

FROG: Fast Response to Optimization Goals for HTTP Adaptive Streaming over SDN

Mustafa Türk, Müge Sayıt, *Senior Member, IEEE*, Ali C. Begen, *Senior Member, IEEE*, and Andreas J. Kasser, *Senior Member, IEEE*

Abstract—In modern video streaming systems, clients typically make independent bitrate decisions without full knowledge of network conditions, often leading to inefficient resource usage and unstable quality. Network-assisted adaptive streaming is rapidly gaining importance in Network and Service Management (NSM), as operators strive to deliver consistently high Quality of Experience (QoE) under dynamic traffic conditions. This paper introduces FROG, a novel, real-time Software-Defined Networking (SDN)-assisted framework designed to coordinate multi-server and multipath HTTP Adaptive Streaming (HAS) using standardized metrics carriage: Common Media Client Data (CMCD) and Common Media Server Data (CMSD). FROG employs an innovative two-stage optimization workflow in which an initial Linear Programming (LP) model rapidly determines feasible bandwidth bounds, server selection, and path capacities, thereby transforming the remaining optimization into a sequential layer-selection process for coordinated quality selection and flow allocation. This decomposition enables sub-second optimization at the scale of thousands of users, while buffer-aware client feedback is integrated to proactively prevent stalls and maintain system stability. Experiments on an emulated SDN testbed demonstrate that FROG achieves QoE comparable to that of the optimal MILP solution on tractable instances while virtually eliminating video stalls, and significantly reduces quality oscillations, outperforming state-of-the-art network-assisted approaches by up to 2.32× under playback-driven evaluation scenarios. Scalability experiments with up to 4,000 clients further demonstrate sub-second optimization runtimes, confirming the practicality of FROG for large-scale deployments.

Index Terms—HAS, SDN, QoE optimization, multipath routing, multi-server delivery, CMCD, CMSD.

I. INTRODUCTION

The rapid evolution of modern communication networks—driven by cloud computing, virtualization, and edge intelligence—has introduced unprecedented complexity in resource management and service delivery. These networks must dynamically allocate resources, monitor performance, and adapt to highly variable conditions to ensure consistent service quality. Among the diverse traffic types, video has emerged as the dominant contributor to global Internet traffic, emphasizing the urgent need for intelligent and automated management mechanisms that optimize bandwidth

utilization and enhance user-perceived QoE [1]. To address these challenges, SDN has gained prominence by providing programmability, centralized visibility, and fine-grained automation, enabling optimization of both Quality of Service (QoS) and QoE across heterogeneous network environments. Network Function Virtualization (NFV) complements SDN by supporting flexible deployment and orchestration of network services in virtualized environments. These technologies are particularly relevant for video delivery, where client-driven adaptation mechanisms such as Adaptive Bitrate (ABR) streaming operate independently of global network control. By decoupling the control and data planes, SDN enables centralized network management and network-assisted optimization of streaming applications. Although NFV is not explicitly utilized in this work, it is widely adopted in modern network architectures to improve service flexibility and resource orchestration.

HAS delivers video by allowing clients to dynamically adjust the requested bitrate according to prevailing network conditions. Although this ABR mechanism improves service continuity, client-driven decisions rely on incomplete network information, which can negatively affect QoE. SDN complements this approach by providing centralized network visibility and control, enabling more informed resource allocation and streaming decisions. HAS has become the dominant method for large-scale video delivery because of its scalability and resilience. However, the decentralized nature of client-driven adaptation introduces inherent limitations: clients lack global visibility, often misestimate available throughput, compete unfairly for bottleneck links, and experience bitrate oscillations or playback stalls that degrade QoE [2], [3], [4]. These problems result from a persistent mismatch between distributed application-layer adaptation behavior and network-level resource management.

To bridge this gap, NSM frameworks increasingly explore network-assisted adaptive streaming, in which the network provides real-time telemetry (e.g., congestion, link utilization, and latency) to guide streaming decisions [5]. Emerging standards such as CMCD and CMSD [6], [7] further support coordination between streaming clients, servers, and CDNs by enabling the exchange of information such as buffer status, bitrate recommendations, and session metadata. As operators deploy Multi-access Edge Computing (MEC) platforms and transition toward 5G/6G, streaming environments are becoming inherently multi-server and multipath [8]. These environments offer rich optimization opportunities but also demand management solutions that can respond reliably and

M. Türk is with the International Computer Institute, Ege University, Izmir, Türkiye. Email: turk.mustafa@hotmail.com.

M. Sayıt is with the CSEE, University of Essex, UK and International Computer Institute, Ege University, Türkiye. Email: muge.sayit@essex.ac.uk, muge.sayit@ege.edu.tr.

Ali C. Begen is with Ozyegin University, Istanbul, Türkiye. Email: ali.begen@ozyegin.edu.tr.

A. Kasser is with the Deggendorf Institute of Technology, Germany and Karlstads Universitet, Sweden. Email: andreas.kasser@th-deg.de, andreas.kasser@kau.se.

at scale.

Optimization-based approaches for HAS provide useful foundations [9], [10]; however, most assume simplified, static settings or single-server architectures. More importantly, the combined decision space of server selection, multipath routing, and layered video quality assignment leads to an NP-complete problem [10], making monolithic MILP formulations infeasible for real-time decision making. Existing network-assisted approaches typically address only a subset of the decision space, focusing on bitrate adaptation, server selection, or path allocation independently. Consequently, they do not provide a unified solution capable of simultaneously addressing these interdependent decisions while maintaining scalability for large-scale real-time deployments. Thus, a key challenge remains: how to deliver coordinated, QoE-aware streaming decisions across thousands of clients without overwhelming the control plane.

To address this challenge, we present FROG, an SDN-assisted framework for real-time layered video streaming that coordinates (i) video quality selection, (ii) server assignment, and (iii) multipath flow allocation. FROG integrates CMCD-style feedback, SDN-based path control, and a scalable two-stage optimization design to enable closed-loop network–application coordination. Our main contributions are:

- **Coordinated multi-server/multipath optimization:** FROG is, to the best of our knowledge, the first operational framework that coordinates layered quality selection, per-segment server selection, and SDN-controlled multipath routing.
- **Two-stage optimization framework:** We propose an LP-based Optimality Programming Model (OPM) that derives feasible bandwidth bounds and path capacities, reducing the original MILP to a sequential layer-selection process executed by the Core Programming Model (CPM) for coordinated quality selection and flow allocation, enabling real-time operation.
- **CMCD-aligned proactive stall mitigation:** Clients provide buffer information, enabling FROG to prioritize at-risk users and prevent playback stalls.
- **Scalability to thousands of clients:** Structural simplification and pruning yield a complexity that grows nearly linearly with the number of clients and layers. FROG maintains sub-second optimization times for up to 4,000 clients, where the evaluation focuses on the computational scalability of the optimization pipeline rather than end-to-end QoE under playback-driven conditions.
- **Substantial QoE improvements:** Evaluation on an emulated SDN testbed shows that FROG significantly improves QoE, reduces oscillations, and nearly eliminates stalls while outperforming several baselines.

The remainder of this paper is organized as follows. Section II reviews related work. Section III presents FROG and its optimization models. Section IV discusses experimental results. Section V concludes the paper.

II. BACKGROUND AND RELATED WORK

HAS has become the dominant method for large-scale video delivery, yet its decentralized, client-driven operation often

leads to inaccurate throughput estimation, unfair bandwidth sharing, and frequent quality oscillations [2], [4], [11]. These limitations have motivated the emergence of *network-assisted* streaming frameworks, which leverage SDN, NFV, and emerging telemetry standards such as CMCD/CMSD to improve coordination between the network and HAS clients. In this section, we revisit prior work across three relevant dimensions and highlight their limitations in relation to our multi-server, multipath framework, FROG.

A. Network-Assisted Adaptive Video Delivery Methods

Several studies investigate how SDN and NFV can enhance HAS by providing global visibility and programmable control over network resources. Early SDN-assisted approaches demonstrated that QoE can be improved by adjusting queue management on OpenFlow (OF) switches [12] or by dynamically mapping QoS indicators to QoE requirements in SDNDASH [13] and its extension [14], which addressed scalability, network overhead, and client heterogeneity. ORL-SDN is another prominent line of work, where online reinforcement learning is integrated with SDN to optimize bitrate adaptation and network resource utilization [15]. Reza et al. [16] demonstrated a practical SDN implementation of the SAND architectural model, highlighting controller-driven networking benefits for assisted systems. Ozcelik and Ersoy proposed CSASDN [17], which incorporates segment duration into centralized optimization, mitigating throughput estimation errors in DASH clients. Several studies have also explored SDN-supported multipath strategies for improving end-to-end throughput and QoE [18], [19]. Despite demonstrating the benefits of SDN assistance, these approaches do not jointly consider server selection and quality assignment, two fundamentally related decisions in large-scale, multi-server deployments.

Beyond traffic engineering, SDN has also been used to improve content delivery by exposing cache availability and bandwidth information to the control plane. SABR [20] leverages SDN to refine ABR decisions based on cache-aware routing, while further work focuses on selecting optimal CDN servers based on real-time network conditions [21]. These systems help avoid bottlenecks and reduce fetch delays, but remain limited to either single-server environments or non-layered video formats.

The combination of SDN and edge computing has recently become an important direction for next-generation video delivery. ES-HAS [22] integrates SDN with edge-assisted fetching to reduce end-to-end latency, and CSDN [23] extends this concept with transcoding capabilities, enabling hybrid fetching and adaptation across distributed edge servers. LEADER [24] introduces collaborative edge resource sharing for improved load distribution, while ARARAT [25] proposes a hierarchical SDN–NFV–edge architecture to reduce latency and enhance QoE across multiple clients. These frameworks highlight the benefits of network-assisted optimization, but they operate largely at a coarse time granularity, do not provide per-segment multipath control, and lack mechanisms for real-time multi-server decision making. Service Function Chaining (SFC) has also been investigated to support

adaptive streaming. SERENA [26] integrates SDN, SFC, and edge nodes to orchestrate flow processing for latency- and bandwidth-sensitive streaming applications. While SFC enriches the design space for deploying adaptive services, these works do not address fine-grained rate selection or path allocation at scale.

Overall, prior studies demonstrate the importance of SDN-assisted orchestration but do not jointly address the three interdependent problems tackled by FROG: (i) segment-level quality selection, (ii) dynamic server assignment, and (iii) coordinated multipath allocation. In contrast, FROG coordinates these decisions through a decomposition-based optimization framework, integrating all three components while scaling to thousands of users with sub-second runtimes.

B. Adaptive Layered Video Delivery

Layered video encoding decomposes a stream into a base layer and multiple enhancement layers, enabling more flexible quality adaptation compared to single-layer formats. Scalable Video Coding (SVC), standardized as an extension of AVC [27], enables efficient adaptation under fluctuating bandwidth conditions and has motivated research on SDN-assisted delivery of layered content. Although SVC has seen more limited industrial adoption than single-layer AVC/HEVC-based DASH due to lower compression efficiency, higher encoding complexity, and coding overhead, its layered architecture offers several advantages for SDN-assisted multi-server and multipath streaming. By representing multiple quality levels within a single scalable bitstream, SVC reduces storage redundancy compared to independently encoded representations. In multi-server deployments, only the required enhancement layers need to be transferred between media servers, reducing inter-server communication overhead. Furthermore, enhancement layers can be delivered progressively after the corresponding base layer has been buffered, allowing video quality to improve as additional bandwidth becomes available while reducing the likelihood of playback interruptions [28]. During congestion, enhancement layers may be selectively discarded while uninterrupted playback is sustained by the base layer. These characteristics provide greater flexibility for bandwidth allocation, multipath transmission, and dynamic path selection, making SVC well suited to the proposed SDN-assisted optimization framework.

Several works have proposed network-assisted algorithms for selecting paths or distributing layers across the network. Gohar and Lee introduced DMPF [29], an SDN-enabled approach that maps MPTCP subflows to optimize the allocation of SVC layers over multiple paths. In emerging mobile and edge-centric environments, Yeznabad et al. [30] allocate radio, storage, and compute resources to maximize QoE, while Kirmizioğlu et al. [31] use SDN-controlled edge units to enhance SVC-based multiparty WebRTC conferencing. These works incorporate layered formats, but they do not address the computationally complex integrated problem of server selection, path allocation, and quality control.

SDN-based video layer assignment has also been studied in [32], where the controller selects codecs and quality

levels according to path conditions. The most comprehensive optimization-based solutions are S²VC [10] and its AVC extension [9], which formulate joint path and quality selection as MILP problems. While these approaches provide strong optimality guarantees in small networks, the combined multi-server, multipath SVC optimization problem is NP-complete [9], [10]. Consequently, their monolithic MILP formulations become computationally intractable for real-time operation, even for moderate client populations.

In contrast, FROG introduces an LP-based decomposition that derives feasible bandwidth bounds, prunes infeasible server-layer assignments, and transforms the remaining optimization from the original MILP into a sequential layer-selection process, enabling scalable, real-time operation. Unlike prior layered streaming systems, FROG coordinates (i) quality selection across multiple servers, (ii) multipath flow allocation, and (iii) buffer-aware prioritization to minimize playback stalls and quality oscillations.

C. CMCD/CMSD-Based Solutions

Recent standardization efforts improve information sharing among streaming clients, servers, and network elements. The CMCD standard enables HAS clients to communicate operational information, such as buffer level, measured throughput, playback state, and requested bitrate, to media servers and CDNs, supporting more informed content delivery and caching decisions [6], [7]. Complementarily, the CMSD standard enables servers and CDNs to provide guidance to streaming clients, including recommended bitrate limits, server load, delivery capacity, and alternative content delivery targets [33]. Together, CMCD and CMSD enable bidirectional information exchange in modern HAS ecosystems, which is particularly relevant to SDN-assisted adaptive streaming. FROG benefits from signaling aligned with CMCD and CMSD through client buffer-state reporting, server-side bitrate recommendations, and delivery redirection.

Several early CMCD-driven studies demonstrated its potential. Bentaleb et al. [34] evaluated CMCD-based feedback for bandwidth allocation, while CADLAD [35] used CMCD metadata to construct device-aware bitrate ladders. CMSD has also been integrated with dash.js clients to explore server-side guidance under realistic deployment scenarios [36]. More advanced systems leverage CMCD/CMSD features with learning-based methods. Bentaleb et al. [37] use reinforcement learning to provide bitrate recommendations based on network behavior and device characteristics. Tran et al. [38] explored CMCD-enabled cellular network APIs to dynamically adjust cache usage under fluctuating conditions.

While these studies highlight the benefits of standardized metadata exchange, CMCD/CMSD has not yet been integrated into a global optimization framework that coordinates server selection, multipath routing, and quality adaptation. FROG is the first framework to embed CMCD/CMSD-aligned buffer metrics and segment metadata directly into its optimization logic, enabling fine-grained, real-time coordination between clients and the SDN control plane. This integration supports proactive stall prevention, more stable quality adaptation, and scalable QoE optimization in multi-server SVC deployments.

TABLE I: Comparison of different studies.

Reference	Layered video	Multi-server	Multipath	CMCD/CMSD	Dynamic Path Selection	SDN Assistance
Farahani et al. [22], [23]	✗	✓	✗	✗	✓	✓
Farahani et al. [25]	✗	✓	✓	✗	✓	✓
Gogar & Lee [29]	✓	✗	✓	✗	✓	✓
Barakabitze et al. [18], Bikas & Sayit [19]	✗	✗	✓	✗	✓	✓
Nguyen et al. [35]	✗	✗	✗	partial (CMCD)	✗	✗
Bentaleb et al. [37]	✗	✗	✗	✓	✗	✗
Erfanian et al. [9]	✗	✗	✓	✗	✓	✓
Tashtarian et al. [10]	✓	✗	✓	✗	✓	✓
FROG	✓	✓	✓	✓	✓	✓

Representative related works are compared in Table I to provide a general overview and highlight the key differences of this study. To the best of our knowledge, FROG is the first study to incorporate multi-server, multipath layered streaming with SDN assistance. Moreover, among similar studies, only FROG utilizes the CMCD/CMSD approach.

III. THE FROG FRAMEWORK

We propose **FROG**—a novel framework designed to enhance QoE for layered video streaming through coordinated segment retrieval from multiple servers over multiple paths. In FROG, an SDN-based network provides real-time feedback and coordination, enabling coherent adaptation decisions to maximize QoE. The framework addresses key challenges, including optimal video quality selection, efficient bandwidth utilization, minimal quality fluctuations, and reduced playback stalls, by leveraging real-time data and advanced optimization techniques to improve resource allocation and user experience.

A. General Architecture

In the proposed architecture, clients are connected to an SDN domain in which multiple servers are available to deliver video segments. Clients may download different segments from different servers, while an SDN controller manages the overall network. The controller hosts an application—the *Optimizer*—responsible for determining segment quality levels, selecting appropriate servers, and optimizing network routes to maximize QoE. This application implements an optimization model that simultaneously coordinates multiple clients, which constitutes the main contribution of FROG.

The overall architecture and communication flow are illustrated in Fig. 1. Fig. 1 also highlights the network edges considered by the optimization model. For the four server-side OF switch topology, the colored dotted lines indicate the feasible routing alternatives. In contrast, for the single server-side OF switch topology, all network edges (shown as solid red lines) are included in the routing decision space, representing all feasible routing alternatives. The Optimizer functions as an interface between the network operator and the video streaming service provider. It periodically retrieves video file metadata—such as file names, sizes, and the number

of segments—along with the Media Presentation Description (MPD) file from media servers (steps 1–2).

When a client joins the system, it first requests and retrieves the MPD file from a default server (steps 3–4). Then, the client sends the number of the next segment to be requested and its current buffer fullness value to the Optimizer (step 5). To minimize playback stalls, the Optimizer prioritizes clients with lower buffer levels during bandwidth allocation. The bidirectional communication between the Optimizer and clients is conceptually aligned with the CMCD/CMSD standards. Specifically, clients report buffer status information as defined by CMCD, while the Optimizer provides video quality information and server redirection recommendations in accordance with CMSD. Although a proxy could be used to collect and forward this information, we assume direct communication between clients and the Optimizer for simplicity. This assumption simplifies exposition and does not affect generality, as the same interaction can be mediated via a proxy or CDN edge.

At the end of the information-gathering period (step 6), the Optimizer collects network statistics from the SDN controller via its REST API. Because the data collection interval and the controller’s update cycle are asynchronous, the Optimizer always uses the most recent available measurements.

To improve estimation accuracy, it also leverages streaming statistics from the previous time slot and excludes transient video traffic to avoid underestimating available bandwidth (hereafter referred to as the “efficient bandwidth” mechanism). Additionally, the Optimizer incorporates the exact segment sizes retrieved from media servers, including a 5.4% network header overhead (Ethernet, IP, and TCP), ensuring that only persistent background traffic informs routing and quality decisions. Using this refined bandwidth information, the Optimizer solves the flow-routing, quality-level selection, and media server assignment problem through a coordinated two-stage optimization framework, achieving more stable quality adaptation and improved QoE.

After obtaining the optimal solution, the Optimizer sends flow assignment requests to the SDN controller (step 7) and notifies the clients of their allocated quality levels and assigned servers (step 8). The SDN controller then

reconfigures the forwarding paths according to the received instructions. Finally, in steps 9–10, clients request and receive video segments from the designated media servers at the assigned quality levels.

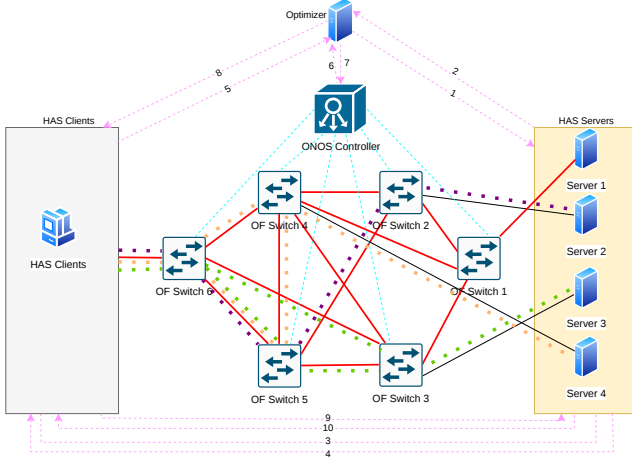


Fig. 1: Topology of the FROG framework, illustrating the SDN-enabled network and the Optimizer.

FROG continuously collects feedback from clients and dynamically adapts video quality levels based on current network load conditions. It operates in a periodic loop, in which the Optimizer gathers system information, solves the optimization problem, reconfigures network paths, and updates all clients within each iteration. This life cycle is illustrated in Fig. 2. During each slot, client requests are aggregated, and at the end of the slot, the Optimizer determines the appropriate quality levels and server assignments while concurrently configuring the forwarding states of the OF switches.

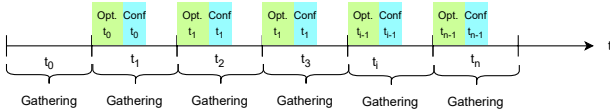


Fig. 2: FROG periodically executes optimization (Opt.) and network reconfiguration (Conf.), informing clients about selected servers and quality layers.

B. Optimization Model for Multi-Server, Multipath HAS

The optimization model is formulated with minimal yet practical assumptions. First, each server is assumed to store all segments and their corresponding quality layers. Second, clients are allowed to retrieve different layers of the same segment from different servers, enabled by CMCD/CMSD-aligned coordination. These assumptions provide flexibility for multipath, multi-server delivery while maintaining segment consistency during playback.

The proposed model enhances QoE by minimizing playback stalls and quality oscillations while improving bandwidth utilization through a coordinated two-stage client-server adaptation framework. It comprises two interlinked optimization models: (i) the **Optimality Programming**

Model (OPM), which determines upper bounds of available bandwidth under end-to-end constraints ($\bar{r}_y^x$), explores feasible multipath routes ($\bar{f}_{ij}^{xy}$), and initializes flow assignments starting from the base layer; and (ii) the **Core Programming Model (CPM)**, a MILP that uses OPM outputs to select the optimal video quality level for each client. Although the problem is NP-complete, we relax it by sequentially fixing video layers to server-side switches according to OPM results, starting with the server-side switch with the lowest available capacity and the base layer. The server-side switch with the largest capacity is left for CPM's optimization, reducing computational complexity while preserving near-optimal assignment.

The overall optimization process is illustrated in Fig. 3 and summarized in Algorithm 1. The notation used in both the OPM and CPM programming models, as well as in the algorithm, is provided in Table II.

1) Optimality Programming Model (OPM)

The objective function of OPM (1) maximizes bandwidth between server- and client-side switches (r_y^x) and the reserved bandwidth on each edge (Γ_{ij}), while minimizing bandwidth usage on the edges (f_{ij}^{xy}). Minimizing f_{ij}^{xy} helps identify shortest paths and avoids unnecessary traffic. The terms $|A_x|$ and $|E|$ are used for normalization to ensure that the objective components remain independent of topology size and connectivity density. The coefficients ξ_1 , ξ_2 , and ξ_3 determine the trade-offs among the objective terms. Their selection and impact on system performance are analyzed in Section IV-G.

$$\min -\xi_1 \sum_{x \in X} \sum_{y \in Y} \frac{r_y^x}{|A_x|} - \xi_2 \sum_{i,j \in O} \frac{e_{ij} \Gamma_{ij}}{|E|} + \xi_3 \sum_{x \in X} \sum_{y \in Y} \sum_{i,j \in O} \frac{f_{ij}^{xy}}{|E|}$$

s.t. constraints: (2) – (6), vars.: $r_y^x, \Gamma_{ij}, f_{ij}^{xy} \geq 0$ (1)

The following constraints ensure feasible multipath routing by enforcing (i) server bandwidth limits, (ii) flow conservation, and (iii) reserved margins for robustness. Constraint (2) ensures that the total traffic delivered by a server-side switch does not exceed the aggregate available bandwidth of its outgoing links:

$$\sum_{y \in Y} r_y^i \leq \sum_{j \in O \setminus X} e_{ij} b_{ij}, \quad \forall i \in X \quad (2)$$

Constraint (3) ensures that each $i \in X$ transmits data at a rate of r_y^i through its neighboring OF switches, excluding other server-side switches. This prevents flows from re-entering other server-side switches, ensuring that data leaves the server region and enters the core network, which reduces the size of the optimization problem:

$$\sum_{j \in O \setminus X} e_{ij} f_{ij}^{iy} - \sum_{j \in O \setminus X} e_{ij} f_{ji}^{iy} = r_y^i, \quad \forall i \in X, \forall y \in Y \quad (3)$$

Constraint (4) ensures flow conservation at intermediate switches ($O \setminus \{X \cup Y\}$):

$$\sum_{j \in O} e_{ij} f_{ij}^{xy} - \sum_{j \in O} e_{ij} f_{ji}^{xy} = 0, \quad \forall i \in O \setminus \{X \cup Y\}, \forall x \in X, \forall y \in Y \quad (4)$$

Constraint (5) guarantees that each client-side OF switch receives the data transmitted by its associated server-side

TABLE II: Notation.

Sets	Description
O	Set of all OF switches, $\forall o \in O$.
E	Set of edges $e_{ij} \in E$, where $e_{ij} = 1$ indicates a direct link between i and $j \in O$.
X	Set of server-side OF switches, $\forall x \in X$.
Y	Set of client-side OF switches, $\forall y \in Y$.
A_x	Set of OF switches adjacent to server-side switch x , excluding switches in X , $\forall x \in X$.
M	Mapping of each $y \in Y$ to its corresponding array of $x \in X$.
C_y	Set of clients connected to client-side switch y .
C	Set of HAS clients, $\forall c \in C$.
Variables	Description
F	Fairness variable, promoting equitable quality allocation and enabling overall quality improvement. Minimizing F balances quality stability, and efficiency, with $F \in [0, 1]$.
r_y^x	Maximum end-to-end available bandwidth from server-side switch x to client-side switch y .
f_{ij}^{xy}	Portion of traffic from $x \in X$ to $y \in Y$ traversing edge (i, j) .
Γ_{ij}	Robustness variable in OPM reserving bandwidth margin on edge (i, j) .
$\psi_{c\ell}^x$	Binary variable indicating whether $x \in X$ sends layer ℓ to client c .
N_c	Normalized count of quality switching events for client c .
I_c	Normalized intensity of quality changes for client c .
Constants	Description
$b_{c\ell}$	Required bandwidth for layer ℓ of client c 's requested segment, $\ell \in L_c$, $c \in C$.
b_{ij}	Available bandwidth between OF switches i and j , $i, j \in O$.
P_c	Priority coefficient of client c based on its buffer level.
L_c	Maximum video quality level that client c can receive.
$\bar{r}_y^x$	r_y^x value derived from the OPM solution.
$\bar{f}_{ij}^{xy}$	f_{ij}^{xy} value derived from the OPM solution.
$\bar{\psi}_{c\ell}^x$	$\psi_{c\ell}^x$ value derived from the CPM solution.

switches:

$$\sum_{j \in O} e_{ij} f_{ij}^{xi} - \sum_{j \in O} e_{ij} f_{ji}^{xi} = -r_i^x, \forall x \in X, \forall i \in Y \quad (5)$$

Constraint (6) limits the total traffic on each edge to the available bandwidth b_{ij} while reserving Γ_{ij} as a robustness margin. The lower and upper bounds imposed on Γ_{ij} prevent degenerate cases in which the optimizer either provides no margin or reserves an excessively large portion of the capacity:

$$\sum_{x \in X} \sum_{y \in Y} f_{ij}^{xy} + \Gamma_{ij} \leq b_{ij}, lb_{ij} \leq \Gamma_{ij} \leq ub_{ij}, \forall i, j \in O \quad (6)$$

Additionally, OPM provides two key benefits beyond supporting CPM: it facilitates buffer management by comparing client demands with end-to-end available bandwidth $\bar{r}_y^x$, and scales $\bar{f}_{ij}^{xy}$ using factor η_y (7) when resources are insufficient even for base-layer delivery, ensuring fairness and mitigating

playback stalls.

$$\eta_y = \frac{\sum_{c \in C_y} b_{c\ell}}{\sum_{x \in X} \bar{r}_y^x}, \quad \forall y \in Y, \ell = 1 \quad (7)$$

Computational Complexity: OPM is formulated as a linear program and can be solved in polynomial time using Interior Point Methods (IPM) [39].

Summary of the OPM. OPM computes feasible end-to-end bandwidth values by coordinating three key variables: r_y^x , which represents the maximum available bandwidth from each server-side switch x to each client-side switch y ; f_{ij}^{xy} , which distributes this bandwidth across all available network paths; and Γ_{ij} , which reserves link-level capacity to ensure robustness under dynamic traffic conditions. Through flow-conservation and capacity constraints, these variables jointly enforce structurally valid multipath routing while minimizing unnecessary traffic. Solving the OPM yields the bounded values $\bar{r}_y^x$ and $\bar{f}_{ij}^{xy}$, which prune the search space of the subsequent CPM. Structurally, the decomposition works because OPM isolates all routing feasibility decisions—those that are computationally most expensive—and resolves them efficiently via linear programming, enabling the CPM to operate solely on quality under known bandwidth limits.

2) Core Programming Model (CPM)

The key objective of the CPM is to determine the optimal video-quality level and server assignment for each client in a way that maximizes overall QoE while ensuring fairness, stability, and feasibility under bandwidth constraints. Building on the end-to-end bandwidth bounds derived by the OPM, CPM minimizes a composite objective that balances three QoE-critical aspects: (i) a fairness variable F , which promotes equitable quality distribution across all clients; (ii) the quality-switch intensity I_c , which reduces large jumps between consecutive segment qualities; and (iii) the number of quality switches N_c , which prevents frequent oscillations. At the same time, CPM enforces SVC layer dependencies, guarantees base-layer delivery, prevents redundant server assignments, respects available bandwidth, and incorporates buffer-aware priorities to proactively mitigate stalls. In essence, CPM provides a QoE-optimized, fair, and stable per-segment quality allocation that remains fully aligned with the network's real-time resource limitations. The coefficient α prioritizes fairness and overall quality improvement by assigning greater weight to F compared to I_c and N_c , while β_1 and β_2 regulate the relative importance of I_c and N_c in the objective function (8).

$$\min \alpha F + \frac{\sum_{c \in C} (\beta_1 I_c + \beta_2 N_c)}{\text{len}(C)} \quad (8)$$

s.t. constraints : (9) – (15)

$$\text{vars. : } F, I_c, N_c \in [0, 1], \text{ and } \psi_{c\ell}^x, v_c \in \{0, 1\}$$

The detailed roles of constraints (9) through (15) are summarized in Table III for clarity.

a) Decodability Constraints:

Constraint (9) enforces that each client receives at least the base layer (layer 1), which is mandatory for video decoding, ensuring decodability as all enhancement layers depend on it.

$$\sum_{x \in X} \psi_{c\ell}^x = 1, \quad \forall c \in C, \quad \forall \ell = 1 \quad (9)$$

TABLE III: Summary of CPM constraints.

Constraint	Purpose/ Type	Simplified Explanation
(9)	Base Layer Guarantee	Every client must receive the base video layer ($\ell = 1$) from exactly one server ($x \in X$).
(10)	Server Uniqueness	Any given layer (ℓ) for any client (c) can be served by at most one server, preventing redundant transfers.
(11)	Layer Hierarchy / Decoding	Ensures that if layer ℓ is received, all preceding layers (1 to $\ell - 1$) must also be received.
(12)	Fairness Regulation	Links the fairness variable F to the client's allocated quality level. Minimizing F balances quality distribution.
(13)	Quality Switch Intensity	Minimizes how much the quality level changes between time slots, normalized by historical intensity.
(14)	Quality Switch	Minimizes the occurrence of quality changes between time slots, normalized by the historical switch count.
(15)	Bandwidth Capacity Limit	The total required bit rate (weighted by buffer priority P_c) cannot exceed the available bandwidth capacity $\bar{r}_y^x$.

Constraint (10) ensures that each layer of a client is transmitted by at most one server-side switch, eliminating redundant transfers:

$$\sum_{x \in X} \psi_{c\ell}^x \leq 1, \quad \forall c \in C, \quad \forall \ell \in \{1, \dots, L_c\} \quad (10)$$

Constraint (11) enforces the SVC dependency chain, which stipulates that a client cannot receive layer $\ell + 1$ unless it has received all layers $\leq \ell$. This constraint guarantees that all lower layers from 1 to ℓ are transmitted before layer $\ell + 1$.

$$\sum_{x \in X} (\psi_{c(\ell+1)}^x - \psi_{c\ell}^x) \leq 0, \quad \forall c \in C, \quad \forall \ell \in [1, L_c - 1] \quad (11)$$

b) Fairness Constraints:

Constraint (12) assumes identical, non-congested client-to-switch access links. Under this assumption, it enforces quality fairness only among clients competing for resources within the SDN-controlled network, rather than accounting for heterogeneous access-link capacities. It regulates the quality level assigned to each client:

$$F + \frac{1}{L_c} \sum_{x \in X} \sum_{\ell=1}^{L_c} \psi_{c\ell}^x \geq 1, \quad \forall c \in C. \quad (12)$$

The variable F acts as a global slack term that is minimized in the objective function. As a client receives more layers, its normalized quality increases, allowing F to decrease, while normalization by L_c ensures uniform treatment of clients with different maximum quality levels. For example, if two clients have $L_c = 4$ and one is assigned three layers while the other receives only one, the latter yields a normalized quality of 0.25, forcing $F \geq 0.75$. Since the objective minimizes F , the optimizer instead equalizes the layer assignments, yielding normalized qualities of 0.5 for both clients (corresponding to a video quality level of $\ell = 2$) and reducing the bound to $F \geq 0.5$.

c) Video Stability QoE Constraints:

Constraint (13) limits the video quality switch intensity (I_c), a variable minimized in the objective function (8). Switch intensity is defined as the magnitude of the quality difference between consecutive video segments. For example, if segment 10's quality level is 2 and segment 11's quality level is 4, the intensity is $|2 - 4| = 2$. In the equation: $\bar{\ell}_c$ denotes the client's previously obtained quality level; $\bar{\mu}_c$ represents the accumulated intensity since the session began; and $I_{max} = \max\{\bar{\mu}_c + L_c - 1 \mid \forall c \in C\}$ serves as the normalization factor. Therefore, the constraint ensures I_c is always at least the normalized intensity, enabling the objective function to penalize large fluctuations.

$$I_c - \frac{\bar{\mu}_c + \left| \sum_{x \in X} \sum_{\ell=1}^{L_c} \psi_{c\ell}^x - \bar{\ell}_c \right|}{I_{max}} \geq 0 \quad (13)$$

Constraint (14) limits the number of quality switches (N_c), which is minimized in the objective function (8). This constraint accounts only for the occurrence of a quality change, independent of its magnitude. In (14), $\bar{v}_c$ denotes the cumulative number of past switches for client c , and $N_{max} = \max\{\bar{v}_c \mid \forall c \in C\} + 1$ is used for normalization. The binary variable v_c indicates whether a switch occurs in the current time slot. The constraint enforces N_c to be greater than or equal to the normalized switch count, enabling the optimizer to minimize the number of switching events. For example, if a client's quality changes between consecutive segments (e.g., from level 4 to 1), then $v_c = 1$, causing N_c to increase regardless of the change magnitude.

$$N_c - \frac{\bar{v}_c + v_c}{N_{max}} \geq 0, \quad \forall c \in C \quad (14)$$

d) Capacity Constraints:

Constraint (15) defines the bandwidth capacity limits between each server-side and client-side switch pair (x, y) using $\bar{r}_y^x$ to determine the values of the $\psi_{c\ell}^x$ variables. FROG proactively mitigates stall risks by incorporating the buffer priority variable P_c , which dynamically increases when a client's buffer level drops below a predefined threshold, enabling a more conservative bandwidth allocation for that client. When the buffer level exceeds the threshold, P_c retains its default value of 1. This does not increase the actual bitrate requirement of the requested video layer; rather, it causes the optimization model to treat the client as requiring more bandwidth than its nominal demand. Consequently, paths with limited residual capacity become less attractive, encouraging the selection of paths with greater available bandwidth. By preserving additional headroom along the selected path, this mechanism improves resilience to bandwidth fluctuations and proactively reduces the risk of playback stalls.

$$\sum_{c \in C_y} \sum_{\ell=1}^{L_c} \psi_{c\ell}^x P_c b_{c\ell} \leq \bar{r}_y^x, \quad \forall x \in X, \forall y \in Y \quad (15)$$

Summary of the CPM. The CPM builds on the bandwidth bounds $\bar{r}_y^x$ provided by the OPM to determine the optimal video quality level and server-side switch assignment for each client. The binary decision variables $\psi_{c\ell}^x$ specify whether server-side switch x delivers layer ℓ to client c , while

the auxiliary variables F , I_c , and N_c capture fairness, quality-switch intensity, and the number of quality changes, respectively. These variables interact through constraints that enforce SVC layer dependencies, guarantee unique server assignment per layer, and maintain bandwidth feasibility using buffer-aware priority weights P_c . Structurally, the CPM works efficiently because routing complexity has already been eliminated by the OPM; thus, the MILP focuses exclusively on allocating feasible layers in a manner that maximizes QoE while maintaining fairness and playback stability across all clients.

Computational Complexity Analysis: After OPM-based fixing of routing decisions and server selection, together with the pruning of infeasible layer combinations, the original MILP formulation is transformed into a sequential SVC layer-selection process performed by the CPM. Owing to the hierarchical dependency enforced by Constraint (11), only contiguous layer assignments starting from the base layer are feasible. Consequently, for a client with L_c available layers, only L_c feasible quality assignments exist instead of the 2^{L_c} arbitrary layer combinations of a general binary formulation. Therefore, the computational effort grows proportionally with the number of clients and available layers, resulting in an overall complexity of $O(|C| \cdot L_c)$. This complexity refers to the sequential layer-selection process following OPM-based fixing and pruning rather than to the complexity of solving the original MILP formulation.

Justification: (i) Sequential Selection: Constraint (11) reduces per-client complexity from $O(2^{L_c})$ to $O(L_c)$; (ii) Bandwidth Pruning: Layer assignment checks $\sum_{c \in C_y} \sum_{\ell=1}^{L_c} b_{c\ell} \leq \bar{r}_y^x$ execute in $O(1)$ using incremental sums; (iii) Server-side Switch Selection: Fixing server-side switches to client-side switches leaves only one server to optimize in CPM, introducing an $O(1)$ operation and avoiding extra complexity.

Empirical Validation: Experimental results confirm the linear $O(|C|L_c)$ scaling behavior, with solution times of approximately 1 s for 4,000 clients, including flow setup and verification stages. However, we do not report QoE metrics for the 4000 client experiments because these tests assume a fully available network up to its capacity and rely on programmatically generated client requests rather than actual playback-driven demand. Therefore, they are intended primarily to validate computational scalability rather than user-perceived performance. Nonetheless, the optimization outputs of the system (i.e., quality-level assignments produced by the solver) remain consistent with those observed in smaller-scale experiments conducted under equivalent capacity conditions, suggesting that the system scales without introducing adverse optimization behavior, although a full QoE analysis at this scale is left for future work. When tested with 4,000 clients under a provisioned capacity of 5 Mbps per client, the optimization stage remains sub-second on average, while the complete optimization and flow-preparation pipeline averages 1.12 s, with an average CPU utilization of approximately 82%.

Computational Complexity Analysis of Pure MILP: Without the decoupling benefits of the OPM phase, the problem reduces to a *Pure MILP*. Even when the sequential layer constraint (11) is applied to linearize

the layer-selection process, the overall model remains *computationally intractable*. This is due to the inherent complexity of the *Multi-Server Multi-Path (MSMP)* routing problem—simultaneously deciding flow routing and server assignment—which corresponds to an instance of the *Integer Multi-Commodity Flow Problem (IMCFP)*, known to be NP-Complete. The total complexity is dominated by the exponential factors related to network routing and server assignment: The total number of Operations = $O(|C| \cdot L_c \cdot 2^{|X| \cdot |E|})$ **Justification:** (i) Sequential Layer Selection: Same effect as in CPM. (ii) Flow-routing Selection (Intractability): Since the model must simultaneously optimize flow across all network edges ($|E|$) for all competing demands, this introduces a dependency on $O(2^{|E|})$ in the exponent, stemming from the path choice combinations. (iii) Server-side Switch Selection (Intractability): Without the server-fixing step, the model must decide which server ($x \in X$) is optimal for each layer, introducing an exponential dependency on $O(2^{|X|})$ in the exponent for the assignment problem.

To demonstrate the substantially shorter solve time achievable with FROG, we compare its runtime with a Pure MILP implementation in Table IV, using a 30-client, single-server setup with the ED-I-1080p SVC dataset and the topology in Fig. 1. Although this configuration is simplified, the Pure MILP formulation remains computationally intractable due to multipath decision variables. In this scenario, optimization and flow configuration must be completed within 2 s to prevent playback stalls (approximately 1.5 s for optimization and 0.5 s for flow setup). Even when the entire 2 s budget is hypothetically allocated solely to optimization—ignoring flow preparation—the Pure MILP still cannot meet this limit, frequently exceeding it by a large margin, as reflected by the “Exceeds” entries in Table IV. Including the flow setup time only worsens the violation. Even when the optimizer is explicitly capped at 2 s, 3 s, 4 s, or 5 s, Pure MILP still fails to converge. The “Max” column reports the longest runtime observed in any single optimization cycle, while “Avg” provides the mean runtime across all cycles, illustrating both worst-case and typical computational effort. Consequently, flow updates are delayed, causing severe stalls and QoE degradation. In contrast, FROG consistently satisfies these timing constraints, confirming its practicality for real-time deployment.

TABLE IV: Runtime and QoE comparison between FROG and Pure MILP.

Method	Limit	Exceeds	Max (s)	Avg. (s)	QoE
FROG	None	0	0.65	0.12	22.94
	2 s	80	2.53	0.97	−13.39
	3 s	76	3.57	1.09	−18.77
Pure MILP	4 s	82	4.62	1.21	−27.64
	5 s	97	5.57	1.32	−41.66
	None	68	553.89	2.58	−160.70

To further assess the solution quality of FROG, an additional experiment was conducted using a small-scale

scenario consisting of 10 clients, for which the *Pure MILP* formulation could be solved to convergence. The experiment used the same network topology and the BBB-I-1080p dataset, with all switch-to-switch links configured at 50 Mbps. To isolate the effect of the decomposition itself, the buffering-aware enhancement was disabled. The results showed that the *Pure MILP* achieved an average QoE value of 32.59, while FROG achieved 32.34, corresponding to a difference of only 0.77%. FROG also achieved a slightly higher average video quality (3.80 versus 3.78), a lower average stall duration (0.16 s versus 0.39 s), and a slightly higher average quality variation (22.60 versus 20.60). These results indicate that the decomposition employed by FROG preserves solution quality closely to that of the optimal MILP formulation while maintaining the computational efficiency required for larger-scale deployments.

We compare FROG with a variant that excludes the CPM module to assess its impact. As shown in Table V, removing CPM leads to a substantial degradation in QoE and fairness, along with a significant increase in quality switches. This indicates that a pure LP-based (OPM) approach, which relies only on instantaneous bandwidth availability, is insufficient for stable QoE delivery. The observed performance loss is mainly caused by the sequential layer allocation process, which is sensitive to client arrival order.

TABLE V: Comparison of FROG and FROG without CPM.

Method	QoE	Video Quality	Quality Change	Stalls (s)	Fairness
FROG	22.08	2.95	29.61	0.27	0.94
FROG-No CPM	15.79	3.15	62.51	0.25	0.90

3) Optimization Algorithm

The coordination between the OPM and CPM models is visualized in Fig. 3, with the steps detailed in Algorithm 1. To coordinate the two optimization models and address the NP-Complete nature of the problem, we propose Algorithm 1. The algorithm relaxes the original formulation by incrementally fixing video layers, enabling tractable solutions while maintaining high QoE across clients and paths. It determines each client's optimal layer quality and corresponding flow path, then communicates the results to clients together with the assigned servers.

Initially, OPM (1) is solved to determine the maximum end-to-end bandwidth $\bar{r}_y^x$ between each $x \in X$ and $y \in Y$, and the results are passed to CPM. Each $M[y]$ —the ordered set of all x s for a given y —is sorted by available bandwidth. For each y , the algorithm iterates over $x \in M[y]$ to assess base-layer sufficiency and fixes the corresponding x - y pair for that layer. When bandwidth is insufficient even for the base layer, $\bar{f}_{ij}^{xy}$ is scaled using the ratio η_y (7) to ensure delivery. The loop terminates upon reaching the highest-capacity switch in $M[y]$, which remains as the optimization variable in CPM. CPM then initializes ψ_{cl}^x based on $\bar{r}_y^x$, fixing infeasible links to 0 and retaining feasible ones as active variables (Rows 1–18). Any remaining unset ψ_{cl}^x variables are expressed as `IlExpr` objects and incorporated into CPM, which is then solved (Rows 19–25). If the CPM yields no

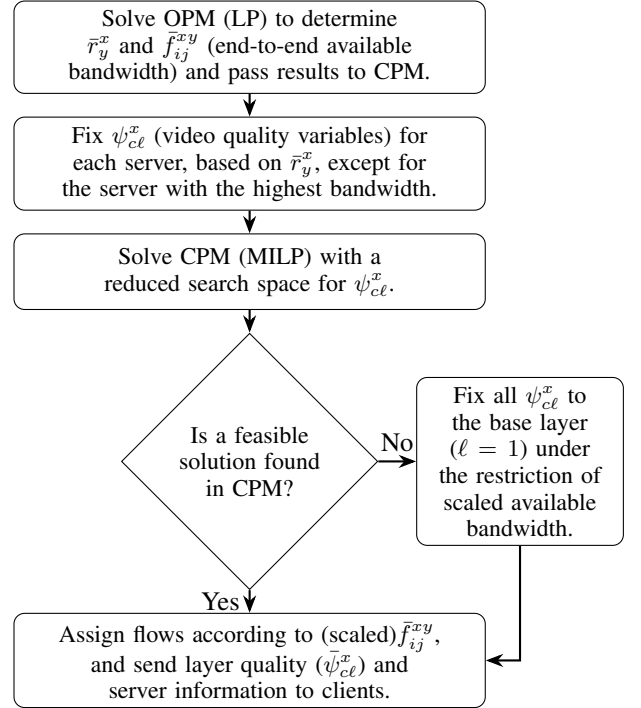


Fig. 3: Workflow of the optimization algorithm integrating the OPM (LP) and CPM (MILP) models.

feasible solution, each $\bar{\psi}_{cl}^x$ is reset to 0, and base layers are allocated to all clients using the available $\bar{r}_y^x$ values. Flow assignments are subsequently finalized according to $\bar{f}_{ij}^{xy}$, and each client is notified of its assigned quality layer and corresponding server (Rows 26–34). Note that flow priorities are incrementally increased at each iteration to ensure smooth transitions between configurations and prevent interference from ONOS's default routing protocols.

C. Deployment Challenges and Practical Considerations

Deploying FROG in operational networks introduces several challenges that differ from controlled SDN testbeds. Production environments typically include heterogeneous switches and controllers with varying SDN capabilities. Differences in OF support, pipeline design, and hardware constraints can limit fine-grained multipath control, requiring careful compatibility assessment and gradual integration. Controller load becomes a key factor. Although FROG's optimization is lightweight, real controllers must also manage routing, telemetry, and security tasks, and frequent re-optimization may introduce overhead or excessive flow churn. Deployment should therefore balance responsiveness with operational stability and avoid unnecessary reconfiguration cycles. The optimization algorithm could be outsourced to an external server, removing a potential bottleneck from the SDN control plane.

FROG relies on accurate and timely bandwidth measurements as well as CMCD-style client feedback. However, real networks often provide telemetry data at coarse intervals, and measurements may be noisy or temporarily inconsistent due to queueing dynamics, congestion, or data-plane sampling delays. Out-of-date information can lead to suboptimal routing or

Algorithm 1: Video quality, multi-server, and multipath assignment algorithm.

Data: $M, \forall r_y^x, \forall f_{ij}^{xy}, \forall \psi_{cl}^x, \forall P_c, \forall b_{cl}, X, Y, C, C_y$

Result: Optimized solution of $\forall \psi_{cl}^x$ and flows

```

1 Solve the objective function of OPM (1)
2 Sort each  $M[y]$  in non-decreasing order based on  $\bar{r}_y^x$ 
3 foreach  $y$  in  $M$  do
4   if  $\eta_y(\text{Eq. 7}) > 1$  then increase each  $\bar{f}_{ij}^{xy}$  for
     current  $y$  as  $\bar{f}_{ij}^{xy} = \eta_y \bar{f}_{ij}^{xy}$ 
5   foreach  $x$  in  $M[y]$  do
6     if  $x$  is the last element of  $M[y]$  then break
7     double  $\text{fixed\_rate} \leftarrow 0.0$ 
8     if  $\bar{r}_y^x == 0$  then
9       foreach  $c \in C_y$  do
10        for  $\ell \leftarrow 1$  to  $L_c$  do  $\psi_{cl}^x \leftarrow (0,0)$ 
11      else
12        for  $\ell \leftarrow 1$  to  $L_c$  do
13          foreach  $c \in C_y$  do
14            if  $\psi_{cl}^x$  is not set to  $(1,1)$  then
15              if  $\text{fixed\_rate} + P_c b_{cl} \leq \bar{r}_y^x$ 
16                then
17                   $\text{fixed\_rate} += P_c b_{cl}$ 
18                   $\psi_{cl}^x \leftarrow (1,1)$ 
19                else  $\psi_{cl}^x \leftarrow (0,0)$ 
19       $x \leftarrow$  last element of  $M[y]$ 
20      foreach  $c \in C_y$  do
21        for  $\ell \leftarrow 1$  to  $L_c$  do
22          if  $\psi_{cl}^x$  is not set to  $(1,1)$  then
23             $\text{Ilo\_Expr} += \psi_{cl}^x b_{cl}$ 
24       $\text{CPM.add}(\text{Ilo\_Expr} \leq \bar{r}_y^x)$ 
25 Solve CPM (8)
26 if “No solution available in CPM” then
27   set each  $\psi_{cl}^x \leftarrow 0$ 
28   foreach  $c \in C$  do
29      $y \leftarrow c$ 's connected  $y$ 
30     foreach  $x$  in  $M[y]$  do
31       if  $\bar{r}_y^x - b_{c1} \geq 0$  then
32          $\bar{r}_y^x = b_{c1}, \psi_{c1}^x \leftarrow 1$ , and break
33 Assign flows according to  $\bar{f}_{ij}^{xy}$ 
34 Send layer & server information to clients

```

bandwidth misallocation, potentially causing stalls or unnecessary quality fluctuations. Incorporating telemetry smoothing or short-term prediction can help compensate for such variability, but operators must still account for the limitations of the underlying measurement infrastructure. Network operators must ensure that dynamic routing and server-selection decisions do not violate operational constraints such as security requirements, inter-cluster traffic limits, or SLA-driven performance guarantees. Multipath distribution and frequent reconfiguration may conflict with policies governing traffic

locality, peering arrangements, or lawful interception. As a result, the optimization logic must be aligned with operational rules, and all flow-update actions must undergo validation to prevent unintended routing behaviors or policy violations.

Multi-CDN or multi-domain environments impose administrative boundaries that limit coordinated path control. FROG performs best when operating within a domain where the operator controls routing; in more distributed settings, cross-domain coordination may need to occur at coarser timescales. Traffic in production networks is inherently unpredictable. Sudden congestion, flash crowds, or partial failures may invalidate previously optimal configurations. Even with built-in robustness margins, the system must handle abrupt shifts through fallback paths and conservative allocations. Finally, client-side heterogeneity remains an obstacle. Not all players support CMCD/CMSD or accept external guidance, which can lead to mixed adaptation behaviors. Ensuring fairness and stability in such environments requires mechanisms that accommodate both cooperative and traditional HAS clients without compromising system-wide performance.

IV. PERFORMANCE EVALUATION

In our evaluation, we aim to answer the following questions:

- **QoE maximization:** How effective is FROG in maximizing QoE across clients under diverse network conditions, particularly when video layers are distributed across multiple servers?
- **Real-time performance:** Is FROG suitable for online optimization without degrading performance, even when server selection is considered alongside path and bitrate decisions?
- **Stall and oscillation minimization:** How much does FROG reduce playback stalls and quality oscillations?
- **Control plane overhead:** What is the control plane overhead and scalability of FROG?

A. Baselines

To answer these questions, we evaluate FROG against four adaptive streaming baselines under dynamic client-server assignments, grouped into two paradigms: (i) S²VC' and S²VC'+ network-assisted HAS clients based on [10], updated with a flow-assignment strategy consistent with this study. Specifically, OPM is used to determine flow paths instead of the original flow-assignment mechanism proposed in [10], which relies on packet tagging before packets leave the media server. This modification was adopted to avoid the additional scalability limitations associated with per-packet tagging while preserving the original quality-selection methodology. The efficient bandwidth mechanism is consistently applied in both FROG and S²VC'+, ensuring that all compared methods operate under the same bandwidth estimation assumptions. Both S²VC' and S²VC'+ run as ONOS-compatible SDN applications. (ii) SARA [40] and baseline DASH—decentralized clients making local bitrate decisions based on buffer occupancy and throughput; SARA uses actual segment sizes, while DASH relies on MPD-file estimates.

B. Video Content and Encoding

All approaches were evaluated using pre-encoded SVC sequences with four quality layers, following DASH-SVC recommendations. Segments are 2 s long, including clips with mixed motion complexity to reflect realistic streaming conditions. BBB-I-1080p and ED-I-1080p SVC video sets were used [41], [42]. Only BBB-I-1080p results are shown here; ED-I-1080p results and the optimizer source code are available online [43].

C. Key Performance Indicators (KPIs)

The following KPIs are used to compare FROG with baselines:

- **Average Delivered Quality Level ($\bar{Q}$):** Let Q_i^c be the quality level received by client c for segment i , and s_c the number of segments. Then,

$$\bar{Q} = \frac{1}{|C|} \sum_{c=1}^{|C|} \frac{1}{s_c} \sum_{i=1}^{s_c} Q_i^c$$

where $|C|$ is the total number of clients.

- **Stall Duration ($\bar{D}$):** Let D_c be the total stall duration for client c . Then,

$$\bar{D} = \frac{1}{|C|} \sum_{c=1}^{|C|} D_c$$

- **Quality Change (ΔQ):** The average absolute change in quality between consecutive segments:

$$\Delta Q = \frac{1}{|C|} \sum_{c=1}^{|C|} \frac{1}{s_c - 1} \sum_{i=1}^{s_c - 1} |Q_{i+1}^c - Q_i^c|$$

- **QoE Value ($\bar{QoE}$):** Computed as

$$\bar{QoE} = \alpha \cdot \bar{Q} - \beta \cdot \Delta Q - \gamma \cdot \bar{D},$$

with empirically chosen weights $\alpha = 10, \beta = 0.25, \gamma = 0.25$. These values follow common practice in HAS evaluation and were verified to preserve relative performance trends.

- **QoE Fairness Index (FI) [44]:** Measures fairness of QoE distribution:

$$FI = 1 - \frac{\sigma}{\sigma_{\max}},$$

where σ is the standard deviation of QoE values and $\sigma_{\max} = (H - L)/2$, with $H = 40$ and $L = 0$. FI is averaged over three runs per scenario.

D. Evaluation Environment

Experiments were conducted on Ubuntu 20.04.6 LTS using Mininet 2.3.0, IBM ILOG CPLEX 22.1.1, and ONOS 3.0.0.b, on a system with an AMD Ryzen 5 5500U CPU and 32 GiB memory. For standard tests, switch-to-switch links were set to 100 Mbps for 20–30 clients and 250 Mbps for 50 clients; client/server-to-switch links were 1 Gbps. Congestion was introduced via limited switch-to-switch capacities, and scenarios with Poisson-distributed link capacities were additionally evaluated. Experiments used two topologies: (i) one server-side switch, and (ii) four server-side switches.

The bandwidth values used in the evaluation are intended to create controlled congestion scenarios rather than to emulate the absolute capacities of modern backbone links. The considered SDN-controlled domain may also represent

regional aggregation or edge-cloud transport where video traffic can still cause congestion. Since the proposed optimization framework depends primarily on the relative relationship between traffic demand and available bandwidth, similar high-utilization conditions may arise in higher-capacity networks under proportional scaling of traffic demand and network capacity.

E. Comparative Results of FROG

We evaluate QoE for FROG and baseline approaches under various bandwidth scenarios. Experiments with Poisson-distributed bandwidth are labeled with the suffix **-P**. QoE is similar for 20- and 50-client setups due to identical per-client bandwidth, while the 30-client setup shows lower QoE because per-client bandwidth is reduced.

FROG achieves the highest QoE, followed by S^2VC^+ , thanks to improved video quality and smoother adaptation. SARA and DASH perform worse, sometimes yielding negative QoE, as both rely on ONOS's default Dijkstra-based forwarding without network-level optimization (see Fig. 4).

Key observations include: (i) using all available paths improves performance compared to relying on the best path alone. Accordingly, FROG, S^2VC^+ and S^2VC^+ , which exploit multiple paths, achieve higher video quality than DASH and SARA. (ii) Efficient bandwidth utilization strongly affects video quality, as seen in S^2VC^+ versus S^2VC^+ . Furthermore, the performance gap between FROG and S^2VC^+ / S^2VC^+ is related to their quality selection mechanisms. Their LP-relaxed formulations determine quality levels in a continuous space and subsequently map the resulting solutions to discrete video representations by rounding down. In contrast, FROG directly optimizes discrete quality selections within the formulation, which avoids potential quality loss due to rounding. In addition, FROG employs a buffering mechanism that supports higher quality levels under varying network conditions (Fig. 5).

Effective bandwidth utilization together with FROG's buffering mechanism contributes to stall-free playback and generally fewer quality transitions compared to the baseline approaches (Figs. 6 and 7). An exception is observed in the 30-client scenario, where DASH exhibits fewer quality transitions; however, this is mainly due to its tendency to select lower (base-layer) representations under congested conditions, which limits adaptation opportunities and results in reduced overall video quality.

Perceptual video quality is further evaluated using VMAF, as summarized in Table VI. The results show that higher bitrate or selected quality levels do not necessarily translate into proportional VMAF gains. For example, in the 30-client scenario, FROG achieves a substantially higher average quality level than S^2VC^+ (approximately 3 versus 1.5), while the VMAF difference is only about 1.6 points. Since both FROG and S^2VC^+ achieve VMAF values above 96, the evaluated schemes operate in a perceptually saturated high-quality region for the BBB-I-1080p sequence. Consequently, performance differences are more apparent in bandwidth utilization, adaptation behavior, and streaming stability than in perceptual quality. Overall, FROG maintains comparable

perceptual quality while supporting efficient multipath, multi-server streaming.

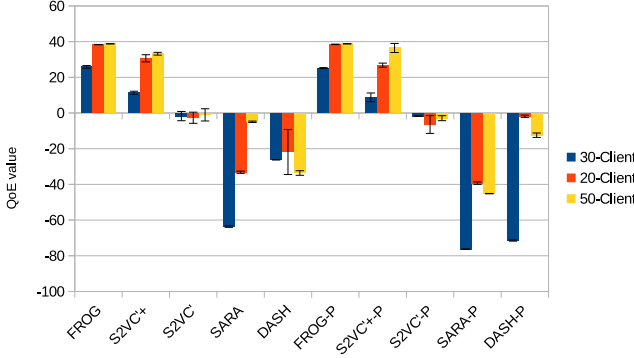


Fig. 4: QoE values for the single server-side setup.

Video stall results are shown in Fig. 7. Both FROG and S2VC+ exhibit almost no stalls. Since both approaches employ similar optimization mechanisms and utilize the same flow assignment configuration for fair comparison, these results clearly demonstrate the effectiveness of FROG's flow assignment approach. In contrast, SARA and DASH show considerably poorer performance due to the absence of network-level support.

We evaluate SARA, DASH, and FROG using a setup with four server-side switches, each connected to a dedicated server. SARA and DASH clients are manually and evenly distributed across these switches. The robust and intelligent utilization of all available paths, effective bandwidth utilization, and the robust buffering mechanism of FROG enable it to outperform both SARA and DASH in the four-server scenario, even though these clients are manually and evenly distributed across the server-side switches. FROG consistently achieves the highest QoE values across all test cases. Although SARA slightly outperforms FROG in video quality for the 30-client scenario (see Fig. 9), its performance in quality variation (see Fig. 10) and stall duration (see Fig. 11) is considerably worse. The DASH client shows the weakest performance in nearly all tests, underscoring the advantage of network assistance. Overall, FROG demonstrates consistently superior QoE compared with all other approaches, primarily due to its integrated network-assisted optimization and flow assignment mechanisms.

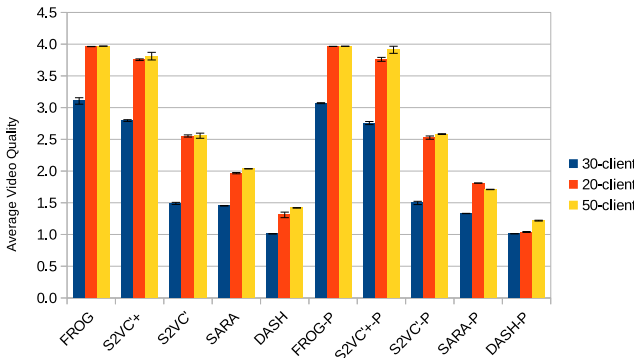


Fig. 5: Average video quality for the single server-side setup.

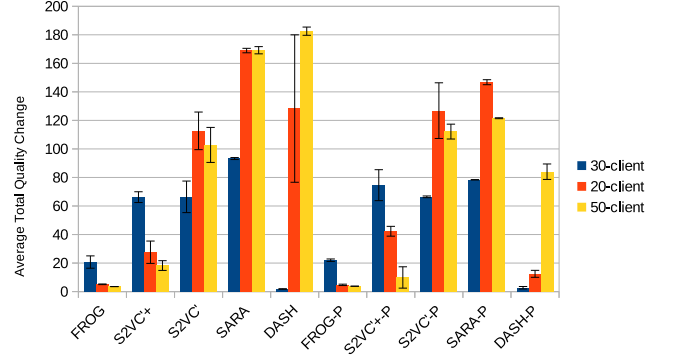


Fig. 6: Average total quality changes for the single server-side setup.

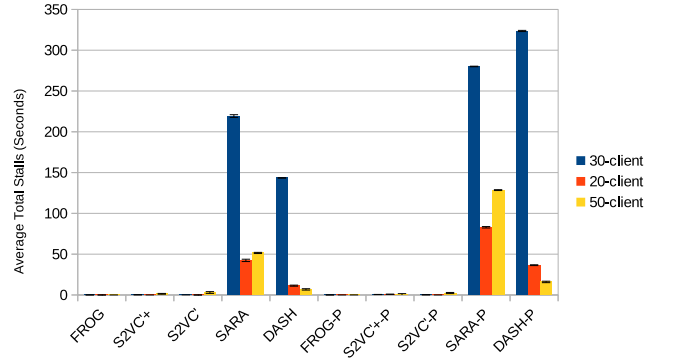


Fig. 7: Average total stalls for the single server-side setup.

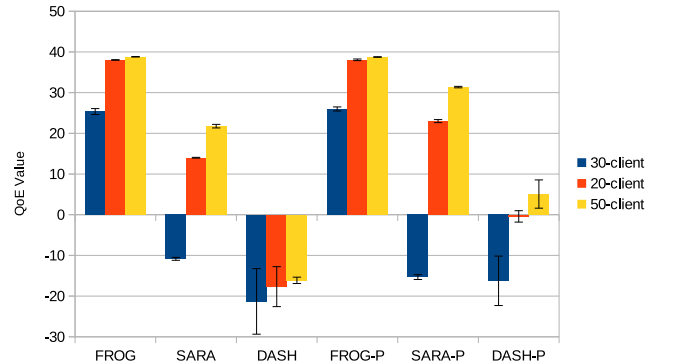


Fig. 8: QoE values for the four server-side setup.

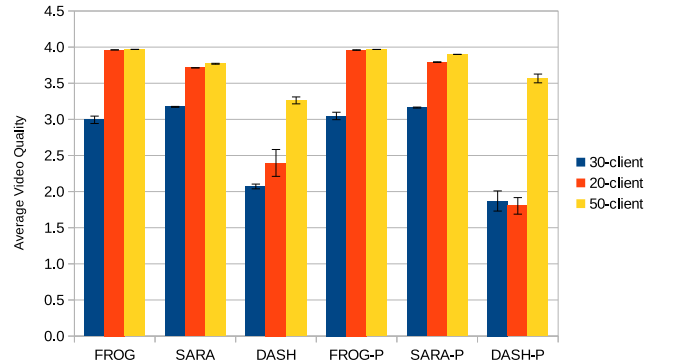


Fig. 9: Average video quality of the four server-side setup.

TABLE VI: Comparison of average VMAF scores under different client loads.

Metric	20 Clients			30 Clients			50 Clients		
	FROG	S ² VC'+	S ² VC'	FROG	S ² VC'+	S ² VC'	FROG	S ² VC'+	S ² VC'
Mean VMAF	97.38	97.27	95.97	96.55	96.29	94.95	97.39	97.37	95.96

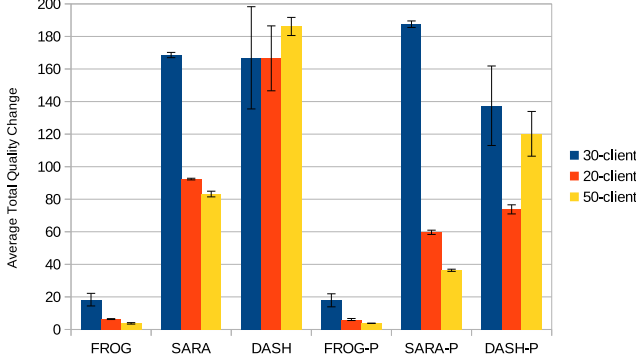


Fig. 10: Average total quality changes for the four server-side setup.

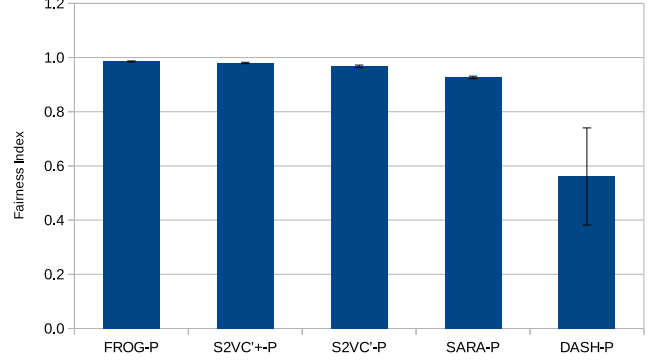


Fig. 12: QoE Fairness Index under Poisson-distributed bandwidth for single server-side setup.

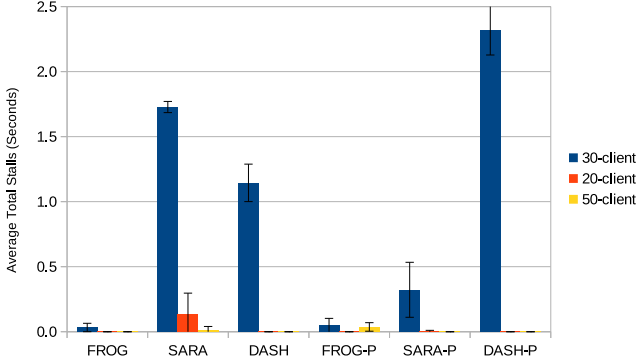


Fig. 11: Average total stalls for the four server-side setup.

We assess client QoE fairness using the *FI* metric [44], averaged over 50 clients in a single server-side configuration. The evaluation is conducted under a Poisson-based bandwidth distribution, representing the most dynamic test scenario. As shown in Fig. 12, FROG achieves the highest fairness, closely followed by S²VC'+ with only a 0.01 difference. These results confirm that FROG delivers superior QoE while preserving a high level of fairness among clients. FROG achieves this result primarily through its robust buffering mechanism, which helps maintain high QoE while preserving fairness. In contrast, the fairness of S²VC'+ is partly influenced by its rounding-down strategy, which tends to reduce quality differences among clients by mapping continuous quality values to lower discrete representations. An important outcome of this study is the observation that the OPM, through its strict bandwidth upper bound formulation, provides a powerful mechanism for simplifying downstream optimization problems. By determining the end-to-end available bandwidth in advance, OPM eliminates the need to recalculate bandwidth availability in subsequent models, enabling faster and more

efficient decision-making.

F. Ablation Study

To isolate the contribution of key design components, we conducted an ablation study considering: (i) removal of the buffer-aware operation to evaluate its impact and to analyze the conditions triggering the P_c variable, given that client buffer levels are monitored (levels 0 and 1); (ii) removal of the buffer priority mechanism (P_c); and (iii) restriction of the number of available paths. A 30-client setup with the ED-I-1080p SVC video set was used. The results are summarized in Table VII and Table VIII. As shown in Table VII, both buffer-aware operation and buffer priority (P_c) consistently reduce stall occurrences. In particular, P_c acts as a lightweight buffer-protection mechanism that increases routing headroom for clients in critical buffer states, thereby improving playback stability under congestion. While the improvement magnitude is moderate in this setting, it remains consistent across runs despite the relatively low congestion level. The parameterization of P_c (1.05 for buffer level 0 and 1.1 for buffer level 1) is intentionally conservative. Larger values were observed to lead to over-protection behavior, which can make the OPM infeasible and force fallback to ONOS best-path routing, degrading overall optimization quality. This confirms that P_c is most effective as a fine-grained stability controller rather than a dominant optimization driver.

The most significant performance impact is observed in the multipath configuration. Reducing the system to two paths (Sw1→Sw3→Sw6 and Sw1→Sw4→Sw6) leads to a clear degradation in all QoE metrics, while a single-path configuration (Sw1→Sw3→Sw6) results in severe quality collapse due to insufficient capacity for even base-layer delivery. These results demonstrate that multipath routing is essential for maintaining service continuity under load.

Overall, the ablation study confirms a clear hierarchy of

TABLE VII: Ablation study of buffer-priority (P_c).

Configuration	Video Quality	Quality Change	Stalls (s)	QoE
Standard FROG	3.03	27.00	0.01	23.55
No Buffer	2.91	28.60	0.07	21.97
No Buffer + No P_c	2.92	30.03	0.30	21.63

TABLE VIII: Ablation study of multipath mechanisms.

Configuration	Video Quality	Quality Change	Stalls (s)	QoE
Single-Path	1.08	14.10	286.96	-64.43
Two-Path	1.97	29.17	17.88	7.93
Three-Path	3.03	27.00	0.01	23.55

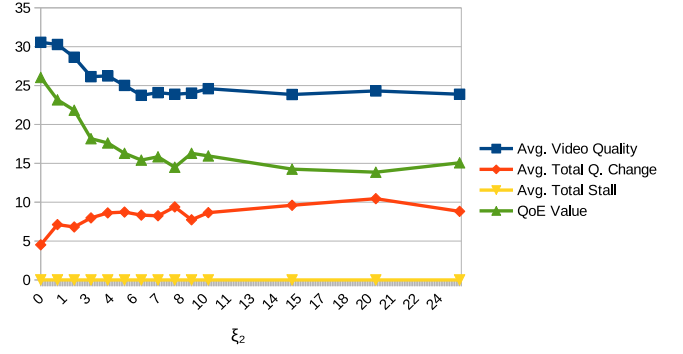
contributions: multipath utilization is the primary performance enabler, buffer-aware operation provides secondary stability improvements, and P_c delivers additional fine-grained stall reduction under congested conditions.

G. Parameter Sensitivity Analysis

To evaluate sensitivity to segment duration, additional experiments were conducted using 4-second segments formed by combining two consecutive 2-second SVC segments. The corresponding layer bandwidth requirements were recalculated, and the optimization was repeated. The resulting performance trends remained consistent with those for 2-second segments, indicating that the proposed framework is robust to the investigated segment durations.

The OPM objective function (1) consists of three weighted terms controlled by ξ_1 , ξ_2 , and ξ_3 . The first and third terms represent competing objectives: maximizing the bandwidth allocated between server- and client-side switches (r_y^x) and minimizing network-edge bandwidth usage (f_{ij}^{xy}), respectively. Since bandwidth allocation is the primary objective, a higher weight is assigned to the first term. The normalization by $|A_x|$ and $|E|$ improves scale consistency across different network topologies. Experimental results show that assigning a larger weight to the third term drives both r_y^x and f_{ij}^{xy} toward zero, resulting in zero end-to-end bandwidth allocation. Accordingly, $\xi_1 = 2$ and $\xi_3 = 1$ are selected to provide a balance between bandwidth allocation and edge utilization. The coefficient ξ_2 controls the trade-off between reserved bandwidth (Γ_{ij}) and edge bandwidth usage (f_{ij}^{xy}). Since reserved bandwidth is limited to 10% of each edge capacity, $\xi_2 \in [0, 10]$ is sufficient to evaluate its practical impact; $\xi_2 = 25$ is additionally considered as an extreme case. The results are shown in Fig. 13. Note that the KPI values used for QoE computation are defined in Section IV-C and are scaled according to the QoE formulation. Increasing ξ_2 from 0 to 7 reduces average video quality due to increased bandwidth reservation, while performance stabilizes beyond $\xi_2 = 7$ as the reservation constraint approaches its upper bound.

To investigate the impact of the weighting coefficients in the CPM objective function (8), the parameter sensitivity analysis is presented in Table IX. Since the normalized objective terms have an upper bound of one, the initial experiment assigned equal importance to the competing objectives by

Fig. 13: Performance evaluation of OPM under varying ξ_2 values.TABLE IX: Impact of α , β_1 , and β_2 on QoE metrics.

Test No	α	β_1	β_2	Video Quality	Quality Change	Stalls (s)	QoE
1	2	1	1	2.41	6.60	0.00	22.46
2	4	1	1	2.89	10.47	0.01	26.31
3	5	1	1	3.04	20.73	0.00	25.23
4	10	1	1	3.08	23.27	0.03	25.00
5	10	1	2	3.08	26.27	0.02	24.27
6	10	2	2	3.07	25.13	0.03	24.36
7	10	2	3	2.87	11.40	0.00	25.85
8	10	0	5	2.87	9.73	0.02	26.30
9	10	5	0	3.00	16.37	0.00	25.94

setting $\alpha = \beta_1 + \beta_2$ (Test 1). Increasing α while keeping $\beta_1 = \beta_2 = 1$ shifted the optimization toward video quality, increasing the average video quality from 2.41 to 3.08 at the cost of more frequent quality changes. For $\alpha > 5$, further increases produced only marginal changes in video quality, indicating a stable operating region.

With $\alpha = 10$, the effects of β_1 and β_2 were further examined. The results show that β_2 , which weights the quality-change penalty (N_c), has a stronger influence on quality stability than β_1 , which weights the quality increment term (I_c). Increasing β_2 substantially reduced the number of quality switches while maintaining competitive QoE, whereas increasing β_1 produced a smaller reduction, providing finer adjustment of quality transitions. These results support the selected parameter values as a balance between video quality, quality stability, and overall QoE.

H. Control Plane Overhead

To evaluate FROG's control plane overhead, we report the OF protocol overhead in Table X. An OFPT_FLOW_MOD message is, conservatively, about 150 bytes. Including TCP/IP and Ethernet overhead of 58 bytes, the total on-the-wire size of a FLOW_MOD message is approximately 208 bytes. We update switch tables every 2 seconds for each video layer according to the optimization results. Considering 4 layers per video, the total control plane overhead for one client is approximately 0.0033 Mbps. Table X shows the overhead for different numbers of clients and switches.

TABLE X: OF protocol message overhead (Mbps).

Number of Clients	Number of Switches		
	6	10	20
20	0.40	0.67	1.33
30	0.60	1.00	2.00
50	1.00	1.67	3.33

We report the average CPU load of the SDN controller and optimizer in Table XI. We observed that ONOS CPU usage remains roughly constant over time, indicating that flow assignments impose minimal load. In contrast, the optimizer's CPU usage varies across iterations, reflecting changes in network conditions and video layer allocations. Implementing four servers reduces the optimizer's computational burden by pre-assigning video layers, making its operation more efficient.

TABLE XI: CPU usage percentages.

# of Server-Sides	# of Clients	Optimizer (%)	ONOS (%)
1	20	3.75	9.20
1	30	7.51	9.69
1	50	4.60	11.22
4	20	1.88	13.15
4	30	7.29	14.29
4	50	3.00	12.99

V. CONCLUSION

In this study, we propose a framework for layered HTTP streaming over SDN that leverages both multipath and multi-server capabilities. The experimental results demonstrate substantial QoE improvements, where the underlying optimization problem—although NP-complete—is efficiently and near-optimally solved via the developed relaxation approach. Our adaptive media delivery framework, **FROG**, achieves up to $2.32\times$ **higher QoE** than its closest competitor, S^2VC^+ , under constrained network conditions using the Elephants Dream (ED) SVC video dataset in a Poisson-distributed, single server-side setup, while almost completely eliminating rebuffering events. FROG primarily utilizes SDN and mathematical optimization models. The proposed method enables efficient utilization of network capacity, resulting in both significant improvements in video quality and notable reductions in quality fluctuations.

These characteristics open promising directions for future research, particularly in the context of **6G networks** and **volumetric video streaming**. The anticipated features of 6G—ultra-high bandwidth, low latency, and intelligent edge processing—align naturally with OPM's capability to pre-calculate and constrain bandwidth usage. Extending OPM-based frameworks to such environments could enable more scalable and responsive streaming systems. Similarly, volumetric video streaming will impose unprecedented demands on network resources, where OPM's ability to pre-determine bandwidth bounds can play a crucial role in ensuring real-time delivery with minimal overhead. Future work will also investigate path-aware and QoE-aware client clustering to improve scalability while preserving

sufficient per-client granularity. In addition, we will explore access-aware fairness metrics that distinguish operator-controlled resource limitations from client-specific access-link constraints.

ACKNOWLEDGMENT

This work was partly funded by the Bavarian Government through the HighTech Agenda (HTA), the Scientific and Technological Research Council of Türkiye (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under Grant Number 115E449.

REFERENCES

- [1] Cisco Inc., "Cisco annual internet report (2018–2023) white paper," Cisco Inc., Tech. Rep., 2023, see especially "Consumer Internet Traffic by Application" section. [Online]. Available: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>
- [2] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over HTTP," *IEEE Commun. Surveys Tuts.*, vol. 21, pp. 562–585, 2019 (DOI: 10.1109/COMST.2018.2862938).
- [3] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *ACM NOSSDAV*, 2012 (DOI: 10.1145/2229087.2229092).
- [4] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *ACM IMC*, 2012.
- [5] R. Kalan and I. Dulger, "A survey on QoE management schemes for HTTP adaptive video streaming: Challenges, solutions, and opportunities," *IEEE Access*, 2024.
- [6] CTA, "CTA-5004: Web application video ecosystem—common media client data," 2020, accessed: Mar. 19, 2024. [Online]. Available: <https://cdn.cta.tech/cta/media/media/resources/standards/pdfs/cta-5004-final.pdf>
- [7] —, "CTA-5006: Web application video ecosystem—common media server data," 2022, accessed: Mar. 19, 2024. [Online]. Available: <https://cdn.cta.tech/cta/media/media/resources/standards/pdfs/cta-5006-final.pdf>
- [8] X. Jiang, F. R. Yu, T. Song, and V. C. M. Leung, "A survey on multi-access edge computing applied to video streaming: Some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, pp. 871–903, 2021.
- [9] A. Erfanian, F. Tashtarian, and M. H. Yaghmaee, "On maximizing QoE in AVC-based HTTP adaptive streaming: An SDN approach," in *IEEE/ACM IWQoS*, 2018.
- [10] F. Tashtarian, A. Erfanian, and A. Varasteh, "S2VC: An SDN-based framework for maximizing QoE in SVC-based HTTP adaptive streaming," *Comput. Netw.*, vol. 146, pp. 33–46, 2018.
- [11] F. Tashtarian and C. Timmerer, "REVISION: A roadmap on adaptive video streaming optimization," *arXiv*, 2024.
- [12] J. W. Kleinrouweler, S. Cabrero, and P. Cesar, "Delivering stable high-quality video: An SDN architecture with DASH assisting network elements," in *ACM MMSys*, 2016.
- [13] A. Bentaleb, A. C. Begen, and R. Zimmermann, "SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking," in *ACM Multimedia*, 2016 (DOI: 10.1145/2964284.2964332).
- [14] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous, "SDNHAS: An SDN-enabled architecture to optimize QoE in HTTP adaptive streaming," *IEEE Trans. Multimedia*, vol. 19, pp. 2136–2151, 2017 (DOI: 10.1109/TMM.2017.2733344).
- [15] A. Bentaleb, A. C. Begen, and R. Zimmermann, "ORL-SDN: Online reinforcement learning for SDN-enabled HTTP adaptive streaming," *ACM TOMM*, vol. 14, pp. 71:1–71:28, 2018 (DOI: 10.1145/3219752).
- [16] R. S. Kalan, M. Sayit, and A. C. Begen, "Implementation of SAND architecture using SDN," in *IEEE NFV-SDN*, 2018 (DOI: 10.1109/NFV-SDN.2018.8725632).
- [17] I. M. Ozelik and C. Ersoy, "Chunk duration-aware SDN-assisted DASH," *ACM TOMM*, vol. 15, pp. 1–22, 2019.
- [18] A. A. Barakabitze, L. Sun, I.-H. Mkwawa, and E. Ifeachor, "A novel QoE-centric SDN-based multipath routing approach for multimedia services over 5G networks," in *IEEE ICC*, 2018.

- [19] S. Bikas and M. Sayit, "Improving QoE with genetic algorithm-based path selection for MPTCP," *IEEE Trans. Netw. Service Manag.*, vol. 21, pp. 3874–3888, 2024.
- [20] D. Bhat, A. Rizk, M. Zink, and R. Steinmetz, "SABR: Network-assisted content distribution for QoE-driven ABR video streaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, pp. 1–25, 2018.
- [21] R. S. Kalan, "Improving quality of HTTP adaptive streaming with server and network-assisted DASH," in *IEEE CNSM*, 2021.
- [22] R. Farahani, F. Tashtarian, H. Amirpour, C. Timmerer, M. Ghanbari, and H. Hellwagner, "ES-HAS: An edge- and SDN-assisted framework for HTTP adaptive video streaming," in *IEEE LCN*, 2021.
- [23] R. Farahani, F. Tashtarian, H. G. Amirpour, C. Timmerer, M. Ghanbari, and H. Hellwagner, "CSDN: CDN-aware QoE optimization in SDN-assisted HTTP adaptive video streaming," in *IEEE LCN*, 2021.
- [24] R. Farahani, F. Tashtarian, C. Timmerer, M. Ghanbari, and H. Hellwagner, "LEADER: A collaborative edge- and SDN-assisted framework for HTTP adaptive video streaming," in *IEEE ICC*, 2022.
- [25] R. Farahani, M. Shojafar, C. Timmerer, F. Tashtarian, M. Ghanbari, and H. Hellwagner, "ARARAT: A collaborative edge-assisted framework for HTTP adaptive video streaming," *IEEE Trans. Netw. Service Manag.*, vol. 20, pp. 625–643, 2023.
- [26] R. Farahani, A. Bentaleb, C. Timmerer, M. Shojafar, R. Prodan, and H. Hellwagner, "SARENA: SFC-enabled architecture for adaptive video streaming applications," in *IEEE ICC*, 2023.
- [27] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, pp. 1103–1120, 2007.
- [28] S. G. Ozcan and M. Sayit, "Multipath transmission aware abr algorithm for svc has," *Computer Communications*, vol. 201, pp. 20–36, 2023.
- [29] A. Gohar and S. Lee, "Multipath dynamic adaptive streaming over HTTP using scalable video coding in software defined networking," *Appl. Sci.*, vol. 10, p. 7691, 2020.
- [30] Y. F. Yeznabad, M. Helfert, and G.-M. Muntean, "QoE-driven cross-layer bitrate allocation approach for MEC-supported adaptive video streaming," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 6, pp. 6857–6874, Dec. 2024.
- [31] R. A. Kırmızıoğlu, A. M. Tekalp, and B. Görkemli, "Distributed virtual selective-forwarding units and SDN-assisted edge computing for optimization of multi-party WebRTC videoconferencing," *Signal Process. Image Commun.*, vol. 130, p. 117173, 2025.
- [32] R. S. Kalan and M. Sayit, "SDN-assisted codec, path, and quality selection for HTTP adaptive streaming," *IEEE Access*, vol. 9, pp. 129 917–129 932, 2021.
- [33] A. C. Begen, "Manus manum lavat: Media clients and servers cooperating with common media client/server data," in *ACM ANRW*, 2021 (DOI: 10.1145/3472305.3472886).
- [34] A. Bentaleb, M. Lim, M. N. Akcay, A. C. Begen, and R. Zimmermann, "Common media client data (CMCD): Initial findings," in *ACM NOSSDAV*, 2021 (DOI: 10.1145/3458306.3461444).
- [35] M. Nguyen, B. Taraghi, A. Bentaleb, R. Zimmermann, and C. Timmerer, "CADLAD: Device-aware bitrate ladder construction for HTTP adaptive streaming," in *IEEE CNSM*, 2022.
- [36] S. Pham, W. Law, A. C. Begen, D. Silhavy, B. Berthelot, S. Arbanowski, and S. Steglich, "Common media server data (CMSD)—update on implementations and validation of key use cases," in *ACM MHV*, 2023 (DOI: 10.1145/3588444.3591031).
- [37] A. Bentaleb, M. Lim, M. N. Akcay, A. C. Begen, and R. Zimmermann, "Bitrate adaptation and guidance with meta reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 23, pp. 10 378–10 392, 2024 (DOI: 10.1109/TMC.2024.3376560).
- [38] T. Tran, D. Gageot, C. Neumann, G. Bichot, A. Tlili, K. Boutiba, and A. Ksentini, "On the benefits and caveats of exploiting quality on demand network APIs for video streaming," in *ACM NOSSDAV*, 2024.
- [39] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *ACM Symp. Theory Comput. (STOC)*, 1984.
- [40] P. Juluri, V. Tamarapalli, and D. Medhi, "SARA: Segment-aware rate adaptation algorithm for dynamic adaptive streaming over HTTP," in *IEEE ICCW*, 2015.
- [41] ITEC, Alpen-Adria Universität Klagenfurt, "BBB SVC DASH dataset," 2015, accessed: Oct. 23, 2023. [Online]. Available: <http://ftp.itec.aau.at/datasets/SVCDASHDataset2015/BBB/I/segments/1080p/>
- [42] —, "ED SVC DASH dataset," 2015, accessed: Oct. 23, 2023. [Online]. Available: <http://ftp.itec.aau.at/datasets/SVCDASHDataset2015/ED/I/segments/1080p/>
- [43] M. Türk, "FROG: Evaluation results for ED-I-1080p SVC video set," <https://github.com/mustafaturk/FROG/tree/main>, 2025, accessed: Oct. 13, 2025.
- [44] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, and M. Varela, "Definition of QoE fairness in shared systems," *IEEE Commun. Lett.*, vol. 21, pp. 184–187, 2016.



Mustafa Türk received the B.S. degree in Finance from Anadolu University, and the M.Sc. degrees in Information Technologies from Sakarya University and Ege University in 2010 and 2016, respectively. He received the Ph.D. degree in Information Technologies from Ege University, Türkiye, in 2024. His research interests include optimization, software-defined networking, network function virtualization, future networks, and video streaming. Additionally, he has worked as a network and system expert for more than 25 years.



Müge Sayit (Senior Member, IEEE) is a Professor of Computer Science at the International Computer Institute, Ege University, Türkiye. Currently, she is at the School of Computer Science and Electronic Engineering, University of Essex, UK. Her research interests include software-defined networking, network function virtualization, future networks, video streaming, and video codecs. She has served as the principal investigator or as a researcher in various R&D projects.



Ali C. Begen is a computer science professor at Ozyegin University in Istanbul and the founder of Networked Media, an innovation and engineering studio focused on IP video technologies. Previously, he was a technical lead at Cisco in San Jose, California. Begen earned his PhD in Electrical and Computer Engineering from Georgia Tech in 2006. His work has earned him numerous academic and industry honors, including an Emmy Award for Technology and Engineering. He is a Streaming Video Technology Alliance (SVTA) Fellow, holds over 40 US patents, and is recognized as a Distinguished Member of the ACM. More details are at <https://ali.begen.net>.



Andreas J. Kessler received his Diploma degree in 1995 in Mathematics and Computer Science from University of Augsburg, Germany, and a Ph.D. degree in 2002 in Computer Science from University of Ulm, Germany. From 2003–2004 he was an Assistant Professor at the School of Computer Engineering, Nanyang Technological University, Singapore. From 2005–2007 he was a lecturer in Computer Science at Karlstads Universitet, Sweden, where he has been a professor in Computer Science since 2007. Since 2023, he has been a professor in Computer Science at Deggendorf University of Technology, Germany, where he leads the laboratory of Intelligent Networks and Systems (INets). Andreas J. Kessler is a Senior Member of IEEE. His research interests include Software Defined Networking, Network Function Virtualization, Network Programmability and Wireless Networks.