Learning-based Joint QoE Optimization for Adaptive Video Streaming based on Smart Edge

Xiaoteng Ma, Qing Li, *Member, IEEE,* Yong Jiang, *Member, IEEE,* Gabriel-Miro Muntean, *Senior Member, IEEE* and Longhao Zou, *Member, IEEE*

***Abstract*—The latest increase in HTTP-based adaptive video streaming over the Internet enables a growing number of clients to compete for a shared bottleneck bandwidth. This competition may affect users' Quality of Experience (QoE) negatively, especially in terms of fairness and stability. This paper presents *Flex-Steward*, a solution that performs multi-client joint QoE optimization for adaptive video streaming during bottleneck bandwidth sharing. Joint QoE optimization refers to improving QoE fairness among clients with various video devices and availing from differentiated services with different priorities. Flex-Steward deploys an adaptive bitrate delivery algorithm based on Neural Networks (NN) and reinforcement learning at the network edge. It relies on a trained NN model to make appropriate bitrate recommendations in terms of video chunks to be requested by clients sharing the same bottleneck bandwidth. Flex-Steward is assessed in comparison with alternative state-of-the-art algorithms under different network conditions using a real-life prototype. Results show how Flex-Steward reduces the unfairness in terms of joint QoE optimization with between 10.9% and 41.7%.**

***Index Terms*—Bitrate Recommendation, Edge Computing, Reinforcement Learning, Joint QoE Optimization**

I. Introduction

LATELY, video streaming traffic has become mainstream in the Internet, especially in terms of bandwidth share [1]. In the context of massive amounts of video traffic, exchanged by very diverse devices, guaranteeing high quality of experience (QoE) for video clients becomes a significant challenge [2]. It is crucial for content providers (CP) to improve QoE for their client services in order to increase client engagement, which profoundly influences their revenue [3].

HTTP-based adaptive streaming (HAS) is becoming the predominant form of video delivery. Some previous works designed efficient adaptive bitrate (ABR) algorithms to optimize the QoE of a single client based on HAS [4]–[8]. In general, these algorithms run on client-side video players and dynamically select the most appropriate bitrate for each video chunk (e.g., 2-second block) based on various client-side observations. These algorithms improve the QoE of a single client without considering other clients that share the common bottleneck bandwidth. Such an approach encourages competition among all flows and results in poor performance in terms of stability, fairness and bandwidth utilization [9]. Such competitions are common at the access point between the edge network (e.g., a campus or enterprise network) [10]–[13] and the public network. Moreover, the latest increase in the use of high-resolution devices has also put pressure on the backbone network, which makes the problems caused by bottleneck bandwidth competition more prominent [14].

To increase their revenues, CPs are keen to realize *joint QoE optimization* among clients sharing the bottleneck bandwidth. Joint QoE optimization refers to improving QoE fairness among clients with diverse video devices and achieving differentiated service for clients with different priorities. Achieving joint QoE optimization targets an increase in the number of clients satisfied with the service offered by CPs. On the one hand, clients should have fair QoE if they are equally important to the CP. However, in general, the bandwidth is fairly shared in terms of QoS rather than QoE [15], which is unfair to clients with high-resolution devices. For instance, two devices with different resolutions have different user-perceived qualities despite supporting video services with the same bitrate. However, they request the same bitrate video content if they observe QoS fairness. Consequently, the lower-resolution client may get excessive bandwidth share, while the higher-resolution client may suffer from poor video quality due to insufficient bandwidth allocation. On the other hand, CPs would like to provide differentiated service among clients to improve their revenue. They provide different priorities for clients based on how much the clients pay for their service. At the same time, a client who pays more to CPs is more likely to quit when getting poor QoE. Therefore, CPs should ensure that high-priority clients have better QoE.

The conventional QoE optimization methods, client-side [5]–[8] or server-side [16]–[22], are end-to-end control solutions and regard the network as a black box. Such approaches have difficulties in measuring the bottleneck bandwidth as well as gathering client and network information. In order to overcome these drawbacks, some network-assisted approaches were proposed [12], [23]–[27]. However, due to the limited computing capabilities in the network, it is difficult to track the client status changes and do the control in real time, resulting in sub-optimal performance in joint QoE optimization.

The latest advent of edge computing supports performance improvement of joint QoE optimization [28]. Some projects such as Open Edge [29] and CORD [30] start to work on edge

X. Ma is with the Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, Shenzhen, China (Email: maxt17@mails.tsinghua.edu.cn).

Q. Li is with Peng Cheng Laboratory (PCL), Shenzhen, China (zoulh@sustech.edu.cn)

Y. Jiang are with the Tsinghua Shenzhen International Graduate School, Shenzhen, China (Email: jiangy@sz.tsinghua.edu.cn).

G.-M. Muntean is with the Performance Engineering Lab, Dublin City University, Ireland (Email: gabriel.muntean@dcu.ie).

L. Zou is with the Southern University of Science and Technology and also with PCL Research Center of Networks and Communications, Peng Cheng Laboratory (PCL), Shenzhen, China (Email: zoulh@sustech.edu.cn).

Corresponding author: Qing Li (liq8@sustech.edu.cn)

computing. They transfer the computing power from the cloud to the edge of the network, where it is closer to the end clients. Such solutions make the edge servers (e.g., DSL-boxes, WiFi access points, and base stations) offer resources through open and standardized mechanisms to any applications, enabling computation at the edge. Since the competition for bottleneck is more likely to occur among clients in the same edge network [10]–[12], [31], CPs can deploy an ABR agent as a smart network function on the edge servers where it is close to the end clients. The ABR agent can collect client and network status data in real time and operate the control to clients according to the CP's business model.

This paper proposes *Flex-Steward*, an innovative solution that teaches ABR of multiple clients how to achieve joint QoE optimization by flexibly adapting to dynamic changes in network conditions. Flex-Steward employs an ABR agent which is deployed as a network function on the edge servers. Compared with the client-side-based ABR schemes, the ABR agent on distributed edge nodes at network edge has information gathered from multiple managed clients near the edge server and reduces the computational pressure on the clients. At the same time, compared with server-side-based ABR schemes, the edge supports a more timely interaction with the clients.

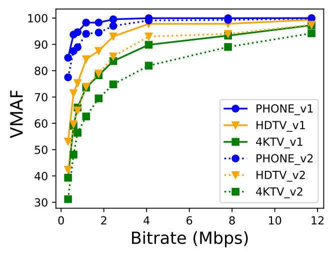
An innovative reinforcement learning (RL) approach is designed to be employed in the Flex-Steward ABR agent to provide bitrate recommendations for the clients. Flex-Steward represents its control policy as a neural network (NN) that maps raw observations (e.g., bottleneck bandwidth samples, all clients' status) to the next chunk's bitrate decision for a client. The NN provides an effective and scalable way to incorporate continuous observations into the control policy, so that Flex-Steward can employ such observations as input, without quantifying their correlation.

Unlike previous end-to-end learning-based ABR algorithms [5], [32], [33], it is challenging to build a numerical simulator to accurately simulate the bandwidth allocation during the bandwidth competitions to pre-train the NN model. Thus, we pre-train the designed RL model for Flex-Steward based on a system prototype with various network conditions and client characteristics. A NN model is then deployed at edge servers to offer bitrate recommendations and perform online training.

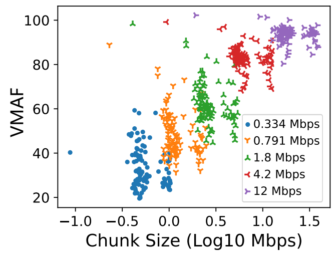
Flex-Steward is evaluated using a full prototype system implementation, considering typical edge network topologies and multiple DASH players based on *dash.js* [34] to emulate request and download processes. Flex-Steward is compared with other algorithms that aim to achieve QoE optimization under different network conditions. We define a utility function to represent the client's evaluation of the service provided by CPs. Compared with other algorithms, Flex-Steward maintains the QoE of all clients at a high level and reduces between 10.9% and 41.7% the utility unfairness. These results demonstrate that Flex-Steward has an excellent performance in achieving joint QoE optimization.

The main contributions of this paper are as follows:

- An intelligent solution to achieve joint QoE optimization for clients sharing the bottleneck bandwidth is designed.



(a) bitrate vs. VMAF.



(b) VMAF vs. chunk size.

Fig. 1: The quality and size characteristics under different devices for typical static video v_1 and dynamic video v_2.

- A RL-based learning algorithm for the ABR agent that learns the control policy through experience data is proposed. Such an experience-driven approach has advantages in terms of learning the client behavior and predicting the effect on QoE caused by bitrate adjustment decisions.
- A general reward function that allows the ABR agent to guarantee QoE fairness among diverse video devices and realize differentiated service offerings according to client priorities in order to increase CP's revenue is devised.
- To validate the performance of all considered ABR algorithms, a full prototype system which extends the *dash.js* reference player [34] was built. Testing results show that Flex-Steward outperforms other algorithms when realizing joint QoE optimization in diverse dynamic network conditions.

Next, Sec.II of the paper highlights the background and motivations of joint QoE optimization. As this paper extends a short conference article appeared in [35], the difference between this paper and [35] is discussed in Sec.II. The system design is detailed in Sec.III and the RL formulation is elaborated in Sec.IV. The RL agent and video delivery system implementation details are presented in Sec.V. The results of extensive evaluation experiments are described in Sec.VI. Sec.VII presents related works in relation to QoE optimization. Finally, the paper is concluded in Sec.VIII.

II. Background and Motivations

A. Adaptive Video Streaming for QoE Optimization

Nowadays, HTTP-based adaptive streaming (HAS) has become the predominant form of video delivery. Videos are encoded into multiple bitrate versions and saved in source servers (i.e., the data center of the CP or its leased CDN cluster). In the video download and playback process that is widely deployed today, the client requests for a chunk n with appropriate bitrate r according to the network status to ensure that the video can be played smoothly with high quality. After receiving the client's requests, the source responds chunks to the client according to its request. However, it is extremely difficult to decide the appropriate bitrate of each chunk for the client due to highly dynamic network conditions (i.e., bandwidth, latency, and packet loss).

To evaluate the effectiveness of bitrate adjustment decisions, the following three QoE-related metrics are usually used [5]–

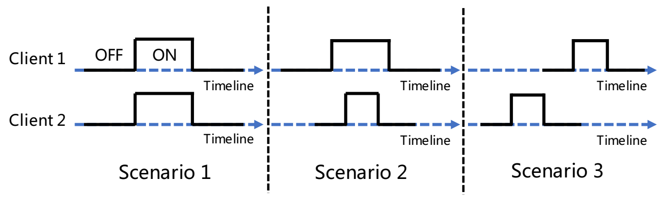


Fig. 2: Three scenarios when two clients sharing the same bandwidth.

[8], [33], [36]–[38]: ***1) Average bitrate/Average perceptual quality*** High bitrate indicates clients have higher perceptual quality, which is associated with better client QoE. Recent work [33], [39] proves that Video Multi-method Assessment Fusion (VMAF) [40] can better characterize clients' perceptual quality. Fig.1a shows the relationship between average bitrate and average VMAF. It illustrates that the marginal improvement in perceived quality decreases at higher bitrates. Besides, the average VMAF varies among videos and devices with different screen resolutions. Note that v_1 is a video that contains mostly static scenes and v_2, represents a video with highly dynamic scenes. ***2) Rebuffering time.*** The rebuffering time is defined as the total time that the video buffer stays empty during video playback, which greatly damages clients' QoE. ***3) Bitrate/Perceptual quality switch.*** Quality fluctuations caused by bitrate switches can also reduce clients' QoE. Recent work [33] also shows that VMAF's fluctuations characterize well the impact of bitrate switches on clients' QoE.

Nevertheless, the control of clients' QoE is a challenging task. First, an ABR algorithm should balance a variety of conflicting QoE metrics [5]. For example, when the network bandwidth is limited, consistently requesting high bitrate chunks increases video quality, but results in rebuffering events which eventually decrease the quality. Secondly, bitrate selection for a given chunk can have a cascading effect on the state of the client. Thirdly, the control decisions available to ABR algorithms are coarse-grained, so that it is difficult to control the client QoE accurately. Finally, the quality and size of different video chunks of the same video are different, as shown in Fig.1b. The video player should weigh the benefits of improving video quality and the risk of rebuffering caused by increasing the chunk size, and make a reasonable bitrate adjustment decision.

B. QoE Degradation Caused by Bandwidth Competition

To increase revenue, CPs hope to increase the number of clients who are satisfied with their QoE given limited resources. However, when multiple clients are competing for the same bandwidth, the clients have a high risk of suffering from poor performance in terms of stability, fairness and bandwidth utilization [9], which makes it difficult for CPs to guarantee high QoE for a wide range of clients. To explain the issue clearly, Fig.2 shows the temporal overlap of the ON-OFF periods among two competing clients as an example. Assume that the bandwidth capacity of the sharing bandwidth is C and a single active connection obtains the whole bandwidth, while two active connections share it fairly. We denote C_1 and C_2 as the throughput observed by two clients. The ON period indicates that the downloading session is active (i.e., the client is downloading a chunk), while the OFF period implies that the downloading process is off.

Scenario 1 shows that the two clients are perfectly aligned, and both of them observe that $C_1 = C_2 = \frac{C}{2}$. However, the fairly shared bandwidth may result in unfair QoE among clients. On one hand, as shown in Fig.1b, the quality and size of chunks are different. When two clients download two chunks of the same quality, but different sizes, the client who downloads the large chunk will experience a long downloading time, which damages the buffer and has a risk of causing rebuffering or quality degradation decisions. On the other hand, the buffer status of clients are also different. Fairly sharing the bandwidth may cause a low-buffer client to suffer from bad QoE. Scenario 2 shows how the ON period of client 2 falls in the ON period of client 1. It may happen when client 2 requests a chunk with a smaller size. Client 1 observes throughput $C_1 > \frac{C}{2}$ and client 2 observes throughput $C_2 = \frac{C}{2}$. Thus, client 1 continues to request a higher bitrate, and client 2 requests a lower bitrate, resulting in unfair QoE distribution among clients. Scenario 3 shows the case that two clients have no overlap of ON periods. Each client monopolizes the available bandwidth during ON period so that $C_1 = C_2 = C$. In this case, both clients overestimate their fairly shared bandwidth, which results in each client's requests for large chunks and causes congestion on the shared link. The congestion makes the clients to reduce their estimated bandwidth in the next request. This phenomenon will cause frequent fluctuations in the bitrate requested and achieved by clients, which leads to instability of their QoE.

Moreover, from the perspective of CPs, the hope is to achieve differentiated services among clients with different priorities in order to improve their revenue. CPs provide different priorities for clients based on how much the clients pay for their services. Therefore, it is expected that high-priority clients are less susceptible to suffer from QoE decrease.

In summary, to retain more clients, CPs should provide fair QoE for clients with the same priority and differentiate service among clients with different priorities according to their business strategy. To achieve theese, CPs should define a utility function to weigh QoE fairness and differentiated services.

C. Introducing Learning-based ABR at the Edge

Several works are proposed to address the QoE degradation problem caused by bottleneck competition [20], [21], [41]–[45]. However, these solutions have limited success in tracking heterogeneous clients' states and realizing joint QoE optimization. As detailed in Sec.II-A, the control of a single client's QoE is a challenging task. Combining the QoE of multiple clients to perform joint optimization is a more difficult task. In this context, this paper proposes an experience-driven ABR to improve the performance by performing joint QoE optimization for video streaming under multi-client bandwidth competition. This approach uses NN to represent the status of multiple clients, and utilizes RL to train the solution to make

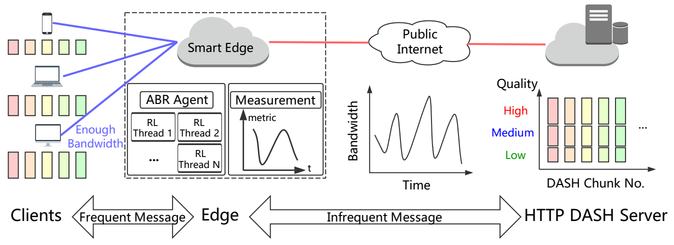


Fig. 3: Video delivery system with edge server.

optimal decisions to achieve joint QoE optimization. The RL trains the policy that can select actions with long-term value through learning multi-dimensional state data, so as to achieve the goal of satisfying the QoE demands of multiple clients as much as possible.

The real-time information collection and real-time control of the managed clients are required to achieve joint QoE optimization. Fig.3 shows a typical video delivery system with an edge server [35]. Compared with the client and the server, deploying the ABR agent at the edge is a better choice for the following reasons:

- **The edge server can have frequent message exchanges with clients it serves with low latency and high stability, which implies that the edge server can monitor the client status in real-time;** this is critical for real-time control over clients. The delivery path between the HTTP DASH Server (DS) and the access point is volatile, so that it introduces high latency and packet loss [13], [46]. To realize fine-grained control over the QoE of clients, the ABR agent should collect fresh client state data to make proper bitrate decisions and send bitrate recommendations rapidly to the corresponding client. On the other hand, the ABR agent only needs infrequent message exchanges with DS, such as the Media Presentation Description (MPD) file and the client's priority, which does not have such strict performance requirements.
- **The edge server can measure the bottleneck performance with a lower cost.** DS has difficulty in perceiving bottlenecks for millions of clients, which introduces high measurement costs. Furthermore, measurement at the client-side does not have a global view of the status of clients, so that the client-side ABR only has sub-optimal performance input in joint QoE optimization. Thus, we deploy the ABR agent at the edge server to locate and measure the bottleneck for groups of clients and realize joint QoE optimization.
- **The edge server has the computing power to perform NN computation and online training**, and moving the ABR algorithm from the clients to the edge server alleviates the load NN computation would put on client devices.

In conclusion, considering diverse potential locations, the edge server is the best place to deploy the ABR agent to manage heterogeneous clients. The edge server are generally deployed by the network providers on a large scale. However, from the perspective of information security and information integrity, the content provider should operate the ABR agent at the smart edge. The network provider can offer resources through open and standard mechanisms to any applications. Thus, the content providers can deploy the agent of Flex-Steward as a smart network function on the smart edges. In this way, the network providers can benefit by renting out edge servers' computing power to content providers. On the other hand, content providers can jointly retain more users by optimizing clients' QoE and increasing revenue.

In the next section, we will describe the design of Flex-Steward in detail. Note that this paper extends a short conference article which has appeared in [35]. The initial conference paper does not consider the chunk-level quality and size variations of a video, which influences the performance of ABR. Additionally, different from [35], this paper refers to the quality-aware QoE model [33], which makes the design of the ABR algorithm consider the quality difference among videos and be closer to the optimization of the real clients' QoE. Finally, the conference paper used a Mininet-based [47] evaluation, whereas this paper validates the performance of Flex-Steward using a full system prototype based on an extended *dash.js* reference player. The feasibility and performance of Flex-Steward is demonstrated with various network traces and client characteristics.

III. Flex-Steward Design

A. Overview

Fig.3 illustrates the principle behind the Flex-Steward solution, which adaptively delivers video from an HTTP DASH Server (DS) of a CP to clients via a smart edge [35]. The "Smart Edge" label indicates the deployment of the proposed intelligent ABR algorithm at the edge servers. The bandwidth between the smart edge and end hosts is generally reliable, while the bottleneck bandwidth competition often occurs in the Public Internet [13], [46]. The Flex-Steward includes the following major components:

1) Clients: There are a variety of device types, as shown in Fig.3. The perceived service quality on various devices is different due to their various resolutions, which implies that different types of devices have different optimal bitrates. A rectangle in Fig.3 represents a single chunk. A larger chunk size implies a larger bitrate. The figure suggests that a client with a high-resolution device should request a higher bitrate to satisfy its quality demand. The client can be a *Thin Client* with simple ABR decision logic at the client-side. (i.e., when the buffer is less than a certain threshold, the client only requests the lowest bitrate). The client requests chunks according to the bitrate recommendations sent by the smart edge. After finishing downloading a chunk, it sends a finish signal to the smart edge and waits for bitrate recommendations for the next chunk. As the transmission delays for bitrate recommendations are much smaller than a chunk's playback duration (generally more than 2 seconds), the extra delay caused by sending bitrate recommendation is ignored.[1]

[1]*ping* was used to measure the round-trip time (RTT) from a PC and a mobile phone to the servers on our campus and data center in our city, respectively. The results show that the RTTs to the server range from 2 to 10 milliseconds, and RTTs to the server in the data center range from 3 to 24 milliseconds.

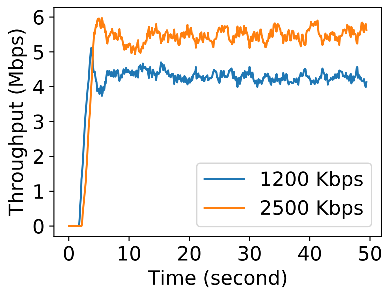


Fig. 4: Throughput of clients requesting different bitrates.

2) Smart Edge: The measurement function and the ABR agent at the smart edge are two main functions that realize joint QoE optimization. On the one hand, the measurement function monitors the network condition from edge network to DS in real-time. On the other hand, the measurement function records the observation of client states by sniffing online client's request packets containing single client information. Then the ABR agent learns the states through this information by NN and makes bitrate decisions for online clients aiming to achieve joint QoE optimization.

The smart edge collects the observation of client states at the beginning of chunk downloading and evaluates the effect caused by the bitrate recommendation after finishing downloading a single chunk. It is important to note that, due to the TCP slow-start-restart enabled [41] and ON/OFF period while DASH chunk downloading, the throughput is related to the bitrate of the downloading chunk. Fig.4 shows the measured throughput of two clients requesting the same video but have different bitrates. Clients requesting a higher bitrate content generally get a higher bandwidth share. Therefore, the bitrate recommendation method can realize differentiated bandwidth resource allocation among clients.

It is worth mentioning that the smart edge pays attention to the joint QoE optimization requesting the same HTTP DASH Server. As long as the bottleneck bandwidth competition occurs between the edge node and the source node, the smart edge can realize joint QoE optimization among clients in the same cluster. We can regard all the links from the smart edge to the source server as a link. The bandwidth competition occurs on the most congested link. The congestion level of the congested link determines the probed delay by the client, which is used as the global information by the smart edge. Therefore, when the bottleneck appears between the smart edge and the HTTP DASH Server, the smart edge can adaptively adjust the bitrate recommendation for clients in its corresponding edge cluster.

3) HTTP DASH Server: The HTTP DASH Server in Fig.3 illustrates the content storage and provision is located in the cloud of the CP. Video chunks encoded at multiple bitrates are stored in this server. Chunks with diverse bitrates are aligned to support seamless quality transitions. The servermay receive thousands of concurrent requests from all over the world.

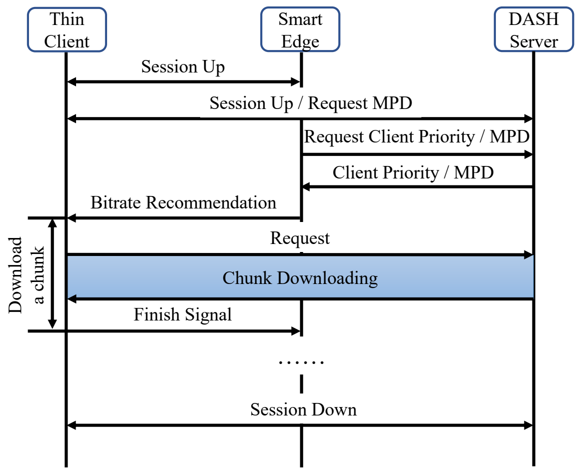


Fig. 5: Flow diagram of a video download session.

B. Adaptive Video Streaming Process

Fig.5 illustrates the flow diagram of the video download session. The client communicates with edge and DASH Server with end-to-end encrypted traffic, respectively. When a client starts to request a video from the server, a ”session up” message that includes the URL of the video and the client information (e.g., device resolution, operating system, and client IP) is sent to the edge and the server to establish the communicating session, respectively. Meanwhile, the server replies with the Media Presentation Description (MPD) file to the client. The MPD file describes chunk information (i.e., timing, URL, media characteristics like video resolutions and bitrates) as the reference of video chunks for clients. We also add the chunk-level quality and size to MPD file to enable quality-size-aware ABR decision-making. On the other hand, if the smart edge receives the ”session up” message form the client and does not find the client's priority or the video's MPD file, it sends an explicit request to the DS.

The client starts to download the video chunk when it received the MPD file. If for some reason (e.g., edge server failure) the client does not receive bitrate recommendations from the smart edge, it uses the buffer-based ABR algorithm [6] run at the client-side to make the bitrate-adaptive decision. Moreover, Flex-Steward excludes clients whose last-mile is the bottleneck based on the client's download speed. If a client downloads a large chunk with low throughput than other clients, it implies that the bottleneck of this client may appear at the last mile. At this time, this client should change Flex-steward to the client-side ABR.

The client sends the request to DS to fetch each chunk. Then DS sends the chunk to the client after receiving the request (”chunk downloading” in Fig.5). When receiving the whole chunk, the client sends a ”finish signal” to the smart edge then the edge sends ”bitrate recommendation” to the client. According to our statistics, 98% the time between the client sending ”finish signal” and receiving ”bitrate recommendation” is under 25ms, which is much smaller than the chunk duration (e.g., 2s). The ”finish signal” contains the client's playing status (i.e., play or resume) and current buffer. The

TABLE I: Notations Table

Notation	Description
i, v	The client id, the video id
$o_i^{(t)}$	The observed state of client i at time t
$a_i^{(t)}$	The determined action of client i at time t
$r_i^{(t)}$	The reward of the action at time t by client i
γ	A factor discounting future rewards, $\gamma \in (0, 1]$
$n_i^{(t)}$	The requested chunk id of client i at time t
N_v	the total chunk number of video v
g, G	The id of the RL thread. The total number parallel threads.
$Q_i^{(t)}$	The QoE of client i at time t
α, β, ζ, λ	The parameters to describe the aggressiveness of each QoE metric
$\tau_i^{(t)}$	The bitrate of the chunk downloaded by client i at time t
$q(\cdot)$	The function that maps the bitrate to perceived quality
$T_i^{(t)}$	The rebuffering time caused by downloading chunk $n_i^{(t)}$
$QL_i^{(t)}$	The QoE $loss$ of client i at time t
q_i^{exp}	The expected quality of the client i
w_i	A weight factor related to the priority of client i
$U_i^{(t)}$	The utility value of client i at time t
$\vec{d^{(t)}}$	The vector that contains delay measurements that indicates network congestion
$\overline{U^{(t)}}$	The utility mean of all online clients
$b_i^{(t)}$	The buffer occupancy of client i at time t
$\vec{p_i}$	The priority list of client i
$\vec{q_n^{(t)}}$	The quality list of all alternative bitrates of the next chunk
$\vec{s_n^{(t)}}$	The size list of all alternative bitrates of the next chunk
K	The number of alternative bitrate at each decision

client stops requesting a new chunk if its buffer occupancy exceeds a threshold. At this time, the client does not have data exchange with edge server. When the client's buffer falls below the max buffer threshold, the client sends a new request to the smart edge for "Bitrate Recommendation" and then it requests the new chunk to the DS. The time of requesting data from edge depends on when the client finishes downloading a chunk from the source server. Note that the request to the smart edge contains the client's playing and buffer status to update the client information recorded by the smart edge. In the end, the client sends the "session down" message to the smart edge, and the server tears down the session. This results in the smart edge ABR agent stop making bitrate-adaptive decisions for the client.

IV. Learning-based Joint QoE Optimization

A. ABR Decision and Model Training Process

As mentioned, Flex-Steward employs a neural network (NN) based reinforcement learning (RL) method for bitrate recommendation to achieve joint QoE optimization. The RL approach attempts to learn an ABR policy from observations through interaction between the ABR agent and the environment. At time t, the agent which makes recommendations to client i makes also observations $o_i^{(t)}$ and chooses an action $a_i^{(t)}$ according to $o_i^{(t)}$. After applying the action, the observation of the environment transitions to $o_i^{(t+1)}$ and the agent receives a reward $r_i^{(t)}$. The goal of learning is to maximize the expected cumulative discounted reward: E[$\Sigma_{t=0}^{\infty}\gamma^t r_i^{(t)}$], where $\gamma \in$ (0,1] is a factor discounting future rewards. Fig.6 summarizes how the NN-based RL makes the bitrate recommendations for client i and updates its NN online.

1) ABR Decision: As is shown in Fig.6, the RL thread makes bitrate recommendations to client i. The decision process is as follows: (1) When the smart edge perceives that client i is going to request chunk n, the "Measurement" function in the smart edge sends its measurement to the waiting queue of the RL thread g. (2) When it is the turn to make a bitrate decision for client i, the queue sends the observation vector $o_i^{(t)}$ to the NN and the memory function (i.e., (o, a, r) memory) of the $(observation, action, reward)$ tuples. The actor-network infers the bitrate recommendation $a_i^{(t)}$ based on $o_i^{(t)}$ [48]. Note that 98% of NN decision is under 10ms. (3) The RL thread g sends $a_i^{(t)}$ to its (o, a, r) memory and client i. (4) The client i fetches the chunk from the source server. (5) The smart edge updates the client measurement with respect to the information contained in the new request. (6) The "Measurement" function sends the reward $r_i^{(t)}$ of action $a_i^{(t)}$ to the (o, a, r) memory of the RL thread g.

2) Model Training Process: To speed up training process and be able to make bitrate recommendations to multiple clients, multiple RL threads are spawn in parallel. The ABR agent contains a central thread and G parallel RL threads. At each decision time for a thread, the thread takes personal observation of the client as input and makes the bitrate recommendation. Note that the client should be in its corresponding client cluster. Thus, each thread is configured to experience a different set of input observations. However, the threads send $(observation, action, reward)$ tuples to the central thread, which aggregates them to generate and train a single RL model. For every fixed number of sequences of tuples of a client, the central agent computes a gradient and performs a gradient decent step. The central thread then updates the actor and critic network and then pushes out the new model to the RL thread which sent this sequence of tuples. This mechanism makes Flex-Steward scalable to cover large number of clients in the edge network.

B. Experience-Driven Network Model

In this section, we formulate the experience-driven model as a RL model. It is represented as a discrete time and action, continuous observation control model, by defining observation $o \in \mathcal{O}$ and action $a \in \mathcal{A}$ spaces and a reward function r. Service quality Q is assessed using the quality-aware QoE model for ABR proposed by Comyco [33], as follows.

1) Definition: QoE $Q_i^{(t)}$ of client i at time t is defined as:

$$Q_i^{(t)} = \begin{cases} \alpha q(\tau_i^{(t)}) - \beta T_i^{(t)} & n_i^{(t)} = 1 \\ \alpha q(\tau_i^{(t)}) + \zeta |q(\tau_i^{(t)}) - q(\tau_i^{(last)})|_+ - \\ \lambda |q(\tau_i^{(t)}) - q(\tau_i^{(last)})|_- - \beta T_i^{(t)} & n_i^{(t)} > 1 \end{cases} \tag{1}$$

where $n_i^{(t)} \in [1, N_v]$ and $\tau_i^{(t)}$ are the sequence number and bitrate of the chunk downloaded by client i at time t and $q(\tau_i^{(t)})$ maps the bitrate to the quality perceived by the client. $T_i^{(t)}$ represents rebuffering time caused by downloading the

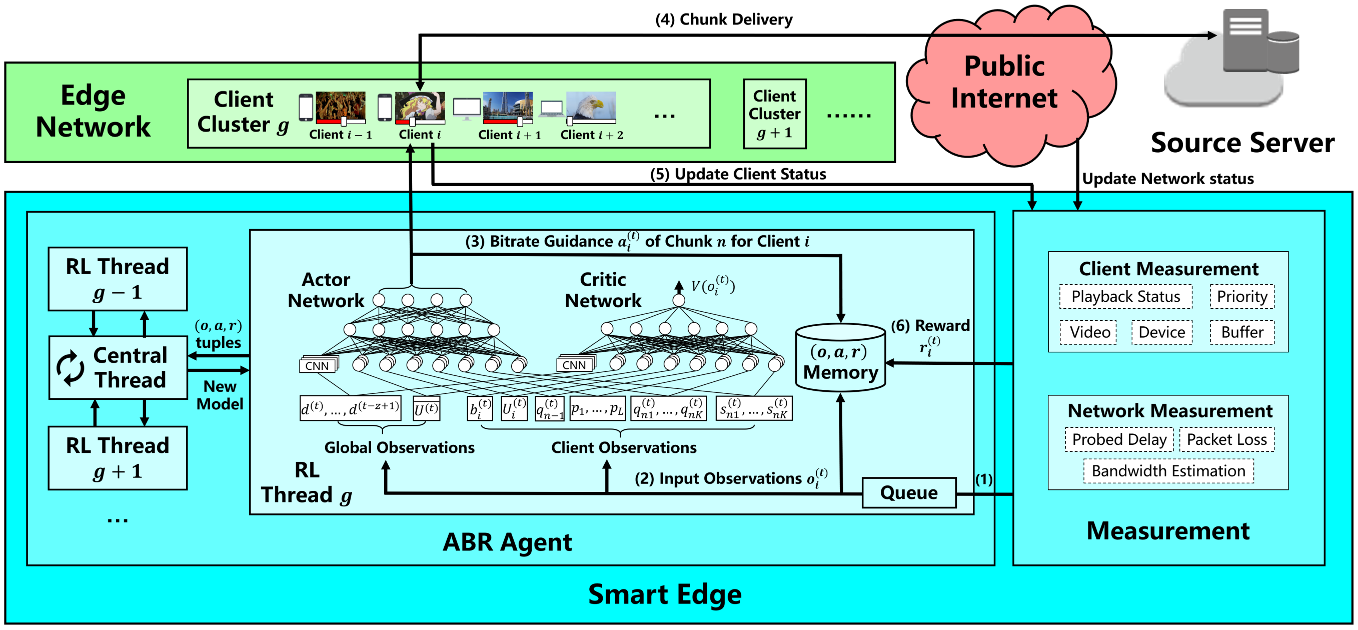


Fig. 6: The diagram of ABR decision and model training process

chunk $n_i^{(t)}$. $|q(\tau_i^{(t)}) - q(\tau_i^{(last)})|_+$ indicates a positive chunk quality switch which means that the chunk switches from a lower bitrate to a higher bitrate, while $|q(\tau_i^{(t)}) - q(\tau_i^{(last)})|_-$ is a negative chunk quality switch. Note that α, β, ζ and λ are parameters to describe the aggressiveness of each metric.

When a client downloads a chunk that reaches its expected quality and does not suffer from rebuffering or quality switch, we state that the client has a satisfactory QoE level. We also define the difference between the client's actual QoE and a satisfactory client QoE level as *QoE loss* $QL_i^{(t)}$ and use it to represent the playback status of client i. The *QoE loss* $QL_i^{(t)}$ of client i at time t is defined as follows:

$$QL_i^{(t)} = \alpha q_i^{exp} - Q_i^{(t)} \tag{2}$$

where q_i^{exp} is the expected quality of the client i. When the client's QoE declines from the satisfactory value, the probability of the client not continue to watch the video increases rapidly [3]. Therefore, in comparison with the QoE value alone, *QoE loss* represents more accurately the probability that a client stops the video playout. Besides, to achieve differentiated service for clients with various priorities, we use the weighted QoE loss as the utility function $U_i^{(t)}$ to reflect the impact on the potential revenue of the CPs when the client i suffers from QoE loss. This utility is defined as follows:

$$U_i^{(t)} = -w_i * QL_i^{(t)} \tag{3}$$

where w_i is a weight factor related to the priority of client i.

2) Inputs: After downloading chunk $n-1$ by client i at time t, the ABR agent located at the smart edge collects network and client state information and transforms them into an observation vector. This vector contains eight dimensions of observation $o_i^{(t)}$=$\{\vec{d^{(t)}}, \overline{U^{(t)}}, b_i^{(t)}, U_i^{(t)}, q_{(n-1)}^{(t)}, \vec{p_i}, \vec{q_n^{(t)}}, \vec{s_n^{(t)}}\}$ as input for the RL model. The observation vector is composed of both global observation and individual observation

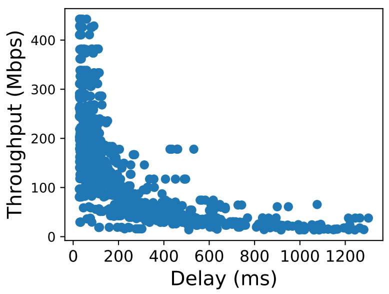


Fig. 7: Bottleneck delay vs throughput.

components of client i, providing relevant information to the adaptive bitrate recommendation process.

Global observation: The global observation includes probed bottleneck delay and mean *QoE loss* for the clients sharing the bottleneck. Multiple clients are selected to send probe packets to detect network delays characterizing the network congestion. As shown in Fig. 7, the transmission delay of the bottleneck bandwidth is approximately inversely proportional to the available bandwidth. We use past $z = 6$ delay measurements $\vec{d^{(t)}} = \{d^{(t)}, ..., d^{(t-z+1)}\}$ to assess the bottleneck congestion level. Pensieve [49] and Comyco [33] suggest that the number of past measurements is set to 8. We try to set k = 4, 6, 8 respectively and find that set k = 6 and k = 8 has equal performance while the reward decreases when k = 4. Thus we set k = 6 in our experiment.

Note that the content providers should select clients with low network fluctuations and high bandwidth as clients who send probe packets. The probe packets are set every one second. In a real-world deployment, multiple clients are needed to probe global congestion. Then we use *box-plot* method to remove outliers and determine the current bottleneck congestion. We use the utility mean of all online clients $\overline{U^{(t)}}$ to indicate the current playback status of all clients.

Individual observation: The individual observation indicates the detailed state of client i when it requests the next chunk. It includes the current buffer occupancy $b_i^{(t)}$; the utility $U_i^{(t)}$ following the download of the last chunk; the quality of the last chunk $q_{(n-1)}^{(t)}$; the priority list $\vec{p_i} = \{p_1, ..., p_l, ..., p_L\}$ where $p_l = 1$ if the priority of client i is l otherwise $p_l = 0$; the quality list $\vec{q_n^{(t)}} = \{q_{n1}^{(t)}, ..., q_{nk}^{(t)}, ..., q_{nK}^{(t)}\}$ and size list $\vec{s_n^{(t)}} = \{s_{n1}^{(t)}, ..., s_{nk}^{(t)}, ..., s_{nK}^{(t)}\}$ of all alternative bitrates of the next chunk. $b_i^{(t)}$, $U_i^{(t)}$ and $q_{(n-1)}^{(t)}$ reflect the client's real-time playback status and $\vec{p_i}$ indicates how important the client i is to the CPs. Note that we regard the priority as a categorical value rather than a scaler value. When we input the priority value as a categorical value, we find that the neural network converges faster and gets a higher reward. So we choose to input the priority as a categorical variable into the neural network. Meanwhile, motivated by Fig. 1a and Fig. 1b, we employ $\vec{q_n^{(t)}}$ and $\vec{s_n^{(t)}}$ as input to the NN model to indicate the potential impact that each action may have on the environment and client's QoE.

3) Action: The ABR agent provides bitrate adaptive recommendations to client i according to the measured observations. The action space includes alternative bitrates for the next chunk. We have a K dimensional action vector for K alternative bitrates.

4) Reward: The primary objective of this system is *maximizing utility fairness* while satisfying the QoE demand. We craft the reward signal to guide the agent towards good solutions for our objective. Specifically, we set the reward $r_i^{(t)}$ at each time t as:

$$r_i^{(t)} = U_i^{(t)} - |\overline{U^{(t)}} - U_i^{(t)}| \tag{4}$$

The reward value includes two metrics: individual utility and utility fairness. The first term indicates the preference of high utility (i.e., low weighted QoE loss). The second term implies fairness among multiple clients' utility. The client i experiences a punishment when its utility is different from the average utility value among online clients. When all online clients have a fair QoE distribution, the second term becomes zero. Note that we normalize $r_i^{(t)}$ based on historical data in the training phase to make the NN easier to converge.

C. The Case of ABR Decision in Realizing Joint QoE Optimization

To make the decision principle of Flex-Steward clearer, we use a case to show the logic of bitrate recommendation in this section. As described in section IV-B2, all threads have the same observation of bottleneck bandwidth as the global observations which indicate the amount of available bottleneck bandwidth and the status of managed clients. Then the client observations help the ABR agent handle the resource allocations among clients. Note that the competition is not among threads but clients. The multi-thread mechanism is to avoid creating a neural network for each client and causing insufficient memory at smart edge. Fig.8 shows how Flex-Steward make the bitrate decisions under perceived total available bandwidth to handle the competition among clients.

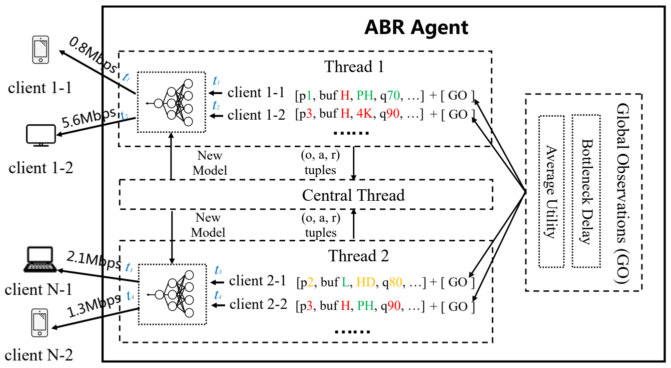


Fig. 8: Flex-steward decision logic

At time t, the ABR agent perceives the total available bandwidth and average utility of online clients, then record them in the Global Observations (GO) module. Assume that the ABR agent estimates that the total available bottleneck bandwidth is about 10Mbps according to the bottleneck delay (Motivated by the results in Fig.7).

On the other hand, there are four online clients connected to the ABR agent. At time t1, thread 1 in ABR Agent receives the bitrate request from client 1-1. Then, thread 1 feeds the client observations and global observations to the neural network (NN) model. Although the buffer of client 1-1 is high, its device resolution and priority are relatively low, so that the model recommends that the client 1-1 requests the chunk at 0.8Mbps. At time t2, thread 1 receives the bitrate request from the client 1-2. Because client 1-2 has high priority and high-resolution devices, the model recommends that client 1-2 request the chunk at 5.6Mbps by NN according to the the information from GO and client observations. Thread 2 also uses the same principle to make bitrate decisions for client 3 and 4 to realize joint QoE optimization by NN based on global and client observations.

V. Flex-Steward Implementation

This section details the implementation of Flex-Steward's RL-based agent and describes the prototype system employed in testing.

A. Reinforcement Learning Agent

TensorFlow [50] is used to implement the RL architecture and train the ABR agent through asynchronous advantage actor-critic (A3C) method [51]. The network architecture, which includes actor and critic networks is shown in Fig.6. In the actor network, we use a 2-hidden-layer fully connected neural network with 400 and 150 hidden nodes in the first and second layers, respectively. Results from the second layer are applied to a softmax function to output the bitrate recommendation. The critic network also uses a 2-hidden-layer fully connected neural network with 250 and 60 hidden nodes in the first and second layers, respectively. However, the final output is a linear neuron without an activation function. During training, we use a discount factor $\gamma = 0.9$, which implies that current actions can be influenced by ten future steps. The central agent updates the actor and critic network after it receives 10 $(observation, action, reward)$ tuples from a client. The learning rate for actor and critic networks is configured to be 2×10^{-4} and 10^{-4}, respectively. Note that we have tried several combinations of NN hyper-parameters, including layer number, node number, learning rate, discount factor, and the number of forwarding steps, and find that the agent performs well for a wide range of hyper-parameter.

B. Video Delivery System

We use one, one, and two x86 servers (running Ubuntu 18.04 system) with two *Intel Xeon E5-2600* CPUs to emulate the source server, edge server, and clients, respectively. The source server is based on Nginx [52], a lightweight and highly stable HTTP server. All the DASH standard videos are saved at the source server. We use the Linux Traffic Control tool[2] to control the sending rate of the source server in order to determine the impact of background traffic on the bottleneck. The smart edge is based on Nginx, uWSGI [53], and Django [54], a common deployment in production environments. Redis [55], a popular in-memory data structure store, is employed to maintain the network and client status. We run multiple *dash.js* players in Google Chrome browsers (v83) on servers that emulate the clients' behaviors. The request packets of the clients are modified to include additional information about the client state, such as buffer occupancy and bitrate of the last requested chunk.

VI. Flex-Steward Evaluation

A. Video and Network Datasets

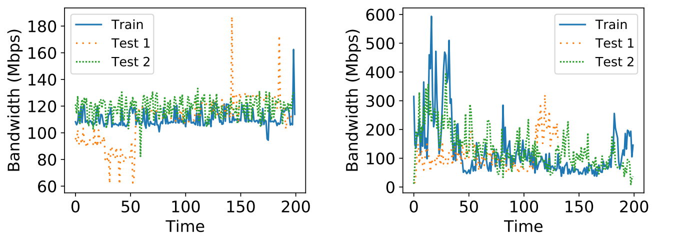
1) Video Dataset: We employ 39 4K videos, from 5 to 10 minutes long which cover content ranging between static and highly dynamic. We use FFmpeg [56] to encode videos into H.264 and MP4Box [57] to convert them to MPEG-DASH chunks of 2-seconds duration. Each video is encoded into 10 discrete bitrates: {334, 595, 791, 1200, 1800, 2500, 4200, 8000, 12000, 24000} Kbps. We use the Video Multi-method Assessment Fusion (VMAF) [40], a state-of-the-art indicator to reflect clients' perceptual quality. The value of VMAF ranges from 0 to 100. Moreover, we utilize VMAF-PHONE, VMAF-HD and VMAF-4K models to evaluate the VMAF value for diverse devices: PHONE, HDTV, and 4KTV, respectively. The resolutions of the reference videos for the three models are 1280×720, 1920×1080 and 4096×2160, respectively. We divide the video dataset into a *training* dataset and a *testing* dataset, which contain 29 and 10 videos, respectively. Both datasets contain multiple types of videos with identical distribution.

2) Network Traces: To simulate dynamically changing background traffic of the bottleneck, we employ the broadband network traces provided by FCC [58] and wireless network traces from [59]. The bandwidth of broadband traces changes smoothly while the bandwidth of the wireless network has a large variance. We have conducted experiments under the above two types of bandwidth scenarios to demonstrate the versatility of Flex-Steward. The dataset contains traces of average throughput with a five-second granularity. We multiply the throughput by 60 to simulate the bandwidth of bottleneck. We divide the network traces into *training* traces and *testing* traces with a ratio of 8:2. Fig. 9 shows a part of the training and test traces of wired and wireless bottleneck in our evaluation.

[2]https://linux.die.net/man/8/tc

TABLE II: Ratio of [device, priority] configurations among clients

Priority \ Device	Configuration 1			Configuration 2		
	PH	HD	4K	PH	HD	4K
1	0.25	0.15	0.1	0.117	0.117	0.117
2	0.15	0.08	0.067	0.117	0.117	0.117
3	0.1	0.067	0.033	0.1	0.1	0.1



(a) wired network traces (b) wireless network traces

Fig. 9: Part of training and testing network traces

B. Experimental Setup

The number of DASH clients ranges from 0 to 100, with the ratio between the number of clients using PHONE, HDTV, and 4KTV is 5:3:2. This ratio was set following a recent Cisco's report [1]. In order to achieve differentiated services, three levels of client priorities are selected: levels 1, 2, and 3, where the highest level being associated with the highest priority. The ratio of clients with priority levels 1, 2, and 3 was also set to 5:3:2. Therefore, the ratio of clients with each [device, priority] combination follows configuration 1 in Table II, where PH, HD and 4K are abbreviations for PHONE, HDTV and 4KTV, respectively. Additionally, to compute the utility value, we set the weight factors w_i in the utility function for priority levels 1, 2, and 3 to 1, 1.2, and 1.5, respectively. Note that the weight can be modified according to the service strategy of the CPs. Finally, according to our test of the RTT between the client and server which is described in Sec.III.A, we set the RTT between the edge server and DS to 24 milliseconds.

C. Methodology

1) Comparison with Baseline Methods: Flex-Steward is compared with the following ABR algorithms which target QoE optimization:

- Client-side ABR already integrated in dash.js: These methods do not consider to optimizing fairness. We use them as competitors to reflect the problems of the current integrated methods in dash.js in achieving QoE fairness.
  - Rate-Based (RB): chooses the highest available bitrate below the predicted throughput, which is predicted using the harmonic mean of the throughout experienced during the last five chunk downloads.

    - Buffer-Based (BB) [7]: mimics the buffer-based algorithm in [7] which uses a reservoir of 3 seconds and a cushion of 11 seconds. i.e. it selects bitrates with the goal of keeping the buffer occupancy above 3 seconds, and automatically chooses the highest available bitrate if the buffer occupancy exceeds 14 seconds.
    - Bola [6]: uses Lyapunov optimization to select bitrates solely considering buffer occupancy observations. Bola is an ABR algorithm widely used in the industry, such as Bilibili[3]. This solution makes clients greedily occupy network bandwidth to obtain high QoE but ignore the QoE fairness among clients. We hope to compare with Bola to show that Flex-Steward can improve the fairness of QoE among clients without reducing the QoE of clients.
- Client-side ABR only considers QoE fairness: We compare Flex-steward with Festive to show the performance on QoE fairness when we adapt the quality-aware model.
    - Festive (Fes) [41]: uses the stateful and delayed bitrate update strategy to decide bitrate for the next chunk. In addition, the next chunk download time is randomized considering buffer occupancy. This strategy aims to improve stability and fairness for clients sharing the common bottleneck bandwidth.
- Methods in achieving joint QoE optimization:
    - Gta [44]: A client-side game-theoretic approach that comprehensively considers the device screen resolution, priority, and requesting video content type to make the bitrate decision to realizing joint QoE optimization.
    - Fineas (Fin) [45]: A collaborative strategy between the client and the server. The server monitors the available bandwidth of the bottleneck and calculates the fairness signal according to clients' priority. The client makes the bitrate decision with reference to the fairness signal and the perceived bandwidth. The fairness signal can be calculated as $\frac{w_i}{\sum_i^M w_i} * \sum_i^M bw_i$, where M is the number of online clients and bw_i is the measured bandwidth by client i.

We choose the above methods because these methods do not shape the traffic in the network and increase the instability of network performance. Most server-side and network-assisted solutions mentioned in our related work section use the equipment in the network to allocate network resources. For instance, the Server and Network-assisted DASH (SAND) mostly use Software-defined Network (SDN) switches to allocate bandwidth to each cluster.

2) Training RL model for Flex-Steward: We pre-trained a RL model for Flex-Steward based on our system prototype before the deployment. During the training phase, the sending rate of the source server changes according to the randomly picked network traces in the training set, and the client picks up the video randomly in the training video dataset. We test the performance of the RL model every 20 minutes based on the test configuration. During the testing phase, the sending rate of the source server changes according to the "Test 1" trace shown in Fig.9a, and the client picks up the video from the testing video dataset. Each testing epoch lasts for 15 minutes, and we record the average reward during the testing to illustrate the performance of the RL model. The training lasts for 12 hours. In the evaluation phase, we deploy the RL model with the highest average reward in Flex-Steward for comparison with other methods.

3) Evaluation Metrics: In order to compare the performance of Flex-Steward with other methods, we evaluate the following metrics in each run.

Average QoE. The average QoE is the mean of QoE values when a client downloads a video. We refer to the QoE definition in Eq.(1) and set α = 0.8469, β = 28.7959, γ = 0.2979, δ = 1.061 according to Comyco [33]. In [33] [61], the author proves that compared with the previous researches on ABR [8], [49], Comyco's QoE is closer to the clients' real QoE, and the results obtained referring to Comyco's QoE model can be fed back to the ABR agent in real time as a reward for RL. Thus, we use Comyco's QoE model to evaluate the performance of ABR decision. Besides, we use ITU-T P.1203 [60], an established QoE model that trained by the subjective test databases to be closer to the clients' real QoE. To measure the impact of video quality on QoE under different resolution devices, we use P.1203 with input scores coming from P.1204-type [62] quality models. ITU-T P.1204.3 is a short-term video quality prediction model that uses full bitstream data to estimate video quality scores on a segment level. The quality score of the P.1204.3 model claims that its predicted score is closer to the Mean Opinion Score (MOS) of humans than VMAF.

Detailed QoE Metrics. We record the average quality, average positive/negative quality switch, and average ratio of the rebuffering time during the evaluation for all clients when all the considered methods are employed in turn.

QoE Fairness Metrics. According to Fig.1a and section II-B, the bottleneck competition may cause QoE unfairness among clients with the same priority. Flex-Steward considers chunk quality and size as inputs to reduce the QoE unfairness among clients with the same priority. To evaluate the quality-and-size-aware ABR of Flex-Steward on improving QoE fairness among clients with the same priority, we calculate the QoE of all online clients every T_{test} = 2 seconds. Then, for clients with the same priority, we figure out both the Jain's fairness [63] and 5th percentile minimum value among their QoE to illustrate the QoE fairness.

Utility Unfairness. We compute the standard deviation of utility defined in Eq. (3) among I online clients at time t by defining the following unfairness index:

$$Unfairness^{(t)} = \sqrt{\frac{1}{I}\sum_{i=1}^{I}(U_i^{(t)} - \overline{U^{(t)}})^2} \tag{5}$$

where $U_i^{(t)}$ represents the utility of client i at time t. $\overline{U^{(t)}}$ is the mean of utility values of all online clients at time t, which is defined as $\overline{U^{(t)}} = \frac{1}{I}\sum_{i=1}^{I} U_i^{(t)}$. Since the VMAF value of

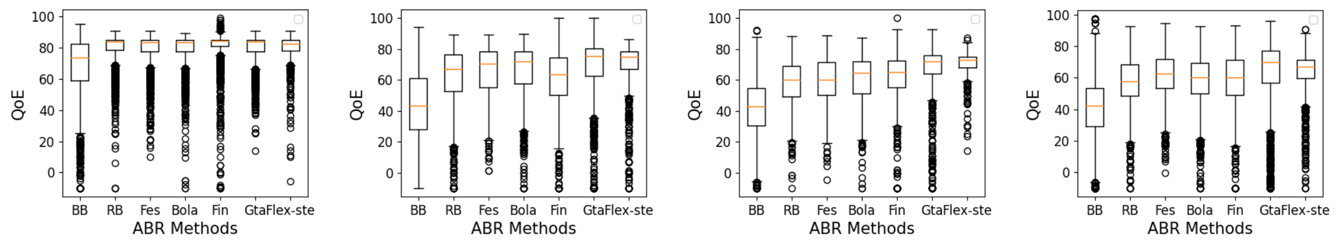
[3]https://www.bilibili.com/

TABLE III: Average QoE under Comyco [33] QoE Model

Conf / ABR	Wired Network Traces									Wireless Network Traces								
	PH1	PH2	PH3	HD1	HD2	HD3	4K1	4K2	4K3	PH1	PH2	PH3	HD1	HD2	HD3	4K1	4K2	4K3
RB	**79.5**	79.4	79.7	61.2	62.8	61.8	57.2	58.7	57.7	**78.9**	77.6	76.3	60.9	**63.9**	62.9	58.2	59.4	58.1
BB	67.7	67.8	67.9	43.0	42.9	42.0	40.2	40.2	40.9	71.6	70.5	72.6	44.3	47.9	50.0	42.3	41.0	46.6
Bola	79.3	79.3	78.9	63.8	61.6	65.7	58.2	60.7	60.1	78.5	**78.0**	**79.0**	63.2	59.7	64.0	58.7	55.8	56.9
Festive	79.3	**80.7**	79.2	62.0	65.9	65.5	61.2	61.8	59.0	77.6	77.1	77.9	59.2	55.3	57.0	56.6	54.5	61.1
Gta	70.8	73.0	79.5	**65.9**	64.6	68.2	63.3	61.0	65.8	78.2	77.8	78.0	**63.7**	61.7	62.0	58.1	55.2	59.1
Fineas	77.3	80.2	**80.5**	60.9	59.6	64.8	58.1	62.8	61.4	77.4	76.4	78.2	56.7	59.6	60.5	57.7	57.1	58.6
Flex-Steward	75.2	78.5	80.3	64.5	**69.6**	**72.6**	**64.8**	**65.7**	**70.2**	75.1	75.7	75.4	60.3	63.7	**66.5**	**63.4**	**63.5**	**68.5**

TABLE IV: Average QoE under P.1203.3 [60] QoE Model

Conf / ABR	Wired Network Traces									Wireless Network Traces								
	PH1	PH2	PH3	HD1	HD2	HD3	4K1	4K2	4K3	PH1	PH2	PH3	HD1	HD2	HD3	4K1	4K2	4K3
RB	3.99	3.99	4.10	3.62	3.64	3.65	3.30	3.37	3.28	3.72	3.90	3.72	3.52	3.59	3.64	3.33	3.13	3.28
BB	3.39	3.40	3.40	3.04	3.04	3.03	2.68	2.68	2.69	3.89	3.95	**4.07**	3.76	3.73	3.60	3.20	3.23	3.29
Bola	**4.08**	**4.07**	4.08	**3.84**	3.71	3.98	3.46	3.67	3.57	3.95	3.84	3.89	3.73	3.61	3.60	3.28	3.13	3.19
Festive	3.99	4.01	4.10	3.76	3.86	3.87	3.65	3.27	3.32	3.80	3.88	4.05	3.50	3.45	3.65	3.38	3.19	3.42
Gta	3.54	3.75	4.09	4.01	**3.96**	3.99	**3.81**	3.77	3.89	**4.03**	**4.03**	3.91	**3.89**	3.87	3.67	3.52	3.24	3.59
Fineas	3.99	4.04	**4.20**	3.78	3.86	3.63	3.61	3.75	3.71	3.94	3.96	3.88	3.49	3.56	3.82	3.42	**3.62**	3.68
Flex-Steward	3.75	4.00	4.08	3.67	**3.96**	**4.03**	3.69	**3.80**	**3.99**	3.75	3.71	3.81	3.75	**3.90**	**3.91**	**3.58**	3.59	**3.88**



(a) [PHONE, Priority 3] (b) [HDTV, Priority 3] (c) [4KTV, Priority 3] (d) [4KTV, Priority 1]

Fig. 10: QoE Boxplot of clients with different characteristics among considered methods

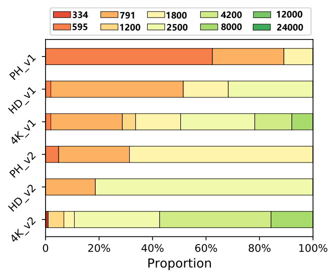


Fig. 11: Distribution of Requesting Bitrate based on Flex-Steward.

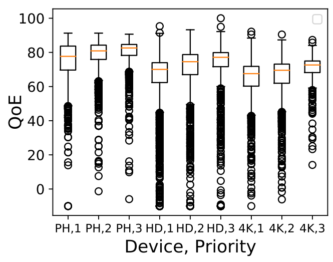


Fig. 12: QoE of Clients with Different Characteristics bassed on Flex-Steward

only a few chunks can reach 100, we set q_i^{exp} in Eq. (2) to 95, which is enough to provide clients with high QoE.

We compute the unfairness index in every T_{test} = 2 seconds. The unfairness metric measures the performance of the considered approaches in enforcing the joint QoE optimization according to the control policy. We apply a fair utility allocation. Note that the lower the unfairness index is, the higher is the utility fairness.

Maximum Weight QoE Loss. The client is more likely to quit when suffering from a high weighted QoE loss. Thus, we measure the maximum weight QoE loss among online clients to indicate the possibility of a client quit watching. The weight QoE loss $WQL_i^{(t)}$ of client i at time t is defined as:

$$WQL_i^{(t)} = \begin{cases} w_i * QL_i^{(t)} & w_i * QL_i^{(t)} <= \alpha q_i^{exp} \\ \alpha q_i^{exp} & w_i * QL_i^{(t)} > \alpha q_i^{exp} \end{cases} \tag{6}$$

Note that when the QoE of a client is lower than a certain value (i.e., αq_i^{exp}), the client definitely quits watching the video. So we define $WQL_i^{(t)}$ as a segmented function.

D. Testing the Performance on Joint QoE Optimization

1) Overall Performance: In order to evaluate Flex-Steward, we compare its performance with that when the other methods are used in turn. To identify the performance changes caused by variations in bottleneck bandwidth, we assess the performance with the bottleneck bandwidth and ”Test 1” traces shown in Fig.9a and Fig.9b, respectively. Table III and Table IV present the average QoE that each method achieves with two types of bottleneck bandwidth under the Comyco [33] and P.1203 [60] QoE model, respectively. To a certain extent, both the results show that Flex-Steward improves QoE fairness among clients with various devices and realizes different services for clients with different priorities.

Table V shows the detailed QoE and QoE fairness among clients with the same client characteristics (i.e., [device, priority] configuration). Fig.10 illustrates the QoE distribution for clients with different characteristics for the considered

TABLE V: Detailed QoE and QoE fairness metrics comparison

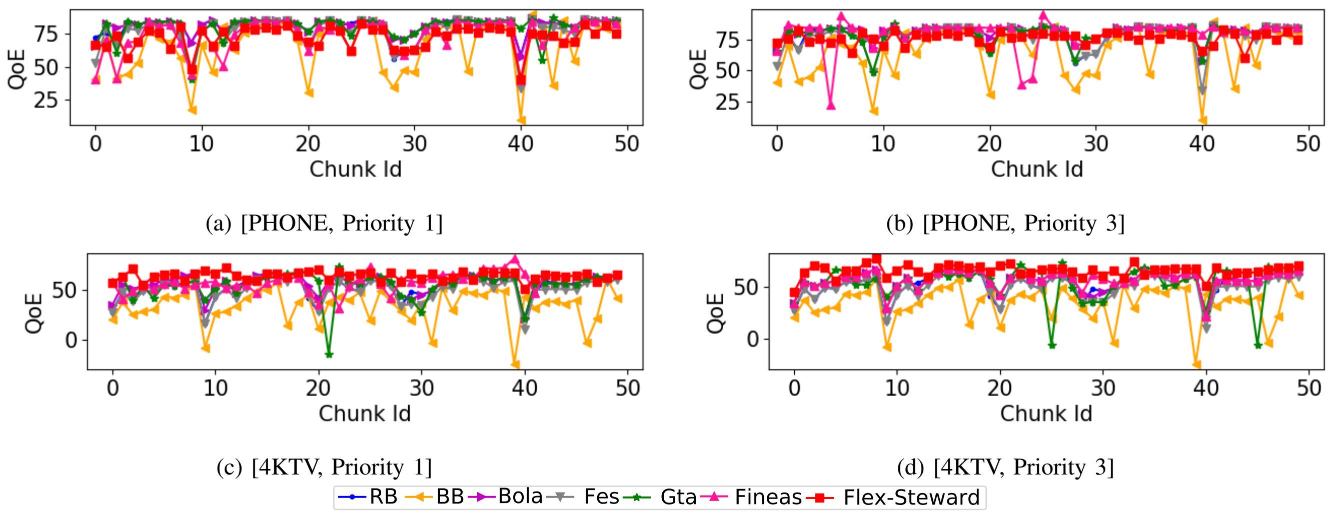
Conf / ABR	Wired Network Traces									Wireless Network Traces								
	PH1	PH2	PH3	HD1	HD2	HD3	4K1	4K2	4K3	PH1	PH2	PH3	HD1	HD2	HD3	4K1	4K2	4K3
(a) Average Quality																		
RB	**95.5**	95.4	95.6	75.4	77.6	76.2	70.3	72.3	71.4	**95.1**	93.9	92.3	75.3	**79.3**	78.1	72.1	73.9	71.8
BB	83.8	83.8	84.0	55.7	55.6	54.8	51.9	51.9	52.5	87.7	86.5	88.5	56.9	61.3	63.9	53.6	52.6	59.2
Bola	95.1	95.3	94.8	78.3	75.6	80.2	71.2	73.9	73.1	94.4	**94.0**	**94.8**	77.6	73.6	78.7	72.2	69.0	69.8
Festive	94.9	**96.3**	94.7	75.8	80.3	79.5	74.4	74.8	72.0	93.6	92.5	93.4	72.7	68.5	70.0	69.1	66.9	74.4
Gta	86.2	88.5	95.1	**80.4**	78.8	83.2	78.1	77.1	80.9	93.9	93.4	93.8	**78.2**	76.3	76.7	71.7	68.5	72.8
Fineas	93.2	96.2	**96.6**	75.1	73.1	79.7	71.5	77.0	75.2	93.5	92.6	94.7	70.6	73.7	75.5	71.6	70.2	71.9
Flex-Steward	90.6	94.3	96.0	79.1	**84.6**	**87.9**	**78.4**	**79.4**	**84.7**	90.4	90.9	90.4	73.8	77.7	**80.9**	**77.0**	**77.2**	**82.6**
(b) Average Rebuffering Ratio (%)																		
RB	1.7	1.7	1.4	2.1	2.1	1.7	1.8	2.8	3.0	3.3	3.9	1.6	2.0	5.6	3.9	4.2	4.7	1.2
BB	**0.1**	**0.1**	**0.08**	**0.09**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.09**	**0.09**	**0.1**	**0.09**	**0.08**	**0.3**	**0.1**	**0.08**	**0.1**
Bola	1.1	1.3	1.1	1.1	0.9	0.7	1.1	1.1	1.3	1.6	2.5	1.7	1.4	1.5	1.9	2.1	2.5	2.0
Festive	0.4	0.4	0.4	0.5	0.3	0.4	0.4	0.4	0.2	2.5	0.3	0.1	1.5	1.7	0.5	0.7	2.5	1.0
Gta	0.6	0.4	0.5	0.5	0.5	0.7	0.4	0.4	0.6	0.3	0.3	0.5	1.1	0.5	1.5	0.7	1.2	0.8
Fineas	0.7	0.6	0.9	0.5	0.7	0.6	0.4	0.6	0.8	1.5	1.7	2.6	2.2	0.8	1.5	2.1	0.4	0.8
Flex-Steward	0.4	0.5	0.5	0.5	0.6	0.8	0.5	0.8	0.9	0.2	0.3	0.3	0.4	0.6	**0.3**	0.9	1.6	0.7
(c) Average Positive Quality Switch																		
RB	**1.22**	1.33	**1.16**	3.14	3.17	3.23	2.61	2.82	2.89	**1.31**	1.59	1.94	3.27	3.07	3.40	2.98	3.04	3.15
BB	4.31	4.25	4.27	6.05	5.98	6.46	5.50	5.58	5.29	3.46	3.64	3.08	5.62	5.53	5.60	4.57	5.36	5.25
Bola	1.25	1.29	1.34	3.01	3.08	2.77	2.59	2.31	2.22	1.43	**1.55**	**1.35**	3.02	3.32	3.23	3.07	3.38	2.54
Festive	1.28	**0.99**	1.20	**2.80**	2.63	2.40	2.34	2.7	2.57	1.53	1.63	1.54	**2.88**	3.30	3.08	2.45	**2.42**	2.19
Gta	2.71	2.51	1.23	3.02	2.86	3.14	4.11	4.99	3.79	1.66	1.69	1.74	3.36	3.93	3.81	3.51	3.80	3.42
Fineas	1.94	1.36	1.37	3.72	4.14	3.59	3.26	3.26	3.24	1.91	2.26	2.12	4.17	4.48	5.35	3.89	4.11	3.80
Flex-Steward	1.92	1.73	**1.16**	3.20	**2.62**	**2.26**	**2.21**	**2.15**	**1.94**	1.81	1.72	1.78	3.01	**2.96**	**2.63**	**2.35**	2.44	**1.78**
(d) Average Negative Quality Switch																		
RB	**1.19**	1.30	1.14	3.01	3.10	3.11	2.45	2.71	2.80	**1.28**	1.55	1.87	3.13	2.98	3.33	2.86	2.94	3.06
BB	4.29	4.24	4.25	5.93	5.88	6.35	5.32	5.42	5.11	3.42	3.58	3.04	5.48	5.51	5.56	4.46	5.23	5.31
Bola	1.23	1.29	1.31	2.89	2.95	2.65	2.43	2.20	2.06	1.38	**1.51**	**1.31**	2.96	3.16	3.18	2.90	3.14	2.43
Festive	1.25	**0.96**	1.16	**2.70**	2.58	2.30	2.27	1.88	2.48	1.50	1.59	1.51	2.78	3.16	2.91	2.34	2.31	2.10
Gta	2.66	2.42	1.17	2.80	2.62	2.93	3.77	4.61	3.53	1.56	1.61	1.68	3.18	3.78	3.69	3.34	3.56	3.23
Fineas	1.91	1.34	1.37	3.59	3.97	3.51	3.15	3.26	3.28	1.90	2.23	2.09	4.04	4.36	5.19	3.76	3.93	3.82
Flex-Steward	1.86	1.65	**1.08**	3.03	**2.38**	**2.11**	**1.93**	**1.86**	**1.73**	1.76	1.64	1.69	**2.76**	**2.73**	**2.44**	**2.10**	**2.18**	**1.63**
(e) Average QoE Standard Deviation																		
RB	9.90	9.97	9.07	18.2	17.9	18.4	14.4	15.8	15.2	10.9	12.4	12.2	18.6	18.7	19.0	16.5	16.6	14.9
BB	18.7	18.4	18.6	23.2	23.2	23.6	19.9	20.3	19.9	17.0	17.6	15.7	23.5	23.3	23.7	19.3	20.4	20.9
Bola	9.88	10.0	10.1	18.3	17.9	16.9	15.0	14.5	15.0	11.3	12.2	10.6	18.4	19.6	18.7	17.4	18.2	16.4
Festive	**9.51**	7.91	9.62	16.7	**15.9**	15.2	13.5	12.9	14.4	12.5	11.6	10.5	19.8	20.8	18.9	15.1	16.6	12.4
Gta	15.9	15.3	8.53	**16.4**	16.4	18.8	20.9	23.1	19.4	10.5	9.99	10.7	18.2	19.7	20.1	18.2	18.6	18.1
Fineas	12.1	10.5	10.7	20.2	20.3	20.2	16.3	17.6	17.3	12.9	14.4	14.4	21.3	22.2	25.2	18.9	19.9	19.5
Flex-Steward	9.99	**8.08**	**6.83**	17.6	16.1	**14.5**	**10.4**	**12.4**	**9.14**	**9.63**	**9.09**	**9.55**	**15.5**	**16.4**	**15.3**	**11.6**	**13.2**	**9.88**
(f) The 5th Percentile Minimum QoE																		
RB	60.2	59.9	62.6	25.3	25.3	25.0	32.4	31.1	28.8	**59.3**	52.0	51.8	24.4	26.2	22.5	29.9	28.1	32.1
BB	30.3	31.2	30.8	-5	-4	-7	0.43	0.11	1.42	35.9	33.3	41.0	0	1.28	0.98	4.79	-2	-3
Bola	59.6	60.2	59.0	27.4	27.4	30.0	31.4	36.0	32.0	56.9	54.5	**58.2**	26.0	21.5	23.4	23.7	21.9	28.0
Festive	**61.2**	**64.2**	60.1	29.7	33.3	35.8	36.1	38.4	33.2	52.9	51.2	55.6	22.0	14.9	20.6	29.1	26.8	38.0
Gta	37.6	39.1	62.1	**32.9**	32.3	29.2	13.0	2.49	15.0	57.0	57.4	57.0	28.5	23.5	22.5	25.1	23.9	24.3
Fineas	51.3	59.7	64.4	28.5	26.9	28.5	19.4	20.5	19.5	53.3	45.6	48.7	16.8	16.9	15.5	23.3	19.7	21.0
Flex-Steward	56.1	63.5	**69.4**	25.2	**37.5**	**40.6**	**40.9**	**46.2**	**52.8**	56.2	**57.8**	56.1	**29.1**	**32.4**	**35.7**	**40.9**	**39.4**	**48.0**

TABLE VI: Average QoE Standard Deviation under the Same Priority

Trace	Wired Network			Wireless Network		
Dev / ABR	P1	P2	P3	P1	P2	P3
RB	17.59	17.25	17.02	17.10	16.72	16.82
BB	18.46	18.21	17.03	22.83	22.83	22.82
Bola	17.35	18.99	17.26	16.62	16.37	15.79
Festive	18.28	19.44	17.60	15.50	14.54	15.03
Gta	17.09	18.18	18.12	17.45	18.36	15.97
Fineas	19.99	19.89	21.19	18.54	17.72	17.92
Flex-Steward	**13.91**	**13.21**	**13.56**	**11.18**	**13.14**	**12.82**

methods. Additionally, Fig.11 and Fig.12 present diverse distribution of requested bitrate by clients with the same priority and the QoE distribution for clients with different characteristics, respectively. Note that v_1 and v_2 respectively imply videos with mostly static and dynamic scenes. There are three takeaways following these results:

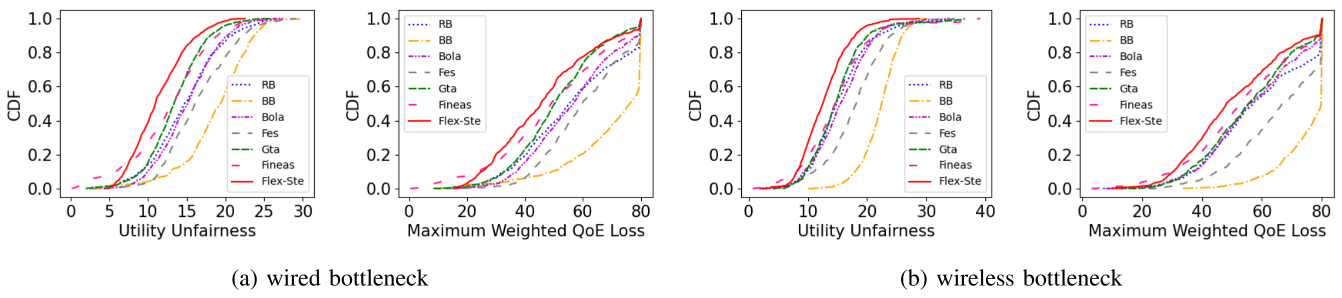
First, we find that when using Flex-Steward, the clients with the same priority but different devices tend to request chunks that support the same quality rather than the same bitrate. As shown in Fig.11, clients watching videos on high-resolution devices tend to request a higher bitrate than clients using low-resolution devices. On the other hand, for the clients with the same priority, those watching highly dynamic videos tend to request a higher bitrate than the clients who watch low dynamic videos. Since RB, BB, Bola, Festive, and Fineas do not take the quality variation into consideration, their bitrate decisions solely depend on their personal perceived network conditions, which results in the fact that clients



(a) [PHONE, Priority 1] (b) [PHONE, Priority 3]

(c) [4KTV, Priority 1] (d) [4KTV, Priority 3]

Fig. 13: QoE variations among clients of different characteristics



(a) wired bottleneck (b) wireless bottleneck

Fig. 14: Comparison of utility fairness and maximum weighted QoE loss under wired and wireless bottleneck

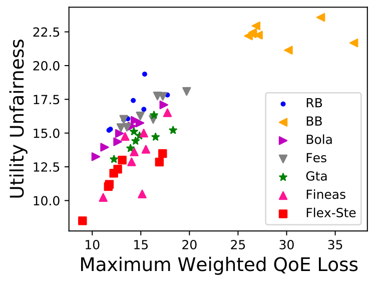


Fig. 15: Performance Comparison

with low resolution devices request high bitrate chunks and waste bandwidth. Since Gta and Flex-Steward take chunk-level perceptual quality into ABR consideration, the QoE of clients watching videos with high resolution devices has improved significantly. In contrast, the QoE of clients watching videos on low resolution devices has only a slight decline as shown in Table III and Table IV.

Secondly, Flex-Steward can achieve differentiated service among clients with different priorities. Table III, Table IV and Fig.12 illustrate that clients with higher priorities can obtain higher QoE. Take the results in Table III as an example, the QoE of clients with priority 2 and 3 is 4.8% and 8.6% higher than the QoE of clients with priority 1, respectively when the wired bottleneck is considered, and 1.9% and 4.2% higher than the QoE of clients with priority 1, respectively when the wireless bottleneck is used. Fig.10c, and Fig.10d and Fig.12 also show that the QoE of clients with high priority is higher and more stable than that of clients with low priority.

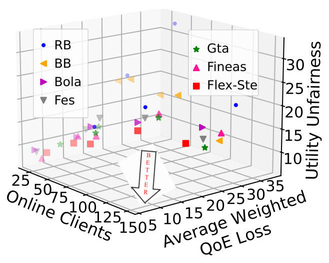
Finally, although the average bitrates selected by Flex-Steward are sometimes lower than those of other methods, it improves the utilization efficiency of bottleneck bandwidth due to its quality-size-aware adaptation and results in fewer negative effects to the QoE of clients (i.e., rebuffering event and quality switch), so that the clients using Flex-Steward maintain a high QoE. On average, Flex-Steward outperforms RB, BB, Bola, Festive, Gta, Fineas by 3.2%, 31.6%, 2.2%, 1.3%, 4.5%, and 3.3% when the wired bottleneck is considered, and 0.7%, 19.1%, 0.1%, 2.9%, 0.1%, and 2.5% for the wireless bottleneck, respectively. Fig.10 shows that Flex-Steward has significantly improved the QoE of a group of clients by slightly reducing the QoE of other group of clients. Comparing Fig.10a, Fig.10b, and Fig.10c, we find that Flex-Steward significantly increase the QoE of clients with high resolution devices, while slightly decreasing the QoE of clients using low resolution devices. Therefore, the overall QoE of clients at the same edge network is maintained at a high level.

2) Single Client Performance: This section discusses the behavior of clients with different characteristics. Fig.13 shows the QoE variations of clients who download the same video over the same bottleneck bandwidth (i.e., wired bottleneck). The quality of video services is assessed in terms of average QoE and QoE stability. According to Fig.13, Flex-Steward has the best performance in achieving differentiated services. Comparing Fig.13a and Fig.13b, we note that the clients with priority 3 have higher QoE stability when compared with the clients with priority 1 using Flex-Steward. Since a low bitrate is sufficient to enable PHONE clients obtain high quality, the QoE of clients with the two priority levels are similar at the stable state. On the other hand, when comparing Fig.13c and Fig.13d, we find that the clients at the priority level 3 have a higher QoE when compared with clients at priority level 1 based on Flex-Steward.

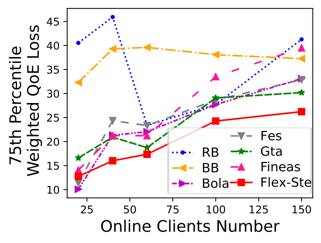
The QoE of clients using Flex-Steward is more stable than that of clients who use other methods. That is because the ABR agent of Flex-Steward is deployed at the edge server where it can perceive the status of the bottleneck and all online clients in real-time, and integrate these factors to make bitrate adaptive recommendations to clients, thereby reducing the risk of QoE fluctuations caused by bandwidth congestion. Additionally, Flex-Steward is based on reinforcement learning, which enables learning through historical experience and as decisions are based on the future state transition probability, Flex-Steward further reduces the risk of QoE fluctuations.

3) Performance of Joint QoE Optimization: This section compares the performance of joint QoE optimization across the considered methods, including QoE fairness for clients with the same characteristics and utility fairness between all clients. As shown in Table.V.e, the average QoE standard deviation of clients with the same [device, priority] configuration using Flex-Steward is significantly lower than those using the other methods. Note that Flex-Steward prefers to improving the QoE and QoE fairness of high-priority client. Thus, for clients with the low priority, the QoE fairness may be lower comparing with other configurations. Besides, one of the objectives of joint QoE optimization is to improve QoE among clients with various devices. Therefore, we measure the QoE fairness of all online clients under the same priority but different devices. The results are shown in Table VI and implies that Flex-Steward improves QoE fairness among clients with same priority. Note that we also calculate Jain's Fairness as an indicator of QoE fairness. Each client group with same priority using Flex-Steward has the highest Jain's fairness. From the experimental results, Flex-Steward surpassed the second place by 0.5%to 3.4%. Due to the space limitation, we only add the Table VI in this paper.

According to Table.V.f, Flex-Steward improves the QoE of the client with the lowest QoE among all clients, helping reduce the risk of the clients quitting watching videos. At the same time, as shown in Fig.14, Flex-Steward has the lowest utility unfairness and the maximum weighted QoE loss when compared with the other methods. Compared with RB, BB, Bola, Festive, Gta, and Fineas, Flex-Steward decreases the average utility unfairness by 24.3%, 37.8%, 21.1%, 25.4%, 11.3%, and 10.9%, respectively, when the wired bottleneck is used, and 23.2%, 41.7%, 17.6%, 24.2%, 11.1%, and 13.9%, respectively, when the wireless bottleneck is employed. The reason Flex-Steward has the lowest utility unfairness and the lowest maximum weighted QoE loss as well as achieve high average QoE is that it reasonably uses the bottleneck bandwidth resources. Flex-Steward selects high bitrates for clients with high-resolution devices or high priorities, while it selects low bitrates for others, increasing the efficiency of the bottleneck bandwidth.



(a) Performance Comparison



(b) 75th maximum weighted qoe loss

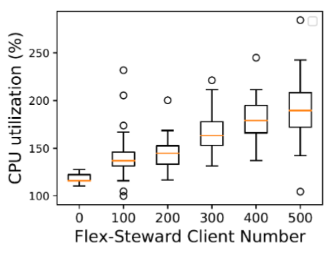
Fig. 16: Performance under Different Online Clients Number

4) Trade Off Between Utility Unfairness and Weighted QoE Loss: We vary the bottleneck bandwidth and the client characteristics in order to evaluate the performance of Flex-Steward under network changes. We conduct 8 epochs of evaluation for each method, and each epoch of evaluation lasts for 15 minutes. After the evaluation of each epoch, we record the average value of utility unfairness and the maximum weighted QoE loss. The bottleneck traces include two test wired traces shown in Fig.9a and two test wireless traces shown in Fig.9b; the two configurations of clients' characteristics are shown in Table II. We get a total of 8 combinations of bottleneck bandwidth and configurations of clients' characteristics. The scatter plot of the tradeoff between average utility unfairness and average maximum weighted QoE loss is shown in Fig.15. Flex-Steward improves utility fairness without affecting too much the maximum weighted QoE in comparison with the other methods. This is as the ABR agent of Flex-Steward takes global information and personal information of the client into consideration so that it reduces the negative effects (i.e., rebuffering ratio and quality switch) on QoE. At the same time, the quality-size-aware adaptation and differentiate service mechanism of Flex-Steward improve the bottleneck bandwidth utilization efficiency.

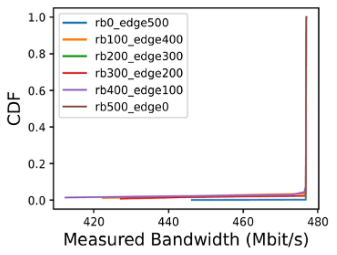
E. Varying Online Clients Number

Meanwhile, we also compare the performance of the Flex-Steward with the baseline ABR algorithms for different online clients number. (i.e., 20, 40, 60, 100, 150). Fig.16a shows the average weighted QoE loss and utility unfairness under different online clients number. Note that during this test, we use the "Test 1" trace shown in Fig.9a and multiply 60 to simulate the bottleneck bandwidth.

As shown in Fig.16a, when the number of online clients is over 40, Flex-Steward has the best performance on Average Weighted QoE Loss. That is because Flex-Steward reduces the QoE of some low-resolution-device clients by reducing the QoE of some high-QoE clients. Flex-steward is more conservative than ohter baseline methods, so when the shared



(a) measure the CPU utilization



(b) measure throughput of source

Fig. 17: Verify the scalability

bandwidth is not large enough to become a bottleneck (e.g., 20 clients), the weighted QoE loss of Flex-Steward is slightly higher than other method. On the other hand, as shown in Fig.16a, the utility unfairness is higher than Gta and BB method. That is because Gta and BB lack the appropriate quality-aware ABR algorithm, so that they tend to request chunks with too small a bitrate to prevent clients from re-buffering. Thus, both Gta and BB methods only maintain a low QoE level for clients, and clients are easy to quit watching videos.

Besides, we evaluate the 75th percentile maximum weighted QoE loss to reflect the risk of clients quitting video watching in the case of varying degree of multi-client bandwidth competition. The result is shown in Fig.16b and illustrates that as the fierce competition for bandwidth intensifies, Flex-Steward can show more obvious advantages.

F. Testing the overhead

As mentioned in section II-C, the edge server needs to receive some messages from clients. Thus, the scalability of Flex-Steward emerges as an issue. To verify the scalability of Flex-Steward, we measure the required resources in terms of computation and overhead bandwidth for running Flex-Steward on edge regarding the different numbers of clients.

1) Computing Resource Comsumption: For computing resource comsumption, we test it on an edge server with two 16-core CPUs. Fig.17 shows the relationship between the CPU utilization of the ABR algorithm on the edge server and the number of online clients. Since the server has two 16-core CPUs, the maximum CPU utilization is 1600%. We measure the CPU utilization of our ABR algorithm every two seconds. When no online clients use Flex-Steward, the edge server loads multiple neural networks but does not make bitrate decisions. At this time, according to Fig.17, when no online clients are using Flex-Steward, the median CPU utilization is 110%, and the 95th percentile CPU utilization is 130%. As the number of online clients increases, the CPU utilization increases. When the number of online clients using Flex-Steward reaches 500, the median CPU utilization is about 172%, and the 95th percentile CPU utilization is about 240%. Take the statistical value of the 95th quantile as the maximum CPU utilization under the current number of clients, the max number of online clients that the edge can provide services can be calculated as:

$$\frac{3200-130}{240-130}=\frac{n_{max}}{500} \tag{7}$$

TABLE VII: Size statistics of exchanged data between the client and various source

Bitrate	Src / Sch	Cloud size	Cloud ratio	Edge size	Edge ratio
334Kbps	only cloud	9.29MB	100%	0	0%
	cloud+edge	9.33MB	98.6%	0.13MB	1.4%
24Mbps	only cloud	628MB	100%	0	0%
	cloud+edge	628MB	99.9%	0.13MB	0.01%

It also is affected by other processes at the edge server and the CPU scheduling strategy of the edge server. Note that the edge node only covers a small cluster of clients (e.g., clients in the same campus, family, or community), which generally do not have tens of thousands of concurrent requests. Moreover, if the edge cluster is large, we can use a server with stronger computing power as the edge server.

2) Bandwidth Overhead: Fig.17b shows the relationship between bandwidth utilization and the number of clients using Flex-Steward. We limit the throughput of the source server up to 500Mbit/s. In this experiment, the total number of clients is set to 500 and divided into two categories. The result is shown in Fig.17b, ‘rb’ indicates clients who request chunks from the source server directly while ‘edge’ indicates edge-assisted video downloading process (i.e., Flex-Steward). We use “ifstat” in Liunx to measure the throughput of the source server. As is shown in Fig.17b, the proportion of clients using the edge-assisted method does not affect the throughput of the source server.

On the other hand, we also investigate the amount of traffic is caused by the communication between the client and the edge. We test a client’s data exchange amount with the edge server and the source server when downloading a video. As shown in Table.VII, the data exchange between clients and the edge node only accounts for a tiny proportion, so the Flex-Steward method only brings a very small traffic overhead even if the clients increase. We take two rounds of experiments. In each round, we use “tcpdump” to capture all packets at the server simulating clients. Table V shows clients’ data size and ratio to the edge server and video source server, respectively. In the first round, the client downloads a low-bitrate video at 334Kbps, and the result shows that the data exchange between the client and edge only accounts for 1.4%. Moreover, the second round of experiment results shows that when the client downloads the same video, the size of data exchange between the client and edge remains almost unchanged. Therefore, the higher the bitrate requested by the client, the lower the proportion of data communication between the client and the edge in the total traffic.

VII. Related Work

The Quality of Experience (QoE) for HTTP adaptive streaming (HAS) is a widely investigated research field. Meanwhile, there are many studies that analyze the problem of multi-client bandwidth competition and propose appropriate bitrate adaptation strategies under different scenarios. According to the deployment location of the bitrate decision agent, the joint QoE optimization scheme can be divided into three types: server-side, client-side and network-assisted solutions.

Server-Side Solutions The source server can have a global view of client status. Meanwhile, servers are easier to allocate their bandwidth resources, and then realize joint QoE optimization for multiple clients from the network level. Akhshabi et al. [20] limited the server's throughput for each chunk to avoid players staying idle during Steady-State, then avoiding vacant bandwidth changes to improve fairness. Marai et al. [21] leveraged a score matrix in throughput allocation to ensure a fair share of the server's bottleneck bandwidth. Vikram et al. [22] designed an end-to-end transport protocol for multi-client video streaming which employs client-server cooperation . Despite their relative positive results, nowadays few joint QoE optimization algorithms are deployed at the server side, mainly because it is difficult to estimate the bottleneck capacity for thousands of access networks. In addition, real-time traffic shaping for thousands of clients is associated with enormous computing costs at the servers, which is not economically sustainable.

Client-Side Solutions Due to simple deployment and scalability, most researchers deploy user experience optimization modules at the client. Spiteri et al. [6] and Huang et al. [7] select bitrates solely considering buffer occupancy based on a heuristic method and a Lyapunov optimization, respectively. Yin et al. [8] and Zhou et al. [64] use a control theory method and a Markov decision method respectively, which combine the buffer and network status perceived by the client to make bitrate adaptation decisions. Furthermore, Mao et al. [5] proposes a deep RL-based ABR method to improve QoE of clients with limited network resources. Kumar et al. [65] proposes an adaptation scheme based on finite-state machine for scalable video coding videos to improve clients' QoE. Recently, following the deployment of image quality evaluation technologies, quality-aware ABR algorithms have been proposed at the client-side, such as [38] and [33], making ABR optimization closer to the optimization of clients' real QoE. Several client-side based approaches supporting multiple HAS were proposed to enhance efficiency, fairness and bandwidth utilization [41], [43]. Jiang et al. designed FESTIVE [41], which decides the next chunk's bitrate and download time by employing a stateful and delayed bitrate update strategy and randomized scheduling. Li et al. designed PANDA [43] that uses a probing mechanism to estimate the available bandwidth and responds to network changes rapidly. Seufert et al. [66] refers to the TCP scheme for improving fairness and proposes a adaptation method to optimize the fairness of multi-client QoE. Bentaleb et al. [44] utilizes a game-theoretic approach at the client-side, which comprehensively considers the screen resolution of devices, clients' priority, and characteristics of the requested video to make bitrate-adaptation decisions. Client-side strategies are easy to deploy without any changes to the network or servers. However, they may suffer from sub-optimal performance due to limited information about other clients' states. It is necessary to find a solution that takes into account scalability and global views of multiple clients.

Network-Assisted Solutions To strengthen the control of multiple users while taking into account the scalability, Thomas et al. [67] proposes the Server and Network-assisted DASH (SAND) architecture. SDN is one of the main enablers for the SAND architecture. Cofano et al. [12] discusses a number of schemes for deploying multi-client ABR based on SDN. Bentaleb et al. [23] [24] proposed a scalable way to deploy an innovative ABR solution based on an SDN network. The solution divides a wide range of clients into multiple clusters and makes bitrate decisions for the cluster based on the cluster characteristics. Lu et al. [25] took the quality information of videos into account to guide the bitrate adaptive decision on a centralized SDN controller. Bagci et al. [27] proposed a dynamic value-based resource allocation method for video delivery over SDN from the perspective of Internet service providers. However, the above strategies are implemented based on network architectures (e.g., SDN) that have not been widely deployed. Meanwhile, this kind of scheme shapes the traffic in the network and increases the instability of network performance. Another type of solution uses the resources of the base station to coordinate the management of multi-user QoE. Another type of solution is aimed at mobile clients and manages multi-client QoE by allocating base station resources [68], [69]. Nowadays, there are also some researches that deploy the heuristic algorithm for joint QoE optimization on the edge server [45]. They utilize the edge server to monitor bandwidth status of managed clients and send guidance signals to clients. However, these solutions are difficult to manage the needs of large-scale heterogeneous clients and make long-term benefit decisions for clients.

VIII. Conclusions

This paper introduces Flex-Steward, an innovative solution that employs reinforcement learning and learns from historical information to achieve joint QoE optimization between video clients sharing a common bottleneck link. Flex-Steward employs a DASH-based adaptive approach and makes appropriate bitrate decisions based on a model trained using observations collected from clients and networks. Extensive experimental results show that Flex-Steward outperforms alternative joint QoE optimization algorithms. Under different network conditions and with diverse client characteristics, Flex-Steward increases QoE level, improves QoE distribution and reduces utility unfairness between video clients. Future work will explore how to use the computing and storage resources at the edge servers and combine edge-side ABR algorithms with efficient caching algorithms to improve joint QoE optimization further.

Acknowledgement

This work is supported by the Key-Area Research and Development Program of Guangdong Province (No. 2020B0101130006), National Natural Science Foundation of China under grant No. 61972189, the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989, the Social Science Fund of Shenzhen under grand No. SZ2020D009, and Science Foundation Ireland Research Centres Programme grants SFI/12/RC/2289_P2 (Insight) and 16/SP/3804 (ENABLE).

REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Forecast and methodology 2018–2023," 2020.

[2] J. Song, F. Yang, Y. Zhou, S. Wan, and H. R. Wu, "Qoe evaluation of multimedia services based on audiovisual quality and user interest," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 444–457, 2016.

[3] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[4] L. Zou, T. Bi, and G.-M. Muntean, "A dash-based adaptive multiple sensorial content delivery solution for improved user quality of experience," *IEEE Access*, vol. 7, pp. 89 172–89 187, 2019.

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

[6] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

[7] 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 *Proc. ACM SIGCOMM*, 2015, pp. 187–198.

[8] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proc. ACM SIGCOMM*, 2015, pp. 325–338.

[9] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when http adaptive streaming players compete for bandwidth?" in *Proc. ACM NOSSDAV*, 2012, pp. 9–14.

[10] V. Sivaraman, T. Moors, H. Habibi Gharakheili, D. Ong, J. Matthews, and C. Russell, "Virtualizing the access network via open apis," in *Proc. ACM CoNEXT*, 2013, pp. 31–42.

[11] S. Sundaresan, W. De Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè, "Broadband internet performance: a view from the gateway," in *Proc. ACM SIGCOMM*, 2011, pp. 134–145.

[12] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, "Design and experimental evaluation of network-assisted strategies for http adaptive streaming," in *Proc. ACM MMSys*, 2016, pp. 3:1–3:12.

[13] A. Balasingam, M. Bansal, R. Misra, K. Nagaraj, R. Tandra, S. Katti, and A. Schulman, "Detecting if LTE is the bottleneck with bursttracker," in *The 25th Annual International Conference on Mobile Computing and Networking, MobiCom 2019, Los Cabos, Mexico, October 21-25, 2019*, S. A. Brewster, G. Fitzpatrick, A. L. Cox, and V. Kostakos, Eds. ACM, 2019, pp. 1:1–1:15.

[14] F. S. Ahamed M M, "5g backhaul: requirements, challenges, and emerging technologies," *Broadband Communications Networks: Recent Advances and Lessons from Practice*, 2018.

[15] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide qoe fairness using openflow-assisted adaptive video streaming," in *Proc. ACM SIGCOMM Workshop on FhMN*, 2013, pp. 15–20.

[16] G.-M. Muntean, P. Perry, and L. Murphy, "Objective and subjective evaluation of QOAS video streaming over broadband networks," *IEEE Trans. Network and Service Management*, vol. 2, no. 1, pp. 19–28, Nov. 2005.

[17] G.-M. Muntean, "Efficient delivery of multimedia streams over broadband networks using qoas," *IEEE Transactions on Broadcasting*, vol. 52, no. 2, pp. 230–235, 2006.

[18] Z. Yuan and G.-M. Muntean, "A prioritized adaptive scheme for multimedia services over ieee 802.11 wlans," *IEEE Transactions on Network and Service Management*, vol. 10, no. 4, pp. 340–355, 2013.

[19] L. Zou, R. Trestian, and G.-M. Muntean, "E3doas: Balancing qoe and energy-saving for multi-device adaptation in future mobile wireless video delivery," *IEEE Transactions on Broadcasting*, vol. 64, no. 1, pp. 26–40, 2018.

[20] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proc. ACM NOSSDAV*, 2013, pp. 19–24.

[21] O. El Marai and T. Taleb, "Online server-side optimization approach for improving qoe of dash clients," in *Proc. IEEE GLOBECOM*, 2017, pp. 1–6.

[22] V. Nathan, V. Sivaraman, R. Addanki, M. Khani, P. Goyal, and M. Alizadeh, "End-to-end transport for video qoe fairness," in *Proc. ACM SIGCOMM*, 2019, pp. 408–423.

[23] A. Bentaleb, A. C. Begen, and R. Zimmermann, "Sdndash: Improving qoe of http adaptive streaming using software defined networking," in *Proc. ACM Multimedia*, 2016, pp. 1296–1305.

[24] 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, no. 10, pp. 2136–2151, Jul. 2017.

[25] Z. Lu, S. Ramakrishnan, and X. Zhu, "Exploiting video quality information with lightweight network coordination for http-based adaptive video streaming," *IEEE Trans. Multimedia*, vol. 20, no. 7, pp. 1848–1863, 2018.

[26] D. Wu, J. Yan, H. Wang, D. Wu, and R. Wang, "Social attribute aware incentive mechanism for device-to-device video distribution," *IEEE Trans. Multimedia*, vol. 19, no. 8, pp. 1908–1920, Apr. 2017.

[27] K. T. Bagci and A. M. Tekalp, "Dynamic resource allocation by batch optimization for value-added video services over SDN," *IEEE Trans. Multimedia*, vol. 20, no. 11, pp. 3084–3096, 2018.

[28] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[29] O. E. Computing. (2017) Open edge computing initiative. [Online]. Available: http://openedgecomputing.org/

[30] CORD, "Cord: extensible service delivery platform," 2018. [Online]. Available: https://opencord.org/

[31] A. Balasingam, M. Bansal, R. Misra, K. Nagaraj, R. Tandra, S. Katti, and A. Schulman, "Detecting if LTE is the bottleneck with bursttracker," in *The 25th Annual International Conference on Mobile Computing and Networking, MobiCom 2019, Los Cabos, Mexico, October 21-25, 2019*, S. A. Brewster, G. Fitzpatrick, A. L. Cox, and V. Kostakos, Eds. ACM, 2019, pp. 1:1–1:15. [Online]. Available: https://doi.org/10.1145/3300061.3300140

[32] T. Huang, R. Zhang, C. Zhou, and L. Sun, "QARC: Video quality aware rate control for real-time video streaming based on deep reinforcement learning," in *Proc. ACM Multimedia*, 2018, pp. 1208–1216.

[33] T. Huang, C. Zhou, R. Zhang, C. Wu, X. Yao, and L. Sun, "Comyco: Quality-aware adaptive video streaming via imitation learning," in *Proc. ACM Multimedia*, 2019, pp. 429–437.

[34] DASH. (2020) Dash.js player. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js?

[35] X. Ma, Q. Li, J. Chai, X. Xiao, S. Xia, and Y. Jiang, "Steward: Smart edge based joint QoE optimization for adaptive video streaming," in *Proc. ACM NOSSDAV*, 2019, pp. 31–36.

[36] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 2001–2014, 2013. [Online]. Available: https://doi.org/10.1109/TNET.2013.2281542

[37] C. Qiao, J. Wang, and Y. Liu, "Beyond qoe: Diversity adaptation in video streaming at the edge," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 289–302, 2021. [Online]. Available: https://doi.org/10.1109/TNET.2020.3032416

[38] Y. Qin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue, "Quality-aware strategies for optimizing abr video streaming qoe and reducing data usage," in *Proc. ACM MMsys*, 2019, pp. 189–200.

[39] Z. Duanmu, W. Liu, D. Chen, Z. Li, Z. Wang, Y. Wang, and W. Gao, "A knowledge-driven quality-of-experience model for adaptive streaming videos," *CoRR*, vol. abs/1911.07944, 2019.

[40] R. Rassool, "VMAF reproducibility: Validating a perceptual practical video quality metric," in *Proc. IEEE BMSB*, 2017, pp. 1–2.

[41] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proc. ACM CoNEXT*, 2012, pp. 97–108.

[42] C. Zhou, C. Lin, X. Zhang, and Z. Guo, "TFDASH: A fairness, stability, and efficiency aware rate control approach for multiple clients over DASH," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 1, pp. 198–211, 2019. [Online]. Available: https://doi.org/10.1109/TCSVT.2017.2771246

[43] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[44] A. Bentaleb, A. C. Begen, S. Harous, and R. Zimmermann, "Want to play dash? a game theoretic approach for adaptive streaming over http," in *Proc. ACM MMsys*, 2018, pp. 13–26.

[45] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. D. Turck, "Qoe-driven rate adaptation heuristic for fair adaptive video streaming," *ACM Trans. Multim. Comput. Commun. Appl.*, vol. 12, no. 2, pp. 28:1–28:24, 2016.

[46] N. T. Baranasuriya, V. Navda, V. N. Padmanabhan, and S. Gilbert, "Qprobe: locating the bottleneck in cellular communication," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments*

and Technologies, CoNEXT 2015, Heidelberg, Germany, December 1-4, 2015, F. Huici and G. Bianchi, Eds. ACM, 2015, pp. 33:1–33:7.

[47] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proc. ACM SIGCOMM workshop on HotNets*, 2010, p. 19.

[48] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. JMLR ICML*, 2016, pp. 1928–1937.

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

[50] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning," in *Proc. USENIX OSDI*, 2016, pp. 265–283.

[51] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. JMLR ICML*, 2016, pp. 1928–1937.

[52] NGINX, "NGINX: High performance load balancer, web server and reverse proxy," 2019. [Online]. Available: https://nginx.org/

[53] uWSGI, "UWSGI: web server," 2016. [Online]. Available: https://uwsgi-docs.readthedocs.io/en/latest/

[54] Django, "Django: web framework," 2019. [Online]. Available: https://www.djangoproject.com/

[55] redislabs. (2020) Redis. [Online]. Available: https://redis.io/

[56] FFmpeg, "Ffmpeg: A complete, cross-platform solution to record, convert and stream audio and video," 2019. [Online]. Available: http://ffmpeg.org/

[57] GPAC. (2020) Mp4box. [Online]. Available: https://gpac.wp.imt.fr/mp4box/

[58] FCC, "Measuring broadband america-eighth report," 2018. [Online]. Available: https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-eighth/

[59] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4g lte dataset with channel and context metrics," in *Proc. ACM MMsys*, 2018, pp. 460–465.

[60] W. Robitza, S. Göring, A. Raake, D. Lindegren, G. Heikkilä, J. Gustafsson, P. List, B. Feiten, U. Wüstenhagen, M.-N. Garcia, K. Yamagishi, and S. Broom, "HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P.1203 – Open Databases and Software," in *9th ACM Multimedia Systems Conference*, Amsterdam, 2018.

[61] T. Huang, C. Zhou, X. Yao, R.-X. Zhang, C. Wu, B. Yu, and L. Sun, "Quality-aware neural adaptive video streaming with lifelong imitation learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2324–2342, 2020.

[62] R. R. Ramachandra Rao, S. Göring, W. Robitza, A. Raake, B. Feiten, P. List, and U. Wüstenhagen, "Bitstream-based model standard for 4k/uhd: Itu-t p.1204.3 – model details, evaluation, analysis and open source implementation," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, Athlone, Ireland, May 2020.

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

[64] C. Zhou, C. Lin, and Z. Guo, "mdash: A markov decision-based rate adaptation approach for dynamic HTTP streaming," *IEEE Trans. Multimedia*, vol. 18, no. 4, pp. 738–751, 2016.

[65] S. Kumar, R. Devaraj, A. Sarkar, and A. Sur, "Client-side qoe management for SVC video streaming: An FSM supported design approach," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 3, pp. 1113–1126, 2019.

[66] M. Seufert, N. Wehner, and P. Casas, "A fair share for all: Tcp-inspired adaptation logic for qoe fairness among heterogeneous HTTP adaptive video streaming clients," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 2, pp. 475–488, 2019.

[67] E. Thomas, M. van Deventer, T. Stockhammer, A. C. Begen, M.-L. Champel, and O. Oyman, "Applications and deployments of server and network assisted dash (sand)," 2016.

[68] M. Anedda, M. Murroni, and G. Muntean, "A novel markov decision process-based solution for improved quality prioritized video delivery," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 1, pp. 592–606, 2020.

[69] I. Triki, R. E. Azouzi, and M. Haddad, "NEWCAST: joint resource management and qoe-driven optimization for mobile video streaming," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 2, pp. 1054–1067, 2020.

Xiaoteng Ma received the B.Eng. degree in electrical engineering from the Huazhong University of Science and Technology in 2017. He is currently pursuing the Ph.D. degree with Tsinghua–Berkeley Shenzhen Institute. He majors in Data Science and Information Technology. His research interests include edge-assisted multimedia delivery, adaptive multimedia and multimedia streaming, and resource allocation on hybrid cloud-edge-client network.

Qing Li (S'10-M'14) received the B.S. degree (2008) from Dalian University of Technology, Dalian, China, the Ph.D. degree (2013) from Tsinghua University, Beijing, China; both in computer science and technology. He is currently an associate researcher professor at Peng Cheng Laboratory, Shenzhen, China. His research interests include reliable and scalable routing of the Internet, software defined networking, network function virtualization, in-network caching/computing, edge computing, traffic scheduling, transmission control, video delivery, etc.

Yong Jiang is currently a Full Professor with Tsinghua Shenzhen International Graduate School. He received his B.S. degree and Ph.D. degree both from Tsinghua University, respectively in 1998 and 2002. He mainly focuses on the future Internet, edge computing, multimedia transmission, AI for networks, etc.

Gabriel-Miro Muntean (M'04, SM'17) is a Professor with the School of Electronic Engineering, Dublin City University (DCU), Ireland, and Co-Director of the DCU Performance Engineering Laboratory. He has published over 450 books, chapters and papers in top-level international venues. His research interests include quality, performance, and energy saving issues related to rich media delivery, technology-enhanced learning, and other data communications over heterogeneous networks. Prof. Muntean is an Associate Editor of the IEEE Transactions on Broadcasting, the Multimedia Communications Area Editor of the IEEE Communications Surveys and Tutorials, and chair and reviewer for important international journals, conferences, and funding agencies. Prof. Muntean is senior member of IEEE, IEEE Broadcast Technology Society and IEEE Communications Society.

Longhao Zou (S'12-M'19) received the B.Eng and Ph.D degrees from Beijing University of Posts and Telecommunications (BUPT), Beijing, China and Dublin City University (DCU), Ireland in 2011 and 2016, respectively. He was a postdoctoral researcher with the EU Horizon 2020 NEWTON Project at DCU. Now he is Research Associate Professor with Southern University of Science and Technology, and also with Peng Cheng Laboratory, Shenzhen, China. His research interests include mobile and wireless communications, adaptive multimedia and mulsemedia streaming, resource allocation and user quality of experience.