Video Streaming Optimizations via Collaborative Multi-CDN Selection with Deep Reinforcement Learning.

Chidambar Joshi

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

December 2025

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Chidambar Joshi**

Entitled: **Video Streaming Optimizations via Collaborative Multi-CDN Selection with Deep Reinforcement Learning.**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Sandra Céspedes

_____ Examiner
Dr. Essam Mansour

_____ Examiner
Dr. Sandra Céspedes

_____ Supervisor
Dr. Abdelhak Bentaleb

Approved by _____
Dr. Joey Paquet, Chair
Department of Computer Science and Software Engineering

_____ 2025

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Video Streaming Optimizations via Collaborative Multi-CDN Selection with Deep Reinforcement Learning.

Chidambar Joshi

Multi-Content Delivery Network (Multi-CDN) strategies are vital to enhancing Quality of Experience (QoE) in adaptive video streaming. The recent content steering standard (ETSI TS 103 998) enables real-time CDN selection by collecting performance statistics from players and CDNs. However, existing rule-based approaches such as round-robin, least connections remain static and often fail to adapt to network dynamics.

To address this, we developed StreamWise, a single-agent learning-based framework for CDN selection. Although StreamWise improved QoE, its coarse-grained design overlooked global operational costs, limiting its effectiveness. Building on this, we propose Cadence, a multi-agent deep reinforcement learning framework that jointly optimizes user QoE and multi-CDN costs. Cadence adopts a Centralized Training with Decentralized Execution (CTDE) paradigm, where per-client agents make fine-grained CDN selections, while a centralized critic coordinates training. Both frameworks are trained on experience trajectories from Pensieve ABR, ensuring realistic adaptation to network and content dynamics.

Through extensive trace-driven emulation experiments, we show that StreamWise improves the average VMAF by 8.5% and achieves a 1.5× higher bitrate, and delivers a 48% QoE improvement over heuristic baselines. Cadence further improves performance, improving VMAF by up to 21%, achieving 1.2× higher bitrate for live streaming, reducing rebuffering events by up to 10×, and lowering multi-CDN operational costs by 35%.

This thesis demonstrates that reinforcement learning-based Multi-CDN frameworks—first with StreamWise and more effectively with Cadence—deliver high-quality, cost-efficient adaptive video streaming at scale, significantly outperforming heuristic and coarse-grained approaches.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Abdelhak Bentaleb, for his unwavering guidance, support, and mentorship throughout my research journey. His expertise, patience, and dedication have been instrumental in shaping my academic and professional growth, and I feel truly privileged to have worked under his supervision.

I am sincerely thankful to the IN2GM Lab for providing an intellectually stimulating environment and fostering a culture of collaboration and academic excellence. I also extend my heartfelt appreciation to my co-author, Jashanjot Singh Sidhu, whose valuable insights and contributions greatly enriched the quality of this research.

My deepest appreciation goes to my family for their unconditional love, encouragement, and belief in me throughout this academic journey. I am especially grateful to my parents, Vasant Joshi and Naina Joshi, for instilling in me the values of education, perseverance, and resilience. I would also like to thank my brother, Sudhanva Joshi, for his constant encouragement and support.

Finally, I acknowledge my home country, India, for providing me with the educational foundation and opportunities that enabled me to pursue my academic aspirations abroad.

In conclusion, this thesis would not have been possible without the support, encouragement, and guidance of the individuals and institutions mentioned above. I remain deeply grateful for their belief in my abilities and their invaluable contributions to my journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

HTTP Adaptive Streaming (HAS) [3] has revolutionized video consumption, now accounting for a significant portion of global internet traffic. This growth is driven by major video-on-demand and live-streaming platforms such as YouTube, Meta, and Prime Video. The widespread adoption of ultra-HD (4K) content and the rapid expansion of 5G networks [4] have further raised viewers' expectations for smooth, high-quality playback with minimal interruptions. Consequently, content providers must continuously optimize their video delivery pipelines to maximize viewer Quality of Experience (QoE) [5, 6], while managing operational costs and scalability requirements.

To meet these demands, most streaming providers rely on Content Delivery Networks (CDNs) [7] that distribute video streams across geographically dispersed servers. CDNs reduce latency, improve scalability, and lower delivery expenses. However, dependence on a single CDN poses notable risks, including outages [8] and performance bottlenecks, which can cause service disruptions and significantly degrade QoE. For example, during periods of intense traffic or CDN failures, a single CDN may falter in delivering consistent performance across regions and networks, jeopardizing smooth playback.

To overcome these vulnerabilities, large content providers increasingly adopt multi-CDN architectures [5, 6, 9, 10]. By integrating multiple CDNs, they enable intelligent traffic routing and

dynamic load balancing across different vendors. This strategy enhances performance and mitigates failure risks, particularly during CDN outages when single-CDN solutions suffer quality of service degradation. Figure 1.1 compares single-CDN and multi-CDN performance, motivating our approach.

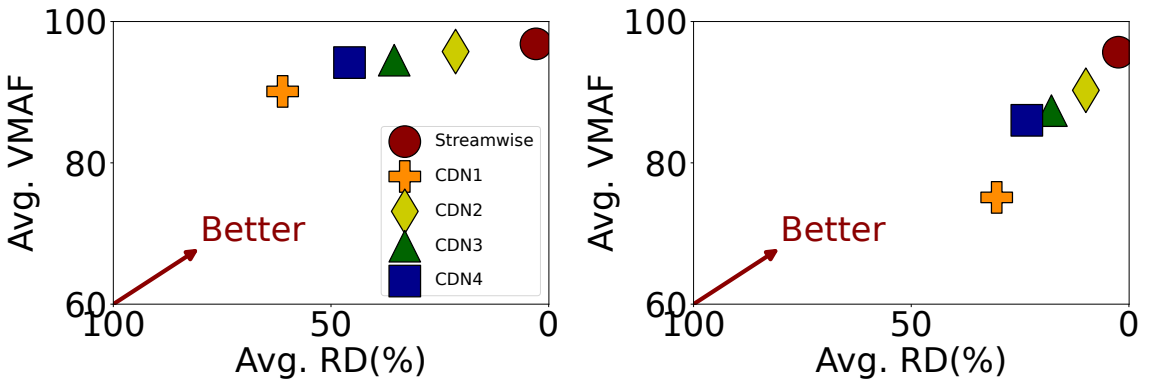

Figure 1.1: Performance comparison of single-CDN versus multi-CDN strategies for Live Low-Latency (LLL:left) and Video-on-Demand (VoD:right) streaming scenarios.

Despite these advantages, current multi-CDN approaches primarily rely on heuristic-based switching strategies, which often struggle to adapt to diverse and rapidly changing network conditions.

1.1.1 Problem Statement

Modern multi-CDN streaming environments demand data-driven adaptation strategies capable of reacting to heterogeneous network dynamics, cost fluctuations, and diverse client conditions. However, the majority of existing solutions still depend on heuristic or statically defined rules for CDN selection. These methods fail to capture the temporal variability of network states and often focus on short-term throughput improvements rather than long-term QoE optimization.

To address these challenges, we developed ***StreamWise***, a deep reinforcement learning (DRL)-based CDN steering framework designed to dynamically choose the optimal CDN in response to real-time feedback. StreamWise leverages an Actor–Critic neural architecture and Proximal Policy Optimization (PPO) to learn effective selection policies that maximize user QoE. While this approach demonstrated substantial improvements over heuristic baselines, its single-agent, group-centric design introduces new limitations. By aggregating Common Media Client Data (CMCD) [11] across user groups, it overlooks regional network heterogeneity, fails to account for differences in user

connectivity, and ignores operational cost implications inherent in pay-per-use multi-CDN models.

These limitations result in uneven resource utilization, unfair quality allocation among clients, and inefficient handling of cost-performance trade-offs. To overcome these issues, we propose ***Cadence***, a multi-agent DRL framework for intelligent CDN selection. Cadence assigns a dedicated agent to each client, enabling decentralized and context-aware decision-making. Through inter-agent collaboration, Cadence jointly optimizes QoE, cost efficiency, and CDN load balance, achieving scalable, fair, and robust video delivery under diverse network dynamics.

1.2 Contributions

This thesis addresses these challenges by developing and evaluating learning-based content steering(ETSI TS 103 998) [6, 12] frameworks for multi-CDN environments. The research began with the design and deployment of StreamWise: a DRL approach for real-time, fine-grained CDN selection that significantly improved QoE metrics compared to legacy and heuristic methods. However, real-world deployment of StreamWise revealed new challenges at scale—such as fair resource allocation, cost-awareness, overloaded CDNs, and an inability to handle highly asynchronous client demand—that exposed the limits of single-agent or group-centric solutions. Motivated by these findings, the thesis introduces Cadence: a collaborative multi-agent DRL framework, where each client is assigned a dedicated agent that collectively optimizes both QoE and delivery cost, prevents resource monopolization, and offers improved resilience during CDN failures.

Major contributions of this thesis are:

- The DRL implementation, and evaluation of StreamWise, a learning-based, single-agent (and group-aggregated) content steering solution for maximizing QoE through real-time CDN switching.
- Experimental identification and analysis of StreamWise's limitations in large-scale, heterogeneous, and cost-sensitive deployments.
- The DRL evaluation of Cadence, a multi-agent, collaborative system for individualized, scalable, cost- and QoE-aware CDN selection, validated by extensive trace-driven experiments

spanning diverse network and streaming scenarios.

- Extensive experimentation covering a wide range of ABR schemes (Pensieve, BBA2, L2A, LoL$^+$), network traces, video content types (with varying encoding parameters), and streaming scenarios (VoD and LLL).

1.3 Outline

This thesis is organized into six chapters that detail the architecture and evaluation of our single and multi-agent DRL models against the state-of-the-art solutions. Chapter 2 examines the critical role of CDNs in serving a global user base. Chapter 3 provides a comprehensive review of the relevant literature and establishes the foundational concepts. Chapter 4 details the design and implementation of the StreamWise DRL framework, while Chapter 5 introduces the Cadence multi-agent framework and its evaluation methodology. Chapter 6 presents the experimental testbed, performance benchmarks, and a comparative analysis of both frameworks. Finally, Chapter 7 summarizes the key findings, underscores the thesis contributions, and proposes avenues for future research with implications for next-generation video streaming systems.

Chapter 2

Content Delivery Networks (CDN) in Scalable Video Streaming

2.1 CDN Concepts

- **Content Provider Origin Server:** The primary web server that hosts and delivers the original source content.
- **CDN Entry Point(s):** Servers within the CDN responsible for retrieving content from the origin server and caching it for distribution.
- **CDN Origin Shield:** An intermediate caching layer designed to protect the origin server from excessive load during high-traffic periods.
- **CDN Edge Servers:** Distributed CDN servers that deliver cached content directly to end users based on proximity and availability.
- **CDN Footprint:** The geographical distribution and reach of CDN edge servers that determines their ability to efficiently serve user requests.
- **CDN Selector:** In multi-CDN architectures, a decision-making component responsible for dynamically choosing the optimal CDN for each request.

- **CDN Offloading:** In Peer-to-Peer (P2P) CDN systems, a mechanism that enables clients to share content among themselves, reducing reliance on edge servers.

2.2 The Scalability Challenge: Origin Server Limitations

To understand the necessity of CDNs, it's essential to address the limitations of a centralized origin server architecture.

- **Network Latency:** The physical distance between a user and the origin server introduces propagation delay, governed by the speed of light in fiber optics. For example, a user in Tokyo requesting content from a server in Virginia will experience a minimum round-trip time (RTT) of over 200 ms before any data processing begins. For video streaming, this delay manifests as long buffering times and slow start-up.
- **Bandwidth Congestion:** A single origin server (or even a clustered data center) has a finite uplink bandwidth capacity. A "flash crowd" event, such as the launch of a popular show or a live sports event, can generate terabits per second of demand, easily saturating the origin's capacity and leading to packet loss, throttling, and service outages for users worldwide.
- **Server Load:** Each video request requires the origin server to perform I/O operations (reading from disk) and packet processing. Under high load, CPU and memory resources become bottlenecks, increasing response times and potentially causing server failure.

In essence, the origin server represents a single point of failure and a significant performance bottleneck. CDNs are designed explicitly to mitigate these issues by distributing the load.

2.3 CDN Architecture: A Distributed System of Points of Presence (PoPs)

A Content Delivery Network is a geographically distributed network of proxy servers and their data centers. The goal is to bring content geographically and logically closer to end-users. The holistic diagram is depicted in the Figure 2.1

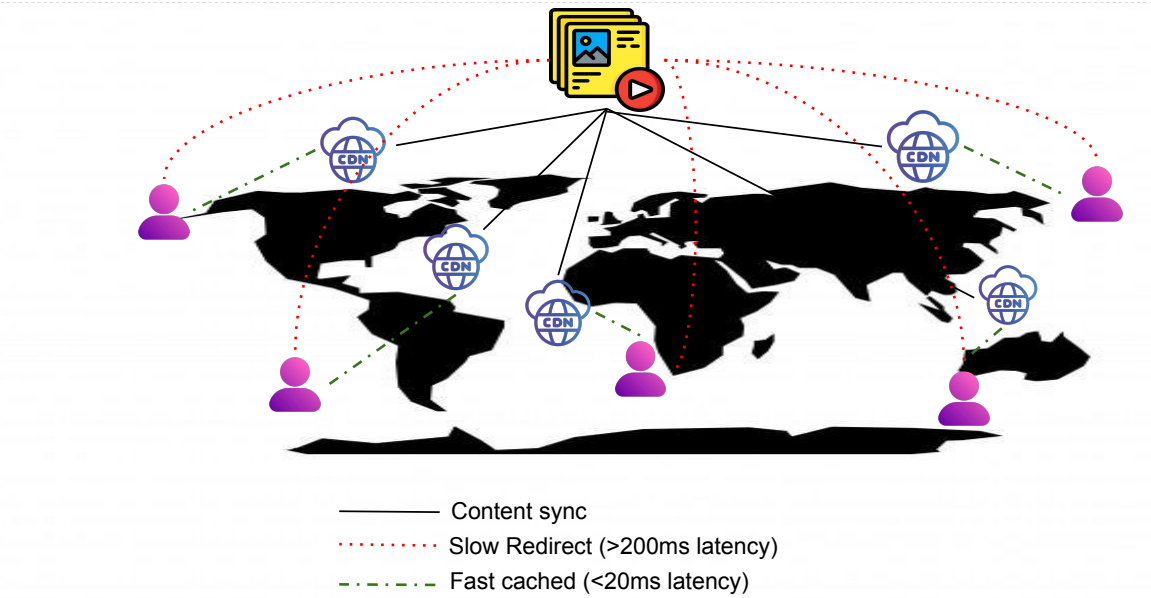

Figure 2.1: Global CDN architecture showing content flow from the origin server through the CDN core to regional PoPs and end users. The diagram illustrates how cached content at edge servers reduces latency by 10x and improves delivery performance.

- **Points of Presence (PoPs):** These are the core building blocks of a CDN. A PoP is a cluster of caching servers strategically located at the edge of the internet, typically within Internet Exchange Points (IXPs) or Internet Service Provider (ISP) data centers in major metropolitan areas.

- **Caching Servers:** Within each PoP, high-performance servers store (cache) copies of popular content. When a user requests a video, the CDN's routing system directs the request to the optimal PoP. If the video is cached there, it is served directly from the edge server. This is called a *cache hit*.

- **The Cache Hierarchy:**

  - **Edge Server (First Tier):** The server closest to the user. On a cache hit, the response is extremely fast.

  - **Mid-Tier / Regional PoPs (Second Tier):** If an edge server experiences a cache miss (the content is not stored locally), it requests the content not from the distant origin, but from a larger, regional mid-tier PoP.

- **Origin Shield (Optional Third Tier):** Some CDNs use a dedicated origin shield server that acts as a single point of contact for the origin, further reducing the load on the origin by aggregating requests from all mid-tier PoPs.

This hierarchical architecture ensures that the vast majority of requests are served from the edge, while the origin server only needs to handle cache-fill requests, which constitute only a small fraction of the total traffic.

2.4 The Content Delivery Workflow

The process of delivering content via a CDN involves several key steps, often transparent to the end-user:

- **Content Prepositioning:** The content provider uploads its video library to the origin server. The CDN then proactively pulls popular content to its PoPs or is configured to pull content on its first request (*cache-on-demand*).
- **User Request Routing (DNS & Anycast):** When a user clicks "play", their client resolves the video URL's domain name. This DNS query is intercepted by the CDN's intelligent routing system, which uses metrics such as the user's IP address (geolocation), real-time network health, and server load to return the IP address of the optimal PoP.
- **Content Serving from the Edge:** The user's device establishes a connection to the selected edge server. The video segments are streamed directly from this local cache.
- **Cache Management:** CDNs employ sophisticated algorithms (e.g., Least Recently Used – LRU, or cost-based algorithms) to manage their finite cache storage, ensuring the most popular content is readily available.

2.5 User QoE Optimization

The primary value of a CDN is its direct and profound impact on User QoE [13], the gold standard for measuring streaming performance.

- **Reducing Latency and Startup Delay:** By serving content from a nearby PoP, CDNs dramatically reduce RTT. This leads to faster TCP connection setup and quicker TLS handshakes, which directly translates to lower Video Start-Up Time (VST), a critical QoE metric.

- **Minimizing Rebuffering Events:** High latency and network congestion cause the video player's buffer to deplete faster than it can be filled, leading to pauses in playback (*rebuffering*). CDNs provide high-bandwidth, stable connections from the edge, maintaining a healthy buffer and minimizing the *Rebuffering Ratio*.

- **Enabling High-Bitrate Streaming:** Adaptive Bitrate (ABR) algorithms [3] in video players dynamically select the best quality video segment based on observed network throughput. A stable, high-throughput connection from a CDN allows the ABR algorithm to select higher bitrate encodes, improving visual quality (Average Bitrate) and reducing quality oscillations.

- **Improving Reliability and Availability:** The distributed nature of a CDN provides inherent fault tolerance. If one PoP or server fails, the routing system can seamlessly direct users to the next best available PoP, ensuring high service availability and reliability.

Chapter 3

Literature Review

3.1 Traditional Switching Methods

Multi-CDN strategies have long been studied to improve video streaming QoE by enhancing reliability, performance, and fault tolerance. Early approaches primarily employed DNS-based switching [14], in which a fixed video URL was dynamically resolved to different CDNs via DNS. While this method's simplicity was attractive, it had substantial limitations. The most critical drawback was the long delay incurred during CDN switching—often lasting several minutes in failure scenarios—due to DNS caching and propagation. Such delays are detrimental to user experience, causing playback interruptions during network fluctuations.

To overcome these latency issues, manifest rewriting techniques [15] were introduced. This approach enables midstream CDN switching by dynamically modifying the content manifest during playback, avoiding costly session resets. Manifest rewriting significantly reduced switch latency compared to DNS-based methods, enabling more responsive adaptation during streaming. Nevertheless, this method was not without challenges: manifest modifications could introduce errors, and switching still depended on delayed detection of CDN failures. Moreover, manifest rewriting was less effective for live streaming scenarios, where continuous, low-latency delivery is paramount.

3.2 Server-Side and Client-Side Switching

Recognizing the limitations in responsiveness and error-proneness, later research explored server-side switching [10, 16] to centralize CDN decision-making. By aggregating data from multiple clients, the server could make informed switching decisions across the user base, giving operators fine control and simplifying deployment. However, server-side methods often relied on aggregated metrics, losing visibility into individual client conditions such as local network constraints or device capabilities. This coarse granularity limited personalization of CDN selection, potentially resulting in suboptimal QoE for users with highly variable network connectivity.

To address this shortcoming, client-side switching [5, 17, 18] empowered video players at the client to collect local, real-time performance indicators—such as buffer occupancy, throughput, and latency—and make autonomous CDN switching decisions. This fine-grained, decentralized approach significantly improved adaptation to heterogeneous network conditions, yielding better user experiences. However, client-side implementations face considerable challenges related to complexity and scalability. Integrating switching logic into diverse proprietary players is difficult, and coordinating switching across a large user base raises operational complexity.

3.3 Content Steering and Heuristic Approaches

3.4 Content Steering and Heuristic Approaches

The release of adaptive streaming standards like DASH and HLS introduced content steering [6, 10, 12], which standardized CDN switching behavior across disparate players. This harmonization reduced integration complexity, enabling providers to implement multi-CDN solutions more consistently and reliably.

Despite this progress, many commercial deployments continue to rely on static load-balancing heuristics [18, 19] such as round-robin (RR) or least-connections (LC) [20]. These algorithms serve as the default traffic steering logic for major infrastructure providers, including *AWS Elastic Load Balancing* [21], *NGINX* [22], and multi-CDN vendors like *CDNetworks* and *Tencent Cloud*, which rely on weighted round-robin to maintain target traffic ratios between providers [23]. Although

straightforward, these heuristics fail to consider the dynamic and heterogeneous nature of network conditions, making them inflexible and often resulting in degraded QoE during congestion or transient failures.

These methods are sensitive to noisy, unstable measurements and struggle to generalize across diverse environments with fluctuating user populations and CDN capacities, mainly due to their rigid, rule-based actions.

3.5 Learning-Based CDN Optimization

To overcome the shortcomings of heuristic methods, learning-based frameworks—particularly those leveraging Deep Reinforcement Learning (DRL)—have emerged. Our solution, StreamWise [24], introduced a DRL-based CDN switching system that combines real-time network and client playback metrics with content steering. StreamWise learns to optimize CDN selection dynamically, providing superior QoE compared to heuristics by proactively reacting to network variations and playback performance indicators.

However, StreamWise's design involves a single learning agent making group-level decisions and does not account for spatial network heterogeneity or regional disparities in CDN infrastructure quality. This reduction in granularity can lead to suboptimal CDN choices for users in specific geographic locations or those disproportionately affected by congestion. Additionally, StreamWise does not incorporate cost into the decision process, limiting practical deployment since multi-CDN providers increasingly face usage-based billing and cost-performance tradeoffs.

To address these scalability and billing challenges, recent literature has shifted towards Multi-Agent Reinforcement Learning (MARL) and cost-aware optimization. Works such as MAARS [25] and NOVA [26] demonstrate that decentralized agents can effectively coordinate edge resources and 360-degree video transmission in heterogeneous environments. Simultaneously, new frameworks like PIRA [27] have highlighted the necessity of balancing high-performance premium CDNs against operational expenditures. However, existing MARL approaches generally focus on edge computing or caching rather than switching, while cost-aware methods often lack the granular, per-user adaptability of a multi-agent system.

In order to bridge these identified gaps and building on StreamWise's foundation, this thesis presents Cadence, a novel multi-agent DRL framework that advances CDN switching by introducing per-user geographic distribution and cost-awareness. By deploying coordinated agents representing individual users and regions, Cadence addresses spatial and network diversity to optimize CDN selection with fine granularity. Furthermore, it explicitly incorporates operational costs alongside QoE metrics, enabling providers to balance user experience and expenditure dynamically.

Chapter 4

StreamWise Solution

4.1 Overview

4.1.1 The Real-World Challenge

Imagine you are watching a live sports event online. The video is delivered to you through a network of servers called CDNs. To ensure smooth playback, streaming providers often use multiple CDNs at once. But which CDN should serve you at any given moment? If the wrong one is chosen, you might experience buffering, low video quality, or even interruptions—especially if a CDN is overloaded or fails.

Traditional approaches (like DNS-based switching or simple round-robin) are too slow or too rigid. They can't react quickly to sudden network changes or individual user needs. This is where StreamWise comes in: it aims to make CDN selection intelligent, fast, and personalized.

4.1.2 Turning CDN Switching into a Learning Problem

To automate and optimize CDN selection, we treat it as a decision-making problem under uncertainty. At every moment, the system must decide: Which CDN should serve the next video segment to this user? The answer depends on many factors—network speed, current buffer, recent failures, delay, and more.

This is naturally modeled as a Markov Decision Process (MDP):

- State: What is the current situation? (e.g., buffer level, recent rebuffering, network speed, which CDN was used last)
- Action: Which CDN to pick for the next segment?
- Reward: How good was the outcome? (e.g., high video quality and no stalls = high reward; buffering or low quality = penalty)
- Transition: How does the system move from one state to another after an action?

The goal is to learn a policy—a rule for picking the best CDN in any situation—that maximizes the long-term reward (i.e., best possible viewing experience over time).

4.2 Neural network architecture

4.2.1 Input State Design

The input state at each decision point is a vector that summarizes the most important, recent information about the streaming session. Specifically, the input state includes:

- **Throughput**: The measured download speed (e.g., in Mbps) for the most recent video segment. This tells the agent how fast data is arriving from the current CDN.
- **RTT** (Round-Trip Time): The time it takes for a request to travel to the CDN and back. High RTT can signal network congestion or distance from the CDN, typically measured in millisecond (ms).
- **Stall** (Rebuffering Event): Whether the video playback stalled (buffered) during the last segment. This is a direct indicator of poor user experience, typically measured in seconds.

4.2.2 The softmax output Layer

After the input state (comprised of throughput, RTT, and stall) is processed by the hidden layers of the neural network, the final decision about which CDN to select is made in the output layer.

This layer uses the softmax activation function, which is a standard choice for multi-class decision problems like CDN selection.

The softmax function takes the raw output values (called logits) from the last hidden layer and converts them into a set of probabilities—one for each possible CDN. The key properties are:

- Probability Distribution: The outputs are all between 0 and 1, and they sum to 1. This means each value can be interpreted as the probability that a particular CDN is the best choice given the current state.
- Decision Making: The CDN with the highest probability is selected for the next video segment. This makes the decision process both interpretable and flexible.

4.3 DRL Framework Implementation

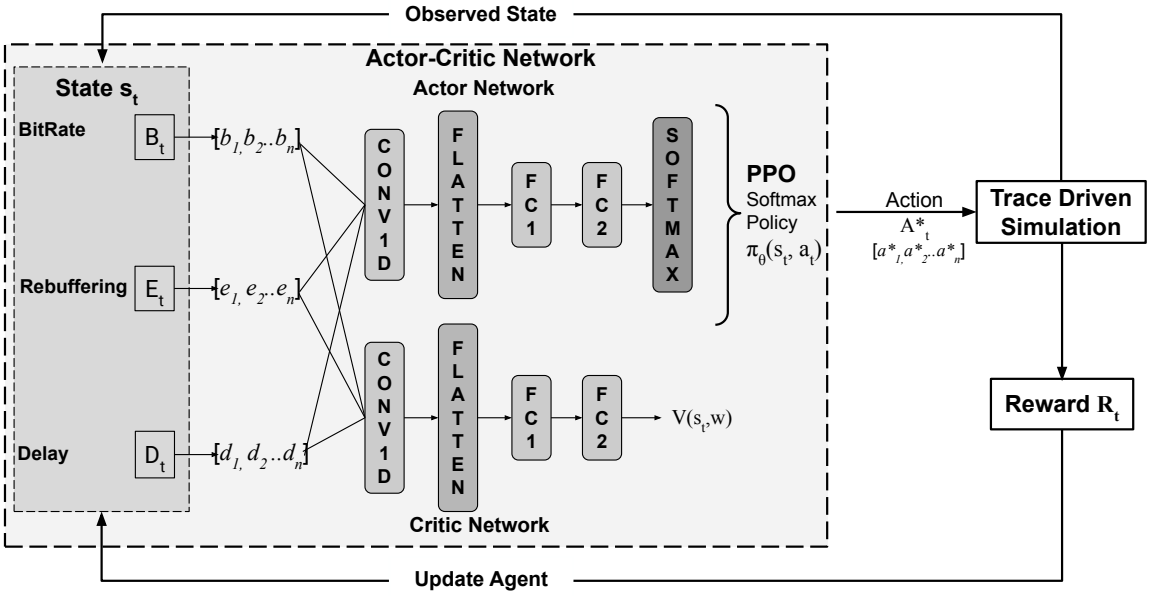

Figure 4.1: StreamWise DRL architecture for adaptive multi-CDN video streaming. The agent observes environment states such as bitrate, rebuffering, and delay, and interacts with multiple CDNs by selecting the optimal one for each segment. The Actor–Critic framework continuously updates policies to maximize QoE by balancing throughput(bitrate), stability, and rebuffering penalties.

4.3.1 Data Pipeline and State Construction

The first step in StreamWise's DRL pipeline is to construct the agent's state—a compact, information-rich vector summarizing the most recent streaming conditions for each CDN. For every decision point (i.e., before downloading a new video segment), the system:

(1) **Loads and parses log data:** Each row in the log contains the observed bitrate (throughput), round-trip time (RTT), and stall (rebuffering) for a CDN.

(2) **Normalizes features:** To ensure stable neural network training, each feature is normalized:

- Bitrate by the maximum available bitrate,
- RTT by a large constant (e.g., 10,000 ms),
- Stall by a constant (e.g., 10 seconds).

This normalization ensures all features are on a similar scale, which is crucial for stable learning.

(3) **Constructs the state tensor:** The normalized values for all CDNs are stacked and transposed to form the input state tensor:

$$\mathbf{s}_t = \begin{bmatrix} \text{bitrate}_t \\ \text{RTT}_t \\ \text{stall}_t \end{bmatrix}$$

This careful state construction ensures the agent has all the information needed to make context-aware CDN decisions, while keeping the input size manageable for efficient learning.

4.3.2 Actor-Critic Neural Network and Action Selection

The core of the StreamWise architecture, illustrated in Figure 4.1, is an Advantage Actor-Critic (A2C) [28] neural network. This network is responsible for both learning and representing the agent's CDN selection policy, mapping system states to probability distributions over potential CDN choices.

(1) **Actor network:** Receives the current state and outputs a probability distribution over available CDNs using a softmax layer. This distribution reflects the agent's confidence in each CDN being the best choice for the next segment.

(2) **Critic network:** Receives the same state and estimates the expected future reward (value function) if the agent follows its current policy. This helps the agent evaluate how good its decisions are.

Action selection is performed by sampling from the actor's softmax output. This means the agent sometimes explores less-likely CDNs (*exploration*), but usually picks the best-known CDN (*exploitation*). This balance is crucial for learning robust, generalizable policies.

The actor and critic are both fully connected neural networks, trained using the RMSProp optimizer. The actor is responsible for decision-making, while the critic provides a learning signal to improve the policy.

The softmax output for CDN i is given by:

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where z_i is the logit for CDN i and K is the number of CDNs.

4.3.3 Training Loop: Reward Design, Advantage Estimation, and Policy Update

The training loop is where the agent learns from experience:

(1) **Reward calculation:** After each action, the agent receives a reward that reflects the streaming QoE. The reward is a weighted sum:

$$r_t = w_1 \cdot \text{bitrate}_t - w_2 \cdot \text{stall}_t - w_3 \cdot \text{RTT}_t$$

where w_1, w_2, w_3 are weights tuned to balance quality, delay, and smoothness. Our results reveal that the optimal values for the weights are $w_1 = 1$, $w_2 = 5$, and $w_3 = 0.5$. Note that the state values passed to the reward function are normalized beforehand.

(2) **Experience collection:** The agent's experience is accumulated in the form of sequential tuples $(\mathbf{s}_t, a_t, r_t, v_t)$ for each episode and stored in a replay memory buffer. Subsequent training utilizes batches of data sampled randomly from this buffer, a mechanism that enhances learning stability by breaking the temporal correlation between consecutive experiences.

(3) **Advantage estimation:** The agent employs Generalized Advantage Estimation (GAE) [28] to compute the advantage A_t for each action. This advantage measures the relative improvement of an action over the state value function $V(s_t)$. The calculation is a recursive function of the TD error, $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$:

$$A_t = \delta_t + \gamma\lambda A_{t+1}$$

where, the parameters γ (discount factor) and λ (GAE parameter) determine the weight of future rewards and the smoothing of the advantage estimate, respectively.

(4) **Proximal Policy update:** The agent's policy is optimized using Proximal Policy Optimization (PPO) [29]. To ensure stable and monotonic improvement, PPO maximizes a clipped surrogate objective function $L_{\text{clip}}(\theta)$ that prevents large, detrimental policy updates. The total loss function also includes a value function loss $L_{\text{value}}(\theta)$ to improve value predictions and an entropy bonus $L_{\text{ent}}(\theta)$ to maintain exploration by discouraging premature policy convergence:

$$L_{\text{total}}(\theta) = L_{\text{clip}}(\theta) + c_v L_{\text{value}}(\theta) + c_e L_{\text{ent}}(\theta)$$

where c_v and c_e are coefficients weighting the respective losses.

(5) **Checkpointing:** The agent's model parameters are periodically saved as checkpoints at a fixed interval (e.g., every 1000 epochs). This practice enables model recovery from interruptions, facilitates resumption of training from a known state, and allows for the evaluation and deployment of intermediate training versions.

Chapter 5

Cadence Solution

5.1 The Multi-Agent Framework

Traditional CDN selection methods rely on either static heuristics or coarse group-based learning, which cannot adapt well to rapidly changing network conditions, heterogeneous user requirements, or operational cost constraints. Cadence overcomes these limitations by assigning a dedicated reinforcement learning (RL) agent to each client, enabling per-user, context-aware decisions.

At every segment download, each agent observes its local streaming conditions—namely, throughput, stall duration, and delay (i.e RTT)—from the available CDNs. These features are normalized across clients and time steps, forming a state vector:

$$s_i^t = \left[b_{i,1}^t, \ldots, b_{i,n}^t,\ e_{i,1}^t, \ldots, e_{i,n}^t,\ d_{i,1}^t, \ldots, d_{i,n}^t \right]$$

where b, e, d denote bitrate, rebuffering, and RTT (delay) respectively for each candidate CDN for client i at time t.

Unlike group-centric models, Cadence uses **Centralized Training with Decentralized Execution (CTDE)** [30]. During training, agents share state information via a global critic, which observes aggregated network and performance metrics:

- **Client-side Experience:** Aggregated per-client metrics (throughput, stall, delay) highlight regional congestion or uneven quality.

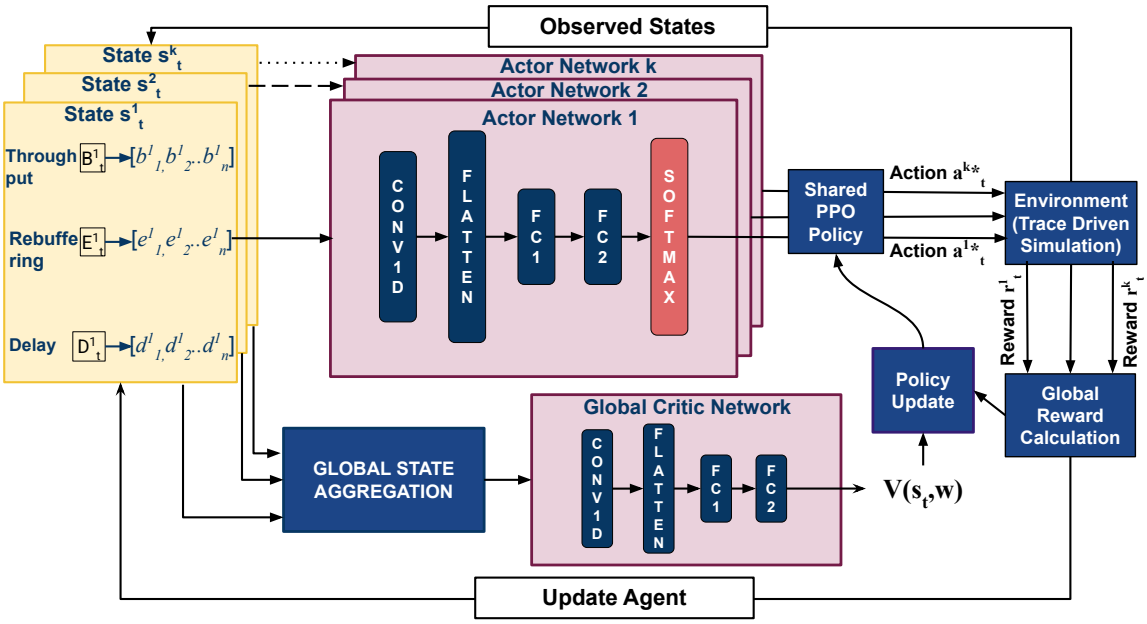

Figure 5.1: Cadence MA-DRL architecture. Multiple actor networks diligently decide the next-best CDN acting individually through a shared Multi-Agent Proximal Policy Optimization (MAPPO) policy. The global critic evaluates the decision by taking the aggregated state input by all agents and updates the policy. The global return represents the aggregated agentic rewards that collectively guide the multi-agent system toward achieving cost efficiency while maintaining a high level of user QoE.

- **CDN Load and Cost:** Real-time statistics for each CDN (current load, delay trends, pricing, selection frequency).

This centralized critic assigns rewards that discourage overloading CDNs and encourage load balancing, cost efficiency, and QoE fairness. Decision-making is decentralized at inference, with each agent independently applying the learned policy. The core architecture of Cadence is depicted in Figure 5.1.

5.1.1 Action Space and Policy

Each agent i selects an action a_t^i, which corresponds to choosing a CDN for its next video segment. The agent's actor network produces a vector of action preferences (logits), which are converted into a probability distribution over the available CDNs using a softmax output layer:

$$\pi(a_t^i \mid s_t^i) = \text{Softmax}(z_t^i) = \left(\frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \right)_{j=1}^{K}$$

where K is the number of available CDNs. Actions are sampled from this distribution, balancing the exploitation of high-probability choices with the stochasticity necessary for coordinated exploration among agents.

5.1.2 Reward Block: Dual Objectives

Our Cadence solution is a multi–agent framework that jointly (i) maximizes end-user QoE and (ii) minimizes multi-CDN operational cost. At decision epoch t, the global reward aggregates client–level QoE terms and system-level penalties.

Notation. Let $\mathcal{N}$ be the set of clients with $|\mathcal{N}| = N$, and $\mathcal{C}$ the set of CDNs. For client $i \in \mathcal{N}$ at time t: b_i^t is achieved throughput, ε_i^t is stall (rebuffering) time or indicator, d_i^t is delivery/latency delay, and $a_i^t \in \mathcal{C}$ is the selected CDN. Let $C(\cdot)$ be the cost of sending a request to a CDN. For CDN $c \in \mathcal{C}$, denote by

$$s_c^t = \sum_{i \in \mathcal{N}} \mathbb{1}[a_i^t = c]$$

the number of selections at time t. Parameter $\theta \in (0, 1]$ is the desired maximum fraction of clients (threshold) per CDN.

QoE terms.

$$\text{Throughput efficiency:} \quad \text{thr}^t = \frac{1}{N} \sum_{i \in \mathcal{N}} b_i^t, \tag{1}$$

$$\text{Stall penalty:} \quad \text{reb}^t = \frac{1}{N} \sum_{i \in \mathcal{N}} \varepsilon_i^t, \tag{2}$$

$$\text{Delay penalty:} \quad \text{delay}^t = \frac{1}{N} \sum_{i \in \mathcal{N}} d_i^t. \tag{3}$$

System-level terms.

$$\text{Cost penalty:} \quad \text{cost}^t = \frac{1}{N} \sum_{i \in \mathcal{N}} C\big(a_i^t\big)\,, \tag{4}$$

$$\text{Overload penalty:} \quad \text{over}^t = \sum_{c \in \mathcal{C}} \max\left(0,\ \frac{s_c^t}{N} - \theta\right). \tag{5}$$

Overall reward. The scalar reward is a tunable linear combination:

$$r^t \;=\; \beta_0 \;+\; \beta_{\text{thr}}\,\text{thr}^t \;-\; \beta_{\text{reb}}\,\text{reb}^t \;-\; \beta_{\text{delay}}\,\text{delay}^t \;-\; \beta_{\text{cost}}\,\text{cost}^t \;-\; \beta_{\text{over}}\,\text{over}^t, \tag{6}$$

where $\beta_\cdot \geq 0$ are hyperparameters that balance quality, latency, fairness and cost. Larger β_{thr} emphasizes throughput, while larger $\beta_{\text{reb}}, \beta_{\text{delay}}, \beta_{\text{cost}}, \beta_{\text{over}}$ discourage stalls, delay, expense, and CDN imbalance.

5.2 Multi-Agent Actor-Critic Training

This work models the multi-CDN selection as a cooperative Multi-Agent Reinforcement Learning (MARL) problem [31, 32], where each client acts as an autonomous agent optimizing a shared performance objective.

The environment is formulated as a Decentralized Markov Decision Process (Dec-MDP) [33], with each agent observing its local state and taking individual actions. The joint action of all agents influences the global system state and yields a shared reward based on aggregated performance metrics.

Training follows the CTDE paradigm. Each agent employs a decentralized actor network for policy learning, while a centralized critic network—having access to global information—evaluates joint actions and provides training feedback.

The algorithm is based on MAPPO, which extends PPO to multi-agent systems. Agents periodically collect environment trajectories, update the critic through value regression, estimate advantages via Generalized Advantage Estimation (GAE) [28], and optimize local policies using a clipped objective.

This framework achieves stable and scalable learning across multiple cooperative agents while preserving decentralized execution in the deployed system.

5.3 Real time CDN state tracking algorithm

5.3.1 Overview

The *Real-Time CDN State Tracking via Media Presentation Description (MPD) Requests* algorithm is designed to continuously monitor the performance of multiple CDNs in parallel during adaptive video streaming sessions. It leverages the *MPD* requests—lightweight metadata queries made before segment downloads—to estimate and update critical network performance metrics, namely **Round Trip Time (RTT)** and **Throughput**. By doing so, the system maintains an up-to-date view of CDN performance without disrupting playback or requiring additional heavy probes, making it suitable for real-time, low-latency CDN selection in both *LLL* and *VoD* modes.

5.3.2 Input and Initialization

Inputs

- **Set of available CDNs ($\mathcal{C}$):** The client has access to multiple CDNs, $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$, each capable of serving the same video content.
- **MPD file size (s_{MPD}):** The size of the MPD file (in bytes). It is typically small but constant, making it ideal for lightweight probing.
- **Average segment size (s_{seg}):** The average size of a video segment, used to interpolate throughput for a segment-level view of CDN performance.

Metric Initialization

Each CDN maintains two rolling metric arrays:

- **RTT[c]:** Stores historical RTT samples for CDN c.
- **Throughput[c]:** Stores throughput estimates for CDN c.

Initially,

$$\text{RTT}[c] \leftarrow \emptyset, \quad \text{Throughput}[c] \leftarrow \emptyset$$

This setup ensures that as the streaming session begins, metrics can be populated dynamically and smoothed over time.

5.3.3 Parallel MPD Request Handling

Parallelism for Real-Time Responsiveness

The core of the algorithm lies in the parallel execution of MPD requests. For each CDN $c \in \mathcal{C}$, an independent MPD request is issued. This enables simultaneous observation of network responsiveness across all CDNs, minimizing measurement latency.

RTT Measurement

The RTT represents the total delay from sending a request to receiving the MPD response:

$$\text{RTT}_c = t_{\text{response}} - t_{\text{request}}$$

RTT provides insights into network latency and serves as an indicator of congestion or propagation delay.

Throughput Estimation

Throughput is estimated using the MPD file size and RTT:

$$\text{T}_c = \frac{s_{\text{MPD}}}{\text{RTT}_c}$$

This approximation assumes that the MPD fetch is dominated by network transfer latency. While coarse, it gives a reliable snapshot of each CDN's instantaneous delivery rate.

5.3.4 Metric Interpolation and Updating

Interpolation Based on Segment Size

Since MPD sizes are much smaller than typical video segments, direct throughput estimates can underestimate CDN capability. Hence, interpolation is performed:

$$\text{Interpolated Throughput}_c = \text{T}_c \times \frac{s_{\text{seg}}}{s_{\text{MPD}}}$$

This normalization aligns the MPD-derived measurements with actual segment download expectations, improving estimation fidelity.

Metric Update Mechanism

After each measurement:

$$\text{RTT}[c] \leftarrow \text{RTT}[c] \cup \{\text{RTT}_c\}, \quad \text{Throughput}[c] \leftarrow \text{Throughput}[c] \cup \{\text{T}_c\}$$

This cumulative update enables the system to capture temporal variations and trends in CDN performance over time.

5.3.5 Temporal Smoothing

Instantaneous network measurements often fluctuate due to transient congestion or routing changes. To counteract this, the algorithm applies temporal smoothing over the last N observations:

$$\text{RTT}_{\text{avg}}[c] = \text{Mean}(\text{RTT}[c][-N:]), \quad \text{Throughput}_{\text{avg}}[c] = \text{Mean}(\text{Throughput}[c][-N:])$$

This stabilizes metric trends, ensuring that CDN selection decisions are not overly sensitive to short-lived anomalies. The value of N controls the responsiveness–stability trade-off.

5.3.6 CDN Selection Logic

With updated and smoothed metrics, the algorithm selects the optimal CDN for the next streaming segment. Typically, this is based on maximizing throughput while minimizing latency:

$$C^* = \arg\max_{c \in \mathcal{C}} \left(\frac{\text{Throughput}_{\text{avg}}[c]}{\text{RTT}_{\text{avg}}[c]} \right)$$

This ensures that the chosen CDN maintains both high delivery rate and low latency, optimizing user Quality of Experience (QoE).

5.3.7 Adaptive MPD Fetch Timing

The frequency of MPD probing depends on the streaming mode:

- **LLL Mode:** Fetch MPD every 200 ms to maintain responsiveness in live conditions.
- **VoD Mode:** Fetch MPD every 500 ms, since VoD allows relaxed timing due to buffering.

This adaptive rate minimizes overhead while maintaining high situational awareness across CDNs.

5.3.8 Termination Condition

The loop continues while the streaming session remains active. Upon completion, metrics are finalized or cleared, and the smoothed statistics can be logged for future analysis.

5.3.9 Algorithmic Complexity and Overhead

Each MPD request is executed in parallel, resulting in an effective per-iteration time complexity of O(1); however, the overall iteration time is determined by the response delay of the slowest CDN. The space complexity is $O(|\mathcal{C}| \cdot N)$, determined by the number of CDNs and stored samples. For practical configurations (e.g., four CDNs and 10–20 samples), the computational and storage overhead remains negligible, making the algorithm well-suited for real-time deployment in adaptive bitrate (ABR) streaming frameworks.

Chapter 6

Evaluation Testbed and Performance

6.1 StreamWise Setup

6.1.1 Experimental Design

All experimental simulations are conducted in a fully containerized testbed, building upon the Vegvisir framework to orchestrate clients, CDNs, and network shapers within isolated Docker environments. This ensures a controlled, reproducible setting for evaluating adaptive streaming under multi-CDN scenarios. Extensive network trace datasets, representing a wide spectrum of real-world conditions (including cellular, WiFi, and congestion-prone paths), drive end-to-end session emulation. Clients use a headless DASH implementation (goDASH [34] over QUIC [35]via mvfst [36]), while network shaping is handled dynamically by tc-netem. To empirically evaluate the StreamWise model under realistic, online conditions, we leveraged the Vegvisir framework [37]. Vegvisir provides a robust and flexible testbed by creating a fully integrated ecosystem where individual components—such as CDN servers, clients, and network shapers—are deployed within isolated Docker containers. We extended the base Vegvisir framework to support the specific requirements of our content-steering research, with key enhancements including:

- **Support for Multiple Topologies:** The testbed was designed to model different real-world deployment scenarios. This includes a *Multi-Server Single-Client (MS-SC)* setup, where multiple CDNs connect to a single client through a single bottleneck link, and a *Multi-Server*

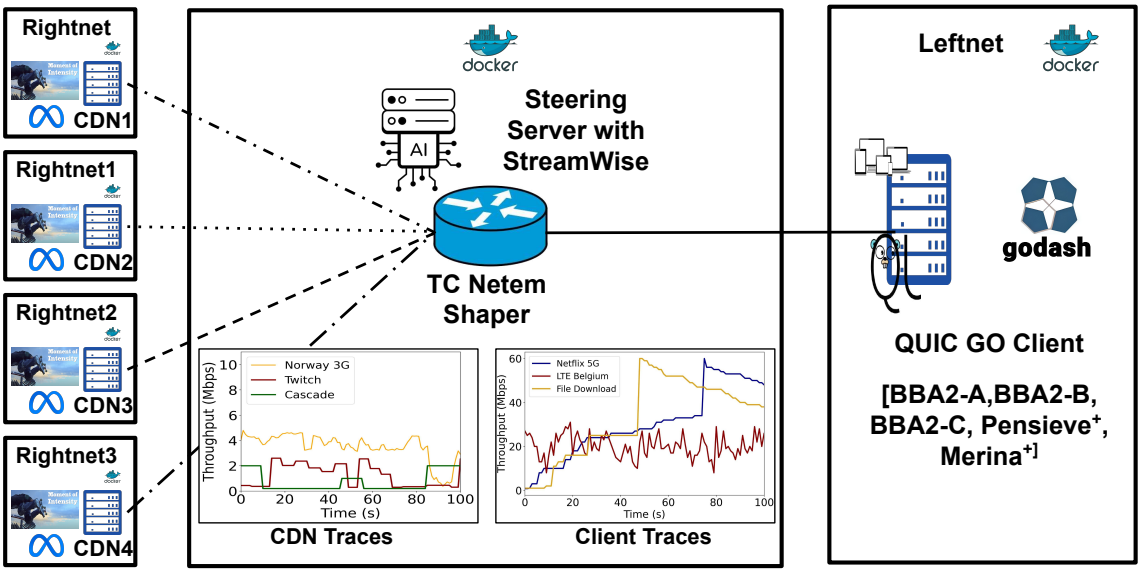

Figure 6.1: StreamWise single-agent evaluation framework illustrating four CDN servers (Rightnet1–4) connected to a QUIC-based goDASH client through a TC Netem [1] traffic shaper. The steering server, running StreamWise, dynamically selects the optimal CDN based on real-time network conditions. CDN and client traces represent the varying throughput patterns used to emulate heterogeneous network environments during evaluation.

Multi-Client (MS-MC) setup, which models more complex mappings (one-to-one or one-to-many) between CDNs and clients, with a shared shaper as the link bottleneck. (Note that in this context, "server" refers to a CDN.)

- **Integration of CMCD/CMSD Protocols:** To facilitate intelligent steering, we implemented the CMCD [11] and CMSD [38] standards. These protocols enable the real-time collection of crucial performance statistics from the clients and CDNs, respectively, providing the essential data feed for the steering logic.

- **Implementation of Content Steering with StreamWise:** The core of our contribution is the integration of the StreamWise algorithm as the content-steering server, which is positioned at the shaper side. This server continuously analyzes the incoming CMCD and CMSD data to make dynamic, data-driven CDN selection decisions.

The resulting end-to-end streaming architecture is depicted in Figure 6.2. The process begins when a client retrieves a MPD file from the origin server. This manifest is specially crafted to inform the client of multiple available CDNs by including several base URLs.

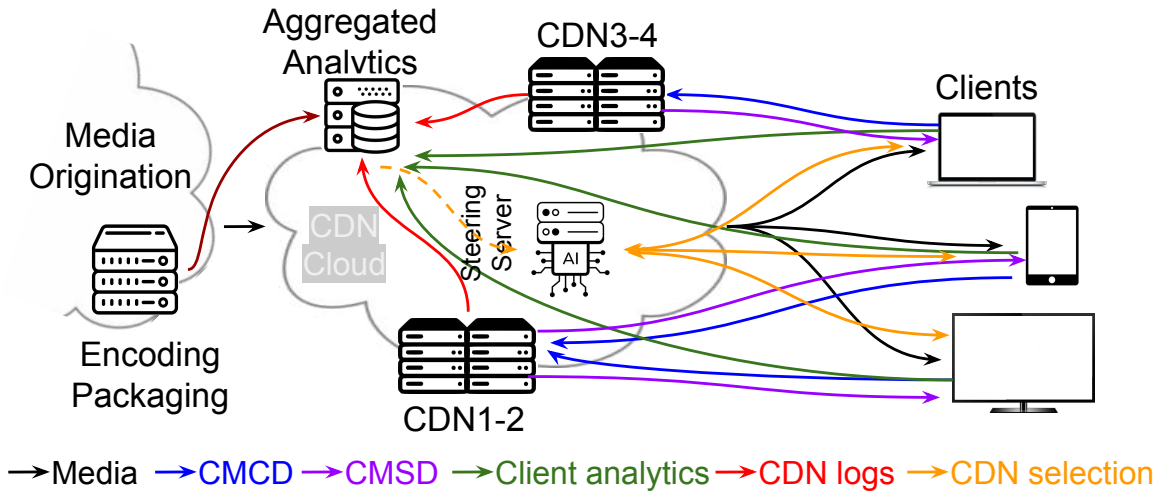

Figure 6.2: The end-to-end streaming architecture with StreamWise as the steering server.

During a streaming session, the content steering server continuously gathers real-time statistics from both clients (via CMCD) and CDNs (via CMSD). This data is fed into the StreamWise model, which computes the optimal CDN for each client. The selected CDN is then communicated to the client through an HTTP response header. This directive ensures that the client's subsequent video segment requests are forwarded to the best-performing CDN, thereby dynamically optimizing QoE based on prevailing network and server conditions.

Offline training for StreamWise's neural network model is conducted using a strictly partitioned set of experience trajectories, entirely disjoint from the traces and content employed during evaluation. Training hyperparameters are optimized through Bayesian search [39], with the selected configurations summarized in Table 6.1 to ensure reproducibility. The discount factor $\gamma = 0.95$ reflects a balanced preference toward near-term rewards while still valuing future returns—a crucial tradeoff in video streaming scenarios that demand adaptive, long-term quality optimization. Learning rates for the actor (2×10^{-4}) and critic (1×10^{-3}) networks were determined to best stabilize policy and value function updates, reflecting the established practice that the critic should learn faster to provide reliable value estimates. The PPO clip parameter $\epsilon = 0.21$ was chosen to constrain the policy update ratio within a range that prevents overly large, destructive updates while maintaining sufficient adaptability, consistent with widely accepted PPO tuning heuristics. Finally, the entropy coefficient $\delta = 1.9$ was empirically set to promote adequate exploration during training, thereby

Table 6.1: StreamWise training/testing parameters.

	Parameter	Symbol	Value
A2C	Discount Factor	γ	0.95
	Advantage estimator	λ	0.9
	Actor & Critic learning rates	ω, $\bar{\omega}$	2 x 10^{-4}, 1 x 10^{-3}
PPO	Clip parameter	ϵ	0.21
	Entropy Coefficient	δ	1.9
	PPO policy update interval, steps	T_θ, n_s	1000, 9
NN	Batch size	B_s	64
	Exploration size, number of episodes	θ, n_e	10, 30
	Number of epochs	itr	90000

enhancing policy robustness and preventing premature convergence to suboptimal behaviors. The trained model is subsequently frozen and integrated into the testbed for live decision-making.

6.1.2 Methodology

This section details the comprehensive experimental framework designed to evaluate the performance of the proposed StreamWise system. The setup was crafted to facilitate a rigorous and fair comparison under scenarios representative of both VoD and LLL streaming, encompassing the definition of video content, network conditions, benchmark algorithms, and performance metrics.

Content and Network Configuration

For the evaluation, we used the Moment of Intensity (MOI) video sequence [40], chosen for its dynamic spatial and temporal complexity, which presents a challenging use case for adaptive streaming algorithms. The content was encoded using AVC (H.264) [41] into a wide quality ladder of ten representations, spanning from 288p at 0.5 Mbps to 2160p (4K) at 40 Mbps, enabling the testing of fine-grained quality decisions.

Two primary streaming scenarios were defined, as summarized in Table 6.2:

- **LLL:** Characterized by a short 2-second segment duration and a constrained 12-second playback buffer to minimize end-to-end latency.
- **VoD:** Employed a conventional 4-second segment duration and a larger 60-second buffer, allowing for greater client-side flexibility.

Table 6.2: Experimental Scenarios and Configuration Parameters

Scenario	VMAF Range	Frame Rate	Bitrate Range (Mbps)	Segment Duration (s)	Max Buffer (s)
A: LLL	37–96	60	0.5–40	2	12
B: VoD	37–96	60	0.5–40	4	60

To emulate diverse real-world conditions, seven publicly available bandwidth traces were used. These were selected to cover a broad spectrum of average bandwidths, variability, and access technologies, including mobile traces (Norway 3G [42], F4G5G [43]) and fixed-line scenarios (Twitch [44], Cascade [45]). Traces were applied both at the CDN side to model last-mile bottlenecks and at the client side to simulate core network congestion.

Benchmark Algorithms

The performance of StreamWise was evaluated against a suite of benchmark algorithms, covering both ABR and CDN switching strategies.

ABR Algorithms. Three state-of-the-art ABR algorithms were selected:

- **BBA2-XLDouble:** A buffer-based heuristic [46] enhanced with a proactive stall-prevention mechanism.
- **Pensieve$^+$** and **Merina$^+$:** Enhanced versions of the learning-based Pensieve [47] and Merina [48] algorithms. Both were retrained for the 10-bitrate ladder used in this study and integrated with BBA2-XLDouble's [46] rebuffering-prevention logic to ensure a fair and stable comparison.

CDN Switching Strategies. StreamWise was compared against a range of heuristic CDN selection approaches:

- **Static Heuristics:** Including Random, Deterministic (fixed allocation ratios), and Round-Robin as baseline strategies with no network awareness.
- **Dynamic Heuristics:** Two simple, network-aware strategies (DH1: highest throughput; DH2: lowest RTT) that leverage the same real-time CDN metrics as StreamWise, serving as direct competitors to isolate the benefit of its sophisticated decision-making.

Performance Metrics

A multi-faceted evaluation was conducted using four key metrics to capture different aspects of the QoE:

- **Perceptual Video Quality:** Measured using VMAF [49] and PSNR [50], with emphasis on the perceptually-oriented VMAF.
- **Ultra-HD Delivery:** The number of segments downloaded in ultra-HD quality (nSeg-UHD) directly measures high-quality service attainment.
- **Overall QoE Score:** The Yin QoE Model [51]) was employed to compute a holistic score. It integrates average bitrate, bitrate switches, and rebuffering duration into a single normalized score (1–5), providing a comprehensive assessment.

The testbed was deployed on a physical machine running Ubuntu 22.04.3 LTS. To ensure isolation and reproducibility, all system components—client video players, CDN servers, and a network traffic shaper—were containerized using Docker and interconnected via Docker Compose networks. Each network trace emulation was executed twice with a duration of approximately 120 seconds, and the results presented are the average of these runs to ensure statistical reliability.

6.1.3 Training Regime and ABR Generalization

The use of Pensieve ABR for generating training trajectories for both StreamWise and Cadence frameworks was a deliberate methodological choice to enforce policy generalization and avoid overfitting to a specific ABR algorithm's heuristic behavior. Unlike conventional algorithms that operate on fixed rules, Pensieve's reinforcement learning-based policy produces a diverse and non-stationary distribution of network interactions. By training our CDN selection agents on these trajectories, we force them to learn from the underlying environmental fundamentals—throughput, latency, and buffer dynamics—rather than the output patterns of a deterministic ABR. This approach ensures that the learned policy is conditioned on the core state variables common to all ABR decision processes, resulting in a CDN selection strategy that is fundamentally agnostic to the client-side bitrate logic and generalizes effectively across different algorithms.

6.1.4 Results and Analysis

MS-SC Simulation

Scenario A: LLL Performance

In the constrained LLL environment, where a low buffer ceiling challenges ABR algorithms, StreamWise's dynamic optimization proved particularly impactful. As evidenced in Table 6.3, StreamWise consistently delivered the best trade-off between high quality and uninterrupted playback. The reported values include the mean and standard deviation across all experimental runs, reflecting performance stability under varying traces and ABR interactions.

For instance, with the LTE Belgium trace [52] and Pensieve^{+} ABR (i.e Pensieve ABR with stall predictor enabled), StreamWise outperformed all competitors, achieving a 7% higher VMAF and enabling 22% more segments to be downloaded in Ultra-HD (UHD). This was a direct result of its ability to select the optimal CDN, which boosted the average bitrate by $1.5\times$ while simultaneously minimizing rebuffering. While static strategies like Random and Round-Robin achieved comparable VMAF scores in some instances, they suffered from significantly more rebuffering, severely degrading the QoE. Deterministic-2 provided a smooth playback but at the cost of unacceptably low video quality, lagging behind StreamWise by a $3\times$ Just Noticeable Difference (JND) [53] in VMAF.

The limitations of simpler dynamic heuristics were also exposed. DH1 and DH2, while matching StreamWise's VMAF in some cases, incurred $\sim 400\%$ more rebuffering. This critical flaw stems from their reliance on a single network metric (throughput or RTT), which prevents them from effectively anticipating and preventing buffer depletion. This pattern held across other ABR algorithms and traces. With Merina^{+}, an inherently more aggressive ABR, DH1's throughput-based strategy initially fetched more UHD segments but frequently triggered stall recovery mechanisms, causing severe quality oscillations and ultimately lower average VMAF. Even with the conservative BBA2-XLDouble ABR, which struggled in the LLL setting, StreamWise's superior CDN steering pushed the quality one level beyond Full HD, achieving a VMAF improvement equivalent to $1\times$ JND over other strategies. Results for the FD [43] and Netflix 5G traces further validated this trend, with StreamWise showing collective VMAF improvements of $\sim 9\%$.

Scenario B: Video-on-Demand (VoD) Performance

Table 6.3: Average results of the QoE and its metrics for different network traces for LLL scenario and content MOI (2s). ↑: higher is better (green), ↓: lower is better (green), lowest performance (red). ± indicates the standard deviation of the respective metric over the averaged results.

			Avg. Perceptual Quality (VMAF)	Avg. Perceptual Quality (PSNR)	Avg. Selected Bitrate (Mbps)	Average RD (%)	Average Yin	Average nSeg-UHD (%)
LTE Belgium	Pensieve+	**StreamWise**	**94.14** ±1.86 ↑	**50.82** ±3.99↑	**30.17** ±6.00↑	0.09 ±0.88	5.00↑	**86.27**↑
		Random	89.60 ±1.16	49.72 ±3.09	23.50 ±6.69	7.21 ±16.53	5.00↑	61.54
		Deterministic-1	89.63 ±1.15	49.67 ±2.84	25.19 ±4.30	1.80 ±2.78	5.00↑	65.91
		Deterministic-2	69.39 ±0.55	45.40 ±2.94	10.38 ±9.80	**0.00**↓	1.00	50.00
		Round-Robin	92.74 ±0.96	50.66 ±3.53	27.69 ±7.50	8.90 ±19.22	5.00↑	61.54
		Least connections	88.70 ±1.15	49.51 ± 2.84	23.93 ±6.00	6.80 ±23.08	1.00	49.36
		DH1	90.84 ±1.36	50.13 ±7.82	27.22 ±5.60	4.40 ±0.37	5.00↑	78.57
		DH2	93.07 ±2.32	50.74 ±4.43	29.42 ±6.24	3.00 ±1.36	5.00↑	82.00
	Merina+	**StreamWise**	**96.84** ±1.64↑	**51.43** ±2.22↑	**36.00** ±6.50↑	**2.87** ±4.32↓	5.00	**97.67**↑
		Random	92.13 ±0.82	50.75 ±3.79	29.00 ±10.44	4.82 ±5.34	5.00	91.67
		Deterministic-1	93.26 ±0.87	50.75 ±3.27	29.21 ±10.37	3.71 ±4.10	5.00	91.89
		Deterministic-2	90.43 ±0.65	51.40 ±3.09	27.11 ±7.23	12.54 ±20.25	5.00	90.97
		Round-Robin	94.43 ±0.65	51.01 ±3.09	31.41 ±9.47	13.18 ±18.41	5.00	93.94
		Least connections	94.43 ±0.65	51.01 ±3.09	31.41 ±9.47	12.61 ±18.35	5.00	93.94
		DH1	78.54 ±0.96	47.96 ±4.76	12.00 ±9.88	7.74 ±4.23	5.00	63.16
		DH2	92.85 ±0.89	48.42 ±2.86	33.27 ±6.32	3.13 ±3.69	5.00	92.87
	BBA2-XLDouble	**StreamWise**	**86.5** ±1.68↑	**48.94** ±2.85↑	**13.39** ±5.44↑	0.00	5.00	-
		Random	81.94 ±1.14	47.09 ±2.17	6.41 ±2.94	0.00	5.00	-
		Deterministic-1	74.36 ±2.06	45.71 ±4.02	4.32 ±2.18	0.00	5.00	-
		Deterministic-2	81.44 ±4.19	46.90 ±2.58	8.17 ±3.83	0.00	5.00	-
		Round-Robin	80.47 ±1.28	46.76 ±2.71	6.58 ±2.60	0.00	5.00	-
		Least connections	80.18 ±0.95	46.51 ±2.61	5.89 ±2.01	0.00	5.00	-
		DH1	83.77 ±1.35	47.55 ±2.70	9.32 ±4.77	0.00	5.00	-
		DH2	80.67 ±2.14	46.76 ±4.27	5.51 ±2.41	0.00	5.00	-
FD	Pensieve+	**StreamWise**	**94.74** ±1.02↑	**51.04** ±3.03↑	**31.33** ±5.71↑	**0.18** ±0.86↓	5.00	**90.16**↑
		Random	78.67 ±0.64	46.78 ±2.60	6.15 ±4.33	13.16 ±18.73	5.00	-
		Deterministic-1	88.70 ±1.15	49.51 ±2.84	23.93 ±6.00	6.80 ±23.08	5.00	49.36
		Deterministic-2	82.84 ±0.43	47.30 ±1.79	8.71 ±3.50	19.42 ±27.14	5.00	45.18
		Round-Robin	89.46 ±1.09	50.56 ±3.73	27.50 ±7.50	9.84 ±14.69	5.00	52.78
		Least connections	82.57 ±1.00	48.45 ±2.68	21.10 ±8.10	22.30 ±28.26	5.00	42.50
		DH1	93.71 ±0.50	50.78 ±1.92	29.44 ±5.19	4.60 ±2.33	5.00	80.65
		DH2	90.71 ±1.72	49.97 ±2.32	28.83 ±5.80	3.60 ±2.29	5.00	64.00
	Merina+	**StreamWise**	**96.84** ±1.64↑	**51.43** ±2.22↑	**36.00** ±6.50↑	**1.73** ±6.25↑	5.00	**97.67**↑
		Random	93.43 ±2.94	49.50 ±3.09	30.90 ±9.80	16.03 ±25.89	5.00	92.97
		Deterministic-1	94.98 ±0.96	51.01 ±4.76	32.25 ±9.00	7.39 ±13.38	5.00	93.30
		Deterministic-2	90.43 ±0.65	51.40 ±3.09	27.11 ±7.20	12.54 ±20.25	5.00	90.97
		Round-Robin	94.89 ±0.87	50.99 ±3.27	32.10 ±9.10	10.51 ±19.69	5.00	94.59
		Least connections	89.29 ±0.89	50.08 ±2.86	9.09 ±6.65	14.84 ±20.75	5.00	85.11
		DH1	79.47 ±1.39	48.16 ±2.81	12.73 ±10.15	8.06 ±3.31	5.00	65.00
		DH2	92.48 ±0.64	48.37 ±4.28	32.22 ±7.15	2.68 ±6.18	5.00	92.06
	BBA2-XLDouble	**StreamWise**	**89.46** ±1.64↑	**49.11** ±3.16↑	**13.34** ±7.8↑	0.00	5.00	-
		Random	80.39 ±1.08	46.86 ±2.74	7.14 ±2.82	0.00	5.00	-
		Deterministic-1	79.05 ±2.51	47.12 ±6.43	8.46 ±6.29	0.00	5.00	-
		Deterministic-2	75.98 ±2.06	46.00 ±4.02	4.85 ±2.44	0.00	5.00	-
		Round-Robin	80.67 ±2.14	46.76 ±4.27	5.51 ±2.41	0.00	5.00	-
		Least connections	80.18 ±0.95	46.51 ±2.61	5.89 ±2.00	0.00	5.00	-
		DH1	85.83 ±1.55	48.24 ±2.66	11.35 ±6.10	0.00	5.00	-
		DH2	85.66 ±1.82	48.08 ±8.6	11.56 ±6.12	0.00	5.00	-

The VoD scenario, with its larger buffer, tests the system's ability to maximize quality without the acute pressure of imminent stalls. As shown in Table 6.4, StreamWise again demonstrated consistent superiority.

For the Netflix 5G trace with Pensieve$^+$, StreamWise secured an 8.5% higher VMAF on average and maintained zero rebuffering. The static heuristics (Deterministic, Round-Robin) clustered together with middling performance, while Random and DH1 lagged significantly. Notably, DH2 performed comparably to StreamWise in VoD, as the larger buffer mitigated its single-metric limitation. However, this performance is not reliable across all conditions. The results were consistent with Merina$^+$ and BBA2-XLDouble, with StreamWise consistently achieving the highest VMAF and PSNR.

This superiority was further confirmed with the LTE Belgium trace, where StreamWise delivered a 7.5% VMAF improvement and a $1.5\times$ higher average bitrate. A key observation was with BBA2-XLDouble ABR, where StreamWise leveraged periods of high throughput to download a greater number of UHD segments than when tested with the Netflix 5G trace, showcasing its ability to capitalize on favorable network conditions. The results for the FD trace, summarized in Figure 6.3 (right), align perfectly with this narrative, reinforcing StreamWise's dominance.

MS-MC simulation

Scenario A: LLL with 10 clients

In the challenging LLL scenario, StreamWise maintained its dominance. As shown in Table 6.5, for the LTE Belgium trace with Merina$^+$, StreamWise achieved a $\sim 4\%$ higher VMAF and the highest PSNR and Yin scores, all while ensuring minimal rebuffering. Static strategies and DH1 produced comparable video quality but suffered from significantly more stalls. DH2, however, proved inadequate at scale, lagging in VMAF by a full JND and incurring $3\times$ more rebuffering than StreamWise due to its simplistic reliance on RTT. This pattern was reinforced with the FD trace, where StreamWise delivered a $\sim 5\%$ VMAF improvement and a $1.3\times$ higher average bitrate. Collectively, in the scaled LLL setup, StreamWise provided an average 4% VMAF improvement with $5\times$ less rebuffering.

Scenario B: VoD with 10 clients

Table 6.4: Average results of the QoE and its metrics for different network traces for VoD scenario and content MOI (4s). ↑: higher is better (green), lowest performance (red). ± indicates the standard deviation of the respective metric over the averaged results.

			Avg. Perceptual Quality (VMAF)	Avg. Perceptual Quality (PSNR)	Avg. Selected Bitrate (Mbps)	Average RD (%)	Average Yin	Average nSeg-UHD (%)
Netflix 5G	Pensieve+	**StreamWise**	**92.13** ±0.80 ↑	49.46 ±2.40	12.44 ±7.01	0.00	5.00	47.62
		Random	78.23 ±1.68	47.02 ±6.41	6.05 ±5.43	0.00	5.00	10.71
		Deterministic-1	89.38 ±0.72	49.28 ±2.51	10.61 ±6.37	0.00	5.00	35.00
		Deterministic-2	87.38 ±0.64	49.02 ±3.82	10.16 ±6.39	0.00	5.00	38.46
		Round-Robin	88.59 ±0.72	49.32 ±2.51	11.26 ±5.67	0.00	5.00	30.00
		Least connections	86.22 ±0.79	48.22 ±4.32	9.86 ±6.60	0.00	5.00	35.90
		DH1	73.49 ±1.28	46.83 ±6.28	8.45±7.40	0.00	5.00	32.14
		DH2	91.66 ±1.28	**50.08** ±6.28 ↑	**18.46** ±5.51 ↑	0.00	5.00	**51.14** ↑
	Merina+	**StreamWise**	**95.66** ±2.22 ↑	**51.24** ±5.09 ↑	**34.14** ±7.88 ↑	**2.42**±0.74 ↓	5.00	**96.15** ↑
		Random	88.80 ±2.27	49.20 ±3.72	28.60 ±10.80	12.20±21.36	5.00	80.26
		Deterministic-1	92.20 ±1.24	50.62 ±4.12	28.17 ±10.59	4.80±2.22	5.00	90.48
		Deterministic-2	91.87 ±1.32	50.52 ±3.33	27.76 ±10.69	9.61±4.32	5.00	90.00
		Round-Robin	92.20 ±1.24	50.62 ±4.12	28.17 ±10.59	10.20±4.72	5.00	90.48
		Least connections	92.20 ±1.24	50.62 ±4.12	28.17 ±10.59	9.95±4.45	5.00	90.48
		DH1	94.80 ±0.91	51.00 ±3.72	32.33 ±8.89	5.16±2.56	5.00	94.44
		DH2	92.19 ±1.32	49.05 ±3.33	30.89 ±8.60	2.72±0.88	5.00	93.00
	BBA2-XLDouble	**StreamWise**	**89.83** ±1.68 ↑	**49.11** ±4.39 ↑	**11.24** ±5.55 ↑	0.00	5.00	**18.92** ↑
		Random	83.50 ±1.06	48.09 ±3.84	7.81 ±5.69	0.00	5.00	3.51
		Deterministic-1	75.83 ±3.72	46.18 ±5.87	5.19 ±3.86	0.00	5.00	1.72
		Deterministic-2	74.74 ±6.08	45.49 ±3.66	4.24 ±2.29	0.00	5.00	3.64
		Round-Robin	77.89 ±0.00	46.01 ±0.00	3.91 ±1.90	0.00	5.00	3.51
		Least connections	77.96 ±1.93	46.12 ±3.68	4.37 ±2.23	0.00	5.00	-
		DH1	87.26 ±1.39	48.49 ±3.08	9.01 ±4.79	0.00	5.00	7.27
		DH2	87.30 ±1.83	48.44 ±3.39	8.94 ±4.36	0.00	5.00	-
LTE Belgium	Pensieve+	**StreamWise**	**89.11** ±0.72↑	**49.21** ±2.51↑	**17.28** ±6.00↑	**0.00** ↓	**5.00** ↑	**60.71**↑
		Random	84.58 ±9.07	47.94 ±6.49	11.83 ± 5.84	**0.00** ↓	**5.00** ↑	34.32
		Deterministic-1	81.87 ±2.45	47.71 ±4.19	6.90 ±5.26	**0.00** ↓	**5.00** ↑	16.25
		Deterministic-2	81.62 ±1.68	47.94 ±4.39	7.87 ±6.04	**0.00** ↓	**5.00** ↑	18.11
		Round-Robin	82.87 ±0.89	48.17 ±3.95	13.20 ±7.71	4.44±1.88	1.00	10.42
		Least connections	88.93 ±1.11	48.79 ±4.23	9.81 ±4.98	10.26±4.74	1.00	22.78
		DH1	64.89 ±1.8	45.19 ±6.67	6.48 ±5.48	**0.00** ↓	**5.00** ↑	25.00
		DH2	84.13 ±0.86	47.43 ±2.52	8.60 ±6.45	**0.00** ↓	**5.00** ↑	36.84
	Merina+	**StreamWise**	**95.40** ±1.95 ↑	**51.16** ±3.40 ↑	**33.59** ↑ ±8.22	3.39±1.18	**5.00** ↑	**95.65** ↑
		Random	85.17 ±0.46	48.68 ±1.98	15.39 ±8.90	31.29±8.96	1.00	74.12
		Deterministic-1	94.54 ±0.65	51.00 ±4.19	32.01 ±9.05	8.06±4.12	**5.00** ↑	94.12
		Deterministic-2	93.68 ±1.21	50.66 ±8.23	30.83 ±9.56	13.95±21.85	**5.00** ↑	92.86
		Round-Robin	91.22 ±0.91	50.41±3.72	26.85 ±10.90	12.26±20.42	**5.00** ↑	88.89
		Least connections	91.59 ±0.93	50.45 ±4.37	27.33 ±10.80	11.84±19.76	**5.00** ↑	89.47
		DH1	94.80 ±0.91	51.00 ±3.72	32.33 ±8.89	5.07±1.56	**5.00** ↑	94.44
		DH2	91.49 ±1.32	48.60 ±3.33	30.46 ±8.60	**2.97**±1.04 ↓	**5.00** ↑	92.60
	BBA2-XLDouble	**StreamWise**	**87.30** ±1.83 ↑	**48.44** ±3.39 ↑	**8.94** ±4.36 ↑	0.00	5.00	**43.33** ↑
		Random	82.41 ±1.83	47.72 ±3.39	7.04 ±5.09	0.00	5.00	3.33
		Deterministic-1	75.15 ±3.72	45.72 ±5.87	4.25 ±2.40	0.00	5.00	25.49
		Deterministic-2	77.69 ±7.30	46.04 ±7.56	4.23 ±2.42	0.00	5.00	6.78
		Round-Robin	77.89 ±0.00	46.01 ±0.00	3.91 ±1.90	0.00	5.00	-
		Least connections	77.96 ±1.93	46.12 ±3.68	4.37 ±2.23	0.00	5.00	-
		DH1	86.23 ±1.81	48.43 ±3.55	8.82 ±5.61	0.00	5.00	23.64
		DH2	83.50 ±1.06	48.09 ±3.84	7.81 ±5.69	0.00	5.00	21.43

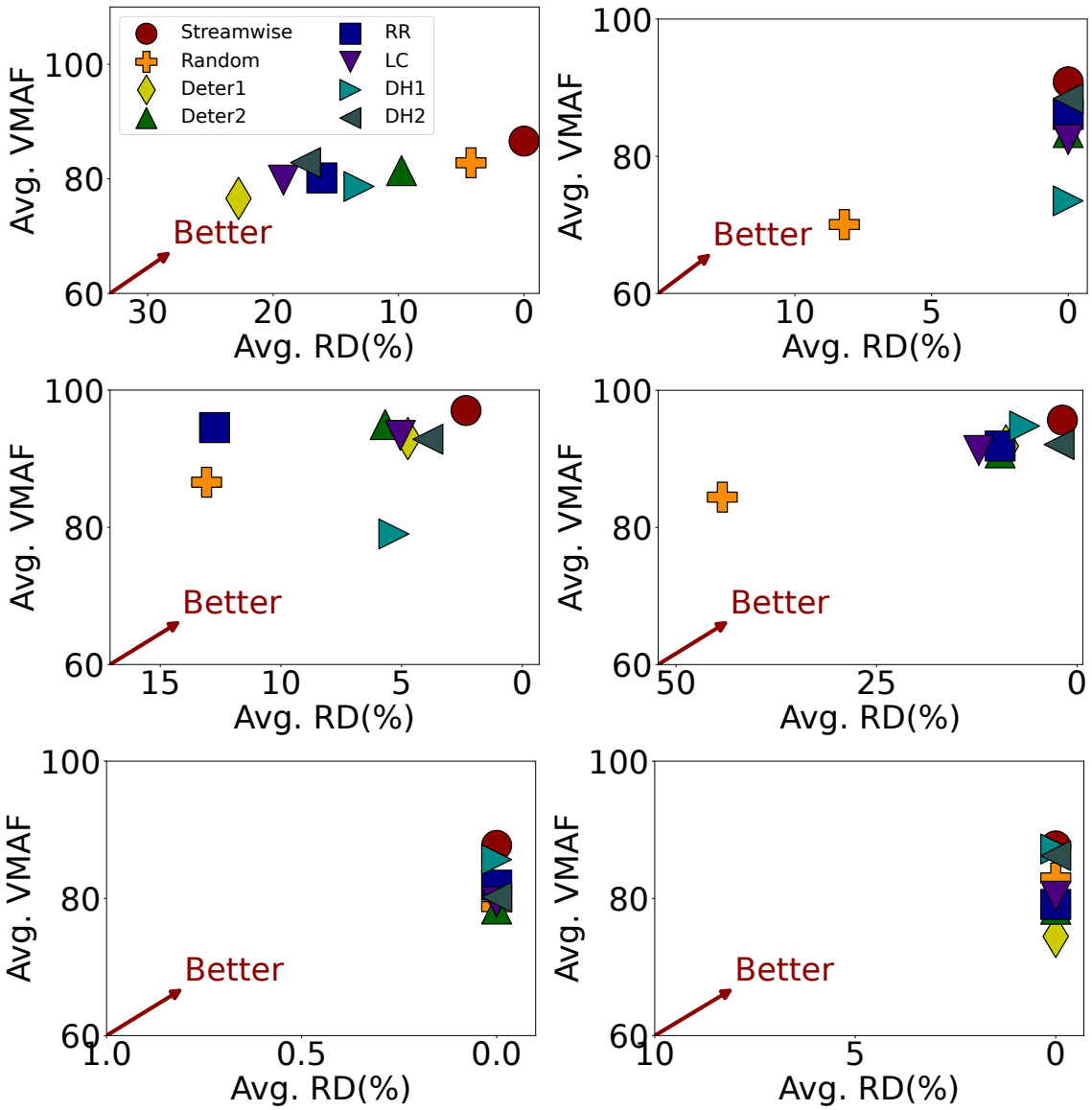

Figure 6.3: VMAF vs Average RD (%) analysis for ABR: Pensieve^{+} (top), Merina^{+} (center), and BBA2-XLDouble (bottom) in LLL mode for Netflix 5G trace (left) and in VoD mode for FD trace(right). *Better* indicates the direction of optimality

Table 6.5: Average results of the QoE and its metrics for different network traces and content MOI (2s). ↑: higher is better (green), ↓: lower is better (green), lowest performance (red), and scenario: LLL.± indicates the standard deviation of the respective metric over the averaged results.

		Avg. Perceptual Quality (VMAF)	Avg. Perceptual Quality (PSNR)	Avg. Selected Bitrate (Mbps)	Average RD (%)	Average Yin	Average nSeg-UHD (%)
LTE Belgium	**StreamWise**	**96.57** ±0.89 ↑	**51.35** ±2.54 ↑	**35.32** ↑ ±7.08	**2.91**±4.38 ↓	5.00	**97.14** ↑
	Random	93.45 ±1.06	50.70 ±4.90	30.15 ±9.54	19.11±26.94	5.00	92.26
	Merina$^+$ Deterministic-1	94.48 ±1.06	50.87 ±3.14	31.77 ±8.68	17.34±24.32	5.00	93.66
	Deterministic-2	92.20 ±1.78	50.42 ±6.15	28.08 ±10.29	15.82±22.94	5.00	90.09
	Round-Robin	94.05 ±0.84	50.84 ±5.30	30.85 ±9.70	20.64±18.06	5.00	93.39
	Least connections	94.86 ±1.05	51.00 ±6.49	32.29 ±8.89	16.56±17.14	5.00	94.59
	DH1	94.93 ±1.86	49.44 ±3.99	33.45 ±6.17	12.87±12.68	5.00	94.00
	DH2	89.76 ±0.81	48.15 ±5.40	25.16 ±11.19	8.87±4.54	5.00	86.67
FD	**StreamWise**	**96.95** ±0.84 ↑	**51.52** ±4.32 ↑	**36.39** ±6.17 ↑	**2.18**±3.98 ↓	5.00	**97.93** ↑
	Random	92.24 ±1.28	50.54 ±5.14	28.80 ±9.64	11.32±12.12	5.00	90.47
	Merina$^+$ Deterministic-1	94.58 ±1.26	50.85 ±3.23	31.95 ±8.60	10.82±11.74	5.00	93.85
	Deterministic-2	93.27 ±1.15	50.85 ±6.00	29.67 ±9.88	9.06±10.24	5.00	92.02
	Round-Robin	93.45 ±0.87	50.69 ±4.10	29.85 ±10.00	20.51±17.97	5.00	92.26
	Least connections	94.00 ±0.83	50.83 ±3.59	30.72 ±9.48	15.53±16.32	5.00	93.01
	DH1	91.99 ±1.26	49.60 ±2.88	28.65 ±5.98	8.40±9.82	5.00	92.15
	DH2	88.44 ±1.32	48.84 ±3.33	23.62 ±11.44	9.74±10.02	5.00	85.00

The VoD scenario further underscored StreamWise's robustness in a shared environment. For the Netflix 5G trace (Table 6.6), StreamWise demonstrated a $\sim$ 5.5% improvement in VMAF, delivering the highest PSNR, Yin score, and average bitrate with negligible rebuffering. While DH1 matched StreamWise's VMAF, it did so at an unacceptable cost, i.e suffering from $\sim 6\times$ more stall time. The LTE Belgium trace closely mirrored these results, with StreamWise again achieving a $\sim$ 5.5% VMAF gain, a $1.5\times$ higher average bitrate, and a $6\times$ reduction in rebuffering while downloading the most Ultra-HD segments.

Statistical Variability Analysis

StreamWise not only improves average video quality, achieving VMAF gains up to 7% over baselines, but also significantly reduces variability (i.e., standard deviation) in key metrics. For instance, on the LTE Belgium trace with Pensieve$^+$ ABR, StreamWise reports a VMAF standard deviation around 1.86, compared to similar or higher deviations in baseline methods. Likewise, average rebuffering times for StreamWise are near zero with very low standard deviation, whereas heuristic approaches exhibit much larger rebuffering averages and higher variability. This low variability is essential because user QoE is strongly influenced by stable, uninterrupted playback; fluctuations in quality or frequent buffering degrade viewer satisfaction even if average quality is comparable.

Table 6.6: Average results of the QoE and its metrics for different network traces and content MOI (4s). ↑: higher is better (green), ↓: lower is better (green), lowest performance (red), and scenario: VoD.± indicates the standard deviation of the respective metric over the averaged results.

			Avg. Perceptual Quality (VMAF)	Avg. Perceptual Quality (PSNR)	Avg. Selected Bitrate (Mbps)	Average RD (%)	Average Yin	Average nSeg-UHD (%)
Netflix 5G	**Merina+**	**StreamWise**	**96.07** ±1.73↑	**51.42** ±4.07↑	**34.89** ±7.39↑	**2.23** ±5.04↓	5.00	**96.79**↑
		Random	91.70 ±1.46	50.29 ±3.92	27.76 ±10.60	22.40 ±18.16	5.00	89.45
		Deterministic-1	88.99 ±1.88	49.91 ±3.55	25.18 ±10.40	16.37 ±17.12	5.00	85.55
		Deterministic-2	93.62 ±1.23	50.84 ±10.25	30.51 ±94.00	12.66 ±12.58	5.00	91.64
		Round-Robin	91.53 ±1.01	50.47 ±3.95	27.29 ±10.79	17.34 ±17.60	5.00	89.41
		Least connections	90.96 ±1.16	50.34 ±3.62	26.64 ±10.88	16.14 ±14.75	5.00	88.62
		DH1	95.75 ±1.18	51.18 ±4.99	34.17 ±7.84	12.93 ±5.66	5.00	93.37
		DH2	86.02 ±1.01	49.49 ±3.51	21.16 ±11.40	8.52 ±8.97	5.00	81.21
LTE Belgium	**Merina+**	**StreamWise**	**95.97** ±0.83↑	**51.26** ±3.31↑	**34.57** ±7.60↑	**2.98**↓ ±5.05	5.00	**96.53**↑
		Random	91.85 ±1.63	50.36 ±4.38	28.33 ±10.25	18.75 ±21.84	5.00	90.12
		Deterministic-1	89.71 ±1.34	49.97 ±5.84	25.69 ±10.70	23.76 ±22.65	5.00	86.63
		Deterministic-2	92.52 ± 1.17	50.53 ±5.43	29.51 ±9.51	21.58 ±21.99	5.00	90.84
		Round-Robin	90.84 ±1.09	50.38 ±6.92	26.42 ±10.97	21.53 ±20.65	5.00	88.33
		Least connections	90.65 ±0.99	50.34 ±7.11	26.48 ±10.74	20.51 ±20.78	5.00	88.14
		DH1	95.64 ±2.07	51.22 ±4.91	34.07 ±7.92	7.95 ±6.09	5.00	96.09
		DH2	85.62 ±1.73	49.21 ±3.49	20.73 ±11.43	3.93 ±27.25	5.00	80.00

By consistently delivering high-quality, smooth streams with minimal performance swings, StreamWise ensures a reliably superior viewing experience under diverse network conditions, making its improvements practically significant for end user QoE.

Comparative Analysis with the Optimal Bound

The ultimate measure of a practical system is its proximity to the theoretical optimum. In this work, the ideal performance ceiling, denoted as `OPT`, is computed via a dynamic programming model that exhaustively explores all possible CDN assignments and playback decisions over the entire session. This approach guarantees the highest attainable QoE under known network and buffer constraints, representing a strict upper bound on achievable performance. As shown in Figure 6.4, which aggregates results over diverse traces and ABR algorithms, our proposed solution achieves near-optimal performance, operating within just a few percentage points of the `OPT` benchmark.

In the demanding LLL scenario, StreamWise operates within a remarkable 2% of the `OPT` benchmark. Similarly, in the VoD scenario, it remains within 3% of the theoretical maximum. These results provide definitive evidence that StreamWise's DRL core is not merely effective but is capable of making highly intelligent, real-time decisions that closely approximate the bounds of achievable performance. This conclusively validates StreamWise as a precision-engineered solution

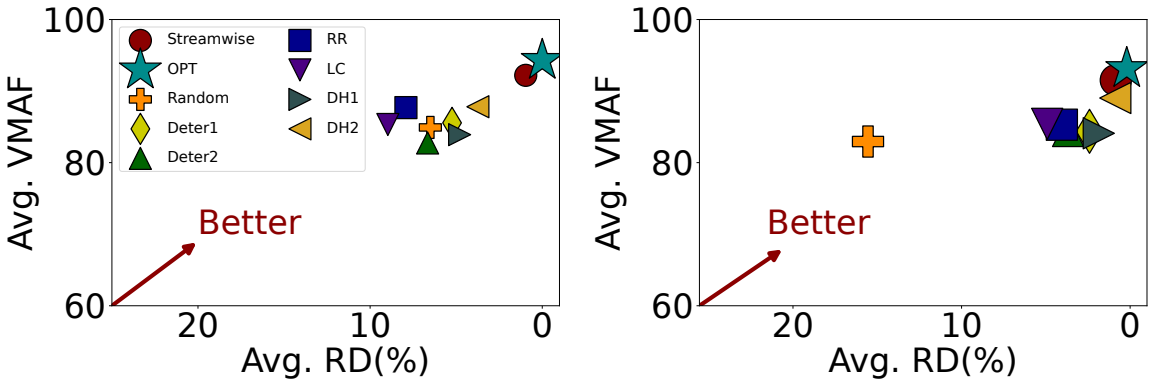

Figure 6.4: Theoretical Optimal (OPT) V/s other competitors: LLL (left) and VoD (right)

.

for optimal CDN selection in dynamic streaming environments.

CDN Switching Frequency

The system's adaptability is quantified by its CDN switching frequency. Across all trace and ABR combinations, StreamWise exhibited an average switching rate of approximately 18% in the LLL scenario and 21% in the VoD scenario, as visualized in Figure 6.5. This consistent activity demonstrates the policy's dynamic response to fluctuating network conditions, continually re-evaluating and selecting the optimal CDN server to maintain performance.

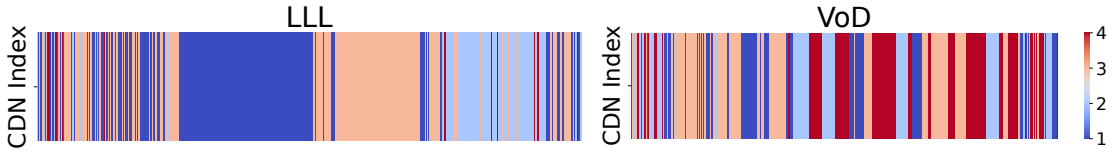

Figure 6.5: Average CDN switching frequency of StreamWise DRL across LLL and VoD modes. The results emphasize the stability of CDN selection decisions, especially under the LLL mode, while highlighting the preference for more cost-effective options in VoD scenarios.

6.1.5 Ablation Studies

CDN with different congestion control

The choice of congestion control (CC) algorithm on CDN back-end servers can significantly impact client Quality of Experience (QoE) [13].To evaluate this, we tested StreamWise in the MS-SC setup (Figure 6.1) using a Netflix 5G trace on the client side with various Adaptive Bitrate

(ABR) algorithms. The servers employed a range of CC algorithms, including BBR, BBR2, Copa2, and Cubic. For brevity, we present results using Merina^{+} as the ABR algorithm. As shown in Figure 6.6, StreamWise outperformed competing methods by approximately 3% in LLL mode and 10% in VoD mode in terms of VMAF score.

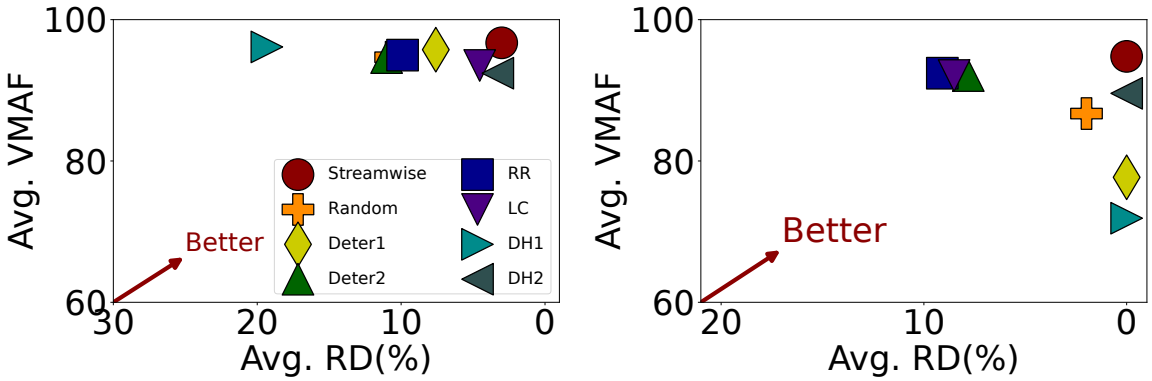

Figure 6.6: Comparison of congestion control (CC) analysis for LLL (left) and VoD (right) modes.

.

Videos with 8K resolution

Method	Avg. Bitrate (Mbps)	RD (%)
StreamWise	**52.17** ↑	1.32
DH1	33.67	3.86
DH2	33.21	3.86

Table 6.7: 8K UHD streaming analysis: StreamWise v/s Dynamic heuristics.

To evaluate the scalability [54] of StreamWise to ultra-high-definition streaming, we conducted experiments using 8K video content. The test video was the chosen to be Atomic Hearts 8K [55] encoded at 120 fps. Due to the demonstrably poor performance of static heuristics with 8K content, we decided to exclude them, and the analysis focused on dynamic methods. As summarized in Table 6.7, StreamWise achieved a significantly higher average bitrate—approximately 56% greater than competitors—while maintaining minimal rebuffering, confirming its efficacy for next-generation video formats.

6.2 StreamWise Integration to Dash.js

6.2.1 System Architecture and Integration

The integration of StreamWise with Dash.js [56] establishes a cohesive framework for intelligent CDN selection through four coordinated algorithms that enable real-time performance monitoring and learning-driven decision making. This architecture enhances traditional adaptive bitrate streaming by introducing server-guided intelligence while maintaining full compatibility with MPEG-DASH [57] standards.

Algorithm 1: Dynamic Request Interception and URL Rewriting serves as the primary integration point with the Dash.js player's request pipeline. This algorithm intercepts every segment request made by the player and orchestrates the complete CDN selection workflow. When invoked, it first calls the state tracking function (Algorithm 2) to capture current network conditions and CDN performance metrics. It then initiates the intelligent CDN selection process (Algorithm 4) which communicates with the DRL model server. Upon receiving the optimal CDN decision, the algorithm dynamically rewrites the request URL, service location, and query parameters to redirect the segment fetch through the selected CDN endpoint. This seamless URL manipulation occurs transparently to the core Dash.js player, ensuring compatibility while introducing intelligent routing capabilities that adapt to real-time network conditions.

Algorithm 2: Comprehensive Multi-Metric State Tracking maintains a real-time, multi-dimensional assessment of CDN performance across three critical metrics. The algorithm continuously monitors RTT through direct measurements from segment requests, providing accurate latency assessment. Throughput estimation is derived from segment size and transfer duration relationships, offering bandwidth capacity insights. Most importantly, the algorithm tracks stall duration by monitoring buffer levels and quantifying playback interruptions, directly capturing QoE impact. For the currently selected CDN, the algorithm uses actual measurement data, while for alternative CDNs, it employs intelligent interpolation based on periodic MPD fetches and historical performance patterns. A snapshot of the updated MPD is shown in Figure 6.7. Note that the `ContentSteering` field in the MPD refers to the default service location, which serves as both

the primary and fallback location if a segment request from the selected CDN fails. This comprehensive state tracking provides the DRL model with rich, temporally-aware contextual information for informed decision making.

```
<BaseURL serviceLocation="Location-1">
https://ftp.itec.aau.at/datasets/mmsys22/3DMark_Night_Raid/4sec/avc/</BaseURL>
<BaseURL serviceLocation="Location-2">http://localhost:7000/</BaseURL>
<BaseURL serviceLocation="Location-3">https://app.3krg.uk/</BaseURL>
<BaseURL serviceLocation="Location-4">http://localhost:5000</BaseURL>
<ContentSteering defaultServiceLocation="Location-2"
queryBeforeStart="true">http://localhost:7000/</ContentSteering>
```

Figure 6.7: An MPD snapshot depicting the CDN deployment across multiple geographical locations, with each BaseURL indicating a specific server endpoint from which the client can request video segments via HTTP call.

Algorithm 3: Proactive Parallel CDN Health Monitoring implements a sophisticated distributed assessment mechanism that evaluates all available CDNs concurrently. Unlike traditional approaches that only monitor the active CDN, this algorithm performs lightweight MPD requests to all CDNs in parallel, measuring actual RTT and deriving throughput estimates. CDNs exhibiting excessive latency (exceeding 1 second threshold) or complete request failures are automatically added to a dynamic blacklist, preventing further requests until recovery is detected. The algorithm continuously reassesses blacklisted CDNs, automatically reintegrating them when performance improves. This proactive monitoring ensures comprehensive network awareness, enabling preemptive switching decisions before performance degradation impacts user experience. The collected metrics undergo temporal smoothing using exponential weighted moving averages to reduce noise while maintaining responsiveness to genuine performance changes.

Algorithm 4: DRL-Guided Intelligent CDN Selection establishes the bridge between client-side adaptation logic and server-side deep reinforcement learning capabilities. This algorithm manages the WebSocket communication channel between the Dash.js client (JavaScript) and the StreamWise DRL server (Python). When invoked, it serializes the current CDN state information and transmits it to the server, where the PPO model processes the multi-dimensional input. The DRL model, trained on extensive streaming sessions, evaluates the trade-offs between throughput maximization, latency minimization, and stall avoidance to determine the optimal CDN selection. The

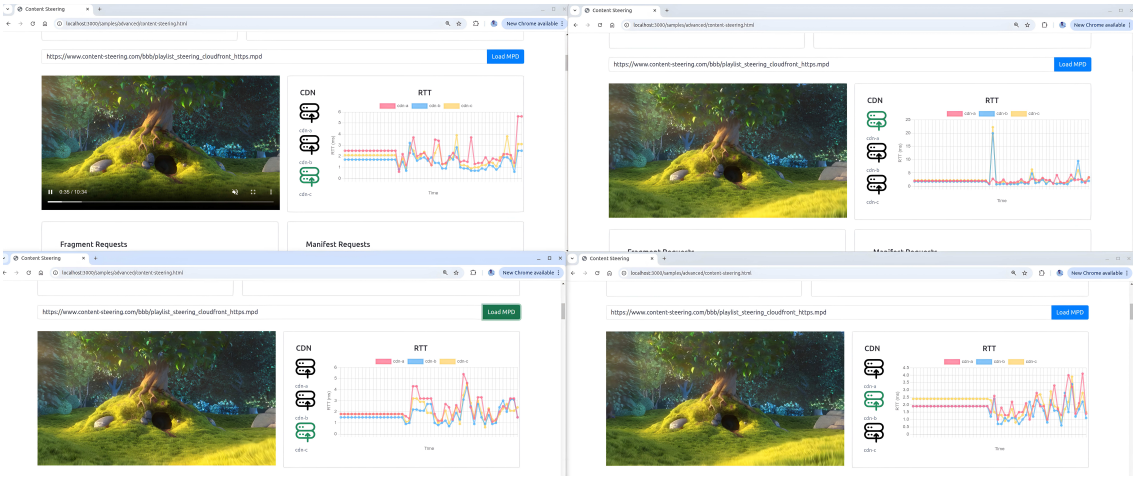

Figure 6.8: CDN switching visualization with real-time RTT tracking using Dash.js players integrated with StreamWise, streaming the BigBuckBunny 2K dataset.

server returns the decision containing the selected CDN's service location and endpoint URL, which the algorithm then integrates into the Dash.js player's media representation path.

Figure 6.8 illustrates the Dash.js player streaming the BigBuckBunny [58] video on all four clients, each integrated with StreamWise at the content-steering server. On the right, the real-time RTT tracking GUI enhancement is shown for each player, displaying the performance metrics across three CDN locations. These locations are defined in the Dash.js content steering server as `cdn-a`: Cloudfront, `cdn-b`: Fastly, and `cdn-c`: Akamai. As depicted in Figure 6.8, StreamWise guides all four clients to make the best CDN selection decisions (highlighted in green). This demonstrates how StreamWise can be seamlessly integrated into Dash.js and scaled to support multiple clients efficiently. Notably, the RTT tracking is part of the GUI enhancement to track each CDN health status and not the sole metric used by StreamWise for CDN selection.

6.2.2 Performance Evaluation

To validate the real-world efficacy of StreamWise, we conducted comprehensive experiments comparing multiple CDN-switching strategies under controlled yet realistic network conditions. Our evaluation framework employed a multi-CDN test environment with four geographically distributed locations, including institutional servers (ITEC), local university machines, and cloud-hosted instances (UK-based Cloudflare CDN [59]), with some locations proxied through 4G LTE

networks to emulate authentic network heterogeneity. Experiments utilized a 150-second video-on-demand session of 3DMark Night Raid content encoded with 15 bitrate levels (100 kbps to 30 Mbps) to thoroughly stress-test adaptation algorithms.

We implemented and compared four distinct CDN selection strategies against our DRL solution: the default single-CDN approach, a dynamic heuristic (DH) based on minimal RTT, round-robin (RR) cycling, and deterministic fixed allocation (Det). All strategies were evaluated using Dash.js's default Dynamic ABR algorithm, which combines BOLA [60] and throughput-based adaptation, ensuring a fair comparison across consistent adaptation logic.

Performance was assessed through multiple QoE metrics including average bitrate, quality switching frequency, CDN switching patterns, and perceptual quality using VMAF. The testbed configuration ensured no rebuffering events occurred, thereby focusing our evaluation specifically on quality optimization and stability aspects of CDN selection. This rigorous methodology provides a solid foundation for comparing StreamWise against established CDN selection approaches under realistic multi-CDN streaming conditions.

The experimental results demonstrate the clear superiority of the our novel framework. As illustrated in Figure 6.10, StreamWise outperforms all other CDN switching strategies across key QoE metrics. It achieves an approximately **78%** higher VMAF than the average of other strategies, selects a higher average bitrate, and ensures smoother playback with fewer quality switches. This performance is a direct result of its intelligent, DRL-based design, which dynamically adapts to real-time network conditions, unlike the reactive or static heuristics of other methods.

The Default strategy, while simple, suffered from frequent and abrupt quality switches. The Dynamic Heuristic (DH) approach, based solely on RTT, proved inadequate in the complex, fluctuating network environment of our testbed (Figure 6.9). The static Round Robin (RR) and Deterministic (Det) strategies failed to adapt in real-time, leading to suboptimal performance.

6.2.3 Results

Figure 6.10 demonstrates the superior performance of StreamWise across three critical QoE metrics. Our DRL-based approach achieves approximately 78% higher VMAF score compared to alternative CDN selection strategies, while simultaneously maintaining higher average bitrate

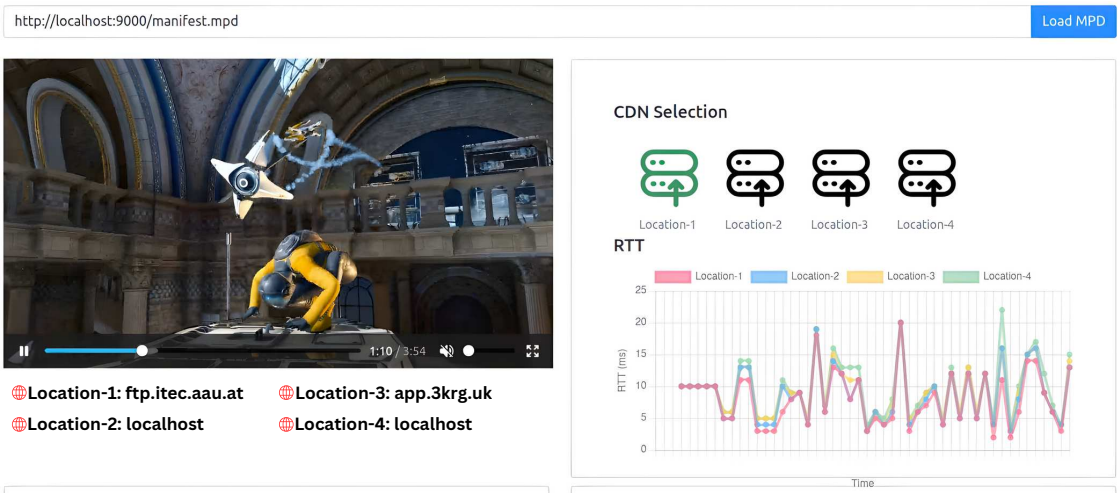

Figure 6.9: [Experimental setup] One client using modified Dash.js player to stream the 3DMark Night Raid video, with RTT tracking mechanism and StreamWise as CDN switching solution at the content steering server.

and significantly reducing quality switches. This performance advantage stems from StreamWise's intelligent adaptation to dynamic network conditions, enabling real-time optimal CDN selection that balances multiple QoE factors.

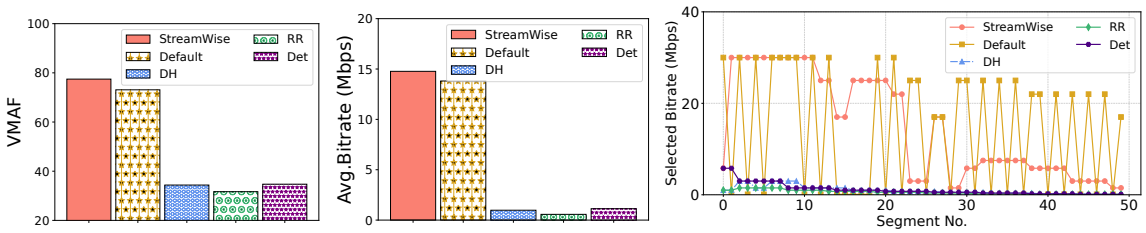

Figure 6.10: QoE comparison for CDN switching strategies; VMAF (left), average bitrate (center) and quality switches (right).

In contrast, the Default strategy (single CDN) exhibits frequent quality fluctuations despite reasonable bitrate performance, while the dynamic heuristic (DH) approach proves inadequate due to its reliance solely on RTT metrics, ignoring available throughput, which often results in the selection of CDNs with low latency but insufficient bandwidth. Static strategies (RR and Det) show limited adaptability, resulting in suboptimal QoE outcomes.

Notably, StreamWise achieves these performance gains with minimal computational overhead, utilizing only 14.96MB RAM for state tracking and delivering CDN decisions within 0.51ms inference time. This efficiency demonstrates the practical viability of our DRL-based approach for

real-time adaptive streaming scenarios.

6.3 Cadence Evaluation

6.3.1 Testbed overview

To validate Cadence under dynamic, large-scale conditions, we developed a high-fidelity experimental framework based on an extended version of the Vegvisir framework [24, 37]. Our enhancement transformed the original single-server-client prototype into a scalable, containerized platform capable of complex multi-agent experimentation. The final architecture, illustrated in Figure 6.11, comprised 70 isolated Docker instances: 60 clients, 4 CDN servers, and 6 traffic shapers, with a minimal total memory footprint of 600MB. All experiments were executed on a high-performance server to ensure consistency.

Realistic network dynamics were emulated using *TC Netem* [1], with content delivered over the QUIC protocol [35] using the goDASH player [34] in headless mode for efficient metric collection. The evaluation covered two core streaming modes: LLL with 2s segments and a 12s buffer, and VoD with 4s segments and a 60s buffer. Each 2-minute run was calibrated to capture essential CDN-client interactions without superfluous overhead.

Operationally, the Cadence steering server assigns a dedicated, lightweight DRL agent to each client. After a brief cold-start initialization where clients are distributed evenly across CDNs, these agents perform decentralized, real-time CDN selection. They leverage CMCD/CMSD telemetry to direct clients optimally, with a negligible inference latency of 0.07ms. This design ensures seamless integration, requiring no modifications to clients or CDNs, while maintaining system-wide stability and near-optimal performance through scalable, asynchronous coordination.

6.3.2 Benchmarking Configuration

This section outlines the key parameters and competitor strategies used to evaluate Cadence. The configuration was designed to ensure a rigorous, generalizable, and fair assessment under diverse and representative conditions.

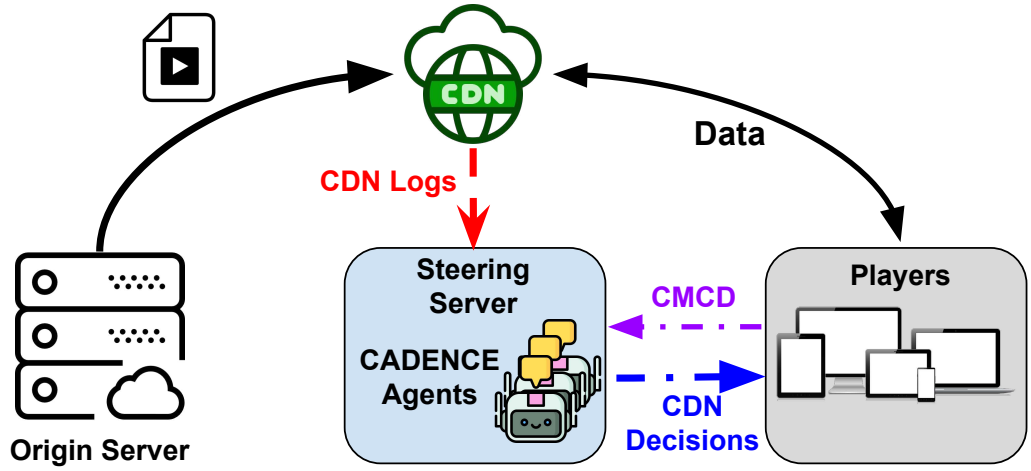

Figure 6.11: Multi-agent End to End architecture for adaptive video streaming. Each agent observes per-stream states and processes them through fully connected layers, and outputs actions via a shared MAPPO [2] policy.

Table 6.8: Client streaming setup.

Video Content	Resolution	Bitrate (Mbps)	FPS	Client Thr.
3D SkyDiver (3DSD) [61]	4K (288p–2160p)	0.5–40	120	50 Mbps
3D Night Raid (3DNR) [40]		0.5–40	60	50 Mbps
Nvidia Asteroid (NVA) [61]		0.5–40	60	30 Mbps
Moment of Intensity (MOI) [40]		0.5–40	60	22.5 Mbps
Of Forest and Men (OFM) [62]	2K (360p–1080p)	0.1–3.5	24	5 Mbps
Big Buck Bunny (BBB) [58]		0.1–3.5	24	5 Mbps

A corpus of six videos—four in 4K and two in 2K resolution—was used to challenge the algorithm with varying spatial and temporal complexity (see Table 6.8). All content was AVC-encoded across ten bitrate levels (4K: 288p–2160p; 2K: 360p–1080p). To simulate a heterogeneous streaming environment, 60 clients were divided into six groups, each streaming a unique video. Crucially, clients within a group shared an identical fixed throughput trace, isolating the performance of the CDN selection logic from network variability.

The evaluation employed a suite of adaptive bitrate (ABR) algorithms, extended within the goDASH player. This included LoL$^+$ [63] for LLL scenarios and the learning-based Pensieve [47] for VoD. A uniform stall prevention mechanism was applied to all ABRs for a fair comparison.

Network Traces: For the online testing, we used four network traces for the CDN-side. The details of the network traces are highlighted in Table 6.9.

Cadence was benchmarked against a comprehensive set of CDN switching strategies, including a retrained, cost-aware variant of the learning-based StreamWise (SA*) algorithm [24], created by

Table 6.9: Network Trace Data

CDN#	Network Trace	Avg. Thrpt. (Mbps)	Std. Dev. (Mbps)
CDN1	Netflix5G [43]	120	60
CDN2	Amazon5G [43]	85	36
CDN3	Twitch [44]	36	13
CDN4	Cascade [45]	80	20

re-training StreamWise with a cost parameter in its reward function. The benchmark also encompasses several heuristic strategies:

- **Deterministic (Det1/Det2):** Fixed allocation ratios (Det1: 30%, 30%, 25%, 15%; Det2: 18%, 18%, 24%, 40% across CDN1–4).
- **Round-Robin (RR):** Cyclical segment allocation across all CDNs.
- **DH1:** Selects the CDN with the highest instantaneous throughput.
- **DH2:** Selects the CDN with the lowest RTT.

These competitors represent the spectrum from static allocation to simple dynamic heuristics, providing a robust baseline for comparison.

6.3.3 Results and Analysis

Impact of CDN Switching Strategy

To comprehensively assess the effectiveness and robustness of the proposed Cadence framework, we evaluate its performance under two representative paradigms: LLL and VoD. These modes differ fundamentally in buffer availability, latency constraints, and stability requirements, making them complementary testbeds for evaluating generalizability. The results across both settings reveal that Cadence consistently reshapes the Pareto frontier of video quality versus cost, while competing strategies remain confined by their structural limitations.

LLL Streaming :

Under stringent LLL conditions, the key challenge is sustaining playback stability while operating with minimal client buffering. Figure 6.12 (left) confirms that all strategies, including Cadence,

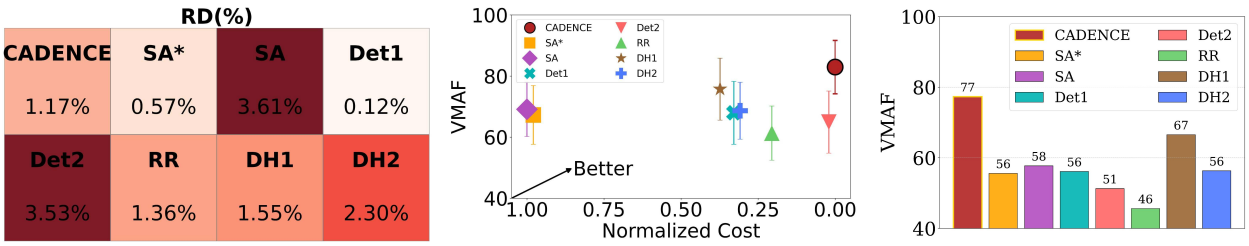

Figure 6.12: RD (%) analysis (left); Avg. VMAF vs Cost tradeoff (middle); and VMAF of 4K clients (right) for ABR:LoL$^+$ in LLL.

maintain rebuffering durations (RD) below 4%, largely due to the stall-prevention capabilities of the underlying LoL$^+$ ABR algorithm. This establishes a level playing field for stability, shifting the focus to the quality-cost trade-off.

As shown in Figure 6.12 (middle), Cadence is the only strategy that jointly improves video quality and reduces multi-CDN costs (*min–max normalized* [64] to $[0, 1]$.). It delivers a 12-unit VMAF improvement (2 JNDs) over the best learning-based alternatives while lowering costs by 45%. This outcome highlights its fine-grained, congestion-aware decision-making, which competitors cannot replicate. In contrast:

- **Learning-based (SA, SA*):** Struggle with coarse-grained selection, leading to CDN overloads or costly recovery switches, inflating costs by up to 70%.
- **Static (RR, Det1, Det2):** Lack adaptability, incurring significant quality penalties (up to 18 VMAF loss) or excessive costs depending on their bias toward expensive or cheap CDNs.
- **Dynamic heuristics (DH1, DH2):** Their single-metric focus (throughput or RTT) results in either inflated costs or degraded quality, lagging behind Cadence by 7–14 VMAF units.

The advantage of Cadence is most striking for 4K streaming, where its congestion-aware design secures a 22-unit VMAF gain (over 3 JNDs) compared to baselines (Figure 6.12, right). This demonstrates that intelligent multi-CDN adaptation is indispensable when throughput requirements are high, while simpler strategies collapse under pressure.

VoD Streaming :

Unlike LLL, VoD clients benefit from large playback buffers, which almost eliminate stalls. Indeed, all strategies maintain rebuffering ratios below 0.5% (Figure 6.13, left). With stability no

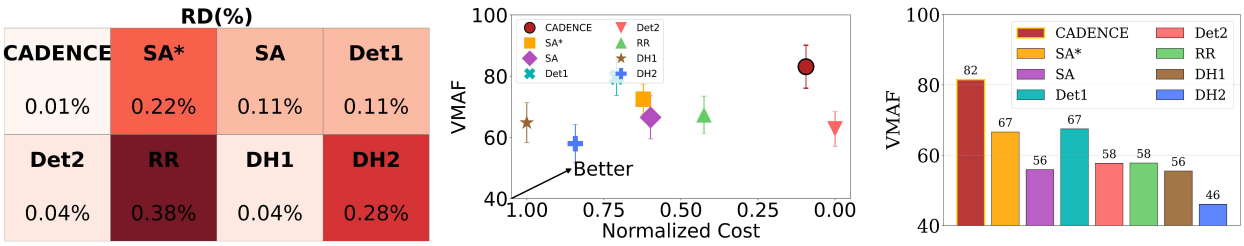

Figure 6.13: RD (%) analysis (left); Avg. VMAF vs Cost tradeoff (middle); and VMAF of 4K clients (right) for ABR:Pensieve in VoD.

longer a differentiating factor, the critical question becomes whether strategies can deliver superior quality without inflating costs.

Here too, Cadence redefines the trade-off. As illustrated in Figure 6.13 (middle), it provides a 17-unit VMAF improvement (over 2 JNDs) over the average baseline while maintaining minimal operational costs. This underscores the transferability of its optimization principles, whereas alternative approaches reveal persistent structural weaknesses:

- **Learning-based (SA, SA*):** Larger buffers reduce the frequency of costly switches, narrowing but not eliminating the cost gap (still 50% higher than Cadence). Their coarse-grained control continues to hinder quality, trailing by at least 6 VMAF units.
- **Static (RR, Det1, Det2):** Det1 approaches Cadence in quality but at a prohibitive 60% cost premium. RR and Det2 minimize costs but sacrifice over 2 JNDs of quality, exposing the rigidity of static allocation.
- **Dynamic heuristics (DH1, DH2):** DH1 becomes the most expensive policy, exploiting large buffers to persistently favor premium CDNs, while DH2 remains low-quality, replicating its poor LLL performance.

Impact of CDN Outages

A principal motivation for employing a multi-CDN architecture is to ensure service resilience against vendor-specific performance degradation or outright failure. To validate Cadence's capability in this critical scenario, we subjected the system to a simulated failure of CDN1 during an LLL streaming session. The failure was modeled as a gradual throughput decay beginning at $t = 40s$,

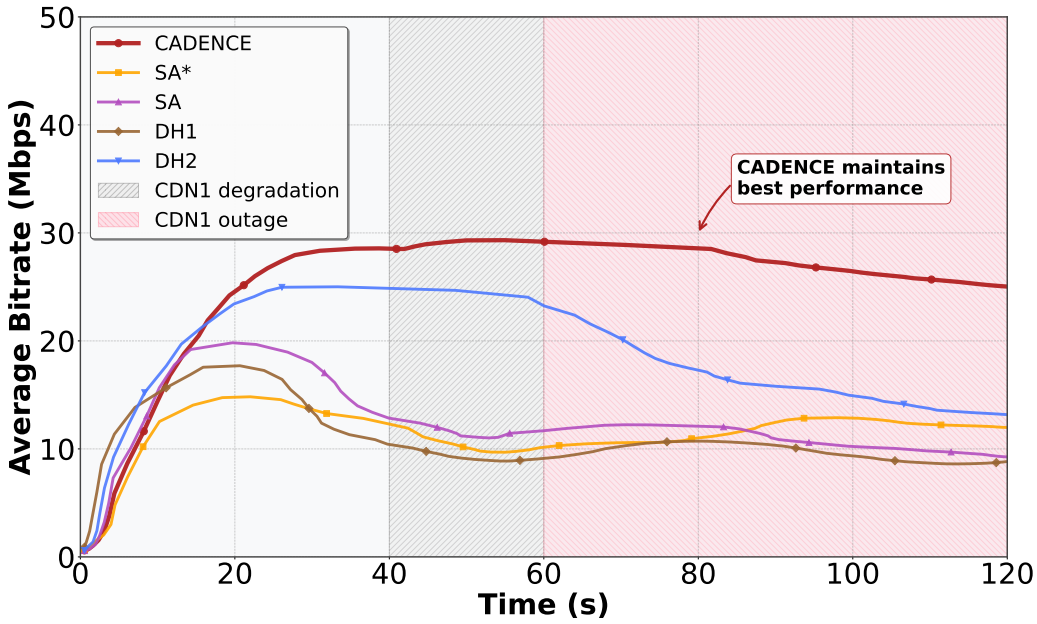

Figure 6.14: Bitrate stability analysis under CDN1 degradation and outage conditions. The proposed CADENCE framework sustains the most stable and highest average bitrate across time, exhibiting strong resilience to network impairments compared to the baselines.

culminating in a complete outage (throughput dropped to ∼100 kbps) at $t = 60s$ that persisted for the remainder of the session.

The experiment immediately exposed the catastrophic fragility of static switching strategies (Det1, Det2, RR). As these policies lack any failure detection mechanism, they continued to route clients to the failed CDN, resulting in a near-total collapse of throughput, severe video quality loss, and indefinite rebuffering. Their fundamental inability to adapt to dynamic network conditions renders them unsuitable for production environments requiring high availability; consequently, they are excluded from the subsequent resilience analysis.

The performance of dynamic strategies during this failure event is detailed in Figure 6.14, which tracks the average bitrate for 4K clients. The results demonstrate a stark divergence in resilience. While competing dynamic strategies (SA, SA*, DH1, DH2) begin to degrade as early as $t = 40s$—coinciding with the initial throughput decay of CDN1—Cadence maintains a stable, high-quality stream.

Specifically, clients managed by Cadence, experience only a minor ∼2Mbps bitrate dip and a negligible VMAF reduction, well within a single JND. In effect, Cadence successfully insulated

end-users from the vendor outage, sustaining nearly identical performance to the no-outage baseline. In contrast, the coarse-grained, single-agent designs of the competing dynamic strategies prevented effective load rebalancing. As CDN1 failed, they were unable to proactively redistribute traffic, causing the remaining healthy CDNs to become overloaded and triggering a cascading QoE collapse.

This robust performance is a direct consequence of Cadence's CDN State Tracker (Algorithm 5.3), which provides continuous, fine-grained monitoring of CDN health. By proactively flagging abnormal RTT growth and throughput instability, the system can preemptively migrate traffic away from a deteriorating CDN before a full outage occurs. This contrasts with reactive strategies that only switch after a failure is detected, by which time client QoE has already been severely impacted.

In summary, this resilience experiment confirms that Cadence not only optimizes for quality and cost under normal conditions but also provides a critical failure-handling capability. Its proactive mitigation strategy ensures uninterrupted, high-quality streaming even during significant CDN outages, a non-negotiable requirement for large-scale, quality-sensitive content delivery platforms.

Sensitivity to State-Update Delays

The efficacy of Cadence's fine-grained, per-segment CDN selection depends on the timely propagation of client state via CMCD reports. To assess its practicality for large-scale, geographically dispersed deployments, we conducted a sensitivity analysis introducing artificial state-update delays. This models the potential latency in propagating client metrics to the central steering server in real-world networks.

We repeated the LLL-mode evaluation under two delay conditions: a 4-second delay (equivalent to two consecutive segments using the same CDN selection) and an 8-second delay (four segments). LLL mode was selected for this stress test due to its strict latency requirements and small playback buffers, which make it especially vulnerable to outdated information.

The results, summarized in Figure 6.15, show that Cadence degrades gracefully under delay. As expected, stale updates reduce decision accuracy, limiting the system's ability to respond to rapid network fluctuations. This manifests as a gradual decline in overall VMAF and a modest increase in multi-CDN operational cost compared to the zero-delay baseline.

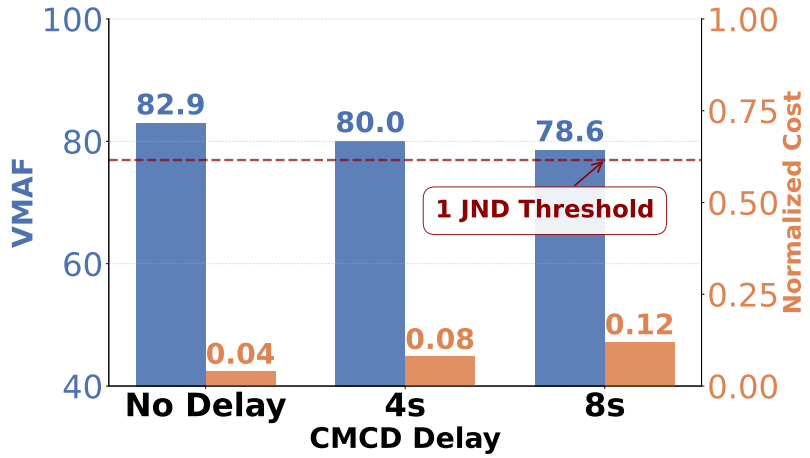

Figure 6.15: Impact of CMCD Delay on Cadence.

Crucially, the degradation remains bounded. Even with an 8-second delay—a significant interval in the context of low-latency streaming—the drop in video quality is contained within a single JND. Furthermore, Cadence continues to outperform all baseline strategies operating with zero propagation delay (cf. Figure 6.12, middle).

This finding underscores an important property: while timely updates are beneficial, Cadence's decision-making is not brittle. Its reliance on both historical performance and learned CDN behavior enables robust operation even with slightly stale information.

In summary, the sensitivity analysis confirms that Cadence maintains performance superiority and operational viability under realistic state-propagation delays. This robustness affirms its suitability for deployment in large-scale, geographically distributed streaming systems.

Scalability Under Increasing Client Load

A core requirement for any production-grade steering system is the ability to sustain both performance and cost-efficiency as client load increases. To evaluate Cadence's scalability, we conducted a series of experiments progressively scaling the number of concurrent clients from 120 to 600 under constrained LLL streaming conditions with 2K content. CDN capacities and aggregate client-side throughput were capped to model system limits and amplify load-induced effects.

The results, summarized in Figure 6.16, reveal two distinct operational regimes. At lower client counts (120 and 300), the combination of high CDN capacity and the modest throughput demands of

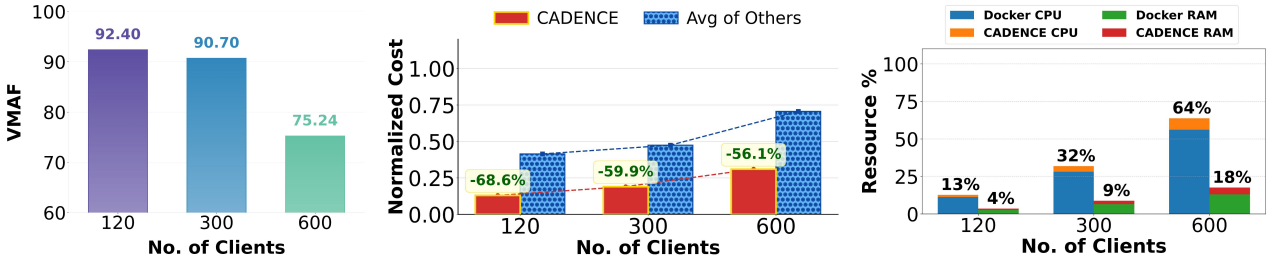

Figure 6.16: Impact of scalability on (a) VMAF (left); (b) Multi-CDN costs (middle); and (c) Resource Consumption. Cadence significantly lowers multi-CDN operational costs, demonstrating on average a fourfold reduction in computational resource usage (CPU and RAM).

2K content creates a non-competitive setting. In this regime, all strategies, including Cadence, reach similar peak VMAF as clients can freely download the highest-quality segments without contention.

The critical differentiator emerges as the system approaches saturation. With 600 clients, the aggregate throughput cap of 500Mbps allocates roughly 1Mbps per client, creating a resource-constrained bottleneck. In this regime, as shown in Figure 6.16 (left), a uniform VMAF drop of approximately 17 units (nearly 3 JNDs) is observed across all strategies. This universal degradation is driven by fundamental network saturation and host-level resource pressure (e.g., CPU scheduling), which forces all ABR algorithms to select mid-to-low bitrate segments. In this scenario, where no CDN can provide a quality advantage, Cadence intelligently pivots its optimization goal from pure quality maximization to cost minimization without sacrificing the already network-limited QoE.

This adaptive capability is clearly demonstrated in Figure 6.16 (middle). As client load increases, the operational cost of all competing strategies rises steadily. Cadence, however, consistently incurs at least 50% lower cost, and its rate of cost increase is the most gradual, showcasing its superior economic efficiency under load.

Furthermore, Cadence achieves this performance with minimal overhead. Figure 6.16 (right) confirms that at the 600-client peak, the system total CPU and RAM usage were 64% and 18%, respectively. Crucially, the Cadence agents themselves consumed only $\sim$10% of total CPU and $\sim$3% of total RAM, attesting to their lightweight, non-intrusive design.

In summary, this scalability analysis demonstrates that Cadence operates effectively across two key regimes: in resource-abundant conditions, it delivers top-tier quality, and in resource-constrained, saturated environments, it automatically optimizes for cost efficiency while maintaining parity in the

network-limited QoE. Coupled with its minimal resource footprint, these findings robustly position Cadence as a scalable and economically sound solution for large-scale multi-CDN deployments.

6.3.4 Ablation Studies

To validate the design choices in Cadence, we conducted several ablation studies. This analysis systematically explores variations in its DRL architecture, including reward function formulation, parameter tuning, and the impact of key constraints. Each variant was evaluated in isolation within the LLL mode using the LoL$^+$ ABR algorithm to precisely measure the effect of individual components. To facilitate a direct comparison across multiple performance dimensions, we introduce a unified scoring metric. This score normalizes the primary objectives—perceptual quality (VMAF) and operational cost—and combines them into a single value:

$$\text{Score} = 0.7 \times \text{VMAFnorm} + 0.3 \times (1 - \text{Costnorm}) \tag{7}$$

This function reflects the design priority of maximizing video quality while managing expenses, assigning a higher weight to VMAF performance. The overall ranking of all variants based on this score is presented in Figure 6.17. The following sections detail the findings for each major design variant.

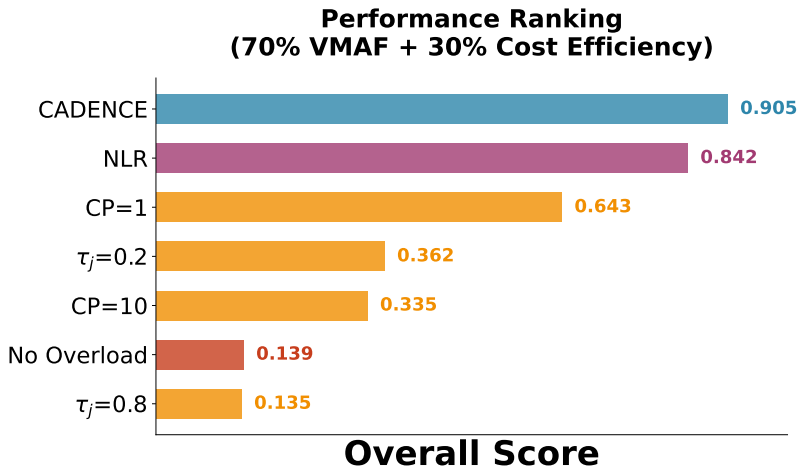

Figure 6.17: Cadence ablation study. The results underscore the trade-offs of different reward formulations,showing that Cadence achieves best balance of perceptual quality and cost efficiency, while overly constrained or relaxed penalty schemes lead to performance degradation.

Reward Function Linearity

We first investigated the impact of reward function complexity. The **NLR** (Non-Linear Reward) variant replaces the standard linear reward function from Eq. (6) with a more complex, non-linear formulation:

$$R = \frac{\omega_0 \cdot R_{\text{thr}} - \omega_1 \cdot R_{\text{reb}} - \omega_2 \cdot R_{\text{delay}}}{\omega_3 \cdot R_{\text{cost}} - \omega_4 \cdot R_{\text{over}}}, \tag{8}$$

with tuned parameters $\omega_0 = 1$, $\omega_1 = 5$, $\omega_2 = 0.5$, $\omega_3 = 5$, and $\omega_4 = 5$.

The results demonstrate the pitfalls of unnecessary complexity. The NLR variant increased the average VMAF by only 3 units—less than one JND, implying no perceptible quality gain for the user. This minor improvement came at a prohibitive cost, increasing operational expenses by 40%. Overall, a non-linear reward function complicates the learning process without yielding a tangible benefit, confirming that the effectiveness of the standard linear reward formulation.

Cost Penalty Sensitivity

The cost penalty parameter (ω_3) in the standard reward function (Eq. (6)) is critical for balancing quality and efficiency. We evaluated two extremes:

- **CP=1 (Low Penalty)**: Weakening the cost constraint ($\omega_3 = 1$) led to a significant performance degradation. VMAF dropped by 6 units (1 JND) while operational costs surprisingly *increased* by 93%, indicating inefficient resource utilization without quality compensation.
- **CP=10 (High Penalty)**: Conversely, an overly strict penalty ($\omega_3 = 10$) successfully reduced costs by 50% compared to the standard Cadence. However, this was achieved at the expense of a severe 20-unit VMAF reduction (over 3 JNDs), significantly degrading viewer QoE.

The standard value of $\omega_3 = 5$ was identified via Bayesian optimization, which efficiently navigates the parameter space. Our results validate that this value strikes an optimal balance, avoiding the inefficiency of a low penalty and the excessive quality sacrifice of a high penalty.

Overload Prevention Mechanism

The overload threshold τ_j is a key for proactive congestion control. We analyzed its sensitivity:

- A **low threshold** ($\tau_j = 0.2$) made the system overly cautious. While it improved cost-efficiency, it led to an overly conservative strategy that reduced VMAF by 16 units (over 2 JNDs).
- A **high threshold** ($\tau_j = 0.8$) reduced sensitivity to impending congestion. This led to actual overloads and a 19-unit VMAF drop (over 3 JNDs).

The optimal value was found to be $\tau_j = 0.4$, which best balances the prevention of overload with the maintenance of high Quality of Experience.

To highlight its necessity, we tested a **No Overload** variant removing the overload penalty. This caused severe congestion and a 22-unit VMAF drop (over 3 JNDs). Although operational cost fell 10% due to continued use of congested, low-cost CDNs, the drastic quality loss makes this trade-off unacceptable. This experiment confirms the overload penalty is essential for maintaining service quality under load.

Chapter 7

Conclusion and Future Work

This thesis explored the evolution and optimization of multi-CDN video streaming systems using DRL, culminating in the design and validation of two frameworks: *StreamWise* and *Cadence*.

StreamWise pioneered the application of DRL to the CDN selection problem in adaptive streaming, demonstrating that an Actor-Critic approach using minimal inputs—RTT, bitrate, and rebuffering duration—can consistently improve average VMAF, bitrate, and reduce rebuffering events in both VOD and LLL scenarios. Experimental results across a wide range of real-world network traces and content types indicated up to 8.5% higher VMAF, a 1.5× average bitrate increase, and a 48% improvement in QoE compared to existing heuristic and static multi-CDN solutions. These findings empirically validate the efficacy of DRL in streamlining intelligent content steering and maximizing QoE for end-users.

Building on the limitations observed in coarse-grained CDN steering and lack of cost-awareness, **Cadence** advanced the state-of-the-art with a novel multi-agent deep reinforcement learning (MADRL) framework. By adopting a CTDE paradigm, Cadence enables per-client agents to make fine-grained CDN choices, dynamically balancing QoE and operational delivery costs. Experimental evaluations with Cadence on a scalable, Dockerized testbed revealed that it can improve VMAF by up to 21%, achieve a 1.2× bitrate improvement for live applications, and lower rebuffering by up to 10×, all while reducing multi-CDN costs by 35%. Importantly, Cadence exhibited strong resilience to CDN failures, scalability under load, and graceful degradation under state propagation delays—pushing system performance close to the theoretical optimum and guaranteeing robust streaming even during

vendor-specific outages.

Collectively, these systems establish that RL-based approaches are uniquely positioned to outperform legacy heuristics by offering cost-efficient, fair, and adaptive content distribution in complex, and dynamic environments. However, this work also surfaces critical challenges on the path to production. The primary limitation lies in the *sim-to-real gap*: while our Dockerized testbed provides a robust validation environment, it cannot fully capture the non-stationarity and correlated failure modes of the global internet, the operational latency of a geographically distributed state tracker, or the complexities of real-world CDN contracts. By decoupling CDN selection from static policies and embedding real-time telemetric feedback loops (via CMCD/CMSD), this thesis delivers foundational contributions to both the theory and practical deployment of agentic AI for video streaming optimization.

Future Work

While the results of this thesis establish a strong baseline for intelligent multi-CDN steering, several avenues remain open for further research and industrial application:

- **Correlated CDN Failures and Non-Stationarity:** Current training assumes independent, uncorrelated CDN failures, yet real infrastructures may degrade simultaneously due to regional outages, peering issues, or BGP instabilities. Extending datasets and models to capture these correlated failure modes will be critical for robust real-world deployments. Integrating MARL paradigms that explicitly incentivize diversification across historically independent CDNs may yield greater resilience. Building on this direction, an open research question emerges: *Can a Graph Neural Network (GNN)-based critic that models shared Internet infrastructure improve resilience to correlated, regional CDN outages compared to an infrastructure-agnostic model?*

- **Hierarchical and Federated Scalability:** The centralized critic in Cadence, while effective at laboratory scale, may face scalability challenges when extended to millions of clients in production, primarily due to communication and synchronization overhead. To address these

limitations, hierarchical or federated reinforcement learning (RL) architectures—such as regional critics that periodically synchronize with a global policy—offer a promising pathway toward hyper-scale, geographically distributed training and deployment. This consideration motivates the following research question: *Can a federated RL approach with periodic model aggregation across regional critics maintain global policy coherence while scaling to millions of geographically distributed clients with non-IID data distributions?*

- **Joint ABR and CDN Optimization:** This thesis modularized ABR control and CDN steering; however, their inherent feedback coupling suggests potential benefits from joint optimization. Suboptimal CDN selections can constrain ABR decisions, amplifying QoE degradation. Future work should therefore investigate end-to-end reinforcement learning (RL) frameworks where bitrate adaptation and CDN selection are optimized concurrently, potentially leveraging graph-based or attention mechanisms to unify network and playback state representations. This leads to an open research question: *Can a multi-objective, multi-agent RL formulation with a monotonic value function factorization outperform a modular architecture by jointly optimizing CDN selection and bitrate adaptation within a single, end-to-end policy?*

- **Dynamic Cost Models and Real-World Contracts:** The cost model employed in this work is static, whereas real-world CDN contracts often involve dynamic or negotiated pricing structures governed by service-level agreements (SLAs) [65]. Future systems should incorporate economic modeling of streaming costs to adapt to fluctuating contract terms and even anticipate or negotiate delivery expenses in real time. To enable a more elastic and economically aware approach, a promising research direction emerges: *Can a hybrid RL–contextual bandit framework dynamically adapt CDN selection in real time to fluctuating spot prices and complex contractual conditions, such as volume tiers and committed-use discounts?*

- **Online Policy Updates and Low-Overhead Inference:** Scaling online policy refinement to production environments introduces practical constraints related to model synchronization, inference latency, and seamless player integration. Advancing research on lightweight model architectures, distributed update strategies, and on-device reinforcement learning (RL) agents could enable broader deployment at scale. This motivates the following research question:

Can model distillation techniques effectively compress the large centralized critic of Cadence into a lightweight, per-client policy capable of executing locally with sub-millisecond latency, without incurring significant performance degradation?

- **Extensive Internet-Scale Evaluation:** While the testbed emulates a wide range of network conditions, it cannot fully reproduce phenomena such as Internet-scale coordinated outages, cross-operator BGP disruptions, or emergent global events. Collaborations with industry partners for large-scale field trials or the use of real-world traffic datasets will be essential to validate, benchmark, and strengthen these frameworks under realistic operating environments. This motivates the following research question: *In a large-scale, live A/B test, does the Cadence policy exhibit statistically significant improvements in QoE and cost compared to a production-grade heuristic system during real-world, Internet-scale disruption events?*

Through addressing these directions, future work can more fully realize the promise of collaborative, agent-driven video delivery—pioneering new benchmarks in efficiency, fairness, and quality for next-generation streaming platforms.

Appendix A

Notations

Symbol	Description
Input State Variables	
$s_t^i = \langle b_{t,i,1}, ..., b_{t,i,n}, e_{t,i,1}, ..., e_{t,i,n}, d_{t,i,1}, ..., d_{t,i,n} \rangle$	State vector for client agent i at time t; bitrate, rebuffer duration, and delay for each CDN $1 \dots n$.
$b_{t,i,j}$	Throughput (bitrate) experienced by client i from CDN j at time t.
$e_{t,i,j}$	Rebuffering duration for client i with CDN j at time t.
$d_{t,i,j}$	RTT/delay for client i to CDN j at time t.
s_t^r	Regional aggregated state representing CDN load, capacity, and QoE metrics.
s_t^g	Global state vector aggregating per-region metrics and CDN network-wide statistics.
Action and Policy Variables	
$a_{t,i}$	Action of agent i at time t; CDN choice for next segment.
π_θ	Policy parameterized by θ, mapping states to action probabilities.
A_i	Action space of agent i (set of available CDNs).
z_i	Logits representing unnormalized action preferences before softmax.
Reward Variables	
R_t	Overall scalar reward at time t.
τ_j	Overload threshold for CDN j (fraction of clients).
Framework Parameters	
N	Set of client agents.
C	Set of available CDNs.
$V(s)$	State value function estimating expected future rewards.
$\hat{A}_t$	Advantage estimate used in policy gradient updates.
γ	Discount factor for future rewards.

Appendix B

Master's Coursework and Contributions

B.1 Master Coursework

Course	Course Code	Semester	Grade
IMAGE PROCESSING	COMP 6771	Winter 2024	A-
MACHINE LEARNING	COMP 6321	Winter 2024	B+
TOPICS/COMPUTER SCIENCE (AI FOR NETWORKED MULTIMEDIA SYSTEMS)	COMP 691	Fall 2024	A-
REINFORCEMENT LEARNING	INTU PLJM	Fall 2024	B

B.2 Contributions

- **C Joshi***, JS Sidhu*, **A Bentaleb**. IMAC: Intelligent Multi-Agent Content Steering for DASH (Accepted at **ACM Mile High Video 2025**) **Equal contribution.*
- JS Sidhu,**C Joshi**, **A Bentaleb**. MAESTRO: Multi-Agent CDN Selector for Video Streaming Optimizations (submitted to **NSDI 2025**)

References

[1] S. Hemminger *et al.*, "Network emulation with netem," in *Linux conf au*, vol. 5, p. 2005, 2005.

[2] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in neural information processing systems*, vol. 35, pp. 24611–24624, 2022.

[3] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over http," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 562–585, 2018.

[4] D. Paul, I. A. Prince, M. S. Islam, M. T. Ahamed, M. N. R. Sarker, and A. Adhikary, "Revolutionizing connectivity through 5g technology," *International Journal of Advanced Engineering Research and Science*, vol. 12, no. 02, 2025.

[5] A. Bentaleb, R. Farahani, F. Tashtarian, H. Hellwagner, and R. Zimmermann, "Which cdn to download from? a client and server strategies," in *Proceedings of the 2nd Mile-High Video Conference*, pp. 135–136, 2023.

[6] Y. Reznik, G. Cabrera, D. Silhavy, S. Pham, A. Giladi, A. Balk, A. C. Begen, and W. Law, "Content steering: a standard for multi-cdn streaming," in *Proceedings of the 3rd Mile-High Video Conference*, pp. 128–128, 2024.

[7] B. Zolfaghari, G. Srivastava, S. Roy, H. R. Nemati, F. Afghah, T. Koshiba, A. Razi, K. Bibak, P. Mitra, and B. K. Rai, "Content delivery networks: State of the art, trends, and future roadmap," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–34, 2020.

[8] A. Medina, "The akamai dns outage and the case for cdn redundancy," *ThousandEyes Blog*,

2024. Accessed: 2024-11-22.

[9] B. Kara and G. Simon, "Power efficient multi-cdn communication over content steering server," in *Proceedings of the 15th ACM Multimedia Systems Conference*, pp. 478–484, 2024.

[10] D. Silhavy, W. Law, S. Pham, A. C. Begen, A. Giladi, and A. Balk, "Dynamic cdn switching-dash-if content steering in dash. js," in *Proceedings of the 2nd Mile-High Video Conference*, pp. 130–131, 2023.

[11] Consumer Technology Association, "CTA-5004: Web Application Video Ecosystem–Common Media Client Data," Sep 2020.

[12] E. T. . 998, "Etsi ts 103 998 v1.1.1:dash-if: Content steering for dash." https://cdn.standards.iteh.ai/samples/67988/93d192067f4d434c9103d1b2317eb4a9/ETSI-TS-103-998-V1-1-1-2024-01-.pdf, 2024.

[13] S. Abbasloo, "Internet congestion control benchmarking," *arXiv preprint arXiv:2307.10054*, 2023.

[14] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of dns-based server selection," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 3, pp. 1801–1810, IEEE, 2001.

[15] R. Seeliger, D. Silhavy, and S. Arbanowski, "Dynamic ad-insertion and content orchestration workflows through manifest manipulation in hls and mpeg-dash," in *2017 IEEE Conference on Communications and Network Security (CNS)*, pp. 450–455, IEEE, 2017.

[16] Y. Reznik, A. Waldron, and G. Cabrera, "Simplifying multi-cdn delivery with hls/dash content steering," in *SMPTE 2023 Media Technology Summit*, pp. 12–15, SMPTE, 2023.

[17] P. Goenka, K. Zarifis, A. Gupta, and M. Calder, "Towards client-side active measurements without application control," *ACM SIGCOMM Computer Communication Review*, vol. 52, no. 1, pp. 20–27, 2022.

[18] A. Bentaleb, P. K. Yadav, W. T. Ooi, and R. Zimmermann, "Dq-dash: A queuing theory

approach to distributed adaptive video streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 1, pp. 1–24, 2020.

[19] C. Ma and Y. Chi, "Evaluation test and improvement of load balancing algorithms of nginx," *IEEE Access*, vol. 10, pp. 14311–14324, 2022.

[20] R. Soni, "Load balancing with nginx," in *Nginx: From Beginner to Pro*, pp. 153–171, Springer, 2016.

[21] Amazon Web Services, *Elastic Load Balancing: Application Load Balancers*, 2024. Default routing algorithm documentation.

[22] F5 NGINX, *HTTP Load Balancing*, 2024. Documentation on Round Robin and Least Connected load balancing.

[23] V. Adhikari *et al.*, "A tale of three cdns: An active measurement study of hulu and its cdns," in *IEEE INFOCOM*, 2012. Analyzes Hulu's use of target ratios and weighted traffic splitting.

[24] C. Joshi, J. S. Sidhu, and A. Bentaleb, "Streamwise: An intelligent content steering for dash," in *Proceedings of the 16th ACM Multimedia Systems Conference*, pp. 35–45, 2025.

[25] Y. Zhang *et al.*, "Maars: Multiagent actor-critic approach for resource allocation in multi-access edge computing," *MDPI Sensors*, vol. 24, no. 23, 2024.

[26] X. Liu *et al.*, "Nova: Neural-optimized viewport adaptive 360-degree video streaming at the edge," *IEEE Transactions on Services Computing*, 2024.

[27] J. Li *et al.*, "Pira: Pan-cdn intra-video resource adaptation for short video streaming," *arXiv preprint arXiv:2510.18606*, 2025.

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

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[30] C. Amato, "An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning," *arXiv preprint arXiv:2409.03052*, 2024.

[31] R. Gorsane, O. Mahjoub, R. J. de Kock, R. Dubb, S. Singh, and A. Pretorius, "Towards

a standardised performance evaluation protocol for cooperative marl," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5510–5521, 2022.

[32] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press, 2024.

[33] A. Beynier, F. Charpillet, D. Szer, and A.-I. Mouaddib, "Dec-mdp/pomdp," *Markov Decision Processes in Artificial Intelligence*, pp. 277–318, 2013.

[34] D. Raca, M. Manifacier, and J. J. Quinlan, "godash — go accelerated has framework for rapid prototyping," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–4, 2020.

[35] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the conference of the ACM special interest group on data communication*, pp. 183–196, 2017.

[36] mvfst, "mvfst quic implementation." https://github.com/facebook/mvfst, 2024. Accessed on Sept. 26, 2024.

[37] J. Herbots, M. Vandersanden, P. Quax, and W. Lamotte, "Vegvisir: A testing framework for http/3 media streaming," in *Proceedings of the 14th Conference on ACM Multimedia Systems*, pp. 403–409, 2023.

[38] Consumer Technology Association, "CTA-5006: Web Application Video Ecosystem–Common Media Server Data," Nov. 2022.

[39] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[40] B. Taraghi, H. Amirpour, and C. Timmerer, "Multi-codec ultra high definition 8k mpeg-dash dataset," in *ACM MM*, 2022.

[41] CS MSU Graphics and Media Lab, "Mpeg-4 avc/h.264 video codec comparison," tech. rep., Moscow State University, Dec. 2005. Available at: http://www.compression.ru/video/codec_comparison/mpeg4_avc_h264.html.

[42] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: Analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, pp. 114–118, 2013.

[43] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: A 5g dataset with channel and context metrics," in *Proceedings of the 11th ACM multimedia systems conference*, pp. 303–308, 2020.

[44] Twitch, "Acm mmsys 2020 grand challenge," in *ACM MMSys*, 2020.

[45] J. Herbots, A. Verstraete, M. Wijnants, P. Quax, and W. Lamotte, "Cross that boundary: Investigating the feasibility of cross-layer information sharing for enhancing abr decision logic over quic," in *Proceedings of the 33rd Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 50–57, 2023.

[46] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 187–198, 2014.

[47] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *ACM SIGCOMM*, pp. 197–210, 2017.

[48] N. Kan, Y. Jiang, C. Li, W. Dai, J. Zou, and H. Xiong, "Improving generalization for neural adaptive video streaming via meta reinforcement learning," in *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 3006–3016, 2022.

[49] R. Rassool, "Vmaf reproducibility: Validating a perceptual practical video quality metric," in *2017 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)*, pp. 1–2, IEEE, 2017.

[50] T. Alexander, "Visual-psnr measure of image quality journal of visual communication and image representation," *Journal of Visual Communication and Image Representation*, 2014.

[51] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 325–338, 2015.

[52] J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "Http/2-based adaptive streaming of hevc video over 4g/lte networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[53] Z. Chen and H. Liu, "Jnd modeling: Approaches and applications," in *2014 19th International Conference on Digital Signal Processing*, pp. 827–830, 2014.

[54] W. Jin, H. Du, B. Zhao, X. Tian, B. Shi, and G. Yang, "A comprehensive survey on multi-agent cooperative decision-making: Scenarios, approaches, challenges and perspectives," *arXiv preprint arXiv:2503.13415*, 2025.

[55] Joumxyzptlk, "Atomic hearts rtx demo," 2024.

[56] DASH-IF, "dash.js." https://reference.dashif.org/dash.js/, 2020. Online.

[57] MPEG-DASH Standard, "Mpeg-dash: Dynamic adaptive streaming over http," 2012.

[58] T. Roosendaal, "Big buck bunny," in *ACM SIGGRAPH ASIA 2008 computer animation festival*, pp. 62–62, ACM, 2008.

[59] P. Jones, "Cloudflare incident on march 21, 2025," Mar. 2025. Accessed: 2025-04-11.

[60] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM transactions on networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[61] J. Otahal, "Joumxyzptlk: High-resolution gaming benchmarks and panoramas," 2025. Accessed: 2025-04-11.

[62] "Of forest and men." http://www.offorestsandmen.org, 2010. [Accessed: Sept 2025].

[63] M. Lim, M. N. Akcay, A. Bentaleb, A. C. Begen, and R. Zimmermann, "When they go high, we go low: low-latency live streaming in dash. js with lol," in *Proceedings of the 11th ACM Multimedia Systems Conference*, pp. 321–326, 2020.

[64] S. Patro and K. K. Sahu, "Normalization: A preprocessing stage," *arXiv preprint arXiv:1503.06462*, 2015.

[65] K. D. Larson, "The role of service level agreements in it service delivery," *Information Management & Computer Security*, vol. 6, no. 3, pp. 128–132, 1998.