Preprint. Cite as: *Eduardo S. Gama, Roberto Rodrigues Filho, Roger Immich, Luiz Fernando Bittencourt "Enabling Adaptive Video Streaming via Content Steering on the Edge-Cloud Continuum" In: Proceedings of IEEE ICFEC, 2024" https://doi.org/10.1109/ICFEC61590.2024.00018*

Enabling Adaptive Video Streaming via Content Steering on the Edge-Cloud Continuum

Eduardo S. Gama†, Roberto Rodrigues-Filho¶, Edmundo R. M. Madeira†, Roger Immich§, and Luiz F. Bittencourt†
†Institute of Computing - State University of Campinas (UNICAMP), Brazil
§Federal University of Rio Grande do Norte (UFRN), Brazil
¶Federal University of Santa Catarina (UFSC), Brazil
eduardogama@lrc.ic.unicamp.br, roberto.filho@ufsc.br, roger@imd.ufrn.br, and {edmundo, bit}@ic.unicamp.br

***Abstract*—One key challenge in Adaptive Video Streaming is the ever-changing edge network conditions at the last mile of access networks. The edge environment is particularly dynamic, influenced by user locations, fluctuation in resource demands and resource capabilities, in contrast to traditional Content Delivery Network (CDN) setups, where content routing decisions are relatively known. To address the dynamism of edge computing environments and to enable applications to better exploit edge-cloud computing resources, this article focuses on content steering technology, a recent addition to adaptive video protocols such as HLS and DASH. We present AVENUE as an architecture for Content Steering Services to orchestrate video delivery dynamically across the Edge-Cloud Continuum. This work designs the principles of the Content Steering Service to create a mechanism that involves two modules - monitoring and selector: The monitoring module captures real-time context metrics, and the selector module chooses an edge server according to the Select Server Algorithm. Our study addresses three steering algorithms with different performance profiles. Numerical results demonstrate that different configurations may yield varying network performance in terms of Quality of Experience (QoE), cache hits, and request load. Moreover, the appropriate selection of heuristics in the Selector module can also have a significant impact, depending on the metric being evaluated.**

***Index Terms*—Adaptive Video Streaming, Edge Computing, Cloud Computing, Edge-Cloud Continuum, Content Steering, Quality of Experience**

I. Introduction

Adaptive video streaming applications often utilize multiple CDNs to deliver end users' content efficiently. 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 streaming systems [1], [2]. 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 [3], [4].

Resource management becomes essential for achieving zero-touch through an efficient steering service to facilitate a shift in how media content is distributed and consumed at the last mile of access networks. The advent of cloud computing and edge computing enables establishing an orchestration in the Edge-Cloud Continuum [5]. This orchestration can enhance end users' viewing experience, optimizing video services for specific needs. Within the Edge-Cloud Continuum, the dynamism at the edge is a critical factor that demands a reevaluation of content steering strategies [6]–[8]. Unlike traditional static CDN setups, where content routing decisions are relatively straightforward, the Edge-Cloud Continuum requires more intelligent, context-aware content steering. This adaptability involves real-time monitoring of edge resources, network conditions, and user preferences to optimize content delivery dynamically. To provide optimal user experiences and efficient resource utilization, content steering must seamlessly switch users' content sources based on the Content Provider's needs.

This work 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 paper 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 the assessment of features related to the network context.

The remainder of the article follows this organization: Section II discusses related works. Section III presents the system model. Section IV introduces the steering mechanism used in the evaluation. Section V details the testbed scenario employed in our experiments and the metrics evaluated in our results. Section VI describes simulation-based tests and results, and finally, Section VII concludes the paper.

II. Related Works

Adaptive Video Services on the edge system have been the subject of several studies [9]–[14]. This section aims to provide valuable insights and contextual understanding of this domain's current state of knowledge.

In the context of Adaptive Video Streamings and with a focus on edge network, Armijo *et al.* [9] introduced SPACE, a system for Segment Prefetching and Caching at the Edge. Leveraging machine learning techniques for segment prefetching, SPACE demonstrated improvements in average bitrate and stall reduction compared to baseline strategies.

Yinxin *et al.* [10] 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 users.

Chen *et al.* [11] 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.

Chen *et al.* [12] investigates 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.

In the work by Xu *et al.* [13], 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.* [14] 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, 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 Content Steering in an Edge-Cloud Continuum setting. We propose, analyze, and evaluate Content Steering and its impact on the network performance and the protocol communication within this environment and the entities involved, exploring its adaptation in real time and leveraging the edge network's advantages.

III. System Model

The Deployed Adaptive Video Streaming workflow is illustrated in Figure 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 [7], [15]. Here, we refer to Adaptive Video Service or Video Edge Service as the interchangeable microservice for the same purpose. Content Steering 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 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 in 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

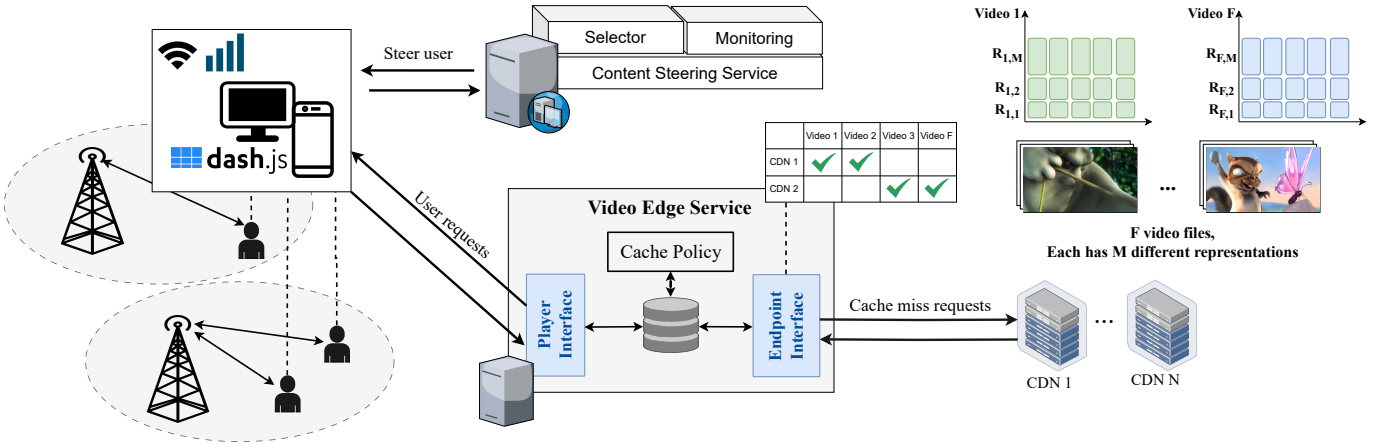

Fig. 1: Workflow Architecture of a Content Steering-Enabled Adaptive Video System.

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. 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 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 use the dash.js player in the Google Chrome browser (version 117.0.5938.88) with headless mode enabled. 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 to facilitate QoE monitoring. This equation incorporates four key metrics: average perceptual quality, quality oscillation frequency, stall event statistics, and perceived throughput. Section V-C 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.

IV. Content Steering Service Enhancement through Adaptive Video Systems

Content steering is an important feature in network management that helps to guide users towards the best video edge servers that can provide multimedia content. In AVENUE, the Content Steering Service is an additional layer that works together with the monitoring and selector modules, giving a comprehensive overview of the system. 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.

A. 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 by 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 their 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 by observing predefined network metrics. This feedback is crucial to ensure efficient network resource use and improve the end-user experience.

B. 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 show the Selector Module's implementation in detail in Algorithm 1.

Algorithm 1: Content Steering Selector

Input: User request req and List of active sessions $sessions$
Output: Edge Server $server$

```
1  # Reatrieve session from req;
2  session ← session.getSession(req)
3  if session is not null and session.is_alive() then
4      return session.getEdgeServer()
5  else
6      session ← createSession()
7      # select edge server for request req
8      server ← SelectServerProblem(req)
9      session.setEdgeServer(server)
10     # Insert session in list of active sessions
11     session.setSession(session)
12     return session.getEdgeServer()
```

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 that the server can generate.

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

Listing 1: Json response

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 1, while the CDN server has a lower priority. The specifications of the EdgeCache are detailed in the PATHWAY-CLONES with an original CDN base, the edge node address rules are given by the URI-REPLACEMENT, with the host replaced by url in HOST. The client receives these instructions with a Time-To-Live (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 nodes availability.

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 [1].

V. Assessing the Edge-Cloud Continuum Scenario

The experiment design focuses on establishing a resilient Adaptive Video System that synergizes with the Content Steering Service. Figure 2 illustrates the scenario used in our experiments. We opted to construct our services (frontend, CDN, Video Edge Services, and Content Steering) to exert complete authority over every facet of its operation. The Mininet-WiFi was used to emulate our envisioned topology at the Edge [16]. At the same time, we deployed the Steering and CDN services in a private Cloud in the Institute of Computing at UNICAMP. The container images are available at https://github.com/eduardogama/AVENUE.git.

A. Edge-Cloud Continuum Scenario 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, being five in each CDN node. We use the reference test videos from the open-source dataset in [17]. 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 [18]. 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 start users requests 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.

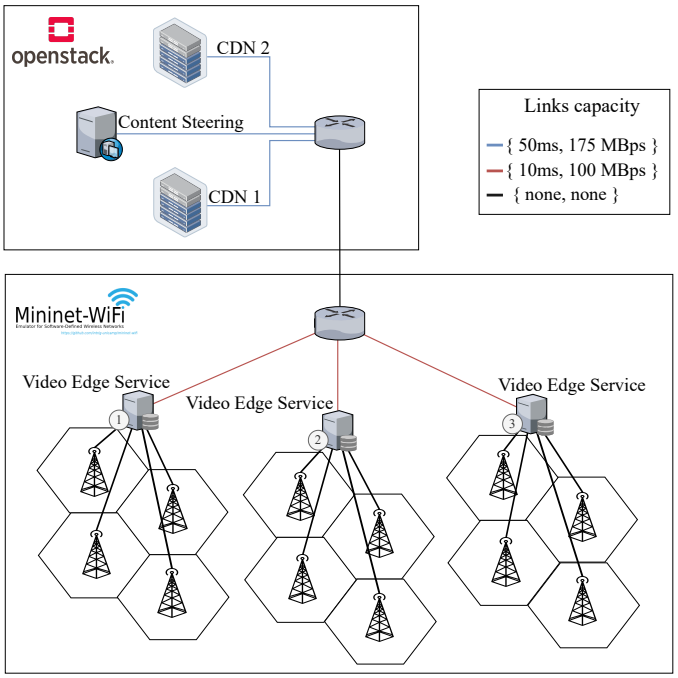

Fig. 2: Scenario a Content Steering-Enabled Adaptive Video System.

B. 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 computing resources 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 I 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.

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

TABLE I: Results of execution time with standard deviation for Server Selection Algorithms.

C. 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 [19]. Equation 1 shows the numerical transformation for each received video segment v_l^k, where $l = [1, ..., L]$ represents the bitrate level and $k = [1, ..., K]$ is the index of the video segment. v_L denotes the maximum bitrate level for video v. In this work, the video dataset considers the representation bitrate range from 100 Kbps (144p) to 17000 Kbps (4K). 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.

$$q_u(v_l^k) = a_1 \cdot \log(a_2 \cdot (v_l^k / v_L)) \quad (1)$$

In order to assess the long-term satisfaction of each user, it is essential to employ a model that incorporates key metrics for quantifying QoE throughout video playback. In this context, we refer to [20], which comprehensively addresses the video segments K and encompasses four metrics.

1) The average perceptual quality of the segments:

$$T_u^q = \frac{1}{K}\sum_{k=1}^{K} q_u(v_l^k)$$

2) The average number of oscillations of quality:

$$T_u^{oq} = \frac{1}{K-1}\sum_{k=1}^{K-1} |q_u(v_l^{k+1}) - q_u(v_l^k)|$$

3) The average number of stall events. In other words, if the playback buffer size is empty, there will be a stall. The average number of stalls S_k can be calculated as:

$$T_u^{sd} = \frac{1}{K}\sum_{k=1}^{K} S_k$$

In Equation 2, the Q_u for each user u can range from 0 to 5, where bad ($Q_u \leq 1$), poor ($1 < Q_u \leq 2$), fair ($2 < Q_u \leq 3$), good ($3 < Q_u \leq 4$), and excellent ($4 < Q_u \leq 5$). By considering these different values, we can more comprehensively and accurately measure the user's satisfaction.

$$Q_u = T_u^q - T_u^{oq} - T_u^{sd} \tag{2}$$

D. 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 3. 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.

$$J = \frac{(\sum_{s=1}^{S} R_s)^2}{S * \sum_{s=1}^{S} R_s^2} \tag{3}$$

VI. Results

This Section presents the numerical results considering the QoE, Cache Hit rate, and Load Fairness.

A. Network Scenarios Evaluation

The experiments depicted in Figure 3 showcase the average QoE on the y-axis with a confidence interval of 95%, calculated using Equation 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 CA, RA, and 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.

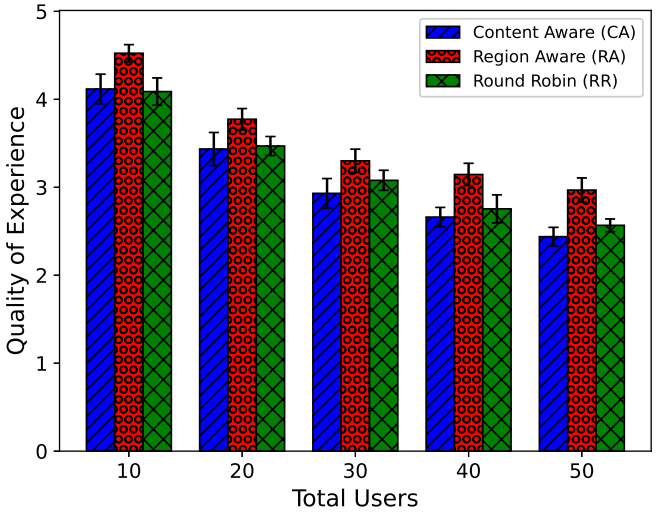

Fig. 3: QoE Performance with CA, RA, and RR (y-axis) and the total number of users in the x-axis. The scenario is illustrated in Figure 1.

Figure 4 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 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

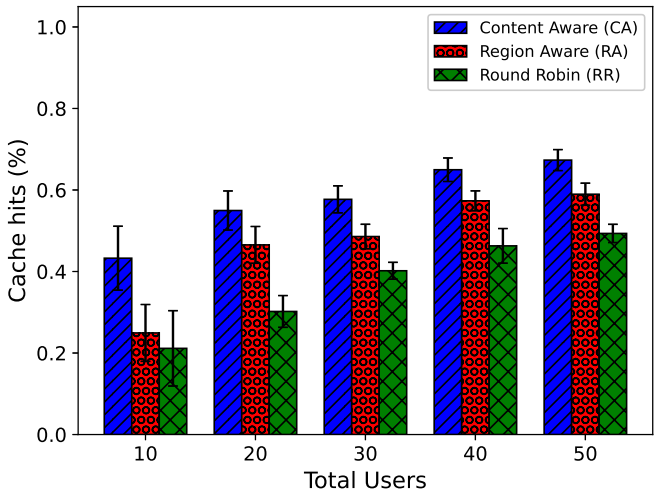

Fig. 4: Cache Hit Performance with CA, RA, and RR (y-axis) and the total number of users in the x-axis. The scenario is illustrated in Figure 1.

steering, highlighting the trade-offs between different fairness guarantees and content delivery efficiency.

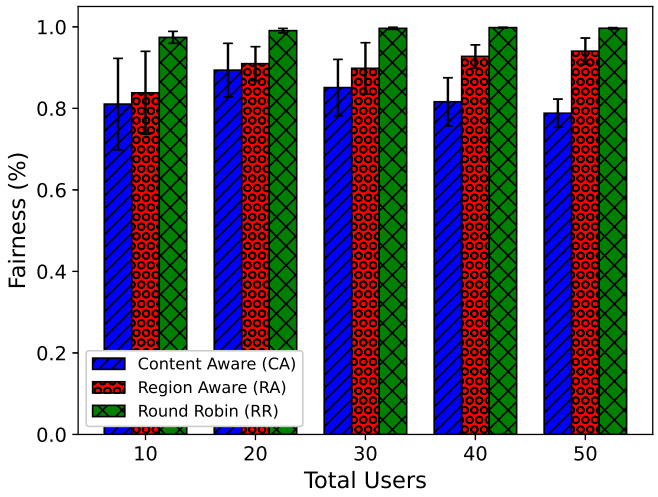

Fig. 5: Fairness Performance with CA, RA, and RR (y-axis) and the total number of users in the x-axis. The scenario is illustrated in Figure 1.

B. Impact on Planning the Adaptive Video Streaming System

An interesting discussion occurs 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 II 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 II). In case of the best load distribution among the servers, the RR is the best choice (see lines 7-9 of Table II). 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.

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

TABLE II: Review of the results considering the performance profiles: QoE, Cache hits, and Fairness load. Two scenarios, with 10 and 50 users, were analyzed.

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.

VII. Conclusion

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

For future work, we plan to deploy a Machine Learning algorithm to exploit the strengths of the studied edge server selection algorithms. This algorithm will continuously learn

and adapt to network conditions and content distribution patterns, identifying the optimal server-select algorithm based on real-time metrics like system load, user location, and content popularity. In this way, The algorithm, thus, will be able to pick the best Server Select algorithm and increases the network's performance.

ACKNOWLEDGMENT

This work was supported by the Innovation Center, Ericsson S.A., and by the Sao Paulo Research Foundation (FAPESP), grant 2021/00199-8, CPE SMARTNESS. This study was partially funded by CNPq and CAPES, Brazil - Finance Code 001.

REFERENCES

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

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

[3] 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.

[4] 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.

[5] 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.

[6] 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] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] Ramon R. Fontes, Samira Afzal, Samuel H. B. Brito, Mateus A. S. Santos, and Christian Esteve Rothenberg. Mininet-wifi: Emulating software-defined wireless networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 384–389, 2015.

[17] 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.

[18] Log distance path loss or log normal shadowing model, 2013. ; accessed in december 17, 2023.

[19] 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.

[20] Abdelhak Bentaleb, Ali C. Begen, Saad Harous, and Roger Zimmermann. Want to play dash?: A game theoretic approach for adaptive streaming over http. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 13–26, New York, NY, USA, 2018. ACM.