the grouping of specific AdaptationSets for advanced use cases, such as tailored content delivery or targeted optimizations. This comprehensive structure underscores the flexibility of DASH in supporting adaptive video streaming, making it suitable for diverse application scenarios in adaptive multimedia systems.

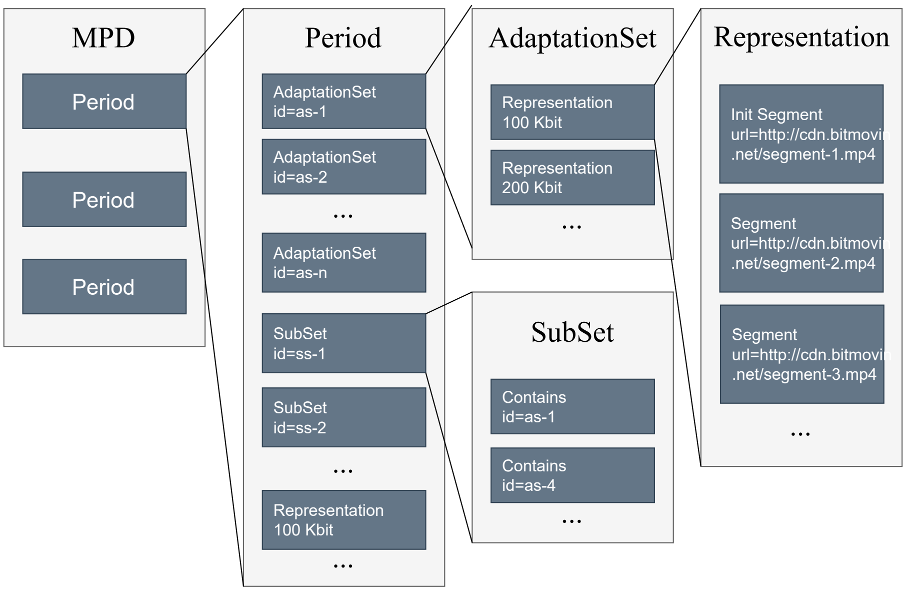


Figure 2.1: Overview of MPD DASH Model.

Application-Network Interaction in Video Streaming

Regarding the HAS paradigm, video streaming applications are traditionally built upon the Transmission Control Protocol (TCP) [25]. Within this paradigm, application-layer ABR logic operates independently from the transport layer. While ABR logic dynamically adjusts streaming quality based on available bandwidth, TCP congestion control regulates data flow to avoid network congestion. However, the lack of coordination between these mechanisms often results in inefficiencies and concerns about fairness. ABR logic, which operates independently at the application layer, tends to adjust the streaming quality too frequently without considering the network conditions monitored by TCP congestion control. This can lead to unfair bandwidth allocation among different sessions and can also cause instability in network utilization.

The emergence of the Quick UDP Internet Protocol (QUIC) transport protocol, as presented in [26], has opened up new avenues for improving the interaction between video streaming applications and transport functionalities. While certain studies advocate for QUIC as a promising solution for video streaming, others engage in debates comparing it to TCP [27]. Furthermore, concerted efforts have been made to enhance QUIC's support tailored to video streaming applications.

QUIC performs better than the TCP protocol, where low-throughput use cases are necessary. However, in some cases, there are degradations compared to TCP and previous generations of HTTP, where the network bandwidth is poor or there is a variation in the bandwidth, as is typically experienced during mobility. The dynamics of application-network interaction for video streaming, particularly concerning integrating emerging transport protocols like QUIC, represent fertile ground for future investigation.

End-to-End HTTP Adaptive Video Streaming Pipeline

Figure 2.4 illustrates the end-to-end process of the HAS paradigm. The process begins with the video source, which provides the raw video content. This source can be a live stream, a camera feed, or a pre-recorded video. The raw video is first passed through an encoder, compressing it into multiple bitrate versions. These versions enable adaptive streaming, allowing the content to adjust dynamically to varying network conditions and device capabilities, ensuring an optimal viewing experience for users.

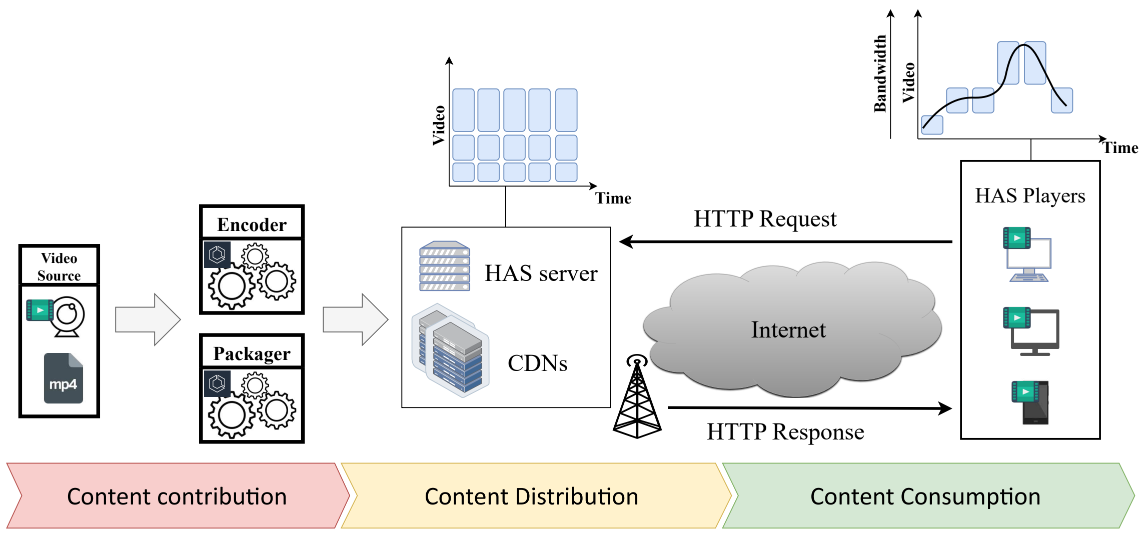


Figure 2.2: Overview of an End-to-End HTTP Adaptive Video Streaming pipeline, illustrating the process from multimedia source acquisition through encoding, segmentation, and Content Distribution servers, culminating in client-side playback.

Next, the encoded video is processed by the packager, which segments it into small, uniform chunks. These chunks are created at different quality levels. Concurrently, the server generates a manifest file, known as the MPD, which provides essential metadata to guide the playback process. The segmented and packaged content is then uploaded to the HAS server, which serves as the central storage and delivery point. The server hosts the video chunks and manifests, making them accessible to users through standard HTTP protocols.

The video content is delivered over the Internet, bridging the server and the end users. Finally, at the user's end, HAS players on devices such as desktops, laptops, or mobile phones dynamically request video chunks. Using ABR algorithms, the players select the appropriate quality level for each chunk, depending on current network conditions and device performance. These algorithms aim to ensure a high QoE for users, even in the presence of bandwidth fluctuations caused by network congestion control, signal strength variations, packet loss, and more. While such fluctuations are common in public Internet environments, they can also occur in private networks, such as home or managed networks, where admission control and various Quality of Service (QoS) tools are typically employed.

2.2 Cloud and Edge Computing

Cloud computing is a model that enables ubiquitous, convenient, and on-demand access to a shared pool of configurable computing resources. This cloud model provides large-scale computational resources with high latency in some ecosystems [28,29]. This model can be servers, storage, applications, and services, which can be rapidly provisioned and released with minimal management effort or service provider interaction. These cloud systems typically utilize servers through which traffic flows via the network core. Additionally, device connections are generally established by static users and stable Internet links [30]. In this way, creating these systems partially addresses the issues of scalability, availability, and interoperability but simultaneously introduces new challenges (e.g., increased latency and core network congestion).

Edge computing aims to bridge the gap between the cloud and end-user devices, addressing the need for processing, communication, and storage resources closer to where the end-users are. This layered model provides ubiquitous access to a shared continuum of scalable computing resources, ensuring high performance and adaptability [31]. By enabling the deployment of latency-sensitive distributed applications and services, edge computing leverages fog nodes—physical or virtual components that act as intermediaries between intelligent end devices and centralized cloud infrastructure-enhancing responsiveness.

Edge nodes can be organized into multi-level architectures—vertically (to support isolation), horizontally (to support federation), or based on the latency between edge nodes and end-users. Edge computing minimizes the response time of supported applications and provides local computing resources to end devices while offering network connectivity to centralized services when needed.

Figure 2.3 depicts a multi-tier network architecture, which is composed of a heterogeneous set of devices and applications using distributed computing resources through multi-access communication technology. Below the cloud layer, the network edge is divided into three layers that are organized hierarchically [3]. Core Network Regional Edge can manage coordination across the distributed infrastructure in this multi-tier ecosystem, for example, in a smart city, followed by the Access Network Edge, which supports a few dozen to maybe a few hundred local nodes at the middle layer of the edge computing infrastructure. The Edge gateways can be distributed on local edge nodes, where the node re-transmits video content employing wired or wireless communication.

Using the edge of the network helps to improve user satisfaction but can also bring negative impacts. When many users request video segments at the edge nodes, these surrogated nodes consider static users and just one edge node tier. These nodes are not ready to deal with the constant changes in the number of users and switching between different Access Point (AP) [32]. It can offer several merits in different types of video streaming services, including:

- **Video-on-Demand (VoD) Streaming:** The implementation of techniques such as most popular content caching or prefetching at the edge in advance enhances the scalability and reliability of VoD services. By offloading demand from cloud-based

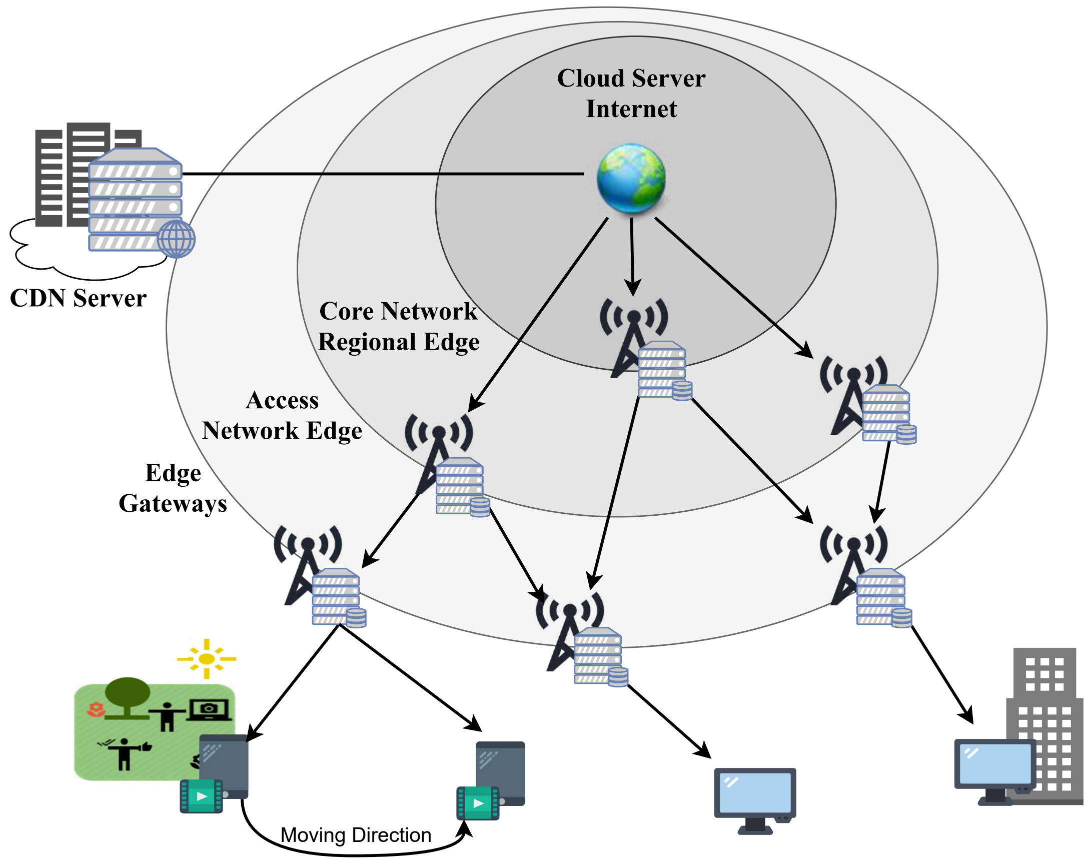


Figure 2.3: A General Overview of the multi-tier network environment.

infrastructure (e.g., origin servers), these strategies facilitate faster content delivery, reduce latency, and enhance overall user QoE.

- **Live Streaming:** By leveraging techniques such as caching, transcoding, and processing can be optimized to reduce latency and improve the video streaming quality. Deploying these processes closer to the end user, where data is generated and consumed.

- **Immersive Video Streaming:** Processing and rendering content close to the user, immersive streaming can significantly reduce latency and improve applications' performance. This is particularly critical for applications such as Augmented Reality (AR) or Virtual Reality (VR), which demand real-time processing and rendering of high-quality video content to ensure seamless user experiences.

Also, there are opportunities for resource management in multi-tier edge/cloud networks to provide video transmission [33,34]. The advantages of nodes closer to end-users can improve the network's functioning. Below, we discuss some insights that can be used in favor of network and provider admin in the infrastructure for video transmission.

- **Improving the User's QoE:** In a multi-tier environment with more than one-tier resource availability, the resources can host the video content near the end-users. This allows for reducing latency and mitigating the load on the network core. The edge nodes combine specific resources to transmit and integrate video streaming services in such communication environments. Within this context, integrating QoE feedback models inside the player creates an opportunity to improve user satisfaction. The results reported in [35] suggest improving the users' satisfaction using the edge multi-tier network is possible. Depending on the video service allocation level, it is possible to provide smooth video playback even in an overloaded network.

- **Potential Bandwidth Saving:** Videos streamed in higher quality use more network bandwidth. Consequently, provisioning from the Cloud will incur high communication expenses in the core network. Delivering part of the video along the network instead of sending the entire video to an edge server or lowering the encoding quality of uninteresting video portions can significantly save bandwidth. Different delivery approaches can have different performances to reduce the uplink bandwidth use. Because of that, it is possible to contemplate an end-to-end design of video streaming, wherein the edge server adapts the video streams based on uplink and downlink bandwidth capacities. Additionally, new forms of video content are being generated today and may present opportunities for bandwidth saving and video services orchestration in edge-cloud infrastructures.

- **Cacheability:** Nodes at the edge provide resources to VoD providers to allocate their caches. The allocated caches make multi-hops within the edge to serve the end user. With this characteristic, different possibilities in the multi-tier network still have to be studied, such as cache allocation, placement, replacement, and selection caches, which usually make decisions in real time. These problems can offer a better video streaming service. In addition, an orchestrator has to consider the user's mobility and the possibility of predicting the direction of movement. As the user changes their trajectory, new caching mechanisms can use this information to optimize streaming video services.

In this thesis, we leverage Edge capabilities to improve user QoE and overall network performance across all proposed contributions. Although the primary application focuses on VoD, the solutions presented are designed with adaptability in mind and can be extended to support live streaming. By addressing key challenges in resource allocation from the cloud to the last mile network [36], our approach aims to deliver seamless and efficient video streaming experiences across diverse application domains.

2.3 Content Steering at the Edge-Cloud Network

Leveraging edge and cloud resources allows for greater scalability and flexibility in content delivery infrastructure. Edge servers can handle localized spikes in demand and deliver content closer to end-users, while cloud resources provide additional capacity and support for global content distribution. Overall, CSS at the network offers speed, scalability, reliability, and personalization, enabling organizations to deliver content more efficiently and effectively to end users worldwide [37, 38].

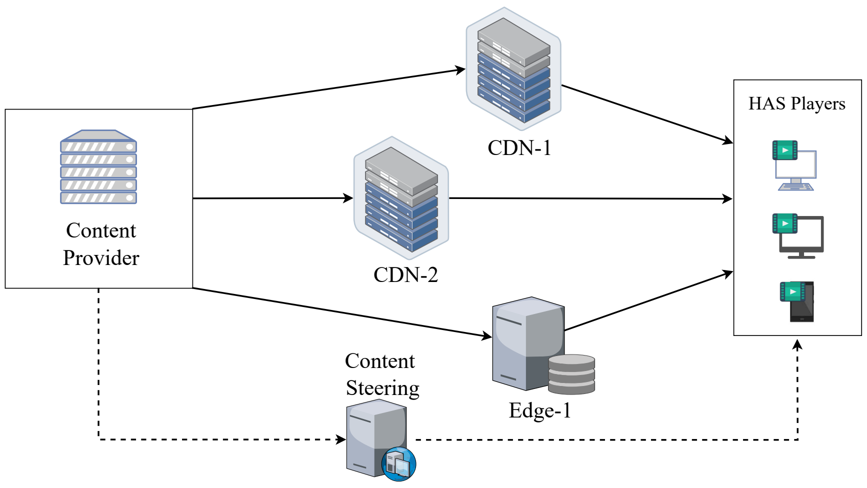


Figure 2.4: Overview of Content Steering Process Architecture: Directing user requests to the content delivery systems.

Figure 2.4 illustrates the Content Steering implementation. The design consists of architecture for video streaming, integrating CDNs, edge computing, and a CSS. The system's core is the CP, working as the origin server, hosting the multimedia content and distributing it to CDN-1 and CDN-2. The architecture can also incorporate Edge Nodes (e.g., Edge-1) as local caching layers, further reducing latency and enhancing delivery efficiency. Content Steering employs its load-balancing strategy based on a list of available servers, dynamically deciding which regional edge servers or CDNs should handle user requests. Upon receiving a request, the CSS node sends the response based on the list of servers by the appropriate Uniform Resource Locator (URL)s. The system's solid arrows represent the multimedia data flow, while the dashed arrows depict the steering mechanism used for load balancing. This design shows great potential in addressing the following limitations:

- **Scalability:** This implementation achieves scalability levels comparable to CDNs or platforms responsible for executing edge functions. By distributing processing tasks across multiple points, it can seamlessly accommodate growing demands without compromising performance;

- **Cost-Efficient Deployment:** Deploying this implementation becomes significantly more economical due to reduced bandwidth and per-request costs at CDNs or edge platforms compared to the higher egress traffic costs associated with cloud platforms. This cost-effectiveness makes it a more viable option for organizations seeking efficient content delivery solutions;

- **Enhanced Responsiveness:** The two-phase implementation enhances responsiveness by enabling lower Time-To-Live (TTL) between clients and servers. This reduced latency facilitates quicker interactions between users and content servers, leading to smoother content delivery experiences.

Reducing response times is paramount for unlocking additional functionalities within the system. When TTL values become shorter than the size of the player's buffer (e.g., 1-30 seconds), it automatically enables QoS and QoE optimizations [39]. For instance, it helps prevent buffering issues and allows clients to access higher-quality streams seamlessly. Moreover, shorter response times are essential for supporting mobile applications, disaster recovery scenarios, and other use cases requiring swift and efficient content delivery.

Table 2.1: Comparison of CDN Switching Methods Based on Key Metrics

Method	**Ease of Use**	**Switch Speed**	**Accuracy**	**Complexity**	**Scalability**	**Best For**
DNS-based	High	Slow	Low	Low	High	Simple systems
Manifest Rewrite	Medium	Medium	Medium	Medium	Medium	Live streaming
Server-side	High	Medium	Low	Low	Medium	Central control
Client-side	Low	Fast	High	High	Medium	User experience
Content Steering Service	Medium	Fast	High	Medium	High	Dynamic CDN switching

CDN switching methods vary significantly in complexity, speed, accuracy, and scalability, each suited to specific use cases [40]. Table 2.1 lists the methods and approaches tried in the past. As easily observed, none of these approaches is perfect. Domain Name System (DNS)-based switching is the simplest to implement, keeping source URLs constant, but it suffers from significant delays—often several minutes—when updating routes. This slow response severely impacts QoE during CDN failures or overloads, making it unsuitable for dynamic environments. Manifest rewrite, on the other hand, enables midstream switching by modifying media manifests, reducing cascading failures, and offering more adaptability. However, it introduces the risk of errors during rewrites and delays while detecting and responding to CDN issues, which can disrupt playback.

Server-side switching simplifies operations with centralized control, where servers manage CDN selection based on collective client data. While easy to deploy, this approach lacks real-time adaptability and fails to account for individual user conditions, limiting its effectiveness in dynamic or heterogeneous networks. Client-side switching, in contrast, excels in accuracy and responsiveness by leveraging real-time metrics from individual clients, allowing seamless midstream transitions. However, its complexity and high implementation costs make it challenging, especially for platforms with restricted or proprietary players.

The CSS is an efficient solution for dynamic CDN switching, integrating speed, accuracy, and scalability. Dynamically selecting CDNs based on real-time client and network data ensures better decision-making traffic distribution for QoE and reliability. It uses standards like DASH and HLS to provide a future-proof mechanism for intelligent CDN switching, optimizing video delivery in diverse network conditions.

2.4 Container management systems

Containerization is a lightweight virtualization technology encapsulating services and applications in small standard units. Unlike traditional machine virtualization, where a virtual machine is instantiated with the entire operating system, libraries, and other dependencies, the container encompasses only the application (or service) of interest and the dependencies necessary for its execution. Scalability is enhanced using this approach, as containers consume fewer resources and generate lower overheads when compared to virtual machines. Each container is defined from an image and built on a stacked-layer structure, where each change in the subsequent layer creates a new layer. Except for the topmost layer, all the others are read-only and can be reused to compose other images. Containers share the same kernel and isolate application processes from the rest of the operating system.

In this context, Docker[1], CoreOS, and other interested parties in the container industry established the Open Container Initiative (OCI) to define standard specifications for container technologies. Currently, OCI proposes three standards: runtime, image, and container distribution. Therefore, OCI allows compatibility among all artifacts produced in compliance with the proposed standards. Among the use cases related to OCI compliance, we highlight the Internet of Things and Edge Computing.

The container technology offers distinguished characteristics from other virtualization approaches: portability, volatility, and resource consumption. Portability is evident in how containers are built, as the application is encapsulated with all the dependencies necessary for its execution, and a container will have the same execution behavior in any environment in which it is provisioned. Regarding volatility, due to the ephemeral nature of the container, all possible data persistence is carried outside the container. Therefore, in case of execution issues related to a failed container, it is not necessary to carry out debugging to identify the fundamental problem affecting its functioning, but rather to destroy it and create a new instance without the need to take care of application data, which can be costly. Finally, as previously described, as it is an environment with a small number of additional processes and services, as opposed to virtual machines, the container consumes fewer resources, optimizing the physical resources available on the host system.

Fig.2.5 shows the main differences between containers about virtualization based on virtual machines.

It is important to note that running containers also require a runtime, the most famous of which is Docker. Container runtimes can be classified into two types: low-level and high-level. Low-level runtimes adhere to OCI specifications. On the other hand, some runtimes have an extra layer because they present more functionalities than the minimum required for just running containers. Examples of these high-level runtimes are Docker, Podman, and CRI-O. A container image should be passed to the runtime for a runtime to execute an image of interest. In the context of containers, images are stored in repositories referenced to the runtime at execution time. These repositories, or Registries, are well-structured and can be public or private. In the first case, the stored images are open to the public. In the second case, the images are made available to a specific audience, for example, one

[1]Docker is a container platform: `https://www.docker.com/`

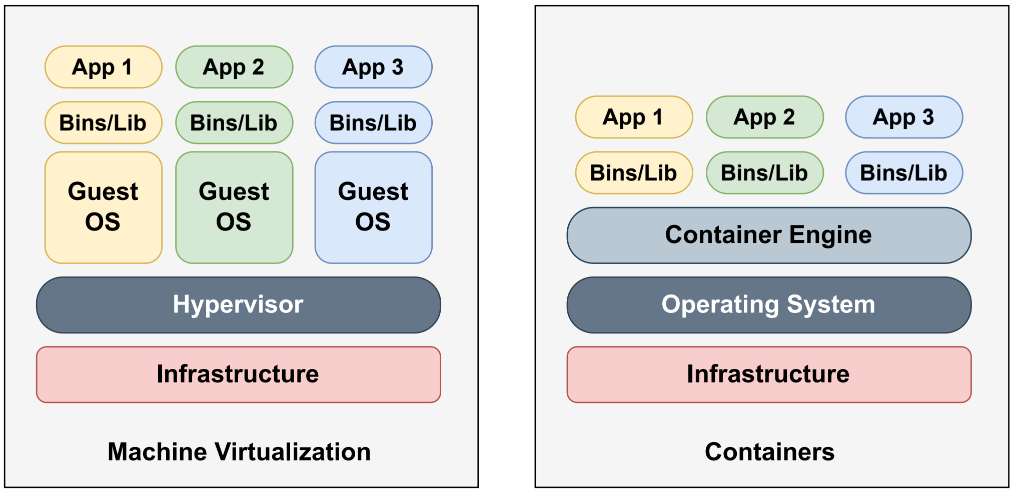


Figure 2.5: Containers vs Virtualization based on Virtual Machines [2].

company employee; therefore, most of the time, they are available only through access credentials. Container image repository implementations include Red Hat's Harbor and Quay.

Containers enable the creation of a new application development paradigm based on microservices. This paradigm enables splitting application components into many relatively small services that, unlike paradigms with monolithic approaches, address the architecture and organization of applications by dividing the whole implementation into small, well-defined, and concise units according to the functions they perform in the application. The microservices communication is performed through well-defined Application Programming Interface (API).

2.5 Quality of Experience

Multimedia content delivery over best-effort networks faces numerous challenges due to fluctuating network conditions. In non-adaptive streaming, videos are typically streamed at a single bitrate from the server. As network conditions degrade, the download rate may fall below the playback rate, leading to buffer depletion and playback interruptions. In contrast, HAS mitigates these issues by encoding video segments into multiple bitrate levels and dynamically selecting the appropriate segment based on current network conditions and the playout buffer level. This approach not only reduces buffering but also optimizes bandwidth utilization, which can play a key role in determining the viewer QoE.

QoE reflects key metrics for assessing the user's overall satisfaction with the delivered content. Unlike purely network-centric metrics, QoE considers the end-user's experience, bridging technical performance indicators with perceptual outcomes. It is shaped by multiple factors such as bitrate adaptation schemes (also known as ABR) and the interaction between application-level control loops and lower-layer mechanisms. Various factors can influence the viewer's perception [41], and we discuss some of these influencing factors.

- **Perceptual Quality:** A higher video bitrate typically provides better video quality.

In HAS, users dynamically adjust video quality during a session, introducing an additional factor influencing the user's QoE.

- **Switching Quality:** As noted in [42], the time spent on each video representation significantly impacts perceived quality. The switching amplitude and frequency between bitrates determines the variation in perceptual quality during a session. Switching amplitude reflects the magnitude of changes in video quality while switching frequency represents the number of bitrate variations throughout the session.

- **Initial Delay:** Every video transmission requires initial data before decoding and playback. This data transfer before video playback is called the initial delay, also known as initial buffering. During this phase, the client does not necessarily need to fill its buffer with a large amount of data, as doing so could result in a prolonged initial delay. However, maintaining a higher buffer level can help the client efficiently avoid buffer underflows during playback.

- **Interruptions:** These are playback freezes caused by the depletion of the buffer. Specifically, a buffer underflow is followed by a buffering period, during which the client must quickly replenish a sufficient amount of video data to resume playback. In some streaming services, rebuffering events may be handled similarly to initial buffering. However, the impact of these two delays differs significantly for users. Unlike initial delay, which occurs before the service begins and is generally anticipated, interruptions happen unexpectedly during playback, making their perception far more negative for users.

QoE assessment models have been proposed in both industry and academia, leveraging objective metrics such as Peak Signal-to-Noise Ratio (PSNR) [43], Structural SIMilarity (SSIM) [44], Video Multimethod Assessment Fusion (VMAF) [45], Mean Opinion Score (MOS) [46], among others [47]. In addition, video quality is also frequently evaluated indirectly through bitrate levels [48, 49]. While these metrics provide valuable insights, they often emphasize different aspects of video quality. For the sake of simplicity, we use logarithmic law over bitrates levels for segment quality measurement, where a video quality model for DASH is proposed as shown in Equations 2.1-2.2. This approach emphasizes a multi-dimensional QoE optimization strategy that balances short-term (e.g., per-segment QoE) and long-term (e.g., session-level QoE trends) objectives.

The short-term model (Equation 2.1) calculates a video quality chunk based on video resolution quality. Each video has N segments and is encoded with L bitrate levels. r_i represents a specific bitrate level. At each step i, the quality of segment i is defined as:

$$q(r_i) = a_1 * log(a_2 * (r_i / r_{|L|})) \tag{2.1}$$

The long-term model considers Equation 2.2 from [48], which consists of four metrics: (a) the average chunk perceptual quality, (b) the average number of quality oscillations, (c) the average number of stall events and their duration, and (d) the startup delay. K

represents the total segments of the video, S_i is the stall duration, and ST_i is the startup delay of user i.

$$QoE_i = \frac{1}{K}\sum_{k=1}^{K} q(r_k) - \frac{1}{K-1}\sum_{k=1}^{K-1} |q(r_{k+1}) - q(r_k)| - \frac{1}{K}\sum_{k=1}^{K} S_k - ST_i \tag{2.2}$$

The QoE_i for each user i can range from 1 to 5, where 1 = bad, 2 = poor, 3 = fair, 4 = good, and 5 = excellent.

2.6 Chapter Conclusions

This chapter examined the mechanisms and technologies that enable adaptive video transmission across network infrastructures. By exploring Cloud and Edge computing, HAS, and CSS, it addressed critical challenges such as latency, scalability, and user mobility. Cloud computing provides centralized scalability, while Edge computing reduces latency by processing data closer to users, enabling responsive, high-quality video delivery. These complementary paradigms create a robust foundation for supporting diverse applications, from VoD to live streaming and immersive media.

The discussion highlighted the importance of HAS, particularly solutions like DASH and HLS, which dynamically adjust video quality based on real-time network conditions. Using ABR algorithms, HAS ensures seamless playback, enhancing QoE by minimizing interruptions, reducing delays, and stabilizing video quality. Additionally, CSS demonstrated how intelligent routing between Edge and Cloud nodes can optimize scalability, responsiveness, and cost efficiency. This approach is crucial for handling dynamic user demands and delivering content reliably under fluctuating network conditions.

Containerization emerged as a key technology for efficient resource management, offering flexibility and scalability through lightweight, modular microservices architectures. The concepts discussed in this chapter—spanning adaptive video streaming, Edge-Cloud integration, and QoE optimization—form a solid foundation for developing innovative solutions. Subsequent chapters will build on these insights, focusing on intelligent resource allocation and real-world implementations to address the challenges of modern video streaming environments.

Chapter 3

Related Work

This chapter introduces the main related works in the scope of the present thesis. We focus on more closely related studies with similar goals. Section 3.1 discusses how Edge-Cloud Networks are organized hierarchically through HAS video streaming systems. Section 3.2 details the work using microservices that address aspects commonly used to create the video streaming architecture. At last, Section 3.3 concludes the chapter.

Tables 3.1 and 3.2, located in Sections 3.1 and 3.2, respectively, summarize the key characteristics of the studies reviewed in this chapter. The column “Proactive” informs whether the proposal takes proactive measures to handle resource management allocation. The “Method” specifies whether the focus is on server-side or client-side solutions, and the “Architecture” highlights the system design, such as hierarchical edge-cloud, cloud-centric, or edge-centric frameworks. Additionally, the “Scaling” column evaluates whether dynamic adjustments to container deployments are considered, and the “QoE” column indicates the use of QoE to optimize the resource allocation. The “Switching” column assesses if the studies explore the impact of switching between video sources or protocols inside the user session. The “Adaptation” refers to the system’s ability to dynamically adjust its operations, resources, or configurations in response to changing conditions such as network fluctuations, user behavior, or resource availability. If a container reaches its load capacity, new instances of that service can rapidly be deployed to the accompanying cluster to help relieve pressure. The services are multi-tenant and stateless with end-users spread across multiple regions. Finally, the “Evaluation” column outlines the methods used to validate the solutions, such as simulations, emulated or testbed trials.

3.1 Edge-Cloud Network-Based HAS Systems

In this section, we examine previous research in video streaming at the hierarchical architecture, specifically focusing on the models of orchestrators capable of effectively managing resources. However, we analyzed these works from the perspective of adaptive video streaming scenarios. Serving content from the edge offers higher application-perceived throughput when the latency savings in receiving the first byte are significant when the CDN transfer is limited by the throughput from higher levels of the CDN hierarchy (or origin) to the edge [30,50,51]. Further, for higher bitrate video, the differences in throughput may be significant enough to impact the video streaming experience. In general, the network hierarchy in video streaming is at the edge, specifically focusing on the models of orchestrators capable of managing services.

For instance, Farahani *et al.* [51] propose the use of an optimization model that employs a service approach with auxiliary caches in their study. The client's requests are served by reverse proxy services placed at the edge and considered the shortest fetch time to attend to the users. Large-scale trials verify the accuracy of the proposed method, and the framework's performance is compared to client-server approaches. It aims to minimize the number of requests forwarded to the origin server. However, the work lacks support for QoE and does not address the potential impacts on service deployment strategies.

The distribution of multimedia services in a hierarchical edge/cloud network also poses significant challenges. Santos *et al.* [52] solve this problem by formulating the service allocation problem as an Integer Linear Programming (ILP), to reduce the number of network nodes while keeping latency in mind. The proposed approach strategically selects the node closest to the user to meet their demands and enhance the overall user experience. Santos *et al.* [53] present an orchestration mechanism called Fog4Video designed to select the edge nodes for downloading video content. The mechanism utilizes user feedback to evaluate video streaming from edge nodes. The work divides the edge into three layers to ensure storage capacity, which improves the average bit rate and significantly reduces monetary costs. Dealing with resource management is an essential requirement in a well-built orchestration. However, the solutions do not account for the operational impacts of deployment scenarios, nor do they address the redirection mechanisms used in their frameworks.

Guan *et al.* [54] demonstrate the performance of the tree-structure model and propose an algorithm with a hit rate improvement in the multimedia content and considerable memory consumption save. The algorithm redirects a user's request for a video to the nearest cache. If the cache node does not receive the video, the edge node forwards it to the upper tier, which forwards the request to the upper tier until it reaches the source. However, suppose any node in the cache network stores the requested video. In that case, it returns the video immediately without forwarding the request to the upper tiers, ensuring an efficient and fast video delivery system. While the work demonstrates effectiveness in static or controlled environments, the work lacks a thorough discussion of its applicability and performance in more dynamic and unpredictable edge scenarios.

In the context of Adaptive Video Streamings and with a focus on edge network, Armijo *et al.* [55] introduced SPACE, a system anticipating the segments requested using Prefetching strategies and Caching at the Edge. Leveraging ML techniques for segment prefetching, SPACE demonstrated improvements in average bitrate and stall reduction compared to baseline strategies. However, real-world testing and evaluation are still necessary to validate the proposed policies and their performance metrics. Additionally, the orchestration between services and users remains unexplored.

Guowaei *et al.* [13] explore the use of multiple CDNs in video distribution and proposes a data-driven user mapping strategy for IPTV services. While the results show a significant QoE improvement in multi-CDN environments, the assumption that all video content is replicated across CDNs is impractical for edge computing scenarios with limited storage. Furthermore, the study does not account for the dynamic nature of edge nodes needed in the Edge-Cloud Continuum.

Taha *et al.* [14] introduce hybrid protocols tailored for adaptive video streaming over edge networks. Their method emphasizes the selection of protocols based on network topology and performance metrics, offering performance improvements in edge environments. However, the protocol selection process is inherently complex, posing significant challenges for real-time implementation in distributed edge scenarios, particularly under dynamic and variable network conditions.

Table 3.1: Comparative Analysis of Related Works.

Work	Arch	Cont	Scaling	Proact	QoE	Switch	Adapt	Eval
[51]	H	✔	✗	✗	✗	✔	✔	Testbed
[53]	H	✗	✗	✗	✔	✗	✗	Simulation
[52, 56]	H	✗	✗	✔	✗	✗	✗	Simulation
[54]	E	✔	✗	✔	✗	✗	✗	Simulation
[55]	H	✗	✗	✔	✔	✗	✗	Testbed
[13]	E	✔	✗	✗	✔	✔	✗	Testbed
[14]	E	✔	✔	✗	✔	✗	✔	Simulation
This thesis	H	✔	✔	✔	✔	✔	✔	Testbed / Emulated

Column abbreviations: Arch = Architecture, Cont = Container, Proact = Proactive, QoE = Quality of Experience, Switch = Switching, Adapt = Adaptation, Eval = Evaluation.
H = Hierarchical; E = Edge; ✔ = Implemented; ✗ = Not implemented; ● = Partially implemented.

In synthesizing the findings from the surveyed literature, it becomes clear that, although progress has been made in executing video streaming services across the edge–cloud continuum, existing solutions still exhibit fundamental flaws in operating video platforms during streaming sessions without suitable node switching. In addition, while some systems have developed QoE-aware scenarios, few go further to enable support for services scaling within the environment. Runtime adaptability is often limited by monolithic design, lack of session-awareness, or lack of a complete orchestration layer that can monitor use, make predictions, and execute decisions in a feedback loop.

3.2 Service Allocation Management

Based on management insights of this domain's current state of knowledge, Carvalho *et al.* [18] and Jeon *et al.* [57] explore ML techniques for container-based strategies to enhance resource utilization in multi-cloud environments. Carvalho's approach introduces a QoE-driven container scheduling mechanism, using ML to infer containers' overall QoE based on their metrics. On the other hand, Jeon's solution focuses on workload forecasting for generic services, enabling proactive container scheduling based on predicted demand. While both solutions improve resource management in cloud environments, they do not consider the dynamics of edge-network interactions and the impact of edge nodes on the end-to-end video delivery experience.

Bentaleb *et al.* [58] proposed a method based on a deep neural network to make predictions about the media that will be used. This method allows the edge server to anticipate demand for cloud-based video segments and make requests in advance. The experimental findings demonstrate a superior performance of the proposed solution over the baseline models within a millisecond time frame. However, their evaluation is based on trace-driven streaming data, which may not account for all possible real-world conditions, such as varying network conditions, user behaviors, and content types.

Shi *et al.* [59] discuss the problem of edge node selection. In order to solve this problem, a proposed edge node selection strategy considers the edge cache's status. Further, the authors proposed a QoE-aware method to optimize QoE directly. Thus, the proposed policies are executed based on users' QoE to effectively utilize the cached content on the server Base Station (BS) as much as possible. Nguyen *et al.* [60] proposed an HTTP-based relay mechanism where the throughput perceived by users is computed while the user receives the segments. This allows the server segment re-transmission during network fluctuations as long as better-quality segments are received. Additionally, the authors utilize the push property of HTTP/2 to retrieve multiple segments sending a single request. The mechanism has reduced the low-quality video time while also slightly improving QoE. However, both works rely on specific, tailored solutions, which introduce complexity when integrating them into other video streaming systems.

Yinxin *et al.* [61] proposed an edge caching scheme that caches the most popular representations and the metadata of less popular segments. This scheme can dynamically determine whether to cache the video chunk with its representations or metadata. Experimental results show the scheme's effectiveness in utilizing idle computing resources at the edge to respond to the most video requests of the most the users. However, the work does not address resource management aspects between edge and cloud servers or the integration of CDN provisioning strategies.

Chen *et al.* [62] presented a three-layer mobile edge network architecture, introducing an edge caching strategy based on user mobility speed and content popularity. The experimental results showcased the superior performance of the caching strategy in terms of average delay and cache hit ratio compared to other classic methods. However, the mechanism for information acquisition is not thoroughly addressed, which could hinder the system's ability to adapt dynamically to changing user demands and network conditions. Furthermore, This limitation suggests that the solution is not tailored to the edge in CDN environments.

Chen *et al.* [63] investigate the multiple bitrate video caching, considering the interplay between processing delay and backhaul transmission. A dynamic programming algorithm and a multiple-bitrate caching algorithm were proposed to select the most appropriate bitrate versions within constrained cache sizes. The numerical results show the effectiveness of the proposed algorithm on content placement. However, the scenarios' deployment operations and redirection mechanisms are not addressed, which are essential for seamless integration and practical implementation in real-world systems.

In the work by Xu *et al.* [64], a joint optimization approach was proposed, along with its decomposition and consideration of system limitations. The selection of the first Closest server as a steering algorithm to attend to users was a notable feature. Meanwhile, Liu *et al.* [65] tackled challenges in minimizing latency for request assignment and resource allocation. Both proposals consider the request load assignment and resource allocation for CDN nodes, showcasing improved performance in total latency and request loads. However, the redirection mechanisms in these works lack real-time adaptability and do not adequately consider users' QoE conditions.

Table 3.2: Comparative Analysis of Related Works.

Work	**Arch**	**Cont**	**Scaling**	**Proact**	**QoE**	**Switch**	**Adapt**	**Eval**
[66]	C	✗	✗	✗	✗	✔	✗	Testbed
[67]	H	✔	✗	✗	✗	✗	✗	Simulation
[59]	E	✗	✗	✗	✔	✗	✗	Simulation
[60]	C	✗	✗	✗	✔	✗	✗	Testbed
[55]	H	✗	✗	✔	✔	✗	✗	Simulation
[61]	E	✔	✗	✔	✗	✗	✔	Testbed
[62]	H	✗	✗	✔	✗	✔	●	Testbed
[63]	H	✗	✔	✗	✗	✔	✗	Simulation
[64]	C	✗	✗	✗	✗	✔	✗	Simulation
[65]	E	✔	✗	✔	✗	✗	✗	Simulation
[18]	C	✔	✔	✗	✔	✗	✔	Testbed
[57]	C	✔	✔	✔	✗	✗	●	Simulation
This thesis	H	✔	✔	✔	✔	✔	✔	Testbed / Emulated

Column abbreviations: Arch = Architecture; Cont = Container; Proact = Proactive; QoE = Quality of Experience; Switch = Switching; Adapt = Adaptation; Eval = Evaluation; H = Hierarchical; E = Edge; ✔ = Implemented; ✗ = Not implemented; ● = Partially implemented.

The main characteristics of reviewed studies in Table 3.2 indicate some common elements in video streaming systems for service allocation. Among the limitations in existing approaches are, for example, insufficient QoE awareness and a lack of real-time adaptability. While the works explore containerization, edge caching, and proactive scheduling, none combine these elements effectively within a unified framework. Only a subset of essential features is considered, whereas the thesis aims to integrate these key aspects.

3.3 Chapter Conclusions

This chapter described the literature solutions for adaptive video streaming within edge-cloud networks. The studies reviewed highlight different methodologies used to optimize video streaming architectures, ranging from orchestration mechanisms and steering strategies to machine learning-driven resource management. Based on the state-of-the-art analysis, some conclusions are taken: *(i)* dealing with a multi-level architecture brings aspects that can help or worsen network performance if not managed correctly. In summary, the previous approaches' efforts to improve users' QoE have focused on reducing traffic load or optimizing multimedia streaming. Thus, these challenges require the composition and development of new service orchestration models; *(ii)* Most of these studies consider resource allocation and latency improvements; they overlook the communication protocol in provisioning Adaptive Video Streaming within the Edge-Cloud Continuum, while others consider static scenarios without the impact of the real-time redirection operations. Our study focuses on implementing an Adaptive Video System to assess the viability of CSS in an Edge-Cloud Continuum setting; *(iii)* Most existing studies overlook the impacts of provisioning adaptive video streaming along the Edge-Cloud Continuum. In contrast, others consider scenarios without accounting for the effects of accurate time redirection in the communication protocol. This thesis addresses this challenge by introducing CSS in the proposed architecture and its impacts on the redirection mechanisms. *(iv)* Another key aspect is the architecture's adaptability to different scenarios within the Edge-Cloud Continuum. Most of the works propose solutions that are optimized for specific conditions but do not generalize for real-world technologies such as HAS communication, observability frameworks and microservices-based deployment strategies. Meeting all the above requirements leverages a dynamic resource allocation, real-time monitoring, and session-aware CSS, ensuring seamless performance across different deployment environments for adaptive video streaming services.

Chapter 4

Adaptive Video Streaming in Multi-tier Edge-Cloud Environment

Adaptive video streaming services are done through the HTTP Servers using the HTTP protocol requests/responses. Original videos are partitioned into segments of equivalent playback time on the server side, and each of these segments has multiple versions that vary in bitrate/resolution/quality [6, 48, 68]. On the user side, the player can request video segments sequentially and adjust the bitrate resolution dynamically to network conditions. The users' QoE is significantly affected by the video quality level segment that the player selects. Considering video streaming platforms working over the network, the final mile of access networks is when QoE deterioration and resource limitations occur for consumers. Nonetheless, edge computing is a hierarchy of neighborhoods in the geographical region where the video content becomes accessible near the end-users. Existing solutions are designed to deliver content so that edge and cloud networks can work together. It alleviates the traffic load from backbone networks to the cloud and, consequently, helps improve users' QoE [30, 32].

Edge computing can accommodate the new demands of video traffic. However, some level of planning is required to leverage edge resources' capabilities. Firstly, the edge node selection problem involves determining the most suitable edge node to host video streaming services for a given user or group of users. For instance, the handover process ensures that users communicate with the nearest AP. However, this process does not inherently ensure the connection is established with the optimal edge node hosting the required video streaming service, potentially resulting in degraded performance. Another important aspect is the need for dynamic scalability in content delivery. CPs must adapt to fluctuations in user demand. When the number of users increases in a specific region, additional edge nodes may need to be provisioned to accommodate the growing load and maintain QoE guarantees. Conversely, when multimedia consumption decreases, the system should consolidate users onto fewer edge nodes, shutting down underutilized nodes to save energy and resources. This scalability demonstrates the importance of a hierarchical and flexible approach to multimedia content delivery [32, 69].

Therefore, content providers or network operators must identify an appropriate edge node to meet users' requirements. This necessity highlights the importance of a decision-making process for edge node selection. Such decisions must be made in real-time to maintain a good QoE and minimize the resource wastage across edge nodes. Moreover, aggregating active users in a minimum number of edge nodes can help balance traffic load [5, 19]. To address these challenges, we introduce the **M**ult**I**-tiered ed**G**e/cloud o**R**chestrator **A**rchitecture for video s**T**r**E**aming (MIGRATE). The proposed architecture dynamically orchestrates services in response to real-time video streaming demands. In addition, we formulate an ILP model that considers the multi-tier edge network. The model aims to maximize infrastructure use without affecting the user's QoE. The main contributions of this chapter are presented below.

1. Introduce MIGRATE orchestrator for video streaming across multiple edge/cloud levels, which takes in consideration the hierarchy of multimedia content transmission. MIGRATE aims to carry out the decision-making of the video streaming service in real time;

2. Proposal of an ILP model and Greedy solution to decide the distribution of connections between users and services;

3. Analysis of the results shows that depending on the optimization strategy used, there is a trade-off between resources and users' QoE.

This chapter is organized as follows: The proposed architecture, MIGRATE, is introduced in Section 4.1, detailing its multi-tier framework, including communication, monitoring, tracking, and optimization components. Section 4.3 presents the experimental evaluation, comparing MIGRATE's performance against baseline approaches in various scenarios. Lastly, chapter 4.4 discusses the insights gained, highlighting the opportunities and challenges of deploying multi-tier Edge-Cloud architectures for adaptive video streaming.

4.1 MIGRATE

The infrastructure supporting edge computing is multi-tiered, as presented in Figure 4.1, consisting of nodes accessible through different levels of network connectivity. These nodes range from specialized servers in the core network to micro-data centers in the radio-access network. The MIGRATE architecture is designed to operate seamlessly in such a multi-tier edge environment, managing multiple video streaming connections across the hierarchy network.

MIGRATE takes advantage of the interactivity between nodes at different tiers to make the best decision within the network. As depicted in Figure 4.1, on the left-hand side, the edge can be divided into various tiers. At the top of this hierarchy are cloud servers, which can be located in either the public or private sector. These clouds can be any video streaming provider or even a local user. The three tiers of the edge network illustrate the hierarchical structure of the edge network. This study examines three edge nodes that provide storage and computation services in order of their communication coverage area. Tier 2 corresponds to the Core Network Regional Edge in this upper multi-level edge, which coordinates throughout the city. The Access Network Edge in

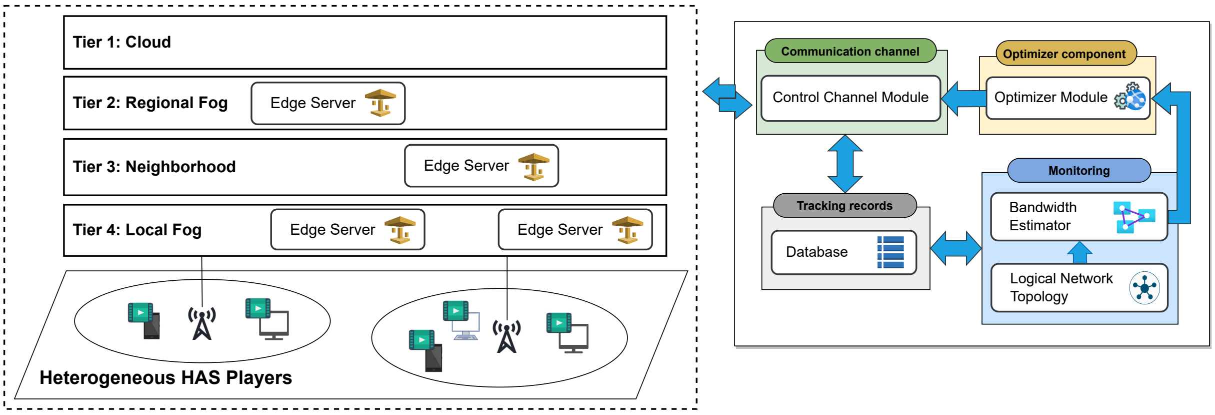


Figure 4.1: A General Overview of the multi-tier network environment on the left side, and the MIGRATE architecture on the right side (Adapted from [3]).

Tier 3 supports a few dozen to a few hundred local nodes on the edge. Tier 4 deploys Edge Gateways on locally hosted edge nodes with limited storage capacity, where the node transmits video data across wired or wireless connections.

Designing a cache hierarchy on edge nodes with an arbitrary number of tiers can present improvements in users' QoE [70]. The architecture mentioned above works toward such advantages by serving the requested content close to the end-user, efficiently forwarding requests between parent edge nodes within the hierarchy, and balancing the video traffic considering hop counts and users attended. In addition, the network core congestion is reduced since it represents an operational overhead for the content provider.

4.1.1 Network and System Model

This section presents an overview of the MIGRATE architecture. The detailed architecture components are then defined, including each component's different functionalities and responsibilities. The orchestrator can monitor the network conditions and allocate resources to ensure video services are deployed on the most appropriate edge nodes. This hierarchical structure enables MIGRATE to make decisions tailored to each level's needs, ensuring efficient resource management and better end-user video streaming quality. The right-hand side of Figure 4.1 shows the MIGRATE software architecture and the necessary interactions for the video streaming components. For the sake of readability, the notations used in this work are summarized in Table 4.1.

Communication Component

The main Communication Component feature works as a control channel for communication between network entities. All inter-entity communication is handled through a channel created by the main class. In this work, the communication functions are offered to redirect the user to an edge cache. Afterward, the optimization component is executed, which involves sending messages through the communication protocol to the users. After receiving a server change notice, users can only communicate with a server that actively

Table 4.1: Notation used in the proposed model

Notation	**Description**
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Non-directed graph representing the network topology
$\mathcal{V}$	Set of vertices (nodes) in the network
$\mathcal{V}_n$	Subset of nodes capable of provisioning and serving content (backhaul infrastructure)
$\mathcal{V}_u$	Set of end-user locations (viewers)
$\mathcal{E}$	Set of edges (links) in the network
$e = (v_x, v_y)$	Edge between nodes v_x and v_y
ω_e	Bandwidth capacity of link e
ψ	Network state
t	Time
$\omega_e(\psi, t)$	Estimated link transfer rate at time t under network state ψ
$\omega_e^T(\psi, t)$	Total available bandwidth capacity of link e at time t under network state ψ
u_k	User k
G	Set of user groups
g_i	User group i
$\lambda : U \mapsto \{LAC, id\}$	Function mapping a user to their real-time location (LAC and cell ID)
LAC	Location Area Code
id	Cell identifier
$\mathcal{JAF}(u_k, \lambda(u_k)) = g_i$	Joint aggregation function mapping user u_k to group g_i based on location
$\mathcal{F}_1$	Subset of nodes below a congested link
$\mathcal{F}_2$	Subset of user groups connected to nodes in $\mathcal{F}_1$
$\mathcal{NUM}(v_j)$	Number of users traversing node v_j
p_j	Importance degree of node v_j to be chosen
$\chi(\psi, v_j)$	Function quantifying the node v_j under network state ψ
$\mathcal{R}_{ij}$	Optimal route between user group g_i and server v_j
$\mathcal{DEM}(g_i, e)$	Bandwidth demand for user group g_i on link e
s_i	Arbitrary multimedia content created by a video server
S_i^t	Video streaming s_i starting at time t_i ($\mathcal{S}_i^T = \{s_i^{t_1}, s_i^{t_2}, ..., s_i^{t_n}\}$)
$\mathcal{CAP}(S_i^T, v_j)$	Cache required for video stream S_i^T on server v_j
$\mathcal{M}_{v_j}$	Cache capacity of edge server v_j
x_{ij}	Binary variable indicating if user group g_i is assigned to server v_j

accepts them. In doing that, this switch server message must be sent so users can initiate requests on an already available server.

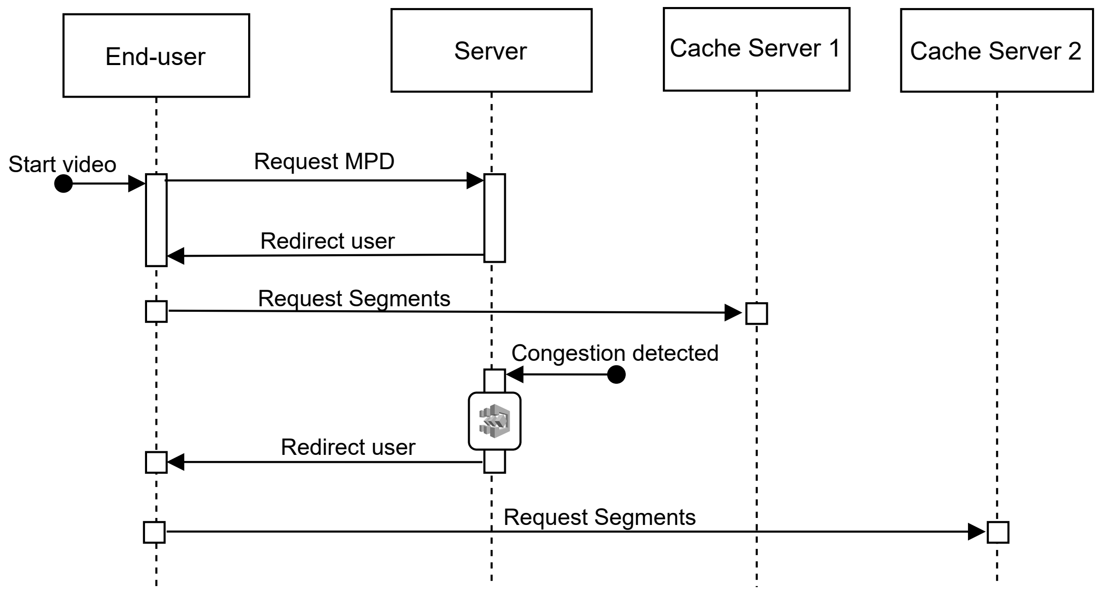


Figure 4.2: Sequence Diagram of the video streaming communication.

A communication flowchart is depicted in Figure 4.2. The client initiates the process by requesting content from a centralized node. The server, in turn, provides the client with an MPD. Using the manifest file, the client begins requesting video segments. Initially, these requests are directed to a primary server. The system actively monitors user feedback and network conditions as the communication progresses. For example, congestion detected in the current communication path can trigger a redirection of the service to an edge cache.

Once redirected, the client requests and receives video segments from the newly assigned edge cache. Further feedback from the client, such as degraded QoE or network performance, may lead to subsequent redirections, either to another edge cache within the same tier or back to a cloud-based server, depending on the system's assessment of resource availability and network conditions. This dynamic and adaptive communication ensures efficient video delivery across different network tiers.

4.2 Monitoring Component

The monitoring component is central to the overall system, continuously observing predefined network metrics that are essential for the MIGRATE orchestration. This component enables real-time awareness of network conditions, ensuring that orchestration decisions, such as the deployment of new edge nodes, are made. In this study, *bandwidth consumption* is considered a key metric [66]. The network topology is modeled as a non-directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of vertices is defined as $\mathcal{V} = \mathcal{V}_n \cup \mathcal{V}_u$. The subset $\mathcal{V}_n$ consists of nodes capable of provisioning and serving content within the backhaul infrastructure of a mobile network operator. In contrast, $\mathcal{V}_u$ represents the set of end-user locations (viewers). The connectivity between nodes is determined by the set of edges $\mathcal{E}$, where exists a link between nodes $v_x, v_y \in \mathcal{V}_n$ such that $x \neq y$ if an edge $e = (v_x, v_y) \in \mathcal{E}$.

Each network link $e \in \mathcal{E}$ is associated with a bandwidth capacity ω_e. The bandwidth estimation process is based on the volume of packets transmitted and received through network interfaces over a specific time period t and under a given network state ψ. The estimated link transfer rate at any time t is denoted as:

$$\omega_e(\psi, t).$$

The total available bandwidth capacity of link e is represented as:

$$\omega_e^T(\psi, t).$$

To ensure network stability and efficiency, the estimated bandwidth consumption must not exceed the total capacity of the link at any given instant t. This constraint is formalized as:

$$\omega_e(\psi, t) \leq \omega_e^T(\psi, t), \quad \forall e \in \mathcal{E}, \quad \forall t. \tag{4.1}$$

Maintaining this constraint is critical for preventing network congestion and ensuring a good QoE for end-users. The system can dynamically adjust resource allocation strategies by enforcing an upper bound on bandwidth utilization, mitigating performance degradation and optimizing traffic distribution across the edge-cloud continuum.

Tracking Component

Most individuals frequently carry their mobile phones with them in daily life. Mobile communication data includes all users' locations. Radio access and application-level data collected from the base stations may be utilized for tracking purposes and supporting the video streaming infrastructure.

The tracking component registers the user's location, generating information on which the system can identify the best network management decision. When a streaming session starts, the Control Channel component first classifies the user into one of the groups based on characteristics extracted from their manifest and associated AP information. These operations can be done with the help of radio access-level data, retrieved from base stations and application-level information. Providing finer-grained insights into user mobility within the streaming ecosystem.

Formally, each user $u_k \in V_u$ is associated with a designated user group. The set of user groups where each $g_i \in G$ represents a distinct group. A user's real-time location can be determined using the Location Area Code (LAC) and cell identifier (id). The function λ maps each user to their real-time location:

$$\lambda : U \mapsto \{LAC, id\}$$

When a new user joins the video streaming system, the insertion algorithm adds the user to a group. We assume that the network data, which includes each user's cell ID connection, can be used to detect an association between an AP and a user, where $\lambda(u_k) = \{LAC_{u_k}, id_{u_k}\}$ represents the exact network point of connection for user u_k.

This work employs a primary *joint aggregation function*, which is defined as a function that combines multiple users from the same group into a single service node. The aggregation process considers LAC and cell id as shared characteristics among users, to simplify the system's optimization and decision-making processes. The joint aggregation function of users to groups is governed by the function $\mathcal{JAF}$, which dynamically maps a user u_k to a group $g_j \in G$ based on their current network position:

$$\mathcal{JAF}(u_k, \lambda(u_k)) = g_i$$

Each set of users belonging to the same group is then aggregated into a single node service. In this way, when a user requests video content, the multimedia server is able to route the user to a particular video provider rapidly. If the viewer is currently watching the video, a redirect message updates the server instead. Thus, the joint aggregation function aids in simplifying the optimization component's computing process.

Optimization Component

The Optimization Component, responsible for making decisions regarding service placement and scalability, is a crucial part of the proposed architecture. This component employs optimization algorithms that consider factors such as the network's current state, the available resources, and the current services' workloads. The system must respond promptly as demand fluctuates to ensure the network remains balanced and efficient.

An ILP model for the optimization component is introduced to demonstrate the flexibility of the proposed architecture. Here, the model decides the number of servers capable of attending to the video demand and distributing users' connections among the edge server points. This model aims to maximize the use of the existing server in order to satisfy the users' QoE following an objective function.

Edge Node Selection and Service Placement

When the Monitoring Component detects a congested link, the Optimization Component is triggered. The edge node selection process begins by extracting relevant data from the Monitoring. The first entry is a subset of nodes defined as $\mathcal{F}_1 \subset \mathcal{V}_1$, where each node $v_i \in \mathcal{F}_1$ below the congested link $e \in \mathcal{E}$. In doing so, a subset of user groups connected to these nodes is also selected, represented as $\mathcal{F}_2 \subset G$. Each selected group g_i is associated with an AP $v_j \in \mathcal{F}_1$ of the network topology.

In a hierarchical model, the upper tiers within the network edge tend to serve more users. To maximize the number of users served by the same node, we used the function defined in Eq. (4.2) that quantifies the importance of selecting each node to deploy the video streaming service, prioritizing nodes that serve the highest number of users. $\mathcal{NUM}(.)$ is a function that returns the total number of users who traverse node v_j, and p_j is defined as:

$$p_j = \frac{\mathcal{NUM}(v_j)}{\sum_{v_x \in \mathcal{F}_1} \mathcal{NUM}(v_x)}, \text{where } v_j \in \mathcal{F}_1$$

Using this formulation, the best candidate nodes for service deployment are identified based on the importance p_j, ensuring that services are allocated to nodes with higher user densities. The same node will serve the maximum number of users by deploying services on nodes with a higher p_j value. The Eq. 4.2 also seeks to limit the number of users through an upper limit. Thus, we can manage the limited resources available at the edge:

$$\chi(\psi, v_j) = \mathcal{NUM}(v_j) * p_j \tag{4.2}$$

Bandwidth and Capacity Constraints

To ensure that the allocated services remain within feasible bandwidth constraints, the following condition must be satisfied for any selected node v_j:

$$\sum_{g_i \in \mathcal{F}_1} \mathcal{DEM}(g_i, e) \leq \omega_e^T(\psi, t), \quad \forall e \in \mathcal{R}_{ij}. \tag{4.3}$$

To support the video services demanded by users, an optimal route $\mathcal{R}_{ij}$ must be chosen between user group g_i and server v_j. $\mathcal{DEM}(.)$ represents the bandwidth demand for g_i, with the indicator function, to effectively make the demand zero when g_i does not pass through link e. While $\omega_e^T(\psi, t)$ denotes the total available bandwidth for link e.

With users accessing the video segments sequentially, the demand for the video content segments in the next time window can be approximately estimated, given the current viewers. This allows efficient management in allocating cache resources to ensure that edge nodes are guaranteed continuous storage. Let s_i be arbitrary multimedia content created by a video server. The video streaming s_i starting at t_i as S_i^t (that is, $\mathcal{S}_i^T = \{s_i^{t_1}, s_i^{t_2}, ..., s_i^{t_n}\}$, where $(t_1, t_2, ..., t_n)$ are timestamps of videos $(s_1, s_2, ..., s_n)$ stored from t_i. The function $\mathcal{CAP}(.)$ calculates the cache of a given video stream, and the cache capacity of the edge server v_j, denoted by $\mathcal{M}_{v_j}$, is given by Eq.(4.4).

$$\sum_{g_i \in \mathcal{F}_2} x_{ij} \cdot \mathcal{CAP}(S_i^T, v_j) \leq \mathcal{M}_{v_j}, \quad \forall v_j \in \mathcal{F}_1. \tag{4.4}$$

The binary variable x_{ij} determines whether a user group g_i is assigned to server v_j:

$$x_{ij} = \begin{cases} 1, & \text{if } g_i \text{ is assigned to server } v_j, \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

Optimization Formulation

It is necessary to select candidates with the highest p_j and who comply with the limits. The proposed problem can be formulated as follows: To find the optimal set of candidates that maximizes the p_j value while satisfying the necessary constraints. This can be achieved by maximizing the objective function in Eq 4.6.

$$\begin{aligned} \max \quad & \sum_{g_i \in \mathcal{F}_2} \sum_{v_j \in \mathcal{F}_1} \chi(\psi, v_j) * x_{ij} \\ \text{Subject to} \quad & (4.1), (4.3), \text{ and } (4.4). \end{aligned} \tag{4.6}$$

The proposed strategy considers the dynamic nature of network resources and adjusts in real-time to changing network circumstances, making it a resilient solution for managing multimedia services in edge-based networks. Once the ILP problem is solved, we receive the nodes on which the video streaming service should be deployed, as well as the connections between nodes $\mathcal{F}_1$ and $\mathcal{F}_2$. Then, the following two procedures are carried out: First, the streaming service is deployed on the available node, and then the communication component updates the active user-service connections.

4.3 Performance Evaluation

Designing a cache hierarchy with an arbitrary number of tiers can present improvements in users' QoE [70]. The architecture mentioned above works toward such advantages by serving the requested content close to the end-user, efficiently forwarding requests between parent edge nodes within the hierarchy, and balancing the video traffic considering hop counts and users attended. In addition, the network core congestion is reduced since it represents an operational overhead for the content provider. First, Section 4.3.1 presents a preliminary outcome achieved by a multi-tier network experiment, assessing the impact of a video streaming service from the player's perspective. After that, Section 4.3.2 analyzes some scenarios without the MIGRATE, where the edge utilization yields similar QoE performance to cloud-based streaming. Finally, Section 4.3.3 shows MIGRATE's performance and insights on the opportunities for caching multimedia content in edge nodes of multi-tier networks.

Experimental Setup and Parameters

We use Adaptive Multimedia Streaming (AMuSt) [71] to implement a video streaming server, and the server is implemented through DASH, with users that allow adaptive video streaming. The AMuSt framework provides a set of applications for producing and consuming adaptable videos based on the DASH standard. DASH functionality is provided by the libdash library [72], an open-source library that provides an interface to the DASH standard. Currently, libdash is the official reference software for the DASH standard. We consider that users are interested in an available video with ten different bit rate representations {235kbps, 375kbps, 560kbps, 560kbps, 750kbps, 1050kbps, 1750kbps, 2350kbps, 3000kbps, 4300kbps, 5800kbps}, which are used by Netflix subsets [73]. Each representation is divided into 2-second segments. All the experiments are executed once with a video of 1600 seconds (800 segments). For simplicity, the multimedia content used in the simulation is deployed beforehand in the edge peering nodes. The Zipf distribution gives the video selection and quality with α equal to 0.7, which ensures that the preference for content is better distributed.

Figure 4.3 illustrates the scenario discussed in this work. We consider a cache hierarchy organized as a binary tree topology with seven nodes and a Cloud Provider connected to the root node. The last four nodes are APs, and the others are edge nodes. The AP nodes are implemented in wireless devices that communicate via IEEE 802.11g at 2.4 GHz. The APs have wired connections to the edge nodes, while end users are connected wirelessly.

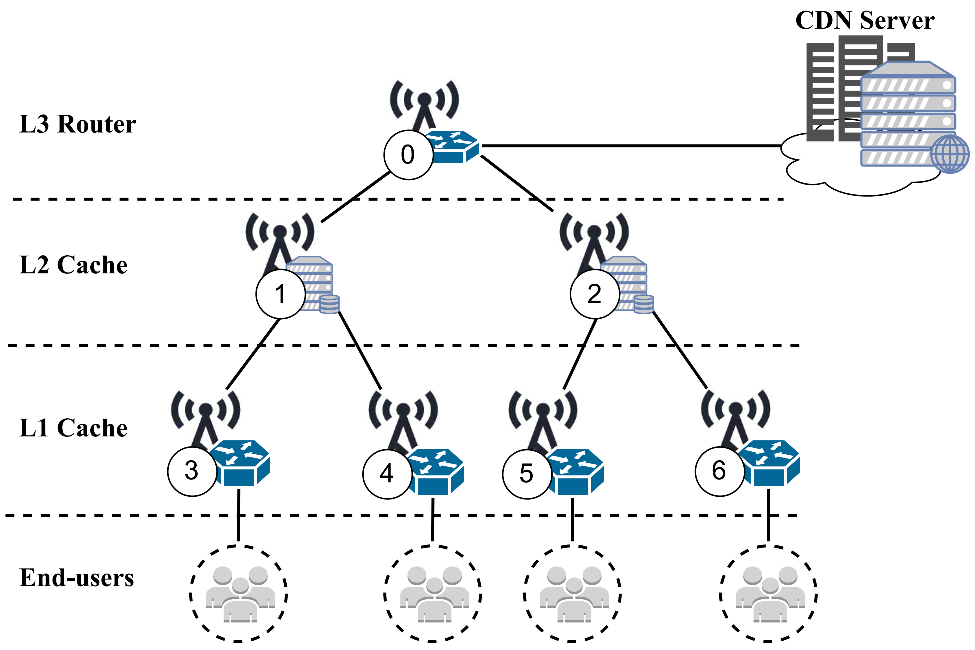


Figure 4.3: Overview of the multi-tier network environment.

Each user connected to the AP is located precisely 8 meters away from the AP. To force the use of edge nodes, the available bandwidth is 20Mbs on links 0-1 and 0-2 and 30Mbs on links 1-3, 1-4, 2-5, and 2-6. This configuration allows us to verify the spread of video streaming services at the edge. The bandwidth in $e_{0,1}$ and $e_{0,2}$ are smaller than in the lower links. In this way, the evaluated solutions can be used to allocate services more efficiently in different conditions.

4.3.1 Impact of a Multi-tier Approach

To illustrate the differences in users' performance requesting a video from cache nodes in different tiers, consider two users requesting the same multimedia content from different layers. Then, an analysis of the impacts over the network is performed on the application context in terms of player metrics. Figure 4.4 illustrates a two-tier network model. The graphs show the results of bitrate, interruptions (stalls), video buffers, and representations switch, respectively, from left to right.

Although a user requests a cache edge node video, it can be located in layers L1 or L2. Each layer level has different network factors, e.g., load, latency, etc. In Figure 4.1, an L1 layer can be interpreted as a personal computer, AP, or Base Station, and this layer transmits the video content through wired or wireless communication channels, whereas in the L2 layer, a specific Edge gateway can be distributed on local edge nodes.

Apart from the initial interruption before the start of the video, neither user experienced the same issue again during the video execution. However, the user who received the video from the nearest edge layer had a higher bit rate than the user receiving the video from the upmost layer. Note that the user receiving the video from the closest layer filled the buffer faster, allowing him/her to keep the video playing at the best possible resolution. On the other hand, the user who received the video from a more distant layer

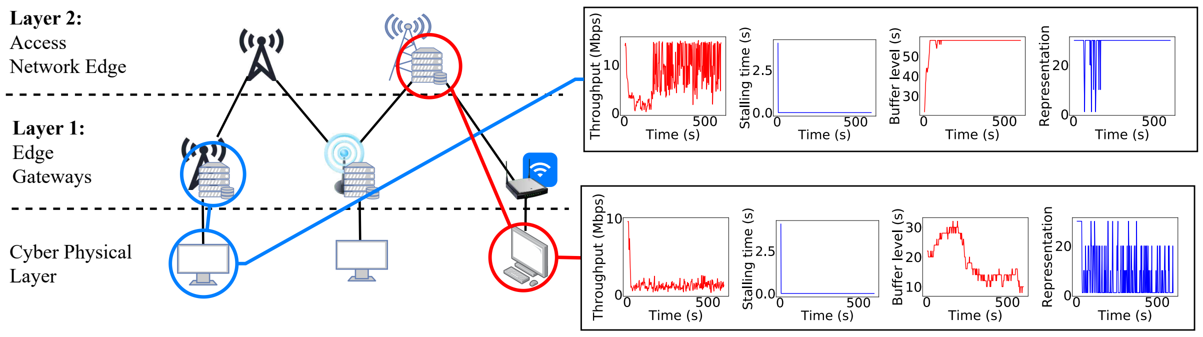


Figure 4.4: Bitrate switches, stalls, buffer size, the startup delay in seconds of a DASH player was requesting a video with ten bitrate levels varying from 50 to 4,500Kbps and from nodes in different tiers.

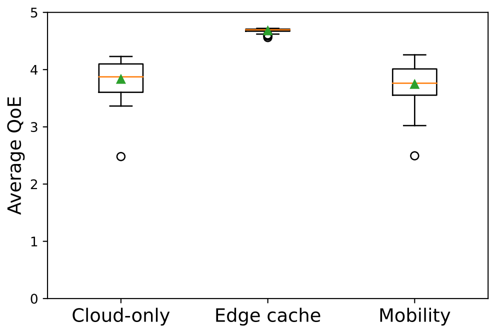
worked with the buffer at the limit and constantly switched resolutions to prevent interruption during the video execution. Despite interruptions in both video playbacks, the transmission from different tiers directly impacts the users' QoE.

4.3.2 Analysis in Multi-tier Edge-Cloud Networks

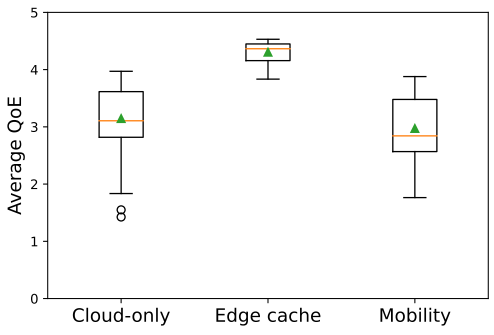
This section analyzes the evaluation of the multi-tier video delivery environment. Three approaches address the impacts identified in the edge/cloud multi-tier network: *cloud-only*, *edge cache*, and *mobile-based* scenarios. The cloud-only scenario uses only the Cloud Provider node to deliver the video content. On the other hand, the edge cache approach uses nodes 1 and 2 as auxiliary nodes to deliver the video. The simulation starts with the users requesting the video from the cloud. When a congested link is detected, the edge cache below the link is turned on. Thereafter, the users receiving the video over the congested link are redirected to the edge nodes 1 and 2.

The number of end-user devices communicating through each wireless AP may change over time due to the mobility of the end-user devices. In practice, the number of end-user devices connecting to the wireless AP changes frequently. To show the problems that can arise in such dynamic load scenarios, a connection change between the APs occurs. We then reran the second experiment, maintaining connections between users and edge nodes 1 and 2, but changing the connection between users and APs, emulating users' mobility. Half the users from each AP move in this mobility scenario. Let i the AP index ($3 \leq i \leq 6$) such that $i < 5$, users from AP_i go directly to AP_{i+2}, otherwise they go to AP_{i-2}.

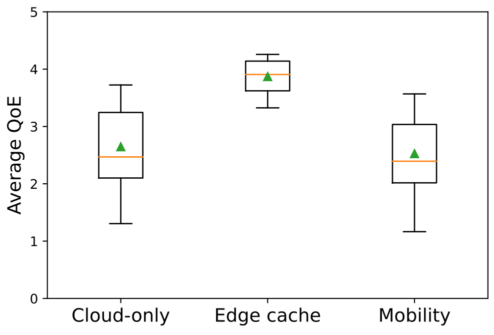
The experiments illustrated in Figures 4.5(a), 4.5(b), and 4.5(c) show the average QoE calculated by Equation 2.2 when there are 15, 20, and 25 users per AP requesting videos in the simulated infrastructure. Each boxplot represents the Cloud-only, Edge cache, and Mobility scenarios. The overall average performance of the Edge cache is better than the Cloud-only and Mobility scenarios, mainly due to the choices of nodes at the edge peering nodes for serving user requests. For instance, in the Edge cache experiment, when congestion occurs in intermediate links $e_{0,1}$ e $e_{0,2}$, the edge peering nodes are activated to serve the end-users below those nodes. This way, the traffic passing through the upward link will be smoothed out, so the users can improve their QoE to an excellent level. Since the users request segments from the closer nodes and with no congestion link in scenarios with Edge cache, as expected, we observe a QoE increase. The performance difference from the Cloud-only and mobility scenarios to the Edge cache experiment is near one level of satisfaction for 15 and 20 users and approximately two satisfaction levels for 25 users.



(a) Average QoE results (15 users per AP).



(b) Average QoE results (20 users per AP).

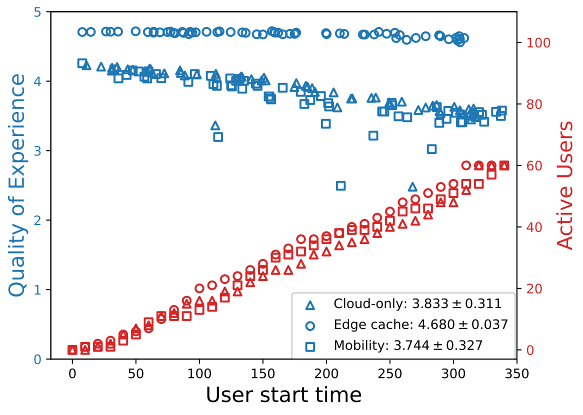


(c) Average QoE results (25 users per AP).

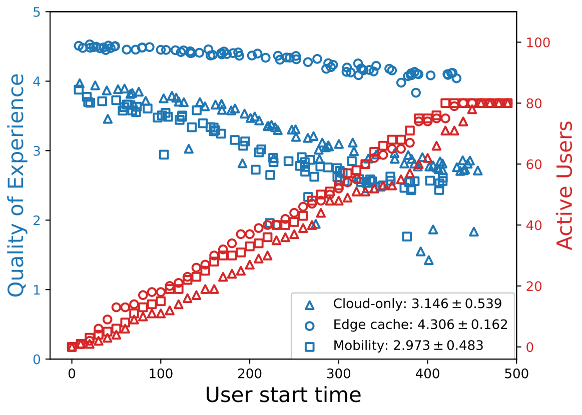
Figure 4.5: Users' QoE average. Each bar group represents a different Planner decision: Full-Edge, Reactive, and Proactive.

Figures 4.6(a), 4.6(b), and 4.6(c) show the final QoE of each user per start time (time to start the segment requests). Each blue tick represents a user, and the y-axis QoE means the final user satisfaction for a fully watched video. Here, we can see the final QoE degradation for each user's entrance during the simulation execution. The red plot represents the number of active users simultaneously. As users reach this experiment's final QoE, it tends to be slightly lower than the previous one. When looking at the final QoE delta between users u_i and u_{i+1}, this seems to be irrelevant, but as we increase the delta, the QoE starts to become considerably different. We can also confirm this behavior by showing that as the number of users increases, the average QoE decreases, and the variation between users increases. Also, note that in the simulation with 25 users per AP, the system already shows a degradation in the quality, which is negatively perceived by the end-user.

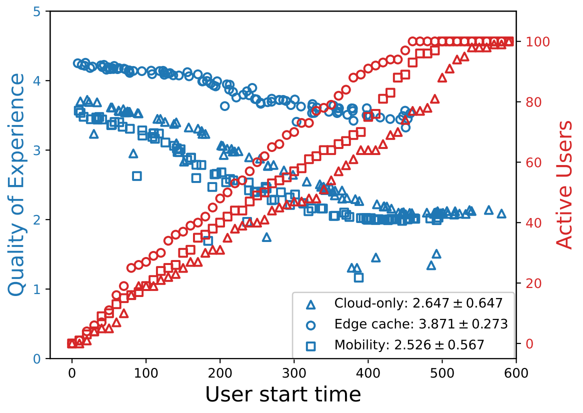
An interesting discussion occurs when the QoE per user is analyzed separately. According to the numerical results, the final QoE worsens as active users increase. However, it is not entirely true for the Edge cache scenario, where the final QoE for each user



(a) Average QoE for each user (15 users per AP).



(b) Average QoE for each user (20 users per AP).



(c) Average QoE for each user (25 users per AP).

Figure 4.6: Average QoE for each user group. Each mark represents the final user QoE for the following approaches: Cloud-only, Edge cache, and Mobility.

remains close. The standard deviation of 0.037, 0.162, and 0.273 for scenarios with 15, 20, and 25 users per AP indicates a closer QoE between the users. Only in the scenario for 25 users per AP, the average QoE dropped with a satisfaction close to regular. This contrasts the other two scenarios, presenting a user's good to excellent satisfaction. The network operates with a high standard deviation for cloud-only and mobility scenarios. As the number of users increases, some outliers start to appear with the worst level of satisfaction, being between poor and inadequate.

Based on these observations, moving the video to the edge can significantly improve the user's QoE. This way, the video transmission system can satisfy users and keep them watching the video until the end. However, suppose there is no correct management of connections in real-time and a dynamic mechanism to tackle a varying load coming, for example, from the mobility of users. In that case, we can conclude that the impact introduced by the AP changes can significantly decrease their QoE. The user experience can get worse even when using the network's edge. Proper multimedia content management can be done by the VoD, which focuses on providing a better QoE for user connection changes. A content migration mechanism at the edge and upper tiers can mitigate the problem. Also, rerouting between the active users' connections and the server nodes may help improve and balance the user's QoE.

4.3.3 MIGRATE performance

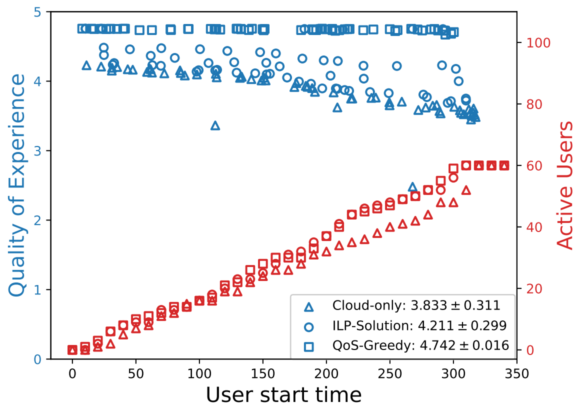
This section presents the performance evaluation of the MIGRATE orchestrator. The evaluation will be conducted by comparing the results obtained from the ILP model to the other two models in the field. The potential results of applying the orchestrator to the mobile operator's network are addressed. Now, three different scenarios: a Cloud-Only scenario without MIGRATE and two scenarios using different strategies to use the available resources in the MIGRATE orchestrator, called QoS-Greedy and ILP Solution.

The *Only-Cloud* scenario only uses the Cloud Provider node to deliver the video content. On the other hand, the QoS-Greedy approach uses the edge network nodes; whenever link congestion is detected, *QoS-Greedy* is activated to choose which edge node to assist in delivering the video. The simulation's initial scenario starts with users requesting video content from the cloud. However, as soon as a congested link is detected, the edge caching mechanism below the link is enabled, and the users receiving the video through the congested link are redirected to the activated edge nodes. This approach aims to alleviate the congestion on the network by reducing the amount of data that needs to be transferred over the congested link and, instead, serving the video content from the edge nodes located closer to the users. The third scenario, *ILP Solution*, uses the proposed architecture to implement the video streaming service, maximizing usage while respecting the user's QoE needs.

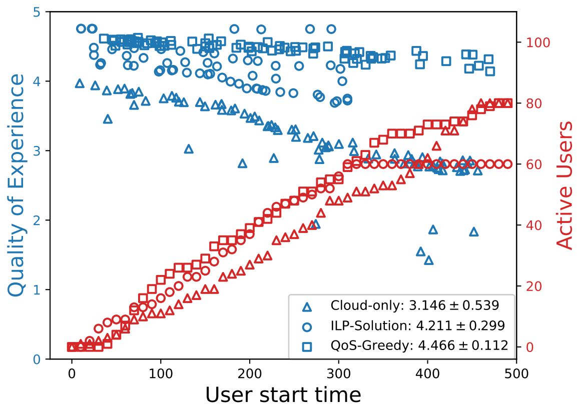
Figures 4.7(a), 4.7(b), and 4.7(c) show the final QoE of each user per start time (time to start the segment requests). Each blue tick represents a user, and the y-axis QoE means the final user satisfaction for a fully watched video. Here, we can see the final QoE degradation for each user's entrance during the simulation execution. The red plot represents the number of active users simultaneously. As users reach this experiment's final QoE, it tends to be slightly lower than the previous one. When looking at the final QoE delta between users u_i and u_{i+1}, this seems to be irrelevant, but as we increase the delta, the QoE starts to become considerably different. We can also confirm this behavior by showing that as the number of users increases, the average QoE decreases, and the variation between users increases. Also, note that in the simulation with 25 users per AP, the system already shows a degradation in the quality, which is negatively perceived by the end-user.

An interesting discussion takes place when the QoE per user is analyzed separately. According to the numerical results, the final QoE worsens as active users increase. However, it is not entirely true for the Edge cache scenario, where the final QoE for each user remains close. The standard deviation of 0.037, 0.162, and 0.273 for scenarios with 15, 20, and 25 users per AP indicates a closer QoE between the users. Only in the scenario for 25 users per AP did the average QoE drop with a satisfaction close to regular, in contrast to the other two scenarios presenting a user's satisfaction from good to excellent. The network operates with a high standard deviation for cloud-only and mobility scenarios. As the number of users increases, some outliers start to appear, with the worst level of satisfaction between poor and inadequate.

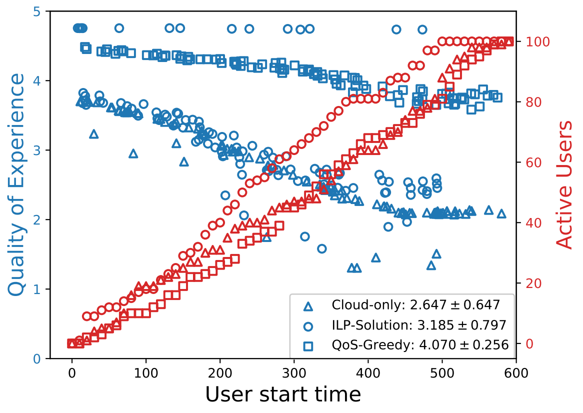
Based on these observations, moving the video to the edge can significantly improve the user's QoE. This way, the video transmission system can satisfy users and keep them



(a) Average QoE for each user (15 users per AP).



(b) Average QoE for each user (20 users per AP).

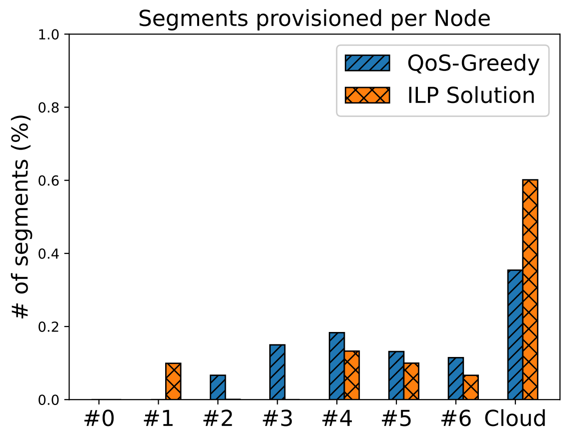


(c) Average QoE for each user (25 users per AP).

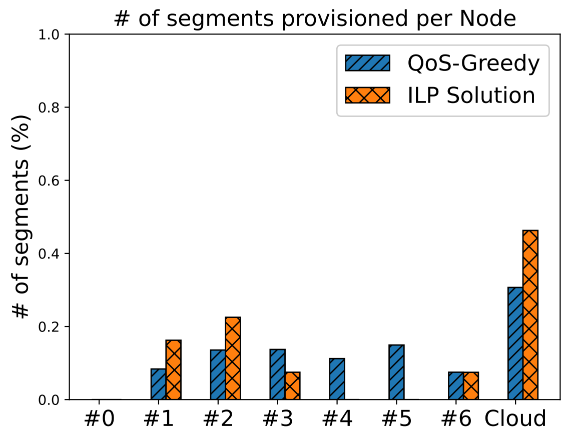
Figure 4.7: Average QoE for each user group. Each mark represents the final user QoE for the following approaches: Cloud-only, Edge cache, and Mobility.

watching the video until the end. However, suppose there is no correct management of connections in real-time and a dynamic mechanism to tackle a varying load coming, for example, from the mobility of users. In that case, we can conclude that the impact introduced by the AP changes can significantly decrease their QoE. The user experience can get worse even when using the network's edge. Proper multimedia content management can be done by the VoD, focusing on providing a better QoE for user connection changes. A content migration mechanism at the edge and upper tiers can mitigate the problem. Also, performing a rerouting between the active users' connections and the server nodes may help improve and balance the user's QoE.

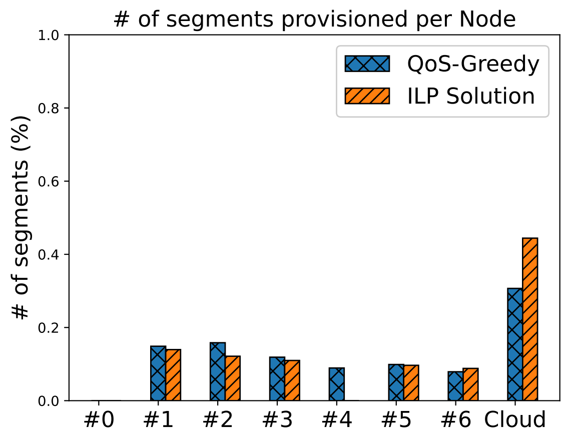
Figure 4.8 (a), (b), and (c) illustrates the distribution of requests between edge servers. For the ILP solution, as the number of hops decreases, the number of requests between layers decreases for the 15 and 20-user scenarios. This can be understood by examining the formal model of the ILP, which only allocates new nodes for video streaming if the current edge servers do not meet the required demand. On the other hand, the QoS-Greedy approach always looks for the closest edge server possible to provide video streaming, resulting in higher consumption of resources. As observed, more video streaming services are activated in the *QoS-Greedy* strategy compared to the *ILP Solution*. This difference in resource usage is reflected in the request distribution between layers and servers, with



(a) 15 users



(b) 20 users



(c) 25 users

Figure 4.8: Number of segments provisioned per node.

the *QoS-Greedy* approach utilizing more resources to ensure a consistent and high level of QoE.

Based on these observations, the simulation results demonstrate the effectiveness of the proposed *QoS-Greedy* approach in providing high-quality video streaming services to users. The *QoS-Greedy* algorithm's selection of the closest edge nodes to the user helps to improve the QoE for users, and the standard deviation of QoE values is lower than for the other algorithms. On the other hand, the results of our analysis show that the *ILP solution* is more efficient regarding resource usage when compared to the *QoS-Greedy* approach. As the number of hops decreases, the number of requests between layers decreases for the 15 and 20-user scenarios. This is due to the formal model of the ILP, which only allocates new nodes for video streaming if the current edge servers do not meet the required demand. In contrast, the QoS-Greedy approach always looks for the closest edge server possible to provide video streaming, resulting in higher resource consumption. Therefore, a trade-off between quality and cost should be evaluated according to the expected demand and resource availability.

4.4 Chapter Conclusion

This chapter introduced MIGRATE and use this architecture to investigate the characteristics of a multi-tiered Edge-Cloud scenario for video streaming services. MIGRATE leverages the strengths of edge computing to enhance resource utilization, optimize content delivery, and improve the QoE for end-users. MIGRATE effectively balances the trade-offs between network efficiency and user satisfaction by enabling real-time decision-making for video content distribution.

The experimental results highlight the significant benefits of multi-tier architectures in reducing latency and improving video playback quality. Users receiving content from lower-tier nodes experience higher bitrates, faster buffer filling, and fewer playback interruptions than those connected to higher-tier nodes. This demonstrates the critical role of localized caching and service delivery in improving QoE. However, the results also reveal a trade-off between resource utilization and QoE. While optimization strategies prioritizing user experience can achieve higher QoE, they require more aggressive resource allocation, potentially leading to increased operational costs. Conversely, resource-efficient strategies may struggle to maintain consistent user satisfaction during peak demand.

Mobility presents a key challenge, as dynamic user movement disrupts resource allocation and reduces QoE due to the lack of real-time rerouting mechanisms. Addressing this issue will require integrating predictive mobility models and real-time resource management techniques into the MIGRATE framework. Despite these challenges, the architecture demonstrates scalability and adaptability, dynamically adjusting to changes in user density and network conditions.

In conclusion, MIGRATE represents a significant step forward in the evolution of adaptive video streaming architectures. Bridging the gap between resource efficiency and user experience offers a scalable solution for meeting the demands of next-generation multimedia services. As edge computing continues to evolve, MIGRATE and similar architectures will play a pivotal role in shaping the future of digital content delivery. However, the continued refinement of algorithms, deeper integration of AI technologies, and adaptation to emerging trends will be essential to fully realizing its potential.

Chapter 5

An Architecture for Content Steering on Edge-Cloud Computing

Adaptive video streaming applications utilize multiple CDNs to efficiently deliver end users' content. These CDNs can duplicate their content catalog across various locations or, more commonly, deploy distributed servers to retrieve content from a centralized shared source. Consequently, distinct URLs are generated for each location, directing end users to the same content source. Apple and DASH-IF have recently made significant progress in standardizing Content Steering Service, introducing this as a new technology to simplify the design of multi-CDN [74, 75]. This service enables content providers to seamlessly transition the user's player between different content sources at startup and midstream. This steering service operates independently of content manifests but requires including the steering server address in the manifest file [39, 76].

This chapter presents the **A**daptive **V**ideo **E**cosystem **N**etwork **U**nifying **E**dge-Cloud Continuum (AVENUE), an architecture designed to manage request load across the Edge-Cloud Continuum. Additionally, it introduces an additional layer in video streaming management through Content Steering. AVENUE does not require custom client plugins, DNS redirects, or Content Management System integration, making it a cost-effective alternative to existing video service switching solutions. The Content Steering Service within AVENUE consists of two fundamental modules: monitoring and selector. The monitoring module gathers real-time context metrics, while the selector module utilizes the Select Server Algorithm to choose an appropriate edge server. This approach ensures a better utilization of Edge-Cloud Continuum resources to provide a high-quality user experience. With proper network integration, this content steering service can address challenges associated with managing video streams, especially in mobile devices where connection quality is susceptible to fluctuations.

The contributions of this chapter are as follows:

- Introducing AVENUE, an architecture designed to facilitate real time decision-making by directing services;
- An experimental testbed to validate the Content Steering Service in an Edge-Cloud Continuum environment. The open-source code is available at `https://github.com/eduardogama/AVENUE.git`;

- Analysis and evaluation of the Content Steering Service: How to adapt the service to the edge necessities and assess features related to the network context.

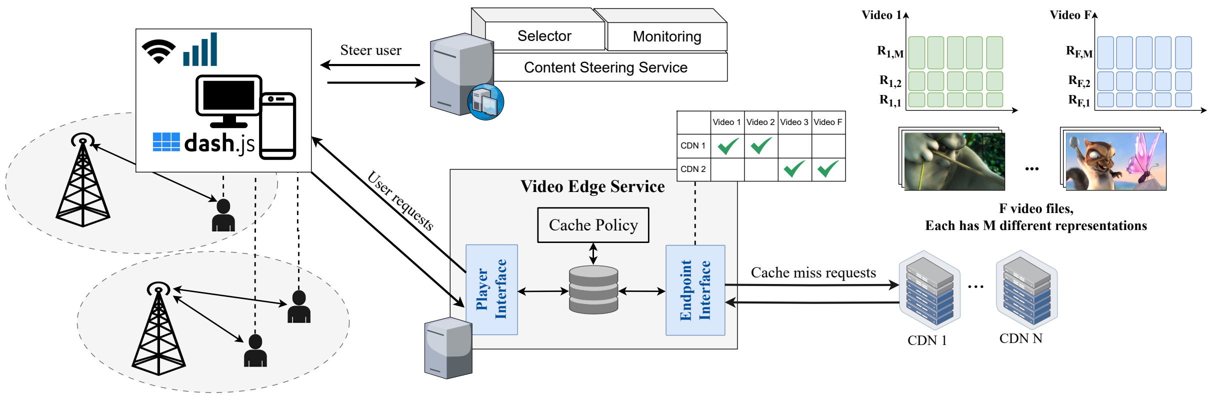
The remainder of the chapter follows this organization: Section 5.1 presents the system model with the steering mechanism used in the evaluation. Section 5.2 details the testbed scenario employed in our experiments, simulation-based tests, and experimental results, and finally, Section 5.3 concludes the paper.

5.1 Seamless Coordination in Content Steering Service

CSS is an important feature in network management that helps to guide users toward the best video edge servers that can provide multimedia content. In AVENUE, the CSS is an additional layer that works with the monitoring and selector modules, giving a comprehensive system overview. In this section, we discuss in detail the monitoring and selector modules and how they work together to ensure the complete functioning of the service.

System Model

The Deployed Adaptive Video Streaming workflow is illustrated in Figure 5.1. The Video Edge Service strategically situates itself in proximity to user demands, capturing video content from CDN sources upon cache misses. This strategic delivery optimizes traffic consumption through the core network and minimizes latency by efficiently storing content at the edge [35,77]. Here, we refer to Adaptive Video Service or Video Edge Service as the interchangeable microservice for the same purpose. CSS is an application-layer solution that empowers Adaptive Video Systems with software-defined adaptability, considering key edge factors like user location, edge capabilities, video content information, and network resource quality. We trace the journey of adaptive video content from the source to the user's player, showcasing how Content Steering dynamically optimizes delivery and the features of the Video Edge Service.



	Video 1	Video 2	Video 3	Video F
CDN 1	✓	✓		
CDN 2			✓	✓

Figure 5.1: Workflow Architecture of a Content Steering-Enabled Adaptive Video System.

Video content provisioning begins with the source from the content provider. This source undergoes encoding into multiple quality levels, defined by their respective bitrate ranges. The content is then segmented into discrete chunks of predetermined duration for independent decoding, enhancing streaming adaptability and efficiency. Static CDNs store these segments in the cloud. The content provider also generates a manifest containing base URLs to video segments and the address for clients to access the content steering server.

When a user requests the manifest from the CDN and receives a directive to contact the **Content Steering Service**, the user's device starts the communication following a standard protocol. Content Steering may leverage a comprehensive global network topology and operate with persistent user sessions in real-time as a key network management component. This stateful approach provides real-time insights into user connections across the Edge-Cloud Continuum and enables the Video Edge Service to operate in an agnostic way. These insights empower the service to optimize video streaming management and make informed decisions with minimal administrative intervention towards a zero-touch network management service. Our deployment involves two modules — monitoring and selector — working together to operate the steering mechanism. The following section describes each module in detail.

An edge node hosts the **Video Edge Service**, with its online status reported to the monitoring module. This service possesses two key communication interfaces: The *Player Interface*, a one-hop connection for players, and the *Endpoint Interface*, which points to one or multiple remote video servers to recover the video segments based on requests from the user's players. The Video Edge Service is a Python-based code that works with Cache-control lib and is adapted to implement different Cache policies. This work initially employs a Least Recently Used (LRU) policy when the cache reaches capacity. In Fig. 5.1, when receiving a request for a video chunk from a user, the *Player Interface* searches for the chunk and, if found in the cache, is delivered directly to the user. Otherwise, a cache miss is generated and the *Endpoint Interface* fetches the chunk from the remote CDN, caches it for future requests, and then the *Player Interface* delivers it to the user.

Mobile users are based on the DASH paradigm, where each client independently executes its own ABR scheme. Based on Equation 2.2, a dedicated module computes the QoE to enable real-time monitoring. This equation incorporates four key metrics: average perceptual quality, quality oscillation frequency, stall event statistics, and perceived throughput. Section 6.2 describes the details of this QoE model. We stored the logs for the last analysis during the simulations.

Our system model design initially adheres to DASH-IF Working Group specifications. Meanwhile, HLS follows identical protocol specifications involving the Content Steering Service. Our design system choice ensures the deployed system's versatility, enabling seamless operation in diverse environments for Adaptive Video Streaming across the Edge-Cloud Continuum.

Monitoring Module

The monitoring module is responsible for capturing real-time context metrics. The metrics are collected periodically, casting requests to a previously informed endpoint. Alternatively, the monitoring module can receive metrics from a predefined entry point port. In this work, our focus is on observing the video content ID and user location. Concerning user location, when a new user joins the platform, the monitoring receives the network details, such as the location area code and cell ID, which accurately identify their current location or stopping point. As for video content ID, the monitoring module handles this as the user requests come with the video content ID. This monitoring data optimizes the content provisioning in two ways:

- Reduced network hops: The system minimizes data travel distance by directing users to geographically closer video providers, reducing latency and network congestion.
- Targeted content caching: Knowing the application contexts allows the routing based on user or server preferences to improve the playback experience or resource allocation.

The monitoring module is central to the overall system. It observes predefined network metrics, which provides crucial feedback to ensure efficient network resource use and improve the end-user experience.

Selector Module

The selector module chooses a video edge node to attend the user requests, achieving a positive impact based on the monitored contexts (e.g., user location, network conditions, video content). This method must achieve a shorter execution time and not cause overhead, which aligns with the vision of the computing continuum, seamless integration of edge computing and cloud. Content targeting capability facilitates efficient and responsive execution. We detail the Selector Module's implementation in Algorithm 1.

Algorithm 1 Content Steering Selector

```
Require: User request req, List of active sessions sessions
Ensure: Edge Server server
 1: session ← session.getSession(req)              ▷ Retrieve session from req
 2: if session is not null and session.is_alive() then
 3:     return session.getEdgeServer()
 4: else
 5:     session ← createSession()
 6:     server ← SelectServerProblem(req)          ▷ Select edge server for request req
 7:     session.setEdgeServer(server)
 8:     session.setSession(session)                ▷ Insert session in list of active sessions
 9:     return session.getEdgeServer()
10: end if
```

The algorithm shown in the pseudocode initiates by fetching the user session from the session list (line 1). If the user session is present and currently active (lines 2-3), the algorithm returns the server that remains active within the specified time threshold until the session expires. The time threshold is the call to the steering server again. In cases where the user session is not found or is inactive, a new session is established (line 4). Subsequently, the algorithm addresses the Selection Server Problem, obtaining the optimal solution servers (line 5). Then, the algorithm updates the user session with the selected servers and provides them as the output for the user's request (line 6). The default time interval threshold between the user and content steering is 10 seconds. When the time threshold is reached, a call to the steering server is made again. This method ensures efficient edge server selection, prioritizing them in the highest order, followed by the subsequent inclusion of CDN servers. Below, we provide an example of a response the server can generate.

```
{
"VERSION": 1,
"TTL": 10,
"RELOAD-URI": "https://steeringserver.com?session=abc",
"PATHWAY-PRIORITY": ["EdgeCache-1", "EdgeCache-2", "CDN"],
"PATHWAY-CLONES": [
   {
       "BASE-ID": "CDN",
       "ID": "EdgeCache-1",
       "URI-REPLACEMENT": {
           "HOST": "https://edgecacheserver-1.com",
       }
   },
   {
       "BASE-ID": "CDN",
       "ID": "EdgeCache-2",
       "URI-REPLACEMENT": {
           "HOST": "https://edgecacheserver-2.com",
       }
   }
]
}
```

Listing 5.1: JSON Response for Content Steering Service. This JSON response provides versioning, TTL, URI, pathway priorities, and pathway clones for dynamic edge and CDN-based adaptive video streaming.

In this example, the response contains two servers identified by the PATHWAY-PRIORITY array. These servers are capable of providing the requested video segments. It prioritizes the EdgeCache server, selected using Algorithm **??**, while the CDN server has a lower priority. The specifications of the EdgeCache are detailed in the PATHWAY-CLONES with an original CDN base, and the edge node address rules are given by the URI-REPLACEMENT, with the host replaced by a URL in HOST. The client receives these instructions with a TTL of 10 seconds, indicating the response interval for requesting the next update. This callback mechanism ensures timely adjustments to the streaming pathways based on network conditions or node availability, enabling its reuse in different parts of the system.

This syntax for steering server responses and client-server interactions remains consistent for both HLS and DASH systems. Consequently, a single server can manage content steering operations for both protocols. For detailed specifications regarding the response format within the Steering response, refer to the guidelines outlined in [74].

Workflow Communication

Figure 5.2 illustrates the workflow of a CSS in the AVENUE architecture, highlighting the interactions between the Cloud, Edge Cache, CSS (comprising Monitor and Selector modules), and the User. The process begins in Figure 5.2-I, where the user initiates video playback. The video playback starts with a request to the Cloud, which responds by providing the manifest file.

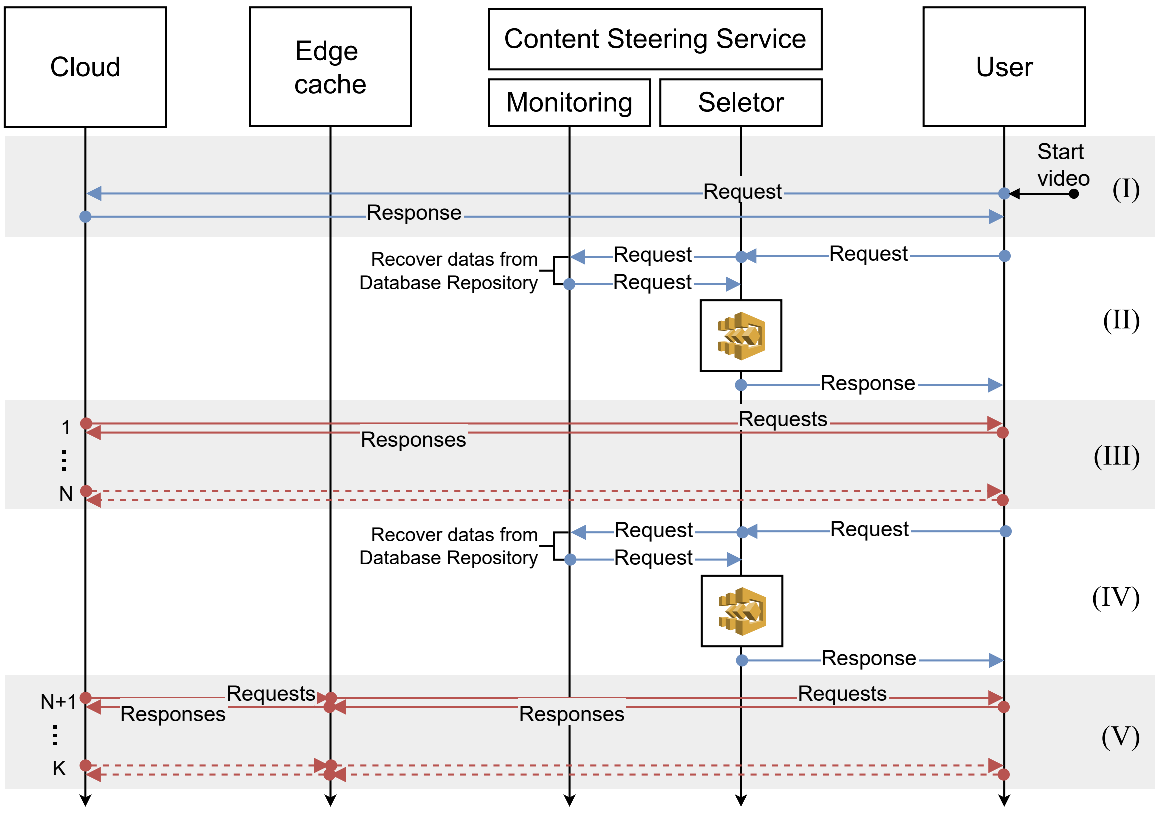


Figure 5.2: Workflow of the CSS in the AVENUE Architecture, illustrating dynamic interactions between the user, edge cache, Cloud, and CSS for the content delivery provisioning.

In Figure 5.2-II, the user contacts the CSS endpoint and requests the video. The CSS retrieves metadata and performance metrics from the Repository to evaluate the optimal content delivery source, either the Cloud or an Edge Cache. Using this information, the CSS executes the Selector algorithm and responds to the user with a JSON message, formatted similarly to the example shown in Figure 5.1. At this stage, these interactions involve control messages rather than video content transmission, focusing solely on determining the most suitable server for handling the user's video requests.

In Figure 5.2-III, the user begins requesting video segments, ranging from segment 1 to segment N. After the TTL expires, as shown in Figure 5.2-IV, the user sends another control request to the CSS. Based on updated metrics, the CSS instructs the video requests to a different node, such as an Edge Cache. Finally, in Figure 5.2-V, the user continues streaming the video, starting from segment $N + 1$, now served by the newly assigned node.

The system dynamically adapts as video segments are delivered, represented by successive requests and responses. Metadata retrieval occurs periodically to reassess and adjust the delivery strategy based on real-time conditions, ensuring continuous optimization. The process seamlessly balances content distribution across the cloud and edge resources, enhancing the user's QoE through efficient decision-making and resource utilization.

5.2 Performance Evaluation

This Section presents the numerical results, considering key performance aspects of the proposed CSS and focusing on QoE, cache hit rate, and load fairness. The evaluation aims to assess different performance metrics of the system, and how the CSS adapts to varying network conditions across the Edge-Cloud Continuum.

5.2.1 Experimental Setup and Parameters

To deploy our cloud nodes, we utilized OpenStack as the orchestrator and employed Kubernetes and Docker to manage the services container. We deployed two CDN nodes and one node for steering and dash.js services, each equipped with a 25G disk, 8G RAM, and four vCPUs at 2.1 GHz, running on CentOS-7. To emulate the Cloud connections, we configured the connection between each node in the Cloud and the root router with 175Mbps bandwidth and 50ms latency using the Linux Traffic Control tool, *tc*. A total of ten video content types were stored, including animation, documentary, and games, five in each CDN node. We use the reference test videos from the open-source dataset in [78]. The codec chosen was Advanced Video Coding (AVC), which is the most used. Since users are interested in a video with twelve different representations, the resolutions range from 144p to 4K, with the representations divided into 4-second segments, and the maximum media duration is 322 seconds.

Our experimental setup in the Mininet-WiFi emulator initiates the establishment of a root router, providing connectivity between the Cloud and the environment. We deploy three video edge servers, each directly connected to the root router with 100 Mbps bandwidth and 10 ms latency. Further, each edge server links the four lowermost nodes representing the Access Points (AP). These links do not incorporate any specific capacity constraints. The AP nodes are realized as wireless devices, employing communication via IEEE 802.11g at 2.4 GHz. The log-distance propagation calculated the signal levels using a Loss Exponent of 2.8, indicative of an Urban Area Cellular Radio environment [79]. All the simulations ran on a local computer Dell G15 5520 12th Gen Intel® Core™ i5-12500H processor with 16GB RAM.

In our scenario, each AP group covers a specific region. These regions have the same popularity video tendency with user preferences for video content denoted by a Zipfian distribution. The Zipfian distribution gives the video selection with α equal to 2.0. It ensures the popularity distribution of video content by the literature. The users' arrivals follow a Poisson distribution arrival rate, and a uniform distribution determines their spatial distribution in 2.0x2.0 kilometers, in which the APs cover the entire region. The users decide their handover based on the Strongest Signal First criterion, wherein they select the AP with the strongest signal. Upon connecting to a specific region, the user's video access frequency adheres to the Zipfian distribution, favoring the request for popular videos from that region.

Mobile users use the dash.js player in the Google Chrome browser (version 117.0.5938.88), enabling headless mode. The Selenium library (version 4.11.2) in Python (version 3.11) implements user behavior. These clients employed the default dynamic-based ABR scheme (abrDynamic.js). A dedicated API was developed following Equation 2.2 to facilitate QoE monitoring. This equation incorporates four key metrics: average perceptual quality, quality oscillation frequency, stall event statistics, and perceived throughput.

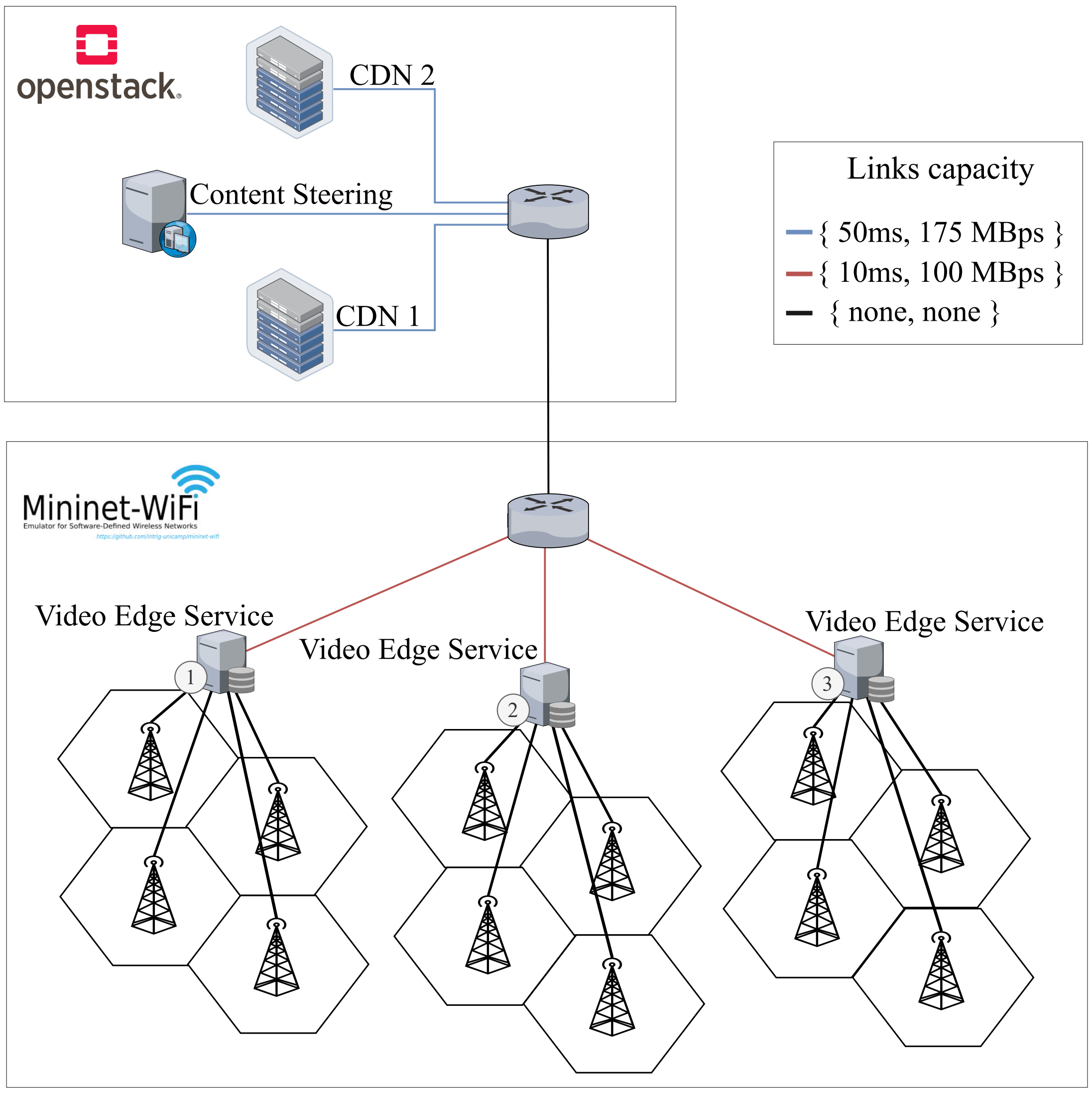


Figure 5.3: Scenario a Content Steering-Enabled Adaptive Video System.

Server Selection Approach

This work investigates three known heuristics tailored for dynamic video delivery scenarios, addressing impact specific features: video content (Content Aware), user location (Region Aware), and load balancing (Round Robin).

- Content-Aware (CA) is designed to mitigate redundant requests for identical videos. Upon the initial request for a video, the system redirects subsequent users seeking the same video to a common edge server. This strategic redirection mechanism optimizes resource utilization, minimizing unnecessary duplication of content in retrieval requests.
- Region-Aware (RA) aims to direct users to the nearest node along the path between the server edge and the user. This methodology dynamically redirects users to the closest server, reducing latency and improving the network's efficiency.
- Round Robin (RR) algorithm assigns each video request to an edge server in a circular manner, ensuring a fair distribution of load among all active edge servers. It prevents overloading individual edge servers for extended periods.

These algorithms must have low execution times to achieve real-time response. Table 5.1 demonstrates that we measure the execution time in microseconds (μ). Hence, the mechanism can operate in real-time with minimal impact and a rapid response time, making it ideal for attending to various user requests.

Table 5.1: Execution time with standard deviation for Selection Algorithms.

Algorithm	**RA**	**CA**	**RR**
Time (μs)	11.67 ± 3.45	7.81 ± 1.11	6.53 ± 1.01

5.2.2 Results and Discussion

This section evaluates the proposed system across key metrics: Perceptual QoE, Fairness, and Network Scenarios. The QoE Evaluation employs a logarithmic model to assess user satisfaction based on segment bitrates. Fairness Evaluation uses Jain's Index to measure request distribution among servers. Finally, Network Scenarios analyzes the impact of increasing user loads on QoE, cache hits, and fairness, highlighting trade-offs among CA, RA, and RR algorithms for content delivery.

Perceptual QoE Evaluation

The QoE metric scores the perceived user satisfaction. To begin, the dash.js API calculates the video quality of each segment based on its bitrate using a logarithmic law [48]. Equation 2.1 shows the numerical transformation for each received video segment v_l^k. The video dataset considers the representation bitrate range from 100 Kbps (144p) to 17000 Kbps (4K) in this work. Constants a_1 and a_2 are defined based on the bitrate representations range. The function $q_u(.)$ ranges from 1 to 5. In this way, $a_1 \approx 0.779$ and $a_2 \approx 613.848$. It provides a more accurate and precise measurement of the user's satisfaction with the video quality. The logarithmic law is commonly used in the literature to model the relationship between the video quality and the bitrate, as it captures the human perception of video quality more accurately than a linear relationship. By considering these different values, we can more comprehensively and accurately measure the user's satisfaction.

Fairness Evaluation

To evaluate aspects related to the Load Balance between the Video Edge Services, we compute the fairness for each server considering the total requests received in Equation 5.1. To quantify this fairness, we employ Jain's Fairness Index, which measures the similarity in Load Balance experienced by individual servers throughout the simulations. The Fairness Index (J) calculates the distribution of total requests (R) received by each server ($s = 1, 2, ..., S$), offering an understanding of the requests equity within the system.

$$J = \frac{(\sum_{s=1}^{S} R_s)^2}{S * \sum_{s=1}^{S} R_s^2} \tag{5.1}$$

Network Scenarios Evaluation

The experiments depicted in Figure 5.4 showcase the average QoE on the y-axis with a confidence interval of 95%, calculated using Equation 2.2, against varying numbers of total users (x-axis) ranging from 10 to 50. Each data point represents the average QoE observed in scenarios employing Content-Aware (CA), Region-Aware (RA), and Round Robin (RR) algorithms, with legends providing information on standard deviation and mean values.

The overall QoE remains comparable among the three algorithms for fewer users (from 10 to approximately 30) scenarios. However, as the number of users increases, the RA algorithm performs better than its counterparts. The strategic selection of the closest edge node to serve users contributes to this improvement. In contrast, RR bases its choices on load distribution between edge servers, and CA considers content in its decision-making process, resulting in both algorithms maintaining similar QoE performance. The distinctions in performance become more pronounced with an escalating number of users, highlighting the efficacy of the RA algorithm in QoE performance under increasing user loads.

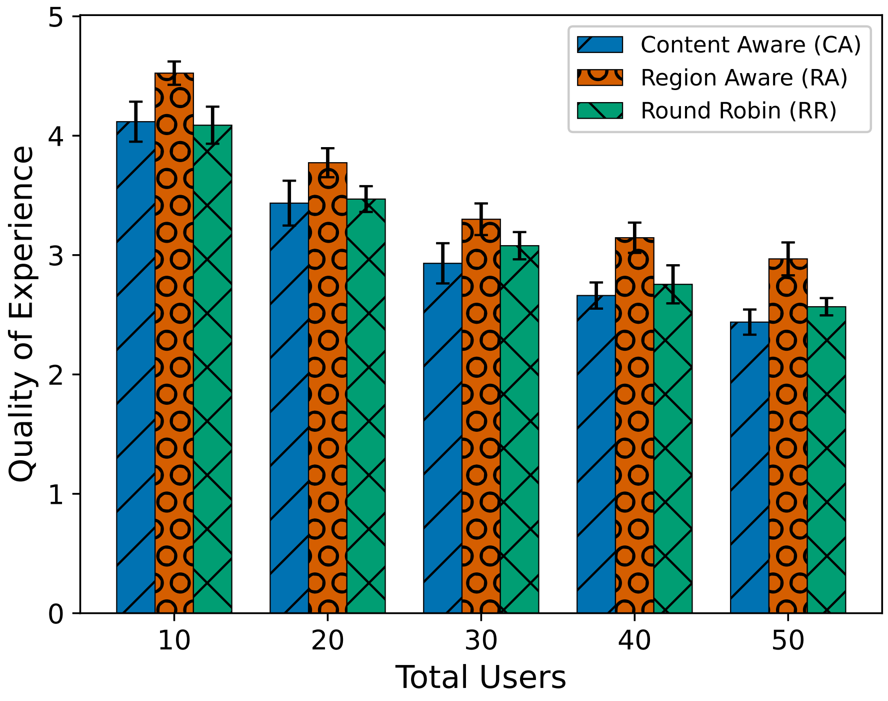


Figure 5.4: QoE Performance with CA, RA, and RR (y-axis) and the total number of users in the x-axis.

Figure 5.5 shows the impact of each algorithm on the cache hit rate. As expected, the CA algorithm exhibits superior performance, achieving a 16% and 34% higher hit rate compared to RA and RR, respectively. This advantage occurs from CA's strategic server selection. CA chooses servers with users requesting the same video content or having made a recent request. This approach optimizes the cache hit rate, minimizing content duplication.

Meanwhile, RA indicates mid-performance, and the RR algorithm demonstrates the least favorable performance with its indiscriminate balancing of requests. This observation aligns with expectations, as the RR approach lacks the contextual awareness of the CA algorithm. Consequently, its cache utilization fails to capitalize on user-specific preferences and content popularity patterns, leading to a comparatively lower cache hit rate.

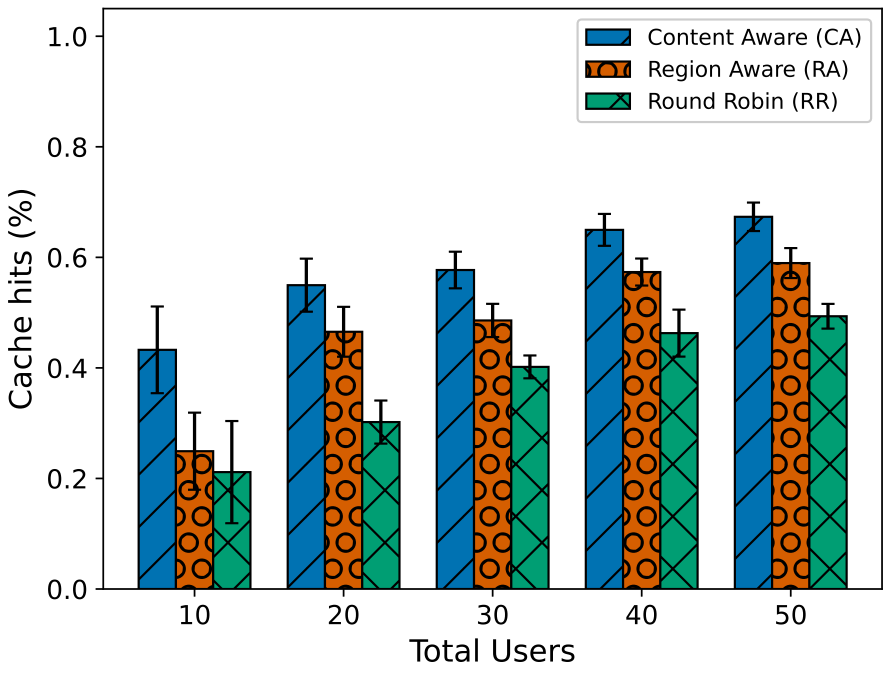


Figure 5.5: Cache Hit Performance with CA, RA, and RR (y-axis) and the total number of users in the x-axis.

Figure 5.6 illustrates the Fairness in distributing video requests across edge servers. RR exhibits the most balanced workload, ensuring equitable distribution and preventing server overload. Conversely, CA can lead to a concentration of requests on specific servers storing popular content, potentially impacting Fairness as the number of users increases. RR outperforms CA and RA by 21% in the 10-user scenario, but this gap narrows to 6% for 50 users due to RA's improved workload distribution. Particularly, CA maintains a steady 20% difference with RR due to its inherent content-driven steering, highlighting the trade-offs between different fairness guarantees and content delivery efficiency.

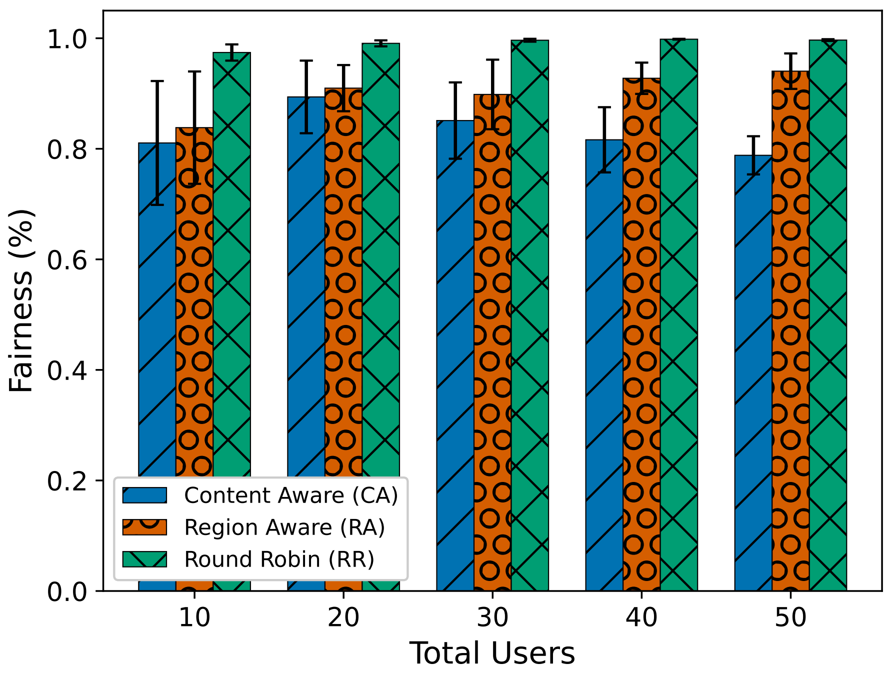


Figure 5.6: Fairness Performance with CA, RA, and RR (y-axis) and the total number of users in the x-axis.

Impact on Planning the Adaptive Video Streaming System

An interesting discussion arises when the impact of the server selector algorithms on network performance is analyzed. Planning Content Steering Service presents different performances depending on the scenario metrics evaluated. Suppose that the Content Provider is interested in ensuring, for the network clients, an average QoE of 3.0 as the minimum acceptable value. For a network with ten users, Table 5.2 shows the network traffic to achieve a QoE of 4.0 for different Server Selector algorithms. However, if we optimize the network to minimize the core network traffic, the CA algorithm becomes the best choice (see lines 4-6 of Table 5.2). In case of the best load distribution among the servers, the RR is the best choice (see lines 7-9 of Table 5.2). For scenarios with high demand, The best one is the RA algorithm. Thus, the content provider can choose the one that best fits the deployment needs.

In summary, each algorithm contributes to network performance in distinct ways. RR ensures the best Fairness and prevents monopolization, RA leverages geographic proximity to return the best users' QoE performance, and CA strategically minimizes the traffic to the core network and increases the number of cache hits. Overall, each algorithm offers distinct advantages and trade-offs, highlighting the importance of choosing the appropriate strategy for specific network scenarios and content distribution patterns.

5.3 Chapter Conclusions

This research focuses on exploring the CSS. Through the exploration and exposition of the CSS Algorithm, we have elucidated the core functionality of the Steering, focusing on

Table 5.2: Review of the results considering the performance profiles: QoE, Cache hits, and Fairness load. Two scenarios analyses with 10 and 50 users.

Line	**Metrics**	**Algorithms**	**Users**	
			10	**50**
1	QoE	CA	4.11 ± 0.25	2.48 ± 0.12
2		RA	4.47 ± 0.13	2.97 ± 0.19
3		RR	4.14 ± 0.15	2.53 ± 0.09
4	Hits (%)	CA	0.40 ± 0.13	0.67 ± 0.04
5		RA	0.29 ± 0.11	0.58 ± 0.03
6		RR	0.17 ± 0.11	0.51 ± 0.03
7	Load (%)	CA	0.81 ± 0.12	0.81 ± 0.02
8		RA	0.81 ± 0.16	0.92 ± 0.06
9		RR	0.97 ± 0.02	0.99 ± 0.01

building a new layer for network management. In the practical implementation, the algorithm can construct a solution based on the Server Select Problem in real time, facilitating seamless communication between servers and clients. It ensures that user requests are not only efficiently processed but also delivered in a deterministic way.

Our study assesses three steering algorithms in dynamic network environments. The evaluated CA, RA, and RR algorithms reveal distinct performance profiles. CA, excelling with strategic server selections based on cached content, consistently outperforms its counterparts, showcasing its efficacy in maximizing cache hit rates and user satisfaction. RA demonstrates scalability, correlating positively with cache hit rates as the user population expands, emphasizing its adaptability to evolving network scenarios.

Chapter 6

Scalable Learning-Based Adaptive Video Streaming Architecture

As discussed and exemplified in previous chapters, the increased demand for adaptive video content affects maintaining QoE guarantees across dynamic network conditions. Traditionally, cloud computing needs to guarantee their customers' performance level, which is pre-established in the SLO specified in the Service Level Agreement (SLA). These agreements often consider QoS metrics, as different applications require distinct types of resource [80, 81]. However, for video streaming services, QoS metrics alone do not completely represent the QoE. For this reason, a QoE-centric approach is desirable to assess video delivery performance accurately [82, 83].

Secondly, this work addresses a real-time allocation of cache services, enhanced by location awareness within an Edge-Cloud network that spans multiple regions. When the cloud can no longer guarantee the required SLO, the architecture addresses the challenges of scaling containers into the Edge. The primary use case includes QoE prediction algorithms, considering QoE as SLO, the architecture anticipates the guarantee violations by a data observation window and uses the Edge to bring the video provisioning closer to the user.

Third, network imperfections along the Edge-Cloud, such as congestion, packet loss, poor wireless signals, and issues at various network layers, can significantly impact video playback. As the overall QoE constantly evolves throughout a streaming session, network variations can cause sudden shifts through the video playback. Consequently, this ongoing QoE fluctuation necessitates real-time monitoring, with an understanding of how these network variations influence the playback.

Our contributions to adaptive video streaming are highlighted below.

- Our main contribution is defining an architecture that leverages existing HAS standards. This architecture tackles the inherent challenges of scaling video services at the Edge, which considers the location-aware coverage of the Edge in multiple regions. This architecture scales video delivery to the Edge during traffic spikes while relying on cloud resources during lower demand periods.

- An ML methodology is defined to deploy models within the architecture. This methodology demonstrates its effectiveness in an emulated scenario. The research employs a LSTM predictive model to forecast QoE.

- The architecture integrates continuous monitoring in real-time. This adaptability is crucial for maintaining the Architecture component's performance in dynamic network environments, ensuring seamless video playback.

- Lastly, this chapter presents a performance evaluation and analysis deployed at an emulated test of this architecture. The results highlight the potential of edge-cloud integration for video streaming services.

The remainder of the chapter follows this organization: Section 6.1 presents an overview of the proposed architecture. Section 6.2 introduces the monitoring used in the environment used in the network. Section 6.3 introduces the planner Service used in the evaluation. Section 6.4 details the testbed scenario employed in our experiments and the metrics evaluated in our results, and finally, Section 6.5 concludes the chapter.

6.1 Proposed Architecture Design Overview

This section gives a general overview of the proposed architecture's components. The architecture adheres to DASH-IF Working Group specifications for creating an independent self-managed adaptive video streaming. Figure 6.1 comprises a shared Database Repository with relevant information and three main components: 1) Domain-Specific Service Monitoring, 2) Planner Service, and 3) Content Steering Service. Multimedia content services at CDN have been abstracted to maintain clarity and emphasize the core functional components. The CDN works in a stateless manner, while the proposed architecture works together to scale cache services at the edge in order to respect SLOs guarantees.

Domain-Specific Service Monitoring

The Monitoring component continuously gathers status updates across various infrastructure layers, from the network plane to the user's media player. This collaboration between different monitoring contexts is crucial for handling potential faults and improving overall service. Indeed, it is mandatory to predict SLO violations and react to such events promptly. The scope of monitored data is diverse in terms of location and type. Monitoring data encompass service-level metrics collected from Monitoring Agents, infrastructure-level information such as computational load, storage usage, and network performance, or user-centric data, such as QoE feedback from end users. All these metrics are transmitted to the Monitoring service through predefined, secure communication channels.

Planner Service

The Planner Service must consider the distribution of workloads between cloud and edge to meet SLO requirements. Here, the component keeps the SLO guarantees. Through monitoring, the planning service acquires relevant insights for scaling video services up or

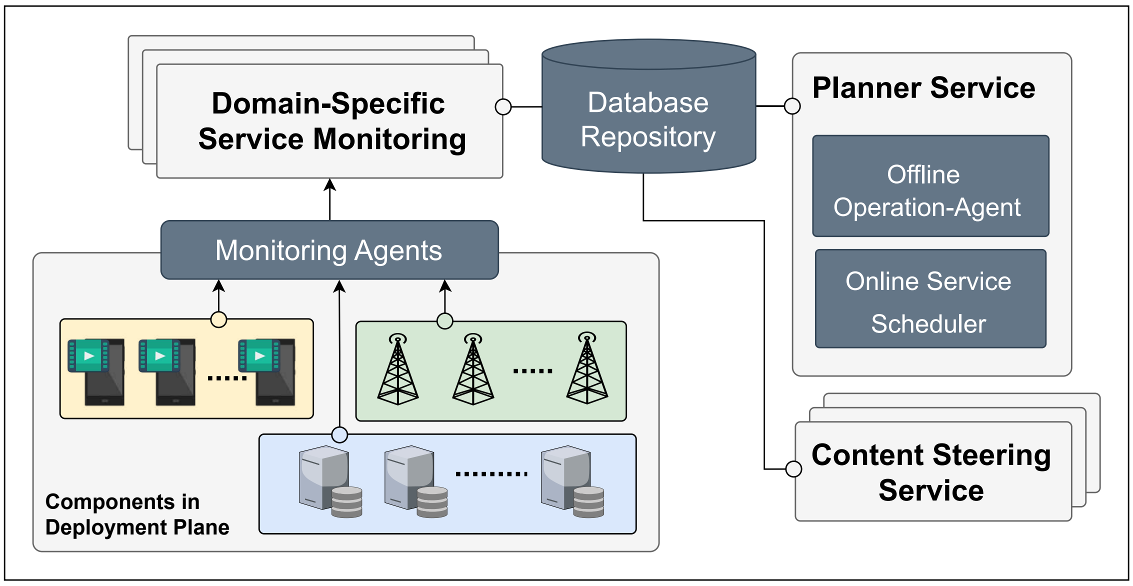


Figure 6.1: An Overview of the Proposed Architecture. Domain-Specific Service Monitoring is responsible for providing continuous monitoring resources. The Planner is responsible for realizing decision-making deployments at the edge. Then, Content Steering works as a load balancer.

down at the network's edge. The proposed Planner component is divided into two phases: **offline** and **online**. The online module functions as an oracle, capable of predicting a specific SLO and making the necessary decisions to auto-scale caching services at the edge when required. In this work, we use ML techniques for forecasting specific metrics that have shown promising results in ensuring SLO requirements. Meanwhile, the offline phase is responsible for training the model. Further details of these phases are discussed in Section 6.3. It dynamically auto-scale up and down the number of containers within the infrastructure in response to changes in demand.

Content Steering Service

The CSS dynamically directs the user's player to the most appropriate video source, whether in the cloud or at the edge. Using the edge of the network with cache capabilities helps improve user satisfaction but can also bring negative impacts. When multiple users request video segments at the edge nodes, these surrogated nodes are considered reliable services, overlooking edge environments' inherent flexibility and dynamic nature. Current video streaming systems are not ready-equipped to handle the variability in the number of deployed edge nodes, leading to potential service degradation under fluctuating conditions. The CSS's ability to access the geographical proximity of the user enhances its decision-making capabilities. This service ensures seamless transitions between different nodes during video playback.

Database Repository

The Database Repository stores the data reported by the Monitor service according to each profile configured at the instances. The information is stored after the Monitoring Service has made a well-defined pre-provisioning.

6.2 Measurement Methodology in Monitoring Structure

This section introduces the methodology for extracting relevant features used to accurately forecast perceptual QoE in the environment in real time. Here, the subsections present a detailed resource metrics collection process, including the approaches for data acquisition and the metrics recorded from the monitor agents. Furthermore, an analysis of the QoE used in the learning dataset is discussed, highlighting both the insights gained and the methodologies applied to ensure the reliability of the dataset.

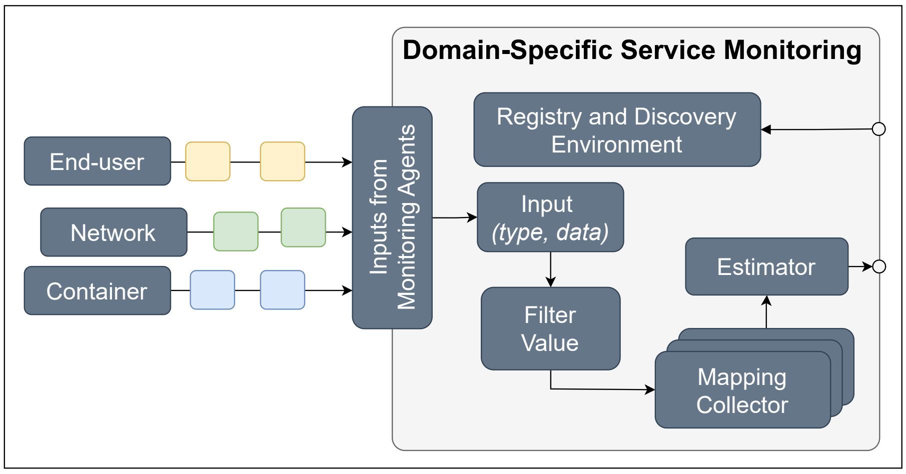

Domain-Specific Service Monitoring
End-user
Network
Container
Inputs from Monitoring Agents
Registry and Discovery Environment
Input (type, data)
Filter Value
Estimator
Mapping Collector

Figure 6.2: The Domain-Specific Service Monitoring is responsible for providing continuous monitoring resources.

Data Collected in the Monitoring Structure

Figure 6.2 details our domain-specific service monitoring in the architecture. Monitoring agents gather data from various sources, including end-users, network, and container components. The end-users are video players that send QoE feedback, network sources with user location and throughput data, and resource usage metrics from containers deployed. Each component sends data independently to the monitoring system.

The received data structures the input data into two fields: type and data. The appropriate insertion method is called based on the type (player, container, network, or network). Player statistics (QoE metrics) follow one format, while container statistics adhere to another. For example, player metrics, such as QoE, are processed and stored in the player database, with additional logic to associate the player's QoE with the node responsible for serving the video stream. This additional logic is defined in the Mapping Collector's specific instructions for each metric received. The data passes through this phase if a metric does not have a Collector.

Regarding container monitoring, when the metrics are inserted in the Database, the service calculates each node's average QoE based on the cumulative QoE values of connected players. Additionally, it records the load on each node, represented by the number of players connected to that node. This information is essential for the Planner Service to dynamically adjust resources in real-time, ensuring that load is balanced across edge and cloud nodes to avoid degradation performance in user experience.

The monitoring also collects and processes network-level metrics, such as bandwidth and latency, which are critical in determining overall network health and its impact on video delivery. The network metrics are classified between the different regions and stored in the network database, helping the system make informed decisions regarding server selection, load balancing, and QoE optimization.

In addition, the registry and discovery environment component maintains a service directory and node geolocation mapping with static data of the infrastructure topology.

Resources Measurement Setup

As previously mentioned, continuous monitoring of resource usage is needed. For our purposes, we used Docker Stats to monitor container resources. Tools such as Nagios and Prometheus can also be readily adapted to collect metrics similar to those from Docker Stats. A Docker API captures node-specific metrics. Table 6.1 provides a detailed breakdown of the collected metrics, categorized into CPU, Disk, Memory, Filesystem, and Network. Seventy-two metrics are gathered for each cache service, reflecting resource utilization.

Table 6.1: Summary of resource categories and their associated metrics used for system performance analysis.

Category	Metrics Description
CPU Data	CPU usage, CPU user, and CPU system, in terms of CPU time consumed per CPU in nanoseconds.
Disk I/O Data	Number of bytes read and written, async and sync operations, and total (sum of reads and writes) from I/O service and daemon.
Memory Data	Memory usage, max usage, cache, and virtual memory data. All values are in bytes.
Filesystem Data	Filesystem capacity, filesystem usage, base usage, and filesystem inodes. All values except inodes are in bytes.
Network Data	Transmit (tx) bytes, receive (rx) bytes, transmit (tx) packets, and receive (rx) packets for each network interface, including eth0 for containers, and cni, flannel, and node interfaces for edge nodes.

Perceptual QoE Estimation

QoE is defined as the degree of satisfaction or annoyance a user experiences with a service, influenced by their expectations and the service's utility. In video streaming, QoE is affected by various factors, including system attributes (like video quality and network conditions) and contextual elements (such as video resolution and stalls). In this work, the video quality of the overall QoE defined as Q^u is in Eq. 2.2. During our empirical evaluation, we observed a consistent pattern in users' QoE variance throughout video playback, as described by Equation 2.2. Figure 6.3 presents a comparative analysis of the difference in QoE values, denoted as ΔQ^u $(= |Q^u_{i+1} - Q^u_i|)$, across the video segment indices. This analysis covers scenarios ranging from 20 to 60 users, using different ABR algorithms. The plot illustrates the fluctuation of ΔQ^u as the segment index i increases, highlighting how the difference between consecutive QoE measurements stabilizes over time.

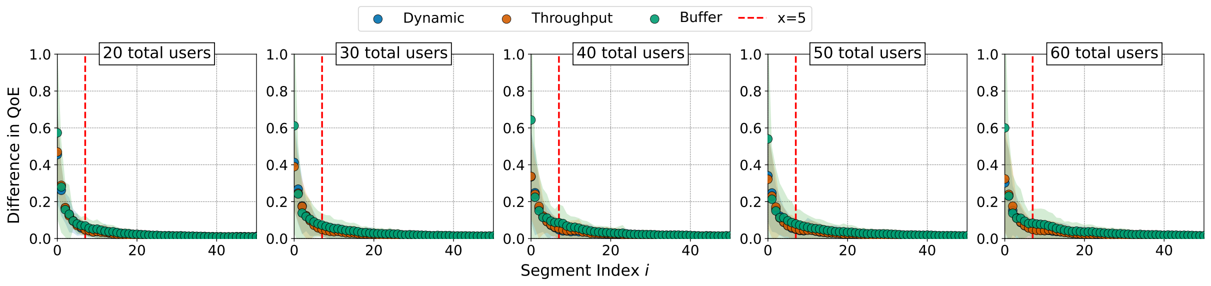


Figure 6.3: Users' QoE difference (ΔQ^u) between consecutive video segments varying user loads (20 to 60 users) and different ABR algorithms.

At the beginning of the video, there is a fluctuation in ΔQ values across all scenarios, with sharp variations evident in the initial segments. These early fluctuations can be attributed to the ABR algorithm in the network's adaptation process and streaming quality adjustments during the session's initial phases. However, as the video progresses, the ΔQ values steadily approach zero, indicating that the differences between consecutive QoE measurements diminish, converging in a more stable QoE.

This behavior suggests that initial stages in user feedback might not accurately reflect the overall performance of the video streaming services, as early-stage fluctuations can give a misleading impression of QoE stability. The user's experience stabilizes as the video session continues, offering a clearer picture of the actual QoE performance over time.

Input Models for QoE Measurement

For further analysis, the mapping collectors organize the input data collected, given its specificities by the monitoring agents, which is then stored in the Database Repository. For a detailed analysis of each node of the QoE provisioned, we need to collect the QoE with the corresponding data from the video-serving node. This association allows us to understand the resource consumption attributed to the server. Moreover, these insights can be further leveraged for ML models in predictive analysis. Our approach involves collecting node samples at every interval $i = 1, 2, \ldots$. We receive a variable number of

QoE values during each sampling interval from different nodes, reflecting the dynamic allocation and switching between different available nodes.

In an Edge-Cloud environment, the set of vertices is $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, where $\mathcal{V}_1$ represents the cloud nodes for provisioning or serving within a mobile operator's backhaul, and $\mathcal{V}_2$ is the edge nodes capable of hosting at least one container. The set of active edge nodes varies depending on the containers deployed across the network and the network demand. Each time an i-th sample is collected from n-th node $v_n \in \mathcal{V}$ with a cache service deployed into v_n. , where content is dynamically available by multiple nodes v_n.

Let $M_{v_n,i}$ be the number of QoE values received from the n-th node during the i-th sampling interval. $Q^j_{v_n,i}$ represents the j-th QoE value for node v_n during the i-th sample, where $j = 1, 2, ..., M_{v_n,i}$. It is important to note that the monitoring process is user-agnostic, meaning that the received QoE values are independent of the particular user player states.

Each sample contains a variable number of QoE values, reflecting the number of users served by v_n during the interval i. The average QoE for each node during a sampling interval is computed as follows:

$$\text{AvgQoE}(v_n, i) = \frac{1}{M_{v_n,i}} \sum_{j=1}^{M_{v_n,i}} Q^j_{v_n,i}$$

Once the samples are computed, they are aggregated and stored in the Database Repository for long-term analysis and system performance optimization. For each sampling interval i, the following data is stored in the database:

$$D_i = \{t_i, \text{Stats}(v_n), \text{AvgQoE}(v_n, i)\}$$

In this expression, Stats(n) refers to the container status of the n-th node based on the setup of the measurement node. In our case, this dataset comprises 72 metrics collected through *docker stats*. Each D_i represents one instance, Contributing to the training of our models. The dataset in the Database Repository supports in-depth data analysis, enabling the identification of long-term trends in QoE and system performance. This, in turn, guides planner strategies within the Edge-Cloud environment, thereby improving the system's efficiency and reliability over time.

6.3 Planner Service Implementation through Adaptive Video Streaming

This section describes the process of deploying cache services within edge nodes coordinated by the Planner Service. Acting as a control mechanism, the Planner interfaces with the edge nodes and the Database Repository, enabling scale cache containers up and down while maintaining SLO guarantees. Therefore, adjusting container deployment based on SLO threshold avoids over-provisioning while guaranteeing performance during peak loads, a critical aspect of edge-cloud resource management.

The Planner component in Figure 6.4 operates through two phases: *Online* and *Of-*

Table 6.2: Table of notations and description.

Notation	Description
$\mathcal{V}$	Set of all nodes.
$\mathcal{V}_c$	Set of active Cloud nodes.
$\mathcal{V}_e$	Set of edge nodes currently active.
v_n	A specific node from $\mathcal{V}$
v_e	an edge node from $\mathcal{V}_e$.
v_c	A specific cloud node from $\mathcal{V}_c$
w	Window of metrics collected for system analysis.
slo	Predicted SLO value.
$slo\_threshold$	Threshold value for the SLO used to trigger actions.
r	A region r of all regions, R.
thr_{cur}	Current throughput value of a resource r_i.
$thr_{\max}$	Maximum throughput value among available resources.
$r_{\max}$	Resource with the highest throughput value.
$\mathcal{E}$	Set of resources already in use (exclusion set).
D_i	Data stored in the centralized database for interval i
Q^u	The QoE value from user u
$Q^j_{n,i}$	The j-th QoE value received in node n during interval i
T	Future time step for prediction
W	Window size of samples used as input in input ML model
$\mathbf{X}$	Input sequence containing W previous samples
$\text{AvgQoE}(v_n, i)$	Average QoE for the n-th CDN node during interval i
$\text{Stats}(v_n)$	Statistics of the n-th CDN node (e.g., throughput, latency)
$g(D_i)$	Data cleaning function given the raw instance D_i

fline. The *Offline phase* initiates with a systematic feature extraction and preprocessing methodology. Our optimization function maximizes overall QoE as our SLO. Upon completing these preparatory steps, the dataset is primed for the training phase of the Machine Learning model. On the other hand, the *Online phase* is responsible for real-time operations, where the decision-making process scales container service within regional edges in response to users' QoE. The components within the Planner service are detailed in the following sections.

6.3.1 Offline Component Operations

We focus on predicting future overall QoE by leveraging data collected from the monitoring service. The offline operations follow the sequence outlined below.

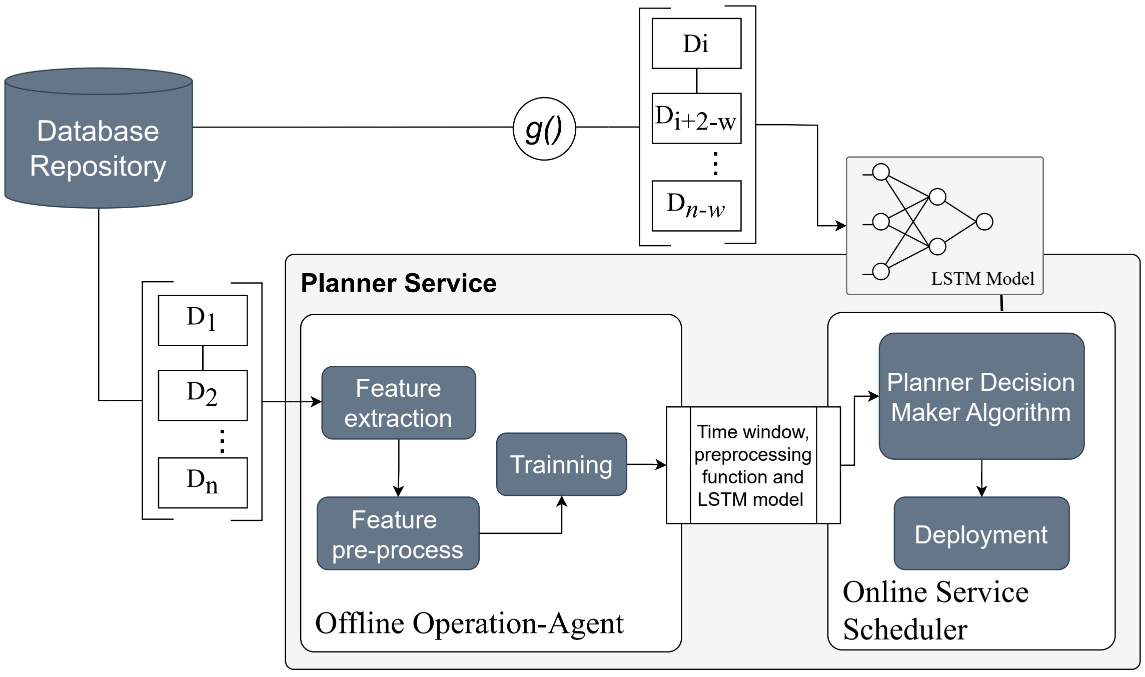


Figure 6.4: The Planner is responsible for realizing the decision-making deployments at the edge.

Dataset Analysis and Features Preprocessing

The Database Repository contains the dataset D. First, we use feature selection techniques to clean the dataset. This stage focuses on reducing dimensionality by eliminating irrelevant or redundant features rather than decreasing the number of data instances. The features with constant values and lower correlations have been removed. After feature selection, the data normalization is applied. Data normalization handles unstructured datasets where values span different ranges. We employ Min-Max scaling to normalize the data into the range $\{-1, 1\}$, defined by $g(D_i) = \tilde{D}_i \in \mathbb{R}$, where $g(\cdot)$ represent the data cleaning and preprocessing function that operates on each instance D_i. Data normalization is essential for mitigating the impact of feature scaling disparities, thereby ensuring equitable feature contributions to the model's learning process.

Model Definition

ML techniques, particularly those based on RNN type, have demonstrated competitive performance compared to other methods when applied to time-series forecasts derived from historical data resource usage [84]. It is a well-documented phenomenon in cloud computing and QoE prediction, suitable for capturing such temporal dependencies. In this way, we opted for an LSTM-based predictive model designed to continuously forecast future aggregated QoE looking at the instances D from Database Repository.

In our approach, the LSTM model is employed to predict future QoE values based on sequences of historical data collected from cloud nodes. The model is trained on input sequences of length w, where each sequence consists of past QoE values and corresponding node statistics collected over w intervals. The raw input data, denoted as $\mathbf{X}$, undergoes a data cleaning and preprocessing stage, $X \xrightarrow{g} \tilde{\mathbf{X}}$, which refines the raw data into a form suitable for model training. This setup enables the model to learn patterns and temporal correlations within the data, as outlined in Equation 6.1:

$$\mathbf{X} = \begin{bmatrix} D_{i-w} \\ D_{i-w+1} \\ \vdots \\ D_i \end{bmatrix} \xrightarrow{g} \tilde{\mathbf{X}} = \begin{bmatrix} \tilde{D}_{i-w} \\ \tilde{D}_{i-w+1} \\ \vdots \\ \tilde{D}_i \end{bmatrix} \tag{6.1}$$

In Equation 6.2, the model outputs a predicted QoE value for a future time step $i+T$, where T represents the prediction horizon:

$$f(\tilde{\mathbf{X}}) \mapsto QoE(i+T) \in \mathbb{R} \tag{6.2}$$

Following training and validation, a threefold $\langle w, g, f \rangle$ is transmitted to the *Online phase*, which operates within the network. This threefold bundle ensures the Planner has all the necessary elements, allowing it to execute the model in a real-time environment.

6.3.2 Online Operations

This phase ingests the threefold bundle, which the decision-making module processes. This process is tightly integrated with the Database Repository, ensuring an up-to-date view of the monitored environment.

The Planner Decision Maker Algorithm (PDMA), highlighted in 3, is embedded within the decision-making component and is responsible for dynamically scaling adaptive video streaming caches at the edge. Its primary goal is to uphold the SLO, defined in terms of QoE guarantees. Operating within an iterative loop, the PDMA continuously monitors container performance, predicts potential QoE issues in the cloud, and makes informed resource allocation decisions to preempt or mitigate performance degradation at the edge.

The algorithm operates in a continuous loop (line 2), iterating over the state of each CDN node (line 3). At each iteration, it retrieves a window of metrics, denoted as $\mathbf{X}$, from v_c using GETWINDOWMETRICS function (line 4). These metrics are then processed via the transformation function $g()$, resulting as $X \xrightarrow{g} \tilde{\mathbf{X}}$ (line 5). The transformed metrics are subsequently passed into a predictive model, accessed through the GETPREDICTOR function, to estimate the QoE of an SLO violation (line 6). If the prediction indicates that the SLO is below the defined threshold, the algorithm triggers the set of actions to maintain the desired QoE (lines 7-8).

The algorithm employs SELECTBESTRESOURCE procedure (line 8) to determine the suitable edge node for cache container deployment. This function assesses the throughput (GETTHROUGHPUT) of the regions R, where the end users (players) are consuming the multimedia content from the cloud v_c. Among the available regions, the region r_i with the highest throughput, which has not yet been allocated, is selected to host the cache

Algorithm 2 Planner Decision Making Algorithm

```
1: procedure PDA
2:     while True do
3:         for $v_c \in \mathcal{V}_c$ do
4:             $\mathbf{X} \leftarrow$ getWindowMetrics()
5:             $X \xrightarrow{g} \tilde{\mathbf{X}}$
6:             $slo \leftarrow$ getPredictor($\tilde{\mathbf{X}}$)
7:             if $slo \leq slo\_threshold$ then
8:                 $v_e \leftarrow$ SelectBestResource()
9:                 DeployCacheContainer($v_e$)
10:            end if
11:        end for
12:        $\mathcal{V}_e \leftarrow$ getContainers()
13:        for $v_c \in \mathcal{V}_e$ do
14:            $v_c \leftarrow$ getParent($v_e$)
15:            if VerifyQoE($v_c, v_e$) $\geqslant slo\_threshold$ then
16:                SchedDeleteContainer($v_e$)
17:            end if
18:        end for
19:        sleep($t$)
20:    end while
21: end procedure
```

container. Afterward, the algorithm proceeds (lines 30-31) to identify and return the edge node within that region. This node, denoted as v_e, is then used to deploy a new cache container (line 9).

After addressing the allocation procedure across the edge nodes, the Planner initiates the deallocation process by retrieving a list of all active containers on the edge nodes (line 12). Then, it iterates over each container (line 13), using the verifyQoE function to assess whether the container's QoE meets the predefined acceptable SLO threshold (line 15). The cache service will be deleted if the container's QoE remains within an acceptable range (line 16). The CSS no longer redirects users to this node, and once all active users

Algorithm 3 Select Best Resource

```
    procedure SelectBestResource
2:      $r_{\max}, thr_{\max} \leftarrow \emptyset, 0$
        for $r_i \in R$ do
4:          $thr_{cur} \leftarrow$ getThroghput($r_i$)
            if $thr_{cur} \geq thr_{\max}$ and $r_i \notin \mathcal{E}$ then
6:              $thr_{\max}, thr_{\max} \leftarrow r_i, thr_{cur}$
            end if
8:      end for
        $v_e \leftarrow$ getEdgeFromRegion($r_{max}$)
10:     return $v_e$                          ▷ Return the best resource
    end procedure
```

have finished consuming the video content from this node, the container is deleted.

6.4 Performance Evaluation

This section describes the experimental evaluation of the proposed multi-tier video delivery architecture, including initial evaluation scenarios, metrics, methodology, and outcomes.

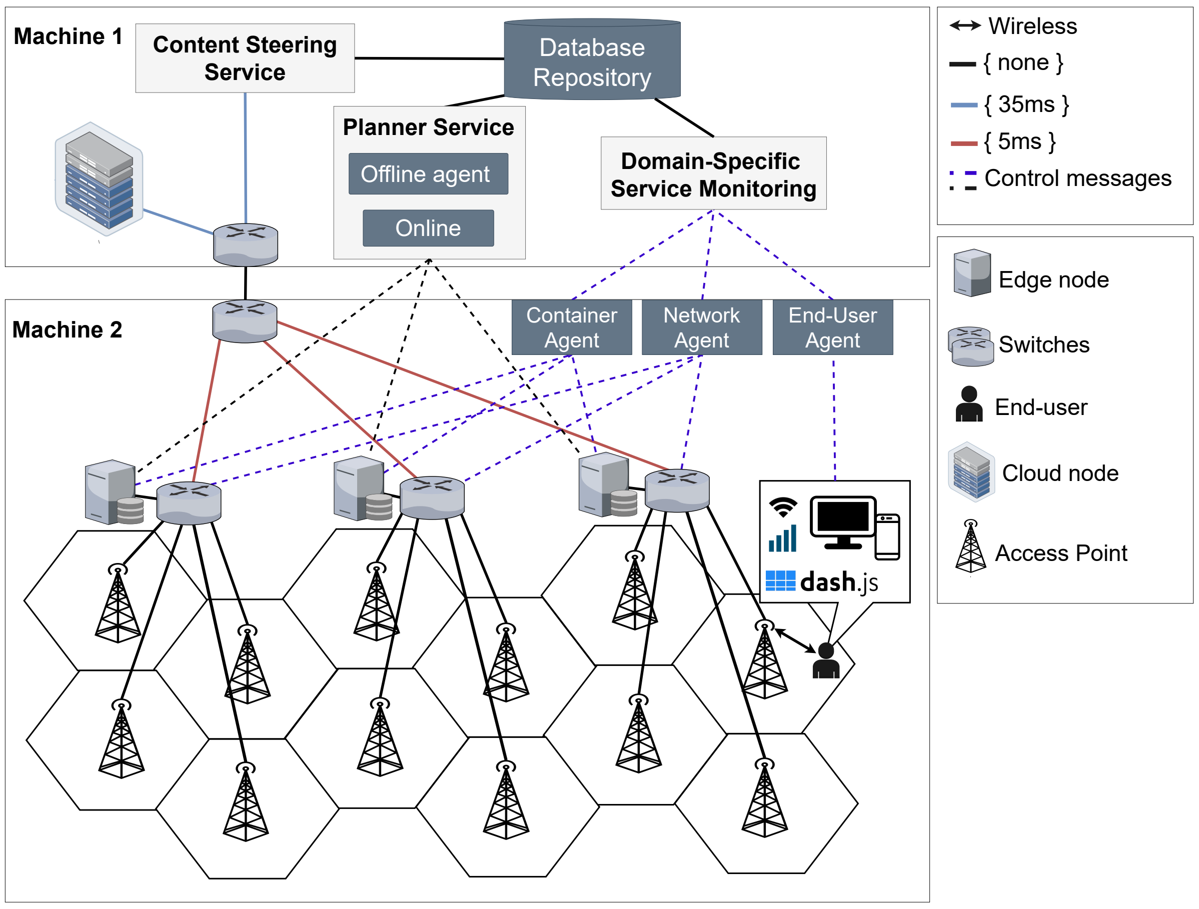


Figure 6.5: Scenario Evaluation Testbed for Adaptive Video Streaming System in Edge-Cloud Continuum.

6.4.1 Experimental Setup and Parameters

Figure 6.5 provides an overview of the emulated scenario engineered to evaluate an Adaptive Video Streaming following HAS standards. At the top, the core components — comprising the DASH FrontEnd, CDN, and the proposed architecture, including the Monitoring, Planner, and CSS — are deployed on a dedicated machine to isolate cloud performance. These components are containerized using Docker, each running independently on an Ubuntu 22.04 LTS environment hosted on a Local machine Intel® Core™ i7-7700 processor with 32 GB of RAM.

The CDN and FrontEnd servers run on an Apache HTTP server [85]. The FrontEnd host dash.js v4.7.4 [86]. The architecture deployment uses a microservices approach, enabling modularity. The architecture's core components, namely Monitoring, the CSS, and the Planner Service, use Python (version 3.11) for implementation. The Planner Service, in particular, leverages TensorFlow and Keras Deep Learning libraries [87, 88], leveraging their deep learning capabilities to execute ML models for adaptive decision-making processes. Using Linux's Traffic Control (*tc*) tool, we configure a backhaul link with a 40 ms latency for outgoing traffic [89].

The CDN stored a video dataset consisting of ten video types, including animation, documentaries, and games. The dataset, sourced from [78], adopted AVC standard, with video files available in twelve different resolutions ranging from 144p to 4K. The videos are segmented into 4-second chunks, with the longest duration totaling 322 seconds. Different resolution choices introduce significant bandwidth demands, mainly for HD and UHD content, pushing the system's capacity to the limit.

Machine 2 represents the edge network environment. We utilized the Containernet-WiFi emulator [90], enabling the accurate simulation of an edge topology, wireless network conditions, and user bandwidth competition. The edge simulation setup was executed on a Dell G15 5520 machine 12th Gen Intel® Core™ i5-12500H processor with 16 GB of RAM.

The system contains the monitoring agents (Container, Network, and End-User Agent). The scenario includes APs providing wireless connectivity to End-Users, who access video content using the dash.js video player. The network components' location, including edge servers, switches, and APs proximity, is mapped into a configuration file with environment information. The monitoring registry stores the components in the database repository.

The edge servers were deployed utilizing Docker-in-Docker (DinD) containers [91], which allow the execution of one Docker container inside another, enabling nested containerization. Based on the latency in [89], each node connects to a central switch with 5ms latency. Each edge node contains four APs, simulated as wireless devices using the IEEE 802.11g standard operating at 2.4 GHz. *Wmediumd* has been adopted to model wireless medium, including bandwidth, latency, and frame loss [92]. The signal propagation followed a log-distance model with a loss exponent 2.8, simulating an urban cellular radio environment [79]. The cache services deployed at the edge operate independently of the overarching architectural components and remain compatible with any caching strategy. However, for the sake of simplicity, all edge caches in this deployment employ the Least Recently Used (LRU) policy for cache replacement.

Comparison Methodology

To evaluate the proposed architecture's performance, we designed a comparison against two approaches: Full-Edge Approach and a Reactive Approach.

The **Full-Edge Approach** serves as a benchmark for assessing SLO violations, particularly due to users' proximity to edge nodes, which typically results in the highest QoE performance with fewer SLO violations. Our objective is to achieve a performance level comparable to this benchmark. Adopting this predictive strategy allows us to anticipate future demand patterns and optimize resource utilization. Specifically, this approach allows us to access edge resources dynamically, only when necessary, reducing unnecessary over-provisioning.

On the other hand, the **Reactive Approach** adjusts resource allocation at the edge only after network SLO violations. This approach is inherently responsive, adjusting resource distribution in reaction to issues. It reacts to the problem after the degradation has already impacted the streaming service, degrading user experience conditions. Consequently, users may experience temporary reductions in video quality or increased buffering while the system recalibrates resource allocation. However, one of its key advantages is that it avoids unnecessary resource allocation, activating additional resources only when performance degrades. This advantage allows us to add this approach as a benchmark to compare network usage performance metrics.

End-users ABR Algorithms

We chose three ABR algorithms—Dynamic, Throughput, and Buffer-based—to represent different strategies in bitrate selection. The Dynamic algorithm is more adaptive and adjusts bitrates based on real-time network conditions, whereas the Throughput algorithm selects bitrates based on the available throughput. The Buffer-based algorithm prioritizes maintaining buffer stability by selecting lower bitrates to avoid playback stalls. These algorithms were chosen for their widespread use in modern streaming applications and their varying optimization goals, allowing us to test the robustness of our architecture under diverse conditions.

Experimental Scenarios

In Figure 6.5, to reflect realistic operating conditions, we tested the system in five load scenarios: Low Load (20-30 users), Moderate Load (30-40 users), and High Load (50-60 users). These scenarios introduced typical network network congestion to our emulated environments. AP groups cover specific regions where video content preferences follow a Zipfian distribution (with $\alpha = 2.0$), ensuring that content popularity distribution adheres to standard models. User requests follow a Poisson arrival process, with spatial distributions modeled uniformly over a 2.0x2.0 km area, which the APs fully cover. The scenario applied the Strongest Signal First (SSF) criterion for handover decisions, where users connect to the AP with the highest signal strength. To ensure the reliability of our results, each experiment is repeated 10 times.

Based on the methodology outlined for measuring server deployment latency in [93], Table 6.3 shows the latency results, calculated as the average of 100 repetitions. The table does not include the times to propagate and setup the snapshot updates, while 5.347 ± 0.585s and 0.204 ± 0.047s are the times for pull and deploy operations, respectively.

Load Forecasting

We performed experiments using the controlled environment to generate the datasets. We varied the latency between the cloud and the users as well. The initial dataset consists of 78 metrics. After applying the operations discussed in 6.3.1, the metrics were reduced to 33. The time series describing the generated metrics was obtained based on the availability

Table 6.3: Average and Standard Deviation of Pull and Deploy Times

Metric	Mean (s)	Standard Deviation (s)
Image Pull Time	5.347	0.585
Deploy Container Time	0.204	0.047

of container metrics, with sampling intervals of 10 seconds. Approximately a total of 120,000 samples were collected. The actual and predicted data of the services request load are available at `https://github.com/eduardogama/QoE-Forecasting-Dataset`.

The dataset was divided into training (70%), validation (20%), and test (10%) subsets. The model was trained using the training set, fine-tuned using the validation set, and evaluated on the test set. To determine the prediction time horizon T, we accounted for four key time delays: t_1, representing the TTL associated with the CSS; t_2, corresponding to the image pull operation; t_3, related to the image deployment process; and t_4, reflecting monitoring updates in the Database Repository. The cumulative delay, defined as $t_1 + t_2 + t_3 + t_4 < 27$ seconds, ensures maximum communication overhead among monitoring, planning, and user-facing entities. Exceeding this threshold introduces latency penalties in coordination. Consequently, we set the time window to 10 intervals and the prediction horizon to 30 seconds.

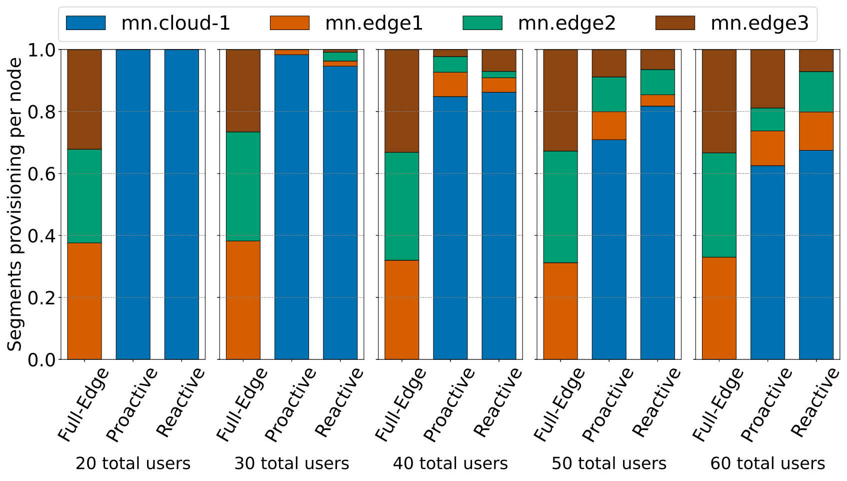
6.4.2 Results and Discussion

Figure 6.6 presents the distribution of video segments delivered by different nodes. In the network, across multiple nodes (mn.cloud-1, mn.edge1, mn.edge2, and mn.edge3) for different resource allocation approaches (Full-Edge, Proactive, and Reactive). The scenarios scale from 20 to 60 total users. Notice that the evaluation of the network load performance for these three possibilities will answer the main question: what is the impact of the proactive strategy (ML model) in the network performance compared to the reactive approach?

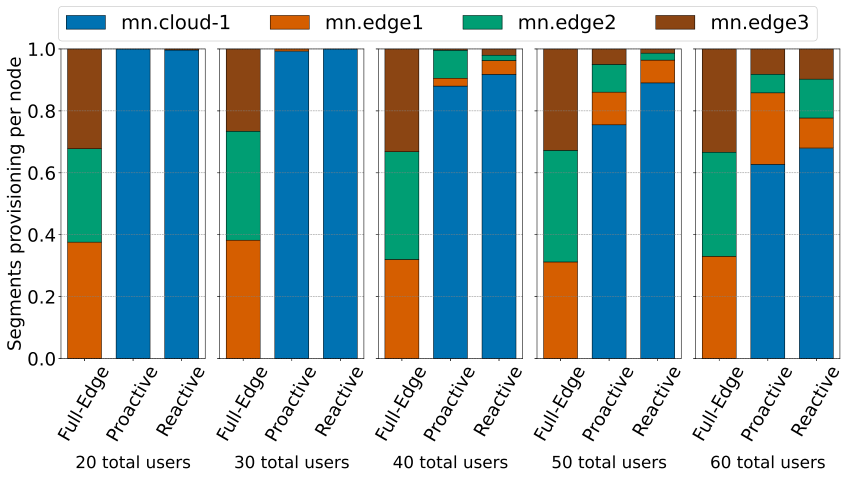
Figures 6.6(a), 6.6(b) and 6.6(c) show the segment distributions and different performance levels are observed when comparing the ABR algorithm results. The Buffer algorithm presents the worst results, while Dynamic and Throughput algorithms perform similarly for 60 users. For the throughput algorithm in the reactive approach, intermediate results indicate an efficient use of edge resources. However, proactive and reactive approaches exhibit similar performances for the Dynamic algorithm.

In the Proactive strategy, when the load reaches 30 users, it starts utilizing the edge nodes, handling approximately 10% of the total segments. As the load increases, the edge nodes are used more extensively, thereby offloading a substantial portion of the traffic from the cloud. With an increasing user count, the proactive approach maintains a balanced utilization, with about $30-40\%$ of segments handled by the edge nodes even for higher user numbers (e.g., $50-60$ users). This highlights how proactive provisioning can effectively leverage edge capabilities, ensuring minimized latency and optimal allocation of network resources.

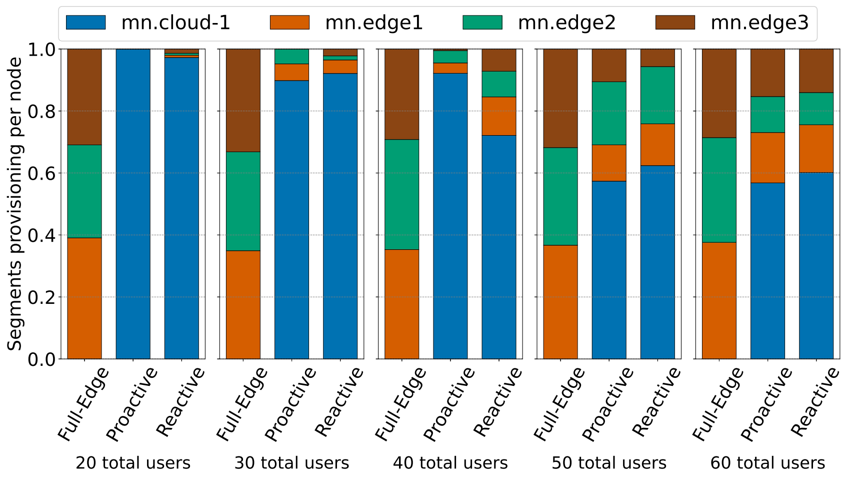
This analysis shows that the Proactive approach balances edge node utilization and



(a) Dynamic Algorithm



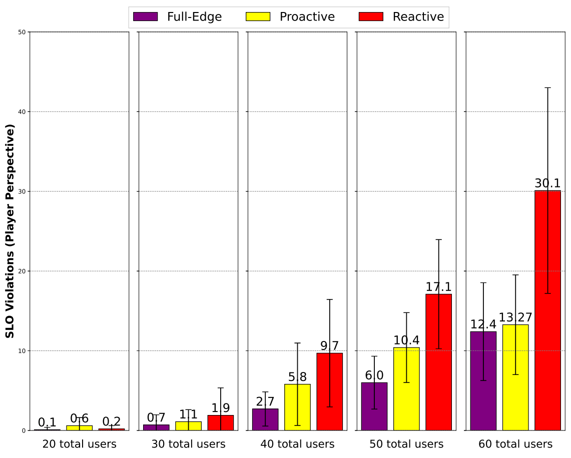
(b) Throughput algorithm



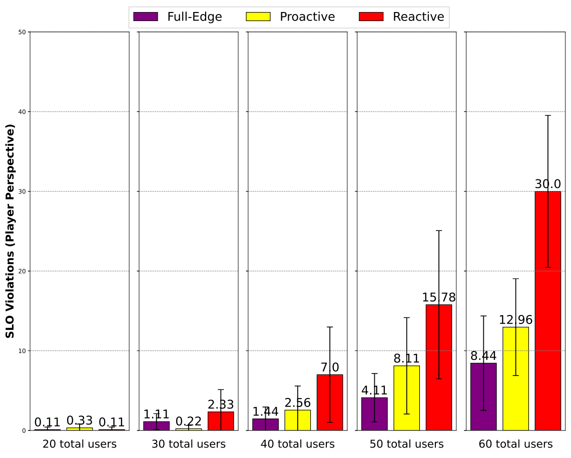
(c) Buffer algorithm

Figure 6.6: Segment distribution across cloud and edge nodes for varying user counts (20–60). Each bar group shows provisioning under different Planner strategies: Full-Edge, Reactive, and Proactive.

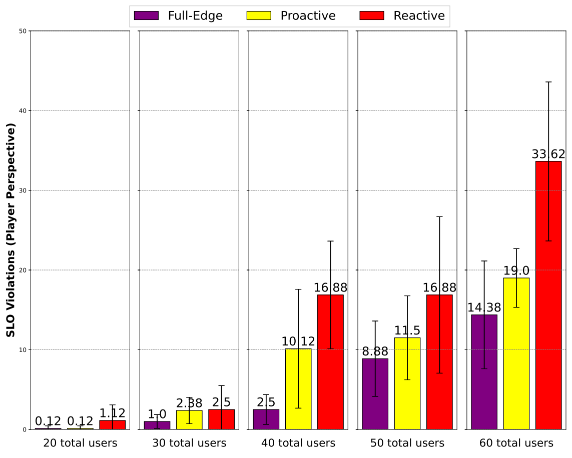
anticipates workload shifts, even compared to the Reactive approach, which only shifts the traffic after the SLO violation occurrences. In the Reactive, only when the SLO threshold is triggered are the edge nodes playing a more significant role in segment delivery. This approach serves as a baseline for comparative analyses, where edge nodes are activated to ensure users receive the video from the closest possible node. The proactive strategy matches the edge node utilization with the Baseline, indicating comparable traffic distribution between the strategies.



(a) Dynamic Algorithm



(b) Throughput algorithm



(c) Buffer algorithm

Figure 6.7: Comparison of SLO violations for different Planner algorithms: (a) Dynamic, (b) Throughput, and (c) Buffer, with varying user counts.

The experiments in Figure 6.7 demonstrate the architecture performance in managing SLO violations from a player perspective varying user loads. As the number of users increases, the architecture's ability to maintain SLOs becomes more challenging and the differences between the strategies become apparent. In this case, our goal is to answer the following questions: Can ML techniques effectively forecast future SLO violations using previous system data features?

Starting with the 20- and 30-user scenarios, all three approaches demonstrate very minimal SLO violations, with values close to zero, indicating that the cloud can effectively manage a low number of users. Overall, the Full-Edge approach consistently performs best (Baseline), providing the cache service as close to the users as possible. With up to 40 users, it maintains minimal SLO violations, indicating its strong capacity to handle increasing demand. Even as the user count grows to 60, the violations increase only moderately, with approximately 12. This stability shows how the behavior from the emulated scenario considers all available edge resources to distribute the load, preventing significant QoE degradation even as the system scales.

The Proactive approach performs nearly as well as Full-Edge in scenarios with 40 or fewer users when the confidence interval is considered. SLO violations remain low. Still, as the user load reaches 50 and 60, there is a notable rise in violations, with 6.3 at 50 and 13.6 at 60 while still competitive compared to Full-Edge, likely due to its ability to predict and scale resources quickly enough to prevent violations. The Reactive approach shows weaker performance, particularly as the number of users grows. The reactive nature of this approach makes it less effective in large-scale adaptive video streaming scenarios, as it cannot preemptively allocate resources to avoid QoE drops.

The Proactive approach shows significant advantages over the Reactive strategy, particularly in scenarios with moderate to high user loads. Anticipating demand and scaling resources before QoE degradation occurs reduces SLO violations by up to 48% compared to a Reactive method. Although the Full-Edge approach performs slightly better under extreme loads, Proactive offers a highly efficient alternative that balances resource optimization and QoE maintenance, making it a compelling choice for adaptive video streaming, especially in dynamic and growing user environments.

6.5 Chapter Conclusions

The proposed adaptive video streaming architecture effectively balances cloud and edge resource utilization across the Edge-Cloud Continuum, optimizing QoE under dynamic network conditions. By proactively anticipating demand shifts and scaling resources, the ML-driven Proactive approach reduced SLO violations by up to 48%. The results highlight that leveraging edge nodes under the proactive strategy offloaded around 40% of traffic from the cloud during peak loads, providing a balanced and efficient resource utilization compared to the reactive approach, which relied heavily on centralized resources. Moreover, while showing consistently high performance, the Full-Edge approach demonstrated that a more adaptive, proactive approach could achieve similar QoE benefits without needing constant resource over-provisioning.

The architecture's adaptability also enables seamless customization for different SLOs and QoE metrics, making it suitable for diverse deployment scenarios within the Edge-Cloud Continuum. This flexibility ensures that the system can dynamically adjust to a wide range of deployment scenarios within the Edge-Cloud Continuum. By incorporating different QoE-driven constraints, the architecture can be fine-tuned to meet the specific requirements of diverse applications, including live streaming, video-on-demand, and real-time interactive media services.

Chapter 7

Conclusions and Future Works

This chapter summarizes this thesis and discusses directions for future research. The objective is to highlight our main contributions and point out some possible directions to proceed with the research to address the drawbacks of the proposed solutions. In this context, we first present the thesis conclusions in Section 7.1. Then, in Section 7.2, we present future directions of this work. Finally, in Section 7.3, we present the publications related to this thesis.

7.1 Conclusions

- **Research Question 1:** How can HAS systems leverage CSS to optimize resource allocation across dynamic Edge-Cloud environments?
  Adaptive video streaming systems can optimize QoE, cache hit rates, and resource allocation in dynamic Edge-Cloud environments by leveraging CSS through modular architectures like AVENUE. These systems integrate a Monitoring Module to capture real-time metrics, such as user location and network conditions, and a Selector Module to assign edge servers using tailored algorithms dynamically. Each algorithm offers trade-offs, with CA excelling in cache optimization, RA prioritizing user satisfaction through proximity, and RR maintaining system scalability by balancing workloads. CSS dynamically optimizes resource utilization and video delivery performance by enabling real-time adaptability and low-latency decision-making.

- **Research Question 2:** Can ML techniques effectively forecast future SLO violations using previous system data features?
  The study uses an LSTM-based predictive model to forecast future aggregated QoE, which serves as the SLO in this context. The model is trained on a dataset comprising 78 metrics, which are reduced to 33 after preprocessing, covering aspects like CPU usage, memory consumption, and network performance. This data is collected from the cloud nodes at 10-second intervals. The LSTM model analyzes these historical data point sequences to learn patterns and correlations, enabling it to predict future QoE values. The effectiveness of this predictive capability is demonstrated in the proactive strategy's performance. By anticipating potential SLO violations based on historical data, the proactive approach allocates resources preemptively,

resulting in a 48% reduction in SLO violations compared to a reactive approach that only responds after violations occur. This result indicates the power of ML techniques in forecasting future issues and enabling proactive mitigation strategies.

- **Research Question 3:** What is the impact of the proactive strategy (ML model) in the network performance compared to the reactive approach?
  Using a ML model, the proactive strategy significantly reduces strain on the network compared to the reactive approach. The proactive strategy shifts around 40% of the traffic from the cloud to edge nodes during peak periods, balancing resource utilization more effectively than the reactive approach. The reactive approach only diverts traffic after an SLO violation, leading to greater reliance on centralized cloud resources. While the proactive and reactive approaches experience minimal SLO violations with low user counts (20-30), the proactive strategy strengthens as the user load increases. It anticipates demand shifts and scales resources accordingly, resulting in fewer SLO violations than the reactive approach. Notably, the proactive approach achieves a 48% reduction in SLO violations compared to the reactive method, highlighting its effectiveness in maintaining the QoE for users, especially in dynamic environments.

7.2 Future Works

Throughout this Ph.D. thesis, a study and an architecture were developed to enhance the users' QoE in adaptive streaming services. Despite these efforts, the video streaming domain continues to face emerging challenges that demand greater focus from network and service management viewpoints. This section highlights these challenges and explores the potential opportunities they present.

7.2.1 Immersive Video Streaming

Immersive streaming technologies like VR and AR are transforming how we consume media, with content like point clouds and 360° video offering interactive and engaging experiences. High-quality QoE for VR streaming requires high-resolution content, fast data transfer, and ultra-low latency. However, VR streaming struggles with bandwidth demands, limiting QoE. Viewport-dependent techniques address this by focusing on the part of the content the user views, dynamically adjusting resolution and bitrate. This reduces bandwidth use while maintaining quality. For example, lower resolution can be applied to peripheral areas without affecting the user experience.

7.2.2 Sustainable HAS System

Video streaming is widely used and energy-demanding, contributing to carbon emissions and environmental issues. In this way, sustainable HAS systems focus on delivering high-quality video experiences while reducing resource consumption and environmental impact. With video streaming accounting for a significant portion of global internet traffic, traditional video streaming services are widely used and high-demanding at the cost of energy consumption, straining data centers, CDN nodes, and edge infrastructure. Sustainable solutions address this by integrating intelligent resource allocation, energy-efficient practices, and adaptive scaling to optimize video delivery and reduce carbon footprints [94,95].

Sustainable HAS systems also embrace renewable energy for powering infrastructure and employ session-aware strategies to prioritize active streams. These systems align video streaming with environmental goals by recycling inactive sessions and optimizing CDNs to minimize redundant data replication. This approach ensures that streaming services can scale to meet user demand while contributing to a more sustainable and eco-friendly digital future.

7.2.3 Learning HAS System

Learning-based HAS systems leverage advanced techniques such as Reinforcement Learning (RL) and Large Language Models (LLM) to optimize video streaming dynamically [96]. Traditional HAS relies on predefined algorithms for bitrate adaptation and resource allocation, which can struggle to adapt to complex, real-time changes in network conditions and user behavior. RL-based systems, however, learn optimal strategies through trial and error, continuously improving their decisions to maximize QoE while efficiently utilizing resources.

7.2.4 Emerging Protocols

QUIC, a cutting-edge transport protocol, offers significant advantages over traditional TCP. By operating over User Datagram Protocol (UDP), QUIC excels in high-latency and lossy network conditions, which is crucial for modern internet applications. Its user-space implementation simplifies deployment and updates, while 0-RTT connection establishment reduces latency. True multiplexing further enhances performance by avoiding head-of-line blocking. Despite these benefits, integrating QUIC into Adaptive Video Streaming solutions remains an under-explored area [26, 27]. In addition, the WebRTC technology stack empowers direct, plugin-free audio and video communication between browsers [97]. Its low latency and high-quality streaming capabilities make it well-suited for real-time applications like video conferencing and live streaming. However, the potential of combining WebRTC with HAS-based methodologies to optimize video streaming performance over the Internet is yet to be fully realized.

7.2.5 Rights Management and Content Security

Integrating Digital Rights Management (DRM) and blockchain technologies enhances security, transparency, and trust in adaptive video streaming systems. DRM ensures controlled access to copyrighted content, but its effectiveness could be amplified by leveraging blockchain for decentralized and tamper-proof rights management [98, 99]. Blockchain could provide a secure and transparent ledger for tracking content ownership, distribution, and usage, ensuring that rights holders are fairly compensated while preventing unauthorized access. Smart contracts could automate licensing and payment processes, reducing administrative overhead and streamlining stakeholder transactions. Additionally, blockchain could be used to verify the integrity of video streams, ensuring that users receive authentic and unaltered content. Combining DRM with blockchain presents opportunities for creating robust, scalable, and fair content delivery ecosystems that address challenges related to piracy, accountability, and user trust in video streaming services.

7.2.6 Satellites

Incorporating Low Earth Orbit (LEO) satellites into Adaptive Video Streaming systems, leveraging their global coverage and high-bandwidth capabilities to address challenges in remote or underserved regions. Satellites could enhance content delivery by acting as supplementary nodes in hybrid Edge-Cloud architectures, enabling seamless streaming even in areas with limited terrestrial infrastructure. Research could focus on optimizing latency, bandwidth allocation, and handover mechanisms between satellite and ground networks to maintain high QoE. Additionally, incorporating ML models to predict satellite link performance and adapt streaming parameters dynamically could further improve reliability and efficiency. By investigating these aspects, the use of satellites could open new opportunities for scalable and resilient video streaming solutions in diverse and challenging environments [100–102].

7.3 Scientific Production

1. E. S. Gama, R. Rodrigues-Filho, E. M. Madeira, R. Immich, and L. F. Bittencourt. Learning-Based Adaptive Video Streaming Architecture for QoE Forecasting on the Edge-Cloud Continuum. IEEE Transactions on Cloud Computing, 2025. **Under review.**

2. Williams, A.; Yadhav, S. V.; Rodrigues-Filho, R.; Gama, E. S.; Madeira, E. R. M.; Bittencourt, L. F.. Handling a Quality of Experience Model. 2023, Brasil. Patente: Privilégio de Inovação. Número do registro: 12564588, título: "Handling a Quality of Experience Model" , Instituição de registro: WIPO - World Intellectual Property Organization.

3. [2] Roberto Rodrigues-Filho, Eduardo S Gama, Marcio Miranda Assis, Roger Immich, and Edmundo Madeira. Content steering: Leveraging the computing continuum to support adaptive video streaming. pages 1–39, 2024.

4. [103] E. S. Gama, R. Rodrigues-Filho, E. M. Madeira, R. Immich, and L. F. Bittencourt. Enabling adaptive video streaming via content steering on the edge-cloud continuum. In 2024 IEEE 8th International Conference on Fog and Edge Computing (ICFEC), pages 35–42, Los Alamitos, CA, USA, may 2024. IEEE Computer Society.

5. [35] Eduardo S. Gama, Natesha B V, Roger Immich, and Luiz F. Bittencourt. An orchestrator architecture for multi-tier edge/cloud video streaming services. In 2023

IEEE International Conference on Edge Computing and Communications (EDGE), pages 190–196, 2023

6. [32] Eduardo S. Gama, Lucas Otávio N. De Araújo, Roger Immich, and Luiz F. Bittencourt. Video streaming analysis in multi-tier edge-cloud networks. In 2021 8th International Conference on Future Internet of Things and Cloud (FiCloud), pages 19–25, 2021.

7. [104]Pedro F. do Prado, Maycon L. M. Peixoto, Marcelo C. Araújo, Eduardo S. Gama, Diogo M. Gonçalves, Matteus V. S. Silva, Roger Immich, Edmundo R. M. Madeira, and Luiz F. Bittencourt. Mobile Edge Computing for Content Distribution and Mobility Support in Smart Cities, pages 473–500. Springer International Publishing, Cham, 2021.

8. [105] F. Pisani, F. de Oliveira, E. S. Gama, R. Immich, L. F. Bittencourt, and E. Borin. Fog computing on constrained devices: Paving the way for the future iot. Advances in Edge Computing: Massive Parallel Processing and Applications, 35:22, 2020.

9. [69] E. S. Gama, R. Immich, and L. F. Bittencourt. Towards a multi-tier fog/cloud architecture for video streaming. In 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), pages 13–14, Dec 2018

10. [106] Carlos A. Astudillo, Tiago P.C. de Andrade, Eduardo S. Gama, Luiz F. Bittencourt, Leandro A. Villas, Edmundo R.M. Madeira, and Nelson L.S. da Fonseca. Internet of things for environmental monitoring based on radio over fiber. In 2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI), pages 1–6, 2018.

11. [107] Paulo Marques, Alexandre P. do Carmo, Valerio Frascolla, Carlos Silva, Emanuel D. R. Sena, Raphael Braga, João Pinheiro, Carlos A. Astudillo, Tiago P. C. de An- drade, Eduardo S. Gama, Luiz F. Bittencourt, Leandro A. Villas, Edmundo R. M. Madeira, Nelson L. S. da Fonseca, Cristiano Both, Gabriel Lando, Matias Schimu- neck, Juliano Wickboldt, Ana P. V. Trevisan, Rafael de Jesus Martins, Raquel F. Vassallo, Felippe M. de Queiroz, Rodolfo Picoreti, Roberta L. Gomes, Cristina K. Dominicini, Víctor García, Rafael S. Guimarães, Rodolfo Villaca, Magnos Martinello, Moises R.N. Ribeiro, Daniel F. Macedo, Vinicius F. Silva, Julio C. T. Guimarães, Carlos Colman-Meixner, Reza Nejabati, Dimitra Simeonidou, Yi Zhang, Frank Slyne, Pedro Alvarez, Diarmuid Collins, Marco Ruffini, Luiz A. DaSilva, and Johann M. Marquez-Barja. Optical and wireless network convergence in 5g systems – an exper- imental approach. In 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), pages 1–5, 2018.

Bibliography

[1] The global internet phenomena report. White Paper, 2024. Accessed 08-abr-2024.

[2] Roberto Rodrigues-Filho, Eduardo S Gama, Marcio Miranda Assis, Roger Immich, and Edmundo Madeira. Content steering: Leveraging the computing continuum to support adaptive video streaming. pages 1–39, 2024.

[3] Denis Rosário, Matias Schimuneck, João Camargo, Jéferson Nobre, Cristiano Both, Juergen Rochol, and Mario Gerla. Service migration from cloud to multi-tier fog nodes for multimedia dissemination with qoe support. *Sensors*, 18(2), 2018.

[4] Xiangbo Li, Mahmoud Darwich, Mohsen Amini Salehi, and Magdy Bayoumi. Chapter four - a survey on cloud-based video streaming services. volume 123 of *Advances in Computers*, pages 193–244. Elsevier, 2021.

[5] Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann. A survey on bitrate adaptation schemes for streaming media over http. *IEEE Communications Surveys & Tutorials*, 21(1):562–585, 2019.

[6] Reza Kalan and Ismail Dulger. A survey on qoe management schemes for http adaptive video streaming: Challenges, solutions, and opportunities. *IEEE Access*, 12:170803–170839, 2024.

[7] Alcardo Alex Barakabitze, Nabajeet Barman, Arslan Ahmad, Saman Zadtootaghaj, Lingfen Sun, Maria G. Martini, and Luigi Atzori. Qoe management of multimedia streaming services in future networks: A tutorial and survey. *IEEE Communications Surveys Tutorials*, 22(1):526–565, 2020.

[8] Roger Immich, Eduardo Cerqueira, and Marilia Curado. Efficient high-resolution video delivery over vanets. *Wireless Networks*, Feb 2018.

[9] Mpeg-dash vs. apple hls vs. microsoft smooth streaming vs. adobe hds, 2015. Accessed on December 17, 2024.

[10] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili. Collaborative multi-bitrate video caching and processing in mobile-edge computing networks. In *2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pages 165–172, Feb 2017.

[11] Z. Ye, F. D. Pellegrini, R. El-Azouzi, L. Maggi, and T. Jimenez. Quality-aware dash video caching schemes at mobile edge. In *2017 29th International Teletraffic Congress (ITC 29)*, volume 1, pages 205–213, Sept 2017.

[12] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck. Optimal vnfs placement in cdn slicing over multi-cloud environment. *IEEE Journal on Selected Areas in Communications*, 36(3):616–627, March 2018.

[13] Guowei Zhu and Weixi Gu. User mapping strategy in multi-cdn streaming: A data-driven approach. *IEEE Internet of Things Journal*, 9(9):6638–6649, 2022.

[14] Miran Taha and Aree Ali. Redirection and protocol mechanisms in content delivery network-edge servers for adaptive video streaming. *Applied Sciences*, 13(9), 2023.

[15] Ehab Ghabashneh and Sanjay Rao. Exploring the interplay between cdn caching and video streaming performance. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 516–525, 2020.

[16] Tomasz Lyko, Matthew Broadbent, Nicholas Race, Mike Nilsson, Paul Farrow, and Steve Appleby. Improving quality of experience in adaptive low latency live streaming. *Multimedia Tools and Applications*, 83(6):15957–15983, Feb 2024.

[17] Xiaoxi Zhang, Haoran Xu, Longhao Zou, Jingpu Duan, Chuan Wu, Yali Xue, Zuozhou Chen, and Xu Chen. Rosevin: Employing resource- and rate-adaptive edge super-resolution for video streaming. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, pages 491–500, 2024.

[18] Marcos Carvalho and Daniel Fernandes Macedo. Container scheduling in co-located environments using qoe awareness. *IEEE Transactions on Network and Service Management*, 20(3):3247–3260, 2023.

[19] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 20(1):416–464, Firstquarter 2018.

[20] Michael C. Thornburgh. Adobe's RTMFP Profile for Flash Communication. RFC 7425, December 2014.

[21] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, July 2003.

[22] Iraj Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia*, 18(4):62–67, 2011.

[23] Roger Pantos and William May. HTTP Live Streaming. RFC 8216, August 2017.

[24] Mpeg-dash (dynamic adaptive streaming over http), 2022. Accessed 08-abr-2024.

[25] Wesley Eddy. Transmission Control Protocol (TCP). RFC 9293, August 2022.

[26] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021.

[27] Michael Seufert, Raimund Schatz, Nikolas Wehner, and Pedro Casas. Quicker or not? an empirical analysis of quic vs tcp for video streaming qoe provisioning. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 7–12, 2019.

[28] Michaela Iorga, Larry Feldman, Robert Barton, Michael J. Martin, Nedim Goren, and Charif Mahmoudi. Fog Computing Conceptual Model - Recommendations of the National Institute of Standards and Technology. Technical report, National Institute of Standards and Technology, March 2018.

[29] Luiz Bittencourt, Roger Immich, Rizos Sakellariou, Nelson Fonseca, Edmundo Madeira, Marilia Curado, Leandro Villas, Luiz DaSilva, Craig Lee, and Omer Rana. The internet of things, fog and cloud continuum: Integration and challenges. *Internet of Things*, 3-4:134 – 155, 2018.

[30] Mukaddim Pathan, Ramesh K. Sitaraman, and Dom Robinson. *Advanced Content Delivery, Streaming, and Cloud Services*. Wiley Publishing, 1st edition, 2014.

[31] Charif Mahmoudi, Fabrice Mourlin, and Abdella Battou. Formal definition of edge computing: An emphasis on mobile cloud and iot composition. In *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 34–42, 2018.

[32] Eduardo S. Gama, Lucas Otávio N. De Araújo, Roger Immich, and Luiz F. Bittencourt. Video streaming analysis in multi-tier edge-cloud networks. In *2021 8th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 19–25, 2021.

[33] Reza Farahani, Mohammad Shojafar, Christian Timmerer, Farzad Tashtarian, Mohammad Ghanbari, and Hermann Hellwagner. Ararat: A collaborative edge-assisted framework for http adaptive video streaming. *IEEE Transactions on Network and Service Management*, 20(1):625–643, 2023.

[34] Jun Du, Chunxiao Jiang, Abderrahim Benslimane, Song Guo, and Yong Ren. Sdn-based resource allocation in edge and cloud computing systems: An evolutionary stackelberg differential game approach. *IEEE/ACM Transactions on Networking*, 30(4):1613–1628, 2022.

[35] Eduardo S. Gama, Natesha B V, Roger Immich, and Luiz F. Bittencourt. An orchestrator architecture for multi-tier edge/cloud video streaming services. In *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, pages 190–196, 2023.

[36] Farzad Tashtarian and Christian Timmerer. Revision: A roadmap on adaptive video streaming optimization, 2024.

[37] Yuriy Reznik, Guillem Cabrera, Daniel Silhavy, Stefan Pham, Alex Giladi, Alex Balk, Ali C. Begen, and Will Law. Content steering: a standard for multi-cdn streaming. In *Proceedings of the 3rd Mile-High Video Conference*, MHV '24, page 128, New York, NY, USA, 2024. Association for Computing Machinery.

[38] Burak Kara and Gwendal Simon. Dynamic content steering services: How to make everyone happy in a multi-cdn world. In *Proceedings of the 3rd Mile-High Video Conference*, MHV '24, page 95, New York, NY, USA, 2024. Association for Computing Machinery.

[39] Ron Zekarias Bo Zhang Biswa Panigrahi Nabajeet Barman Stuart Hicks Ted Krofssik Andrew Sinclair Adam Waldron Yuriy Reznik, Guillem Cabrera. IMPLEMENTING HLS/DASH CONTENT STEERING AT SCALE. Technical paper, IBC 2023, 2023.

[40] D. Silhavy S Pham A. Giladi A. Balk A. Begen Y. Reznik, G. Cabrera and W. Law. CONTENT STEERING: A STANDARD FOR MULTI-CDN STREAMING. Technical paper, IBC 2024, 2024.

[41] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys Tutorials*, 17(1):469–492, 2015.

[42] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner. Assessing effect sizes of influence factors towards a qoe model for http adaptive streaming. In *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 111–116, Sep. 2014.

[43] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[44] Zhengfang Duanmu, Kai Zeng, Kede Ma, Abdul Rehman, and Zhou Wang. A quality-of-experience index for streaming video. *IEEE Journal of Selected Topics in Signal Processing*, 11(1):154–166, 2017.

[45] Reza Rassool. Vmaf reproducibility: Validating a perceptual practical video quality metric. In *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–2, 2017.

[46] Robert C. Streijl, Stefan Winkler, and David S. Hands. Mean opinion score (mos) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2):213–227, 2016.

[47] Xiongkuo Min, Huiyu Duan, Wei Sun, Yucheng Zhu, and Guangtao Zhai. Perceptual video quality assessment: A survey, 2024.

[48] Weiwen Zhang, Yonggang Wen, Zhenzhong Chen, and Ashish Khisti. Qoe-driven cache management for http adaptive bit rate streaming over wireless networks. *IEEE Transactions on Multimedia*, 15(6):1431–1445, 2013.

[49] Peter Reichl, Bruno Tuffin, and Raimund Schatz. Logarithmic laws in service quality perception: Where microeconomics meets psychophysics and quality of experience. *Telecommun. Syst.*, 52(2):587–600, February 2013.

[50] Ehab Ghabashneh and Sanjay Rao. Exploring the interplay between cdn caching and video streaming performance. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 516–525, 2020.

[51] Reza Farahani, Farzad Tashtarian, Alireza Erfanian, Christian Timmerer, Mohammad Ghanbari, and Hermann Hellwagner. Es-has: An edge- and sdn-assisted framework for http adaptive video streaming. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '21, page 50–57, New York, NY, USA, 2021. Association for Computing Machinery.

[52] Fillipe Santos, Roger Immich, and Edmundo Madeira. Multimedia microservice placement in hierarchical multi-tier cloud-to-fog networks. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1044–1049, 2021.

[53] Hugo Santos, Derian Alencar, Rodolfo Meneguette, Denis Rosário, Jéferson Nobre, Cristiano Both, Eduardo Cerqueira, and Torsten Braun. A multi-tier fog content orchestrator mechanism with quality of experience support. *Computer Networks*, 177:107288, 2020.

[54] Yu Guan, Xinggong Zhang, and Zongming Guo. Caca: Learning-based content-aware cache admission for video content in edge caching. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, pages 456–464, New York, NY, USA, 2019. ACM.

[55] Jesús Aguilar-Armijo, Christian Timmerer, and Hermann Hellwagner. Space: Segment prefetching and caching at the edge for adaptive video streaming. *IEEE Access*, 11:21783–21798, 2023.

[56] Fillipe Santos, Roger Immich, and Edmundo R.M. Madeira. Multimedia services placement algorithm for cloud–fog hierarchical environments. *Computer Communications*, 191:78–91, 2022.

[57] Jueun Jeon, Sihyun Park, Byeonghui Jeong, and Young-Sik Jeong. Efficient container scheduling with hybrid deep learning model for improved service reliability in cloud computing. *IEEE Access*, 12:65166–65177, 2024.

[58] Muhammad Jalal Khan, Abdelhak Bentaleb, and Saad Harous. Can accurate future bandwidth prediction improve volumetric video streaming experience? In *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1041–1047, 2021.

[59] Wanxin Shi, Qing Li, Ruishan Zhang, Gengbiao Shen, Yong Jiang, Zhenhui Yuan, and Gabriel-Miro Muntean. *QoE Ready to Respond: A QoE-Aware MEC Selection Scheme for DASH-Based Adaptive Video Streaming to Mobile Users*, page 4016–4024. Association for Computing Machinery, New York, NY, USA, 2021.

[60] Minh Nguyen, Christian Timmerer, and Hermann Hellwagner. H2br: An http/2-based retransmission technique to improve the qoe of adaptive video streaming. In *Proceedings of the 25th ACM Workshop on Packet Video*, PV '20, page 1–7, New York, NY, USA, 2020. Association for Computing Machinery.

[61] Yinxin Li, Haiyan Tu, Guorong Zhou, Ting Li, Yunfeng Wang, Kai Liang, Zhigang Wang, and Liqiang Zhao. Design and implementation of adaptive-bitrate-streaming-based edge caching. In *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, pages 1–5, 2022.

[62] Yingwen Chen, Hujie Yu, Bowen Hu, Zhimin Duan, and Guangtao Xue. An edge caching strategy based on user speed and content popularity for mobile video streaming. *Electronics*, 10(18), 2021.

[63] Ninghao Chen, Weiwei Xing, Di Zhang, Min Guo, and Limin Gao. Multi-bitrate video caching and processing in edge computing: A stackelberg game approach. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[64] Kai Xu, Xiang Li, Sanjay Kumar Bose, and Gangxiang Shen. Joint replica server placement, content caching, and request load assignment in content delivery networks. *IEEE Access*, 6:17968–17981, 2018.

[65] Haolin Liu, Xiaoling Long, Zhetao Li, Saiqin Long, Rong Ran, and Hui-Ming Wang. Joint optimization of request assignment and computing resource allocation in multi-access edge computing. *IEEE Transactions on Services Computing*, 16(2):1254–1267, 2023.

[66] Abdelhak Bentaleb, Ali C. Begen, Roger Zimmermann, and Saad Harous. Sdnhas: An sdn-enabled architecture to optimize qoe in http adaptive streaming. *Trans. Multi.*, 19(10):2136–2151, October 2017.

[67] Zhilong Zhang, Danpu Liu, and Yaxiong Yuan. Layered hierarchical caching for svc-based http adaptive streaming over c-ran. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2017.

[68] Helle Sjøvaag, Ragnhild Kr. Olsen, and Raul Ferrer-Conill. Delivering content: Modular broadcasting technology and the role of content delivery networks. *Telecommun. Policy*, 48(4), July 2024.

[69] E. S. Gama, R. Immich, and L. F. Bittencourt. Towards a multi-tier fog/cloud architecture for video streaming. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 13–14, Dec 2018.

[70] Omer Rana, Manjerhussain Shaikh, Muhammad Ali, Ashiq Anjum, and Luiz Bittencourt. Vertical workflows: Service orchestration across cloud & edge resources. In *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 355–362. IEEE, 2018.

[71] Christian Kreuzberger, Daniel Posch, and Hermann Hellwagner. Amust framework - adaptive multimedia streaming simulation framework for ns-3 and ndnsim, 2016.

[72] C. Mueller, S. Lederer, J. Poecher, and C. Timmerer. Demo paper: Libdash - an open source software library for the mpeg-dash standard. In *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–2, July 2013.

[73] Per-title encode optimization, 2015. ; accessed 20-novembro-2019.

[74] Content steering for dash, 2022. Accessed 28-oct-2023.

[75] Roger Pantos. HTTP Live Streaming 2nd Edition. Internet-Draft draft-pantos-hls-rfc8216bis-14, Internet Engineering Task Force, November 2023. Work in Progress.

[76] Daniel Silhavy, Will Law, Stefan Pham, Ali C. Begen, Alex Giladi, and Alex Balk. Dynamic cdn switching - dash-if content steering in dash.js. In *Proceedings of the 2nd Mile-High Video Conference*, MHV '23, page 130–131, New York, NY, USA, 2023. Association for Computing Machinery.

[77] Bruce M. Maggs and Ramesh K. Sitaraman. Algorithmic nuggets in content delivery. *SIGCOMM Comput. Commun. Rev.*, 45(3):52–66, jul 2015.

[78] Babak Taraghi, Hadi Amirpour, and Christian Timmerer. Multi-codec ultra high definition 8k mpeg-dash dataset. In *Proceedings of the 13th ACM Multimedia Systems Conference*, MMSys '22, page 216–220, New York, NY, USA, 2022. Association for Computing Machinery.

[79] Log distance path loss or log-normal shadowing model. `https://www.gaussianwaves.com/2013/09/log-distance-path-loss-or-log-normal-shadowing-model/`. Accessed on December 17, 2023.

[80] Afzal Badshah, Anwar Ghani, Shahaboddin Shamshirband, Giuseppe Aceto, and Antonio Pescapè. Performance-based service-level agreement in cloud computing to optimise penalties and revenue. *IET Communications*, 14(7):1102–1112, April 2020.

[81] Sehrish Nadeem, Noor ul Amin, Sardar Khaliq uz Zaman, Muhammad Amir Khan, Zulfiqar Ahmad, Jawaid Iqbal, Ajab Khan, Abeer D. Algarni, and Hela Elmannai. Runtime management of service level agreements through proactive resource provisioning for a cloud environment. *Electronics*, 12(2), 2023.

[82] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Qoe-driven resource allocation for massive video distribution. *Ad Hoc Networks*, 89:170–176, 2019.

[83] Jesus Arellano-Uson, Eduardo Magaña, Daniel Morato, and Mikel Izal. Survey on quality of experience evaluation for cloud-based interactive applications. *Applied Sciences*, 14(5), 2024.

[84] Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadegh. Were rnns all we needed?, 2024.

[85] Apache http server project. `https://httpd.apache.org/`. Accessed on October 15, 2024.

[86] Dash javascript player v4.7.4. `https://reference.dashif.org/dash.js/v4.7.4/samples/dash-if-reference-player/index.html`. Accessed on October 15, 2024.

[87] Tensorflow. `https://www.tensorflow.org/`. Accessed on October 15, 2024.

[88] Keras: Deep learning for humans. `https://keras.io/`. Accessed on October 15, 2024.

[89] Batyr Charyyev, Engin Arslan, and Mehmet Hadi Gunes. Latency comparison of cloud datacenters and edge servers. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, 2020.

[90] Containernet: Mininet fork that allows to use docker containers as hosts in emulated networks. `https://github.com/ramonfontes/containernet/`. Accessed on October 15, 2024.

[91] Docker inside docker. `https://medium.com/@shivam77kushwah/docker-inside-docker-e0483c51cc2c`. Accessed on October 15, 2024.

[92] Wireless network emulation with mininet-wifi. `https://mininet-wifi.github.io/`. Accessed on October 15, 2024.

[93] Lorenzo Giorgi, Carlo Puliafito, Antonio Virdis, and Enzo Mingozzi. Service continuity in edge computing through edge proxies and http alternative services. *IEEE Open Journal of the Communications Society*, pages 1–1, 2024.

[94] Burak Kara, Gwendal Simon, Bruno Tuffin, Jerome Vieron, and Ali C. Begen. Studying green video distribution as a whole. In *Proceedings of the First International Workshop on Green Multimedia Systems*, GMSys '23, page 7–9, New York, NY, USA, 2023. Association for Computing Machinery.

[95] Robert Seeliger, Stefan Pham, and Stefan Arbanowski. End-to-end optimizations for green streaming. In *Proceedings of the First International Workshop on Green Multimedia Systems*, GMSys '23, page 10–12, New York, NY, USA, 2023. Association for Computing Machinery.

[96] Zhiyuan He, Aashish Gottipati, Lili Qiu, Xufang Luo, Kenuo Xu, Yuqing Yang, and Francis Y. Yan. Designing network algorithms via large language models, 2024.

[97] Yueheng Li, Hao Chen, Bowei Xu, Zicheng Zhang, and Zhan Ma. Improving adaptive real-time video communication via cross-layer optimization. *IEEE Transactions on Multimedia*, 26:5369–5382, 2024.

[98] Yan Li and Zheng Wan. Blockchain-enabled intelligent video caching and transcoding in clustered mec networks. *Security and Communication Networks*, 2021(1):7443260, 2021.

[99] Noor Adil Abdel Moneim and Methaq Talib Gaata. Video data authentication using a smart contract published on the blockchain. In *2022 Fifth College of Science International Conference of Recent Trends in Information Technology (CSCTIT)*, pages 93–99, 2022.

[100] Rohan Bose, Saeed Fadaei, Nitinder Mohan, Mohamed Kassem, Nishanth Sastry, and Jörg Ott. It's a bird? it's a plane? it's cdn!: Investigating content delivery networks in the leo satellite networks era. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, HotNets '24, page 1–9, New York, NY, USA, 2024. Association for Computing Machinery.

[101] Chao Wang, Yiran Zhang, Qing Li, Ao Zhou, and Shangguang Wang. Satellite computing: A case study of cloud-native satellites. In *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, pages 262–270, 2023.

[102] Matthieu Petrou, David Pradas, Mickaël Royer, and Emmanuel Lochin. Unveiling youtube qoe over satcom using deep-learning. *IEEE Access*, 12:39978–39994, 2024.

[103] E. S. Gama, R. Rodrigues-Filho, E. M. Madeira, R. Immich, and L. F. Bittencourt. Enabling adaptive video streaming via content steering on the edge-cloud continuum. In *2024 IEEE 8th International Conference on Fog and Edge Computing (ICFEC)*, pages 35–42, Los Alamitos, CA, USA, may 2024. IEEE Computer Society.

[104] Pedro F. do Prado, Maycon L. M. Peixoto, Marcelo C. Araújo, Eduardo S. Gama, Diogo M. Gonçalves, Matteus V. S. Silva, Roger Immich, Edmundo R. M. Madeira, and Luiz F. Bittencourt. *Mobile Edge Computing for Content Distribution and Mobility Support in Smart Cities*, pages 473–500. Springer International Publishing, Cham, 2021.

[105] Flavia Pisani, Fabiola de Oliveira, Eduardo S Gama, Roger Immich, Luiz F Bittencourt, and Edson Borin. Fog computing on constrained devices: Paving the way for the future iot. *Advances in Edge Computing: Massive Parallel Processing and Applications*, 35:22, 2020.

[106] Carlos A. Astudillo, Tiago P.C. de Andrade, Eduardo S. Gama, Luiz F. Bittencourt, Leandro A. Villas, Edmundo R.M. Madeira, and Nelson L.S. da Fonseca. Internet

of things for environmental monitoring based on radio over fiber. In *2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI)*, pages 1–6, 2018.

[107] Paulo Marques, Alexandre P. do Carmo, Valerio Frascolla, Carlos Silva, Emanuel D. R. Sena, Raphael Braga, João Pinheiro, Carlos A. Astudillo, Tiago P. C. de Andrade, Eduardo S. Gama, Luiz F. Bittencourt, Leandro A. Villas, Edmundo R. M. Madeira, Nelson L. S. da Fonseca, Cristiano Both, Gabriel Lando, Matias Schimuneck, Juliano Wickboldt, Ana P. V. Trevisan, Rafael de Jesus Martins, Raquel F. Vassallo, Felippe M. de Queiroz, Rodolfo Picoreti, Roberta L. Gomes, Cristina K. Dominicini, Víctor García, Rafael S. Guimarães, Rodolfo Villaca, Magnos Martinello, Moises R.N. Ribeiro, Daniel F. Macedo, Vinicius F. Silva, Julio C. T. Guimarães, Carlos Colman-Meixner, Reza Nejabati, Dimitra Simeonidou, Yi Zhang, Frank Slyne, Pedro Alvarez, Diarmuid Collins, Marco Ruffini, Luiz A. DaSilva, and Johann M. Marquez-Barja. Optical and wireless network convergence in 5g systems – an experimental approach. In *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–5, 2018.