QAVA: QoE-aware Adaptive Video Bitrate Aggregation for HTTP Live Streaming based on Smart Edge Computing

Xiaoteng Ma, Qing Li, *Member, IEEE,* Longhao Zou, *Member, IEEE,* Junkun Peng, Jianer Zhou, *Member, IEEE,* Jimeng Chai, Yong Jiang, *Member, IEEE* and Gabriel-Miro Muntean, *Senior Member, IEEE*

***Abstract*—Currently video streaming in heterogeneous network environments is affected by limited network bandwidth availability and consequent low and variable user Quality of Experience (QoE) levels. In particular, for the case of live video streaming, a very high number of end-clients request content at the same time, generating huge concurrent traffic, and putting pressure on the existing network infrastructure. An approach which helps address this issue is deployment of emerging edge computing technologies to smooth the live streaming traffic and improve QoE by adapting client bitrates and caching content at the edge server. In this context, this paper proposes a novel QoE-aware Adaptive Video bitrate Aggregation scheme for HTTP live streaming based on smart edge computing (QAVA). As an intelligent proxy server, a "smart edge" which deploys QAVA aggregates all the traffic requested by clients for the same live streaming service and adapts their bitrates based on network conditions, client states and video characteristics. The adaptation is performed based on a Deep Reinforcement Learning (DRL)-based algorithm, which is also proposed. The QAVA DRL algorithm is trained and modeled based on a real client experience dataset. The experimental evaluation results presented in this paper show how QAVA outperforms other state-of-the-art adaptive bitrate algorithms in terms of average QoE and QoE fairness.**

***Index Terms*—HTTP live streaming, edge computing, video adaptation, bitrate aggregation, reinforcement learning, QoE**

I. Introduction

THE live video streaming industry has experienced a huge growth in the last few years [1]. In addition to the natural demand for higher video quality, lower re-buffering and fewer quality switches, the live streaming clients have critical Quality of Experience (QoE) [2] requirements in terms of low latency in the current dynamic network conditions, which are different from those of traditional Video on Demand (VoD) services.

In the architecture supporting currently live video streaming services, only a few data centers, hosted by Content Providers (CP), are deployed in a core-regional network to serve millions of end-clients [3]. Therefore, it is no surprise that the large amount of generated traffic makes very challenging to guarantee high client QoE for the live video streaming services. Adaptive Bitrate (ABR) algorithms are generally employed to enhance QoE. However, existing ABR approaches have limitations which include the following ones. Some ABR solutions greedily consume large amounts of bandwidth by selecting the highest bitrates possible [4]–[6], affecting the stability and fairness of client QoE. Other ABR schemes do not consider the client device type and video characteristics, wasting precious network bandwidth and negatively affecting device performance. For example, such schemes would display higher quality videos on low-resolution devices [7] or would play videos containing many static scenes with low temporal and spatial complexity encoded at very high bitrates [8]. In general, these existing ABR solutions cannot ensure that the QoE of all live video streaming clients remains at a high level.

Moreover, end clients located in the same area are likely to request similar video content. Especially as the emerging 5G network solutions will encourage the exchange of a large amount of traffic and the use of rich media formats such as omnidirectional, 4K/8K, immersive video content [9], [10]. Therefore, redundant multimedia transmissions may consume huge network resources under the traditional network architecture, affecting the latency and efficiency of video distributions [11]. This situation would be exacerbated by the large-scale user requests of live video services [12].

The emerging edge computing technologies [13]–[15] are offering new possibilities to improve the QoE of live video streaming by alleviating transmission redundancy and reducing bandwidth competition. Taking the 5G Multi-access Edge Computing as an example, it helps to aggregate the large-scale nonredundant client requests and allocate the video traffic intelligently, which highly releases the traffic pressure on the links of 5G User Plane Function, 5G core and the corresponding Internet Content Delivery Network (CDN) servers. Compared to traditional data centers, the edge computing servers are deployed widely at the network edge, "closer" to end clients. By utilizing various intelligent mechanisms, the edge computing servers can handle client requests, predict network conditions, and optimize QoE more accurately and efficiently.

This paper proposes QAVA, a smart QoE-aware adaptive video bitrate aggregation scheme for HTTP live streaming

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

Q. Li is with Peng Cheng Laboratory (PCL), Shenzhen, China (Email: liq@pcl.ac.cn).

L. Zou and J. Zhou are with the Southern University of Science and Technology and also with Peng Cheng Laboratory (PCL), Shenzhen, China (Email: zoulh@sustech.edu.cn; zhouje@sustech.edu.cn).

J. Peng, J. Chai and Y. Jiang are with the Tsinghua Shenzhen International Graduate School, Tsinghua University and also with Peng Cheng Laboratory (PCL), Shenzhen, China (Email: pjk20@mails.tsinghua.edu.cn; chaijm17@126.com; jiangy@sz.tsinghua.edu.cn).

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

Corresponding author: Qing Li (Email: liq@pcl.ac.cn)

based on edge computing. QAVA is deployed at the edge nodes of an access network where bandwidth competition mostly happens [16]. By monitoring network performance and availing from edge storage and computation, QAVA provides live video services to all the clients within the same access network, at improved QoE levels. Specifically, QAVA first aggregates the demands for the same video from end clients, then requests the content at an appropriate bitrate from data centers, and finally delivers it to the clients. However, QAVA needs to overcome variations in network conditions, diversity of client behaviors and characteristics, and difficulty in controlling client QoE. In order to address these, QAVA employs a Deep Reinforcement Learning (DRL)-based control policy to adjust video bitrate selections intelligently in real-time based on network conditions, client states, and video characteristics.

In order to assess the performance of QAVA, a prototype based on Nginx [17], uWSGI [18] and Django [19] is employed.The performance of QAVA is evaluated under different network conditions. The results show how, when compared with several state-of-the-art ABR approaches based on the edge nodes, QAVA improves average QoE by between 7% and 64% and QoE fairness by between 19% and 52%.

The main contributions of this paper are as follows.

- The paper formulates the QoE-aware adaptive video streaming aggregation and globally optimizes video bitrate adaptation problems to maximize users' QoE and minimize QoE unfairness, utilizing edge computing.
- QAVA adaptive algorithm based on a novel DRL model is proposed to perform intelligent bitrate adaptation during video streaming. A general reward function that allows QAVA to improve the QoE fairness among multiple online clients and ensures that the QoE of the clients remains at a high level under dynamic bottleneck bandwidth is also introduced.
- QAVA is assessed using a full system, employed to train and validate QAVA and its performance. The experimental results indicate that QAVA outperforms several other state-of-the-art ABR solutions, in terms of average QoE and QoE fairness.

The remainder of this paper is organized as follows. Section II gives a brief review of related works. Section III describes the proposed QAVA framework. In Section IV, the bitrate aggregation problem for HTTP live video streaming that considers both QoE and QoE fairness among online clients is formulated. The original problem is solved using a DRL model which makes the intelligent bitrate aggregation decisions. DRL is described in Section V. Prototype implementation is described in Section VI and QAVA performance evaluation is presented in Section VII. Finally, conclusions and future work directions are discussed in Section IX.

II. Related Works

Major related works are discussed next by focusing on ABR solutions for multimedia transmissions and DRL-based schemes for efficient distribution of multimedia and other traffic types.

A. ABR Solutions for Multimedia Transmissions

Researchers have been working on finding solutions for improving the efficiency of multimedia transmissions for many years. Several studies have proposed server-side ABR solutions [20], [21] and more recently client-side ABR schemes [4]–[6], [8], [22] in different contexts. In general, server-side solutions perform fairer bandwidth sharing, but as they rely on client feedback, they may introduce latency in the adaptation process. Muntean et al. [20] proposes the QOAS scheme, which uses an innovative estimation of client perceived quality in the ABR feedback loop and Muntean et al. [21] applies bitrate adaptation in the presence of wireless loss. The clients are the better position to perform the adaptation based on performance data which the client can direct access. K. Spiteri et al. [4] use Lyapunov optimization to select bitrates solely considering buffer occupancy and Zhou et al. [5] propose a Markov Decision based scheme for bitrate adaptation. Moldovan et al. [23] designs a novel DQAMLearn method that aims to support a good learner QoE under educational multimedia content. Mao et al. [6] and Huang et al. [8] utilize DRL techniques to learn an ABR control policy instead of using fixed rules for Video on Demand (VoD) services and live streaming, respectively. These algorithms do not know other client state information, and therefore may suffer from unstable and unfair performance distribution among clients mostly due to their bandwidth sharing competition.

In order to make use of a global view of the multi-client state, several network-assisted QoE-aware bitrate aggregation and joint optimization algorithms were proposed [24]–[30]. Cofano et al. [24] allocates network bandwidth slices to video streams, or guides bitrate selections by using a network controller and Software Defined Network (SDN) switches. Bentaleb et al. [25] leverages SDN capabilities of assisting large-scale heterogeneous clients in making better adaptation decisions. By using a central coordinator to receive quality and buffer level from clients and publishing the aggregate statistics, Lu et al. [26] helps clients make bitrate decisions. By adding a tracker that records all client states at the server-side, Detti et al. [27] solves the unstable performance caused by proxies/caches. Ma et al. [28] determines bitrates for live streaming clients based on their service levels and manages well network bandwidth sharing, ensuring fairness. However, such methods cannot realize fine-grained state tracking and control for many clients, so low bandwidth utilization and poor QoE caused by bandwidth competition still exist. Ma et al. [29], Zhang et al. [30] and Shi et al. [31] propose QoE optimization frameworks for VoD services based on the smart edge. However, such models are not applicable to the low-latency demands of live streaming.

B. DRL for Improving Transmission Efficiency

Recently, several DRL-based methods are proposed to address various problems of multimedia delivery systems. Jawad et al. [32] proposes a DRL-based framework to decide the most suitable routing algorithm to be applied on the QoS-based traffic flows to improve QoS provisioning. Comsa et al. [33] introduces a hierarchical DRL method to support optimized

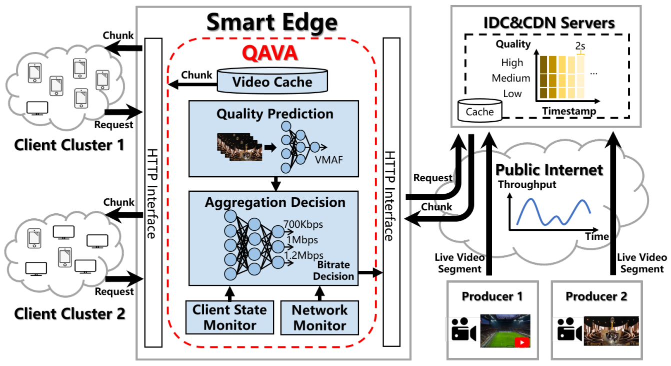


Fig. 1. QAVA-based System Overview.

network resource allocation for video delivery. Zhang et al. [34] employs DRL to find an effective proactive caching policy for multi-view 3D videos in the fifth generation (5G) networks. Zhang et al. [35] utilizes DRL to dynamically adapt to the variation of both client traffic and CDN performance to efficiently schedule large-scale clients. Yeo et al. [36] uses DRL to better leverage the advantages of combining video super-resolution with multimedia transmission. The above works indicate the potential of DRL for applications in multimedia transmissions. The multimedia transmission system generally can get immediate feedback on the state of the environment, enabling the DRL agent to interact with the environment in real-time. Due to the diversity of state features, DRL utilizes the deep neural network to model the states, which can incorporate more dimensional features than traditional modeling approaches, thus allowing for better state representation.

Apart from multimedia transmissions, DRL has also been applied to resource scheduling for multitasking in diverse environments. Chinchali et al. [37] applies DRL to cellular network traffic scheduling and enables mobile networks to carry 14.7% more data with minimal impact on existing traffic delivery quality. Mao et al. [38] uses DRL to design a multi-resource management scheduler to minimize average job slowdown or completion time. Chen et al. [39] develops a two-level DRL system to handle flow-level traffic optimizations in data centers. The above schemes demonstrate that DRL can achieve intelligent scheduling among multiple tasks, thus reducing the negative impact caused by multitasking competition.

This related work discussion demonstrates that DRL is very beneficial for proposing solutions for highly efficient transmissions, especially of multimedia content. However, DRL is very sensitive to the design of states, actions, and rewards, and the above DRL model are difficult to migrate to release multi-clients competition for HTTP live streaming. In this paper, we will discuss and design how to take advantage of DRL to alleviate the problem of multi-clients resources competition for HTTP live streaming.

III. QAVA Framework Design

A. Challenges

QAVA mainly faces three practical challenges that increase the difficulty of designing a good bitrate adaptation algorithm which involves aggregation:

- **Predicting the current state of the network is challenging.** The available network bandwidth changes dynamically over time. In this case, the adaptive bitrate aggregation algorithm needs to accurately predict the network available bandwidth and respond quickly to network changes, which is challenging.
- **Tracking clients' behavior is challenging.** The arrival and exit of a client introduce performance fluctuations to the services of other clients who share the bottleneck bandwidth. However, predicting when clients join or leave is very difficult.
- **Controlling client QoE is very difficult.** First, the algorithm should balance a variety of conflicting QoE metrics (e.g., perceptual quality, re-buffering event, quality switch, latency and chunk skip), not only for a single client, but also for multiple clients, which is a difficult task. Besides, since the content of a requested chunk during live video streaming is generated in real-time, its perceptual quality is difficult to measure in advance. This increases the difficulty of controlling the QoE of clients. Second, the bitrate selection for a given chunk can have a cascading effect on a client. For example, overestimating the available bandwidth may cause the client to choose a higher bitrate, while a long download time may cause the client to select a lower bitrate when making a bitrate decision for the next chunk, resulting in oscillations of client QoE. Third, the control decisions available to the current ABR algorithms are coarse-grained, thus it is difficult to control client QoE accurately.

In order to overcome these challenges, this paper introduces a new DRL-based control policy based on edge computing for adaptive video delivery. The proposed solution improves the video transmission efficiency in diverse network conditions.

B. System Overview

QAVA is deployed as a smart network function at the edge node (called "smart edge"). The *smart edge* is located at the entrance of the access network or at the edge of an Internet Service Provider (ISP), where it maintains stable communication with end clients. It is the best place to realize bitrate aggregation for live video streaming, mostly due to the following advantages:

- **Network Perception.** The bottleneck bandwidth competition is more likely to happen among clients in the same access network [16]. The edge closed to clients can perceive the bottleneck performance and client state with a lower cost. A solution deployed here can improve QoE fairness among clients.
- **Storage.** The edge node can collect client request messages, temporarily store requested video chunks, and deliver bulk content to end clients with low delay and high stability to eliminate redundant transmissions and reduce bandwidth resource consumption.
- **Computation.** The edge has the computing power to apply complex computation of quality prediction and bitrate aggregation. Moving the DRL-based ABR algorithm deployment from the clients to the edge node removes

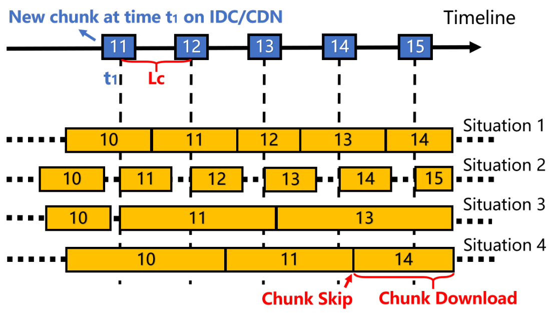


Fig. 2. Typical Chunk Download Situations.

the load of ABR computation from the client devices. Each client device is required to compute simple ABR metrics and select predicted quality levels of video when QAVA is not available. Otherwise, the edge-based QAVA will gather information on the global network conditions, differentiate the client requests, and then make the final selection for the cluster accordingly, while the clients' own ABR is disabled.

- **Locality.** The popularity of video content is spatially local, and local end-clients are likely to request the same video services [11]. The *smart edge* can make bitrate aggregation decisions according to the regional popularity of the video content to satisfy better clients' demands.

Fig. 1 illustrates the live video delivery with the *smart edge*. First, Video Producers (VP) upload their video segments to the Internet Data Center (IDC) or CDN servers through the public Internet in real-time. Then the IDC/CDN servers encode these video segments into multiple bitrates and save them in cache servers. The clients who use the same type of devices (e.g., HDTV and phone are considered in this paper) and watch the same video are clustered together. The clients of a cluster send chunk requests to the *smart edge*. The *smart edge* makes the bitrate aggregation decision and requests a specified bitrate chunk and broadcasts it to the clients after finishing its download.

QAVA architecture includes five modules that help achieve intelligent QoE-aware adaptive bitrate aggregation: (1) The **Client State Monitor Module (CSMM)** collects HTTP requests from clients and monitors online client QoE in real-time based on the information contained in their requests. (2) The **Network Monitor Module (NMM)** records the throughput of the bottleneck bandwidth every T seconds. (3) The **Quality Prediction Module (QPM)** uses the deep Nerual Network (NN) to infer the future chunk's quality. (4) The **Aggregation Decision Module (ADM)** is the core decision module in QAVA and makes bitrate decisions based on DRL for live video streaming. ADM utilizes the data collected by CSMM, NMM, and QPM, and makes bitrate decisions for each chunk of each video. (5) The **Video Cache (VC)** is used to cache live content, reducing the number of redundant transmissions dramatically. Section V describes the module functionality.

C. Video Streaming

Fig. 2 illustrates four typical chunk download situations during live video streaming. A new chunk is generated by the IDC/CDN servers every L_c seconds. A block on the dotted lines implies the download of a chunk, and a block on the timeline axis indicates the latest chunk at a certain time. In situations 1, 2 and 3, the clients always request the latest chunks. Situation 1 has enough bandwidth for smooth downloading with some delay, but no chunk skip. Situation 2 has surplus bandwidth and therefore it does not suffer from delay and chunk skips. For ABR solutions with no global view of client states, the idle periods of Situation 2 may cause other clients to overestimate the available bandwidth, resulting in oscillations of client QoE. Situation 3 has scarce bandwidth with unavoidable delay and chunk skips. Frequent chunk skip events may cause a severe QoE decrease. Consequently, chunk skip events are suppressed until the chunk to be requested falls behind more than P chunks compared with the latest one. Situation 4 illustrates the suppressed request behavior for the case of $P = 1$.

TABLE I
NOTATIONS USED IN SECTION IV

Notation	Description
M^t	The number of online clients at any time t
N^t	The number of online videos at any time t
K	The number of bitrate levels
b_{nk}	The bitrate value of the bitrate level k for the video n
x^t_{mn}	Binary: 1 if the client m is watching the video n at any time t; 0, otherwise
y^t_{nk}	Binary: 1 if the aggregated bitrate level is k for the video n at any time t; 0, otherwise
l^t_m	The chunk bitrate requested by the client m at time t
$q(l)$	The Perceptual quality of the video chunk l
sw^t_m	The absolute value of quality switching between the current chunk at time t and the last chunk
r^t_m	The re-buffering time for m during the request at time t
sk^t_m	The number of chunk skips for the client m from the current request at time t to the next request
e^t_m	The real-time latency of the client m at time t
Q^t_m	The QoE for client m at time t
Qh^t_m	The high QoE unfairness factor of m at time t
Ql^t_m	The low QoE unfaireness factor of m at time t
$\mathcal{Y}^t$	The set of requested bitrate levels for all videos at time t
W^t	The bottleneck bandwidth

IV. Problem Formulation

In order to understand the challenges of QoE-aware adaptive video bitrate aggregation, we formulate the problem as a linear optimization problem. In this way, we explain the problem more clearly. Besides, based on the problem formulation, we analyze the problem complexity and shortcomings of using optimization methods to solve it. We also discuss the necessity and advantages of using DRL to solve this problem. The notations in this section are summarized in Table I.

Assume that M^t and N^t are the number of online clients and videos at any current time t, respectively. Each video is encoded into K bitrate levels, and b_{nk} ($n \in [1, N^t]$, $k \in [1, K]$) represents the bitrate value of the bitrate level k for the video n. Two binary variables x and y denote the video viewed by a client and the aggregation decision for a video, respectively. $x^t_{mn} = 1$ ($m \in [1, M^t]$, $n \in [1, N^t]$) indicates that the client m watches the video n at any time t. $y^t_{nk} = 1$ ($n \in [1, N^t]$, $k \in [1, K]$) indicates that the aggregated bitrate level is k for the video n at any time t. Following the proposed specific objective QoE models in [6], [40], the detailed metrics and definition of QoE at time t are stated next.

Perceptual Quality: The Video Multi-Method Assessment Fusion (VMAF) [41] is used to evaluate the perceptual quality of a video chunk. The VMAF Development Kit (VDK) includes the VMAF models covering mobile phone and HDTV viewing conditions. The mapping between bitrates and VMAF for mobile phone and HDTV are denoted as $q_{ph}(\bullet)$ and $q_{hd}(\bullet)$, respectively. It is worth mentioning that the phone model is also suitable for laptops, TVs, etc. The bitrate of the chunk requested by the client m is defined as l_m^t = $\sum_{(n,k)}^{(N^t,K)} x_{mn}^t y_{nk}^t b_{nk}$. Thus, the perceptual quality of the chunk requested by the client m is $q_{ph}(l_m^t)$ when the client m watches the video by phone.

Quality Switch: The penalty of quality switch is denoted as sw_m^t = $|q(l_m^t) - q(l_m^{last})|$, where $q(l_m^{last})$ indicates the quality of the last requested chunk. The Quality Switch represents the quality variation of the video chunk and penalizes the impacts on the watching smoothness, which is computed by the edge.

Re-buffering: Clients send the buffer information to QAVA through the requests for a new chunk. Let f_m^t be the latest buffer size at any time t, which is received by the edge from the client m. Each new request from m updates f_m^t according to the real-time buffer size of m. Let t_m^{latest} be the time when the edge received the latest request of m. Let $d_m^t = t - t_m^{latest}$ be the duration from the time of receiving the latest chunk request to any current time t. Thus, the re-buffering time computed by the edge-side for the client m is r_m^t = $|d_m^t - f_m^t|_+$ (i.e., in seconds). The re-buffering implies any stalling event in the video streaming for each client.

Chunk Skip: Let z_m^t be the sequence number of the chunk being requested by the client m at any time t. The edge makes the decision for the client m whether to skip some chunks after the current requested chunk (z_m^t). If the edge instructs the client to request the next chunk of z_m^{next}, instead of the next chunk in order (i.e., $z_m^t + 1$), the number of skipped chunks is denoted as sk_m^t = $z_m^{next} - z_m^t - 1$. The corresponding QoE penalty for the video playback non-continuity, namely the chunk skip information, will be computed by the edge. Therefore, the edge will instruct the client to request the appropriate next chunk after the computation of potential QoE. Note that the client's perception of chunk skip is also correlated with the video content characteristics. We have done a subjective experiment to explore the relationship between the number of skipped chunks and client perception for different videos and we have found that the continuity of video content plays a major role in client perception. Therefore, in this paper, we use the number of skipped chunks to represent the impact of chunk-skip events on client QoE. The subjective experiment studying the chunk-skip influence on client's QoE is described in Section VII-D.

Latency: Assume that the client m is watching the video n. As shown in Fig. 3, the latency (i.e., in seconds) of the client m computed by the edge-side is e_m^t = $|f_m^t - d_m^t|_+$ + $(c_n^t - (z_m^{last} + 1))L_c$ + $(t - t_n^{new})$, where c_n^t is the sequence number of the latest chunk of the video n recorded in the client-side, L_c represents the time interval of new chunks appearing in the IDC/CDN servers collected by the edge-side, z_m^{last} is the sequence number of the last requested chunk

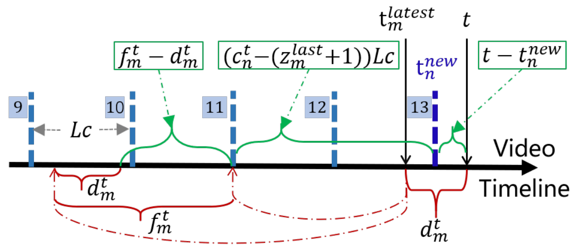


Fig. 3. An example of latency. The latest request from the client m received by the edge at time t_m^{latest} contains the information of f_m^t and z_m^{last} (i.e., 10 in the example). Besides, in this case, $C_n^t = 13$ and $z_m^{last} = 10$.

and t_n^{new} is the generating time of the latest chunk of the video n at the IDC/CDN servers collected by the edge-side as well. The values of f_m^t and z_m^{last} are updated by the request received by the edge at time t_m^{latest}. The Latency defined in the proposed QoE model consists of the startup delay (i.e., when the first chunk arrives) and other delay features within the video streaming procedure.

Related to the QoE model, we follow the industry definition [42] The QoE Q_m^t for client m at time t is represented as:

$$Q_m^t = q(l_m^t) - \alpha_1 sw_m^t - \alpha_2 r_m^t - \alpha_3 sk_m^t - \alpha_4 e_m^t, \quad (1)$$

where α_1, α_2, α_3 and α_4 are the non-negative term weights, indicating how each component affects client QoE. A relatively small α_1 indicates that the user is not particularly concerned about video quality variability; A larger α_1 is, the more effort is made to achieve smoother changes of video quality. A large α_2 indicates that a user is deeply concerned about Re-buffering. If users care about the video content playback continuity, we set α_3 to a relatively large value. In cases where users prefer low latency, we employ a larger α_4. Generally, the values of $q(l_m^t)$, sw_m^t, r_m^t, sk_m^t and e_m^t would vary in [0,1] depending on the evaluation results.

CPs would like to offer satisfactory or good QoE to an increased number of clients to improve revenue. Those with poor QoE are more likely to quit watching, which leads to a decline in CP's revenue. Based on the above considerations, we introduce the unfairness factors of Qh_m^t and Ql_m^t to indicate the QoE unfairness caused by bandwidth competition. Qh_m^t implies how much higher the QoE of the client m is, compared with other clients of higher QoE than the client m. Qh_m^t is larger if there are more clients with lower QoE than the client. Qh_l^t implies how much lower the QoE of the client m is, compared with other clients of lower QoE than the client m. Qh_l^t is larger if there are more clients with higher QoE than the client m. The definitions are as follows:

$$\begin{aligned} Qh_m^t &= \sum_{Q_p^t < Q_m^t} \frac{Q_m^t - Q_p^t}{M^t - 1} \\ Ql_m^t &= \sum_{Q_p^t > Q_m^t} \frac{Q_p^t - Q_m^t}{M^t - 1}. \end{aligned} \quad (2)$$

Qh_m^t and Ql_m^t can be calculated at any time t. The optimal goal is that all clients have equal QoE and thus $Qh_m^t = Ql_m^t = 0$, which implies that clients share bandwidth resources fairly with respect to their individual QoE.

Based on the above analysis, CP expects to improve the QoE fairness among online clients and guarantee that client QoE remains at a high level by deploying QAVA at the *smart edge*. Therefore, we have the following optimization objective that maximizes the sum of QoE and minimizes the sum of QoE unfairness. Since QAVA takes control of client QoE by choosing a bitrate for the chunk to be requested, the control variable for this optimization problem is defined as $\mathcal{Y}^t = \{Y_1^t, ..., Y_n^t, ..., Y_{N^t}^t\}$, which represents the set of requested bitrate levels for all online videos at time t, where Y_n^t is a one-hot vector and is denoted as $\{y_{n1}^t, ..., y_{nk}^t, ..., y_{nK}^t\}$.

Thus, the bitrate adaptation problem at time t on QAVA can be formulated as:

$$\max_{\mathcal{Y}^t} \sum_{m=1}^{M^t} (Q_m^t - \eta_1 Qh_m^t - \eta_2 Ql_m^t), \tag{3}$$

s.t.

$$x_{mn}^t, y_{nk}^t \in \{0,1\}, \forall m \in [1, M^t], \forall n \in [1, N^t], \forall k \in [1, K] \tag{4}$$

$$\sum_{n}^{N^t} x_{mn}^t \le 1, \forall m \in [1, M^t]; \sum_{k}^{K} y_{nk}^t \le 1, \forall n \in [1, N^t] \tag{5}$$

$$\sum_{n}^{N^t} \sum_{k}^{K} y_{nk}^t b_{nk} \le W^t, \tag{6}$$

where η_1 and η_2 are weighted parameters to tune the penalty of the QoE unfairness [43]. Although due to the property of symmetry, $\sum_{m=1}^{M^t} Qh_m^t = \sum_{m=1}^{M^t} Qh_l^t$, in the real scenario, the decision is made incrementally for each live streaming video n. Therefore, the parameters of η_1 and η_2 are retained for the problem formulation to maintain consistency in expression with the following part of this paper. Eq. (3) indicates the objective of QAVA, which is to maximize the QoE of online clients and minimize the QoE unfairness among all clients. Eqs. (4) and (5) guarantee that each client chooses at most one video and QAVA chooses at most one bitrate among chunks with the same content at time t. Eq. (6) indicates that the total bitrates requested by all clients should not exceed the bottleneck bandwidth W^t.

The problem can be reduced to a multi-dimensional knapsack problem. However, solving this problem through traditional linear optimization is a significant challenge when the number of videos increases. In addition, the decision at time t will affect the user experience in the future. For example, a too high bitrate may cause rebuffering or increase the real-time latency in the future. Therefore, employing DRL trained with real experience data is a potential approach to solve this problem in real-time. More significantly, it can learn from the training data for a better decision, which fully considers the impact on the future QoE.

V. DRL-based Adaptive Aggregation Decision

QAVA uses DRL to make adaptive bitrate aggregation decisions. At time t, the agent which makes adaptive bitrate aggregation decisions for the video n observes the state s_n^t and chooses an action a_n^t based on s_n^t. After applying the action, the state of the environment transitions to s_n^{t+1} and the agent receives a reward R_n^t. The goal of learning is to maximize the expected cumulative discounted reward: $E[\sum_{t=0}^{\infty} \gamma^t R_n^t]$, where $\gamma \in (0, 1]$ is a factor discounting future rewards. A3C [44] is employed and is formulated as a discrete time and action, continuous state model, by defining the state $s \in \mathcal{S}$, the action $a \in \mathcal{A}$ and the reward function R, which will be measured and computed by the modules introduced briefly in Section III-B. A3C uses an actor-critic model, where the critic network outputs the estimated value $V(s^t)$ of state s^t and the actor network outputs the probability distribution of each action $\pi(a^t|s^t)$. A detailed description of these components and working procedures for Section III-B modules follows.

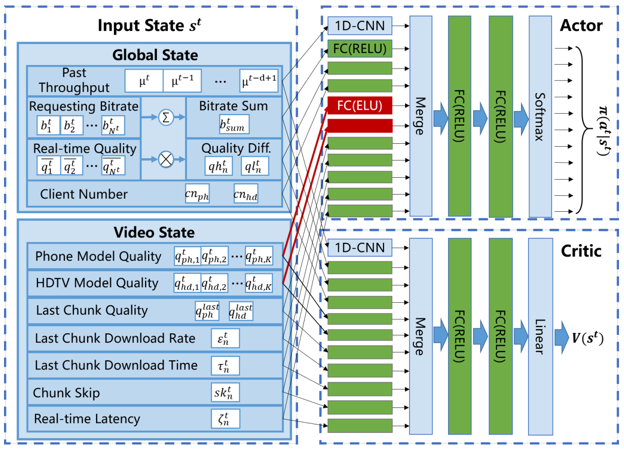


Fig. 4. The Proposed Deep Reinforcement Learning Model Design.

TABLE II
NOTATIONS USED IN SECTION V

Notation	Description
s_n^t	The Observed state of the video n at time t
a_n^t	The determined action of the video n at t
R_n^t	The received reward of the video n at t
$V(s^t)$	The estimated value of state s^t
$\pi(a^t\|s^t)$	The probability distribution of actions
$\vec{\theta}_g^t$	The global state
d	The number of sample periods for bandwidth estimation
qh_n^t	The high quality unfairness factor of the client m at t
ql_n^t	The low quality unfairness factor of the client m at t
$\vec{\theta}_v^t$	The video state
$\overline{Q_n^t}$	The avg. QoE of the clients watching the video n at t

A. States

QAVA aggregates the requested bitrates of a cluster of clients into one bitrate. When QAVA receives the first request for a new chunk of the video n, ADM makes a bitrate decision based on the real-time state s^t, including network conditions, client states, and video characteristics collected from NMM, CSMM, and QPM. Then all the following clients in the same cluster adopt this bitrate. As shown in Fig. 4, we divide the input state into *global state* and *video state*.

Global State: To understand the bottleneck throughput (i.e., available bandwidth) and the state of clients in real time, NMM and CSMM collect the global state into a vector

$\vec{\theta_g^t} = \{\vec{\mu^t}, b_{sum}^t, \vec{q_d^t}, \vec{cn^t}\}$. NMM measures the bottleneck throughput of the past d sample periods and the sum of all downloading chunks' bitrates, which are denoted as $\vec{\mu^t}$ and b_{sum}^t, respectively. At the same time, $\vec{q_d^t}$ and $\vec{cn^t}$, which relate to clients, are measured by CSMM. $\vec{q_d^t}$ contains qh_n^t and ql_n^t, which indicate the comparison between the average perceptual quality of the video n and those of other videos. qh_n^t implies how much higher the average perceptual quality of the video n is than those of other videos, while ql_n^t indicates how much lower the average perceptual quality of the video n is than those of other videos. The definitions of qh_n^t and ql_n^t are:

$$qh_n^t = \sum_{q_n^t > q_p^t} \frac{\overline{q_n^t} - \overline{q_p^t}}{N^t - 1}, \quad ql_n^t = \sum_{q_n^t < q_p^t} \frac{\overline{q_p^t} - \overline{q_n^t}}{N^t - 1}, \tag{7}$$

where $\overline{q_n^t}$ and $\overline{q_p^t}$ are the average perceptual quality of the video n and p, respectively. Besides, to assist the QoE trade off between phone clients and HDTV clients, we input the client number vector $\vec{cn^t}$ to the neural network, which contains the number of online clients using phone and HDTV, respectively. The deep NN learns the best aggregation decision under different phone and HDTV clients through historical experiences. The global state indicates the shared bandwidth status and the status of clients watching other videos. With this information, DRL can avoid using the shared bandwidth greedily for a single video.

Moreover, for some CPs, clients can be subscribers on different price tiers of a streaming service. Under this circumstance, QAVA can perform the aggregation on a per-tier basis. The deep NN makes the aggregate bitrate for each tier respectively. From the perspective of business strategy, to obtain more benefits, different QoE weights can be set for clients of different tiers, so as to achieve flexible QoE control between users of different tiers. However, this problem involves complex network economic models and client behavior pattern analysis, which is beyond the scope of this paper.

Video State: A seven-dimension vector $\vec{\theta_v^t} = \{\vec{q_{ph}^t}, \vec{q_{hd}^t}, \vec{q_{last}^t}, \epsilon_n^t, \tau_n^t, sk_n^t, \zeta_n^t\}$ is used to indicate the characteristics of the chunk to be requested and smoothness of the video download process. Due to the different screen resolution of devices among clients, the perceptual quality of the same video can also be different. Thus, to assist the QoE trade-off between phone clients and HDTV clients, we input $\vec{q_{ph}^t}$ and $\vec{q_{hd}^t}$, which are the vectors of the chunk's predicted quality of all K bitrates based on the phone model and HDTV model, respectively, which are predicted by QPM. The last five items represent the client states that are influenced by downloading the past chunks and are recorded by CSMM. $\vec{q_{last}^t}$ contains the quality of the last requested chunk based on the phone model and HDTV model, respectively. ϵ_n^t and τ_n^t are the download rate and download duration of the last chunk, respectively. sk_n^t is the number of skipped chunks caused by the download of the last chunk. ζ_n^t indicates the average real-time latency of all the clients at time t. The client buffer level is not included in the states. As in the live streaming transmission system, the real-time latency is an implicit indicator for the client buffer level. For example, a high client buffer level means a high real-time latency.

B. Actions

The agent in ADM makes the bitrate decision for the next chunk based on the measured states. The action space is a K-dimension vector for K alternative bitrates.

C. Rewards

When the agent for video n in ADM requests a new chunk z_n^t, it computes the reward R_n^t according to the last chunk z_n^{last}. A reward function is introduced to measure how the impact of the last action is in line with our objective. Specifically, the reward R_n^t is set at time t as:

$$R_n^t = \overline{Q_n^t} - \eta_1 qh_n^t - \eta_2 ql_n^t, \tag{8}$$

where $\overline{Q_n^t}$ is the average QoE of the clients watching video n, which is monitored by CSMM. The definitions of qh_n^t and ql_n^t are detailed in Eq. (7). In the problem formulation part, the unfairness factors (i.e., Qh_m^t and Ql_m^t) are defined based on the QoE differences among clients, according to Eq. (2) and (3). In this part, we replace the QoE differences with the perceptual quality differences to calculate the unfairness factors, because the negative effects of QoE (e.g., re-buffering events) cause drastic oscillation of QoE, making the model of deep NN difficult to converge. It is worth noting that because R_n^t contains the QoE $\overline{Q_n^t}$, the result of DRL targets improving QoE and not only the perceived quality. Each online agent in ADM aims to maximize their rewards so that the proposed objective can be realised.

D. DRL Model

As is shown in Fig. 4, we propose the DRL model for the intelligent real-time decision of online video streaming. For actor-network, we first use the Fully Connected (FC) layer to preprocess the data of different features, so that each feature has the same dimension. For throughput prediction, we use 1D-CNN to capture time-series features; for video quality, we use ELU [45] as the activation function of FC to enhance the model's sensitivity to video quality which frequently changes during video playback. Other features use the general RELU activation function for FC. We employ a two-layer FC with RELU for the core learning model, which is fast enough to guarantee the real-time requirement and the accuracy of online decision-making. Unlike the actor-network, the critic network uses RELU as the activation function of the preprocessing layer of the video quality because the critic network should not be too sensitive to the policy gradient. Besides, the final output of the critic network is a linear function to regress the value of the state s^t. The experiment results in section VII-F1 verifies the effectiveness of this model.

VI. Prototype Implementation

To validate the performance of QAVA, the QAVA-based video delivery system with the quality prediction and DRL agents is implemented.

TABLE III
TEST SETTINGS

Description	**Value**
(a) Quality Prediction Model	
The number of videos in train/test dataset	42/6
The learning rate	10^{-4}
(b) Deep Reinforcement Learning Model	
The discount factor γ	0.99
The learning rate of actor network	3×10^{-4}
The learning rate of critic network	3×10^{-3}
(η_1, η_2) in the reward function Eq.(8)	(0.8, 1)
(c) Implementation	
The Poisson arrival rate λ	0.08
Network Traces	FCC [46]
$(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ in QoE model (Eq.(1))	(1, 0.5, 0.3, 0.03)
The number of hosts	20
Maximum cached chunk number for each video on smart edge	2
The chunk skip tolerant factor P	0
The duration of a chunk (in seconds)	2

A. QAVA-based Video Delivery System in the Real Test-bed

The real testbed involved three x86 servers configured with two *Intel Xeon E5-2600* CPUs to support the live video delivery process. All three servers run *Ubuntu 16.04* system, and are used for video source server, *smart edge*, and end-clients, respectively. The video source server is based on Nginx [17], a lightweight and highly stable HTTP server. QAVA prototype is mainly written in Python 2.7 based on Nginx, uWSGI [18], and Django [19], a common deployment in production environments[1]. After QAVA receives a HTTP request from a client, QAVA makes a bitrate aggregation decision, requests the chunk from the source server, sends the chunk to the client and temporarily stores in VC if the request is the first one for a new chunk. Otherwise QAVA sends the chunk stored in VC directly to the client. In order to guarantee the real-time video streaming service, the storage capacity in the *smart edge* is set to 2 video chunks. To simulate the dynamic bandwidth between the IDC/CDN servers and the *smart edge*, we utilize the Linux Traffic Control tool to control the sending rate of the video source server. We create 20 virtual hosts to simulate the clients. The client request packets are modified to include the clients' current buffer occupancy.

B. Quality Prediction Agent

In order to predict the perceptual quality of the next chunk that is the input of the DRL Video State (i.e. Phone Model Quality or HDTV Model Quality shown in Fig. 4), we refer to the NN architecture in QARC [8] to extract the features of the past video chunks and implement the agent in TensorFlow [47]. We pass past 2 chunks, each of which sampled 12 frames, so totally 24 frames with a size of [96, 64] with 3 channels are inputted into the feature extraction layer. It consists of a convolutional layer with 64 filters, each of size 3 with stride 1, a max pooling layer with a 3×3 filter, and another convolutional layer with the same settings, a max pooling layer with a 2×2 filter and a fully connected layer with 256 nodes. Then, we pass 24 256-dimension vectors to two gate recurrent unit (GRU)

[1]QAVA code is shared on Github: https://github.com/chaijm/QAVA/.

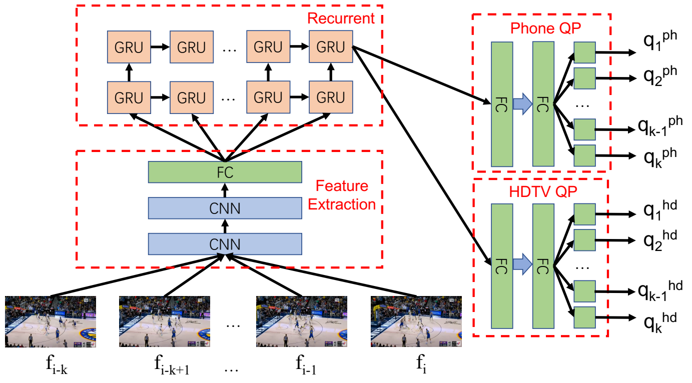


Fig. 5. The Quality Prediction Model.

layers with 256 hidden units. In addition, we connect GRU layers' output with a 2-dimensional device vector through the 2-hidden-layer fully connected layer with 513 and 256 nodes respectively to generate a 10-dimension vector, in which each value represents the VMAF-based perceptual quality score normalized in [0,1] for all alternative bitrates using different devices (i.e., Phone and HDTV). Note that the nodes of the neural network use "RELU" as the activation function except for the output layer of Phone Quality Prediction (QP) and HDTV QP parts. The output layer of the Phone QP and the HDTV QP parts utilize the "linear" activation function. Additionally, the Adam gradient optimizer with a learning rate 10^{-4} is used to train the prediction network. The filter number, feature dimension number, and learning rate are the best parameters used in multiple sets of experiments. The critical parameters' values of the quality prediction agent are summarized in Table.III(a).

C. Deep Reinforcement Learning Agent

A3C [44] is employed to realize parallel bitrate decisions for all videos and the agent is implemented in TensorFlow [47]. Each agent in ADM uses the NN-based actor-critic model to represent the policy $\pi(a|s)$. QAVA feeds an 11-dimension vector (i.e., all the items in $\vec{\theta_g^t}$ and $\vec{\theta_v^t}$) into the NN. In the actor network, the dimension containing the throughput measurements of the past $d = 8$ sample periods is passed into a one-dimension convolutional layer with 128 filers, each of size 4 with stride 1. The other 10 elements are each passed into a fully connected layer with 128 nodes. Then we splice all the output and pass them into a 2-hidden-layer fully connected layer with 1024 and 512 nodes respectively. The second layer's results are applied to the Softmax activation function to output the probability distribution of the policy $\pi(a|s)$. The selection of the activation function is detailed in the next section. The critic network has the same NN architecture as the actor network except that its final output is a linear neuron without activation functions. The discount factor $\gamma = 0.99$ and the learning rate for the actor and the critic network are configured as 3×10^{-4} and 3×10^{-3}, respectively. η_1 and η_2 in Eq. (8) are set to 0.8 and 1, respectively. We have tried extensive (η_1, η_2) combinations in order to identify the combination which results in good performance. The ablation study of η_1 and η_2

is detailed in Section VII-E1. The critical parameters' values of the deep reinforcement learning agent are summarized in Table.III(b).

VII. Performance Evaluation

A. Dataset

Video dataset: The quality prediction model is trained on a large-scale video dataset containing music videos, cartoon, and short movies, which includes two public datasets from [48] and [49] as well as a self-collected video dataset of Tencent music [50]. In order to guarantee the diversity of videos, we obtain 48 videos from these three sources. Among them, there are 4, 16 and 28 videos from [48], [49] and [50], respectively. We have used 42 and 6 different video sequences to train and test both the quality prediction agent and the deep reinforcement learning agent, respectively. The video diversity enables agent independence from particularities of a single video sequence. The length of these videos ranges from ten seconds to tens of minutes, and the resolution is configured to 1920 × 1080 by following the instruction of VMAF (version 0.6.1). These videos are encoded by H.264 and MPEG-DASH using the FFmpeg tool [51]. Each video is encoded into 10 discrete bitrates: {334, 396, 522, 595, 791, 1000, 1200, 1500, 2100, 2500} Kbps. Each chunk represents about a 2 second video.

Network traces: A bandwidth trace dataset is created from two public datasets: a broadband dataset provided by the FCC [46] and a HSDPA mobile dataset collected in Norway [52]. The dataset contains average throughput trace at 1 second granularity. We generate two one-hour throughput traces from the FCC dataset (e.g. vary from 100Mbps to 200Mbps) and the HSDPA dataset (e.g. vary from 0Mbps to 10Mbps) following the [6], respectively. To simulate the bottleneck bandwidth, we adjust the values of the throughput traces according to the number of tested videos. Then we use the Linux Traffic Control tool to simulate the dynamic bandwidth between the video source server and the *smart edge* according to the generated throughput traces.

Client behavior dataset: One million pieces of raw data are generated using the described testbed, which represents the behavior of all online clients sharing the bottleneck. The *global state* and *video state* are extracted from the raw data as the input of the DRL model. By using the generated 40-hour data to train the model, the whole model training process is completed in 30 minutes. To model the patterns of client requests, we assume that each client follows a Poisson arrival process with λ = 0.08 (The value of λ refers to the setting in [24]) and selects a video based on a uniform distribution.

B. Methodology

The following baseline methods which fetch video content directly from the IDC/CDN servers are considered:

- Rate-Based (RB): A client-side ABR that chooses the highest available bitrate below the harmonic mean of the past five-chunk download data rate.
- BOLA [4]: A client-side ABR that uses Lyapunov optimization to select bitrates solely considering buffer occupancy observation.

The following advanced edge node approaches are considered demonstrating advantages of the proposed DRL solution:

- Rate-Based with Cache (RBC) and BOLA with Cache (BOLAC): All clients implement RB and BOLA algorithms, respectively. The edge node only sends a request to the IDC/CDN servers containing the bitrate of clients' first request for a video chunk.
- Tracker (TKR) [27]: A tracker-based approach aims to solve the unstable performance caused by proxies/caches.

C. Evaluation Metrics

Average QoE: The definition of QoE for the client m is shown in Eq. (1), where t is the time that the client m finishes downloading a chunk. Then the average QoE is the mean of all QoE values during the one-hour network trace test. The parameters $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ in the QoE definition are set to (1, 0.5, 0.3, 0.03), respectively. We refer to [6] [40] [42] to set the relatively balanced values of each parameter, which is better to perform the comparisons for the proposed QAVA and the other benchmarks in this paper. Clients and CP can tune the weights of the parameters according to their preferences. We have performed an ablation study for the weights of different components of *Eq.* (1) to illustrate how the QoE model can be adjusted to suit various scenarios. The study and its results are shown in Section VII-E2.

Detailed QoE Metrics: The quality, quality switch, rebuffering ratio, chunk skip frequency and latency are measured to offer a deep dive of the performance of all approaches.

Unfairness: The standard deviation σ^t of QoE among M^t online clients at any time t is used to indicate QoE unfairness. This is defined as:

$$\sigma^t = \sqrt{\frac{1}{M^t}\sum_{m=1}^{M^t}(Q_m^t - \overline{Q^t})^2}, \tag{9}$$

where Q_m^t is the QoE of the client m and $\overline{Q^t}$ is the mean of QoE of all active clients at any time t, defined as $\overline{Q^t} = \frac{1}{M^t}\sum_{m=1}^{M^t} Q_m^t$.

Minimum QoE: The minimum QoE of all the clients is considered as the client suffering from the worst QoE is most likely to quit watching.

D. Subjective Experiment of Chunk-skip Events

To explore the relationship between the number of skipped chunks and client perception, we have invited 110 volunteers to participate in a subjective experiment. We have chosen three videos from the video dataset and cut out 20-second video clips from each video. Then we used the clips of each video to generate 15 test videos, respectively. Each test video has a chunk-skip event from a particular playback position, and the number of skipped chunks is between 1 and 3. The volunteers have watched 45 test videos and rated them from 0 to 4 based on their perceptions when watching the video. A score of 4 indicates that the volunteers are satisfied with the video playback process, while a score of 0 implies that they dislike the chunk-skip event. We normalize the volunteers' scores and calculate a Mean Opinion Score (MOS) for each

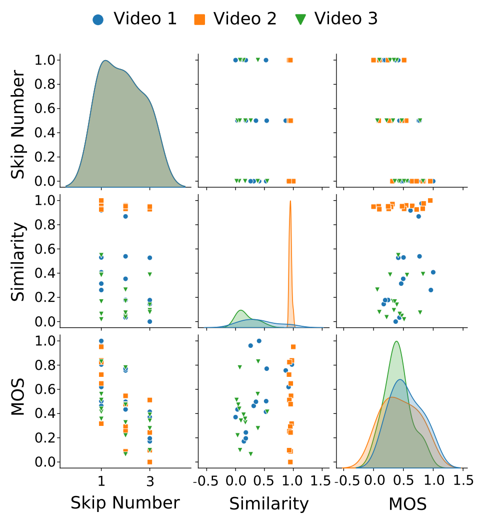


Fig. 6. Correlation between chunk skip number, frame similarity and MOS

TABLE IV
ABLATION STUDY FOR (η_1, η_2) COMBINATION

ID	(η_1, η_2)	Average QoE		Unfairness	Minimum QoE
		PH	TV		
1	(0, 0)	0.677	0.620	0.180	0.380
2	(0.08, 0.1)	0.734	0.667	0.136	0.505
3	(0, 1)	0.772	0.668	0.136	0.535
4	(0.4, 1)	0.720	0.648	0.160	0.436
5	(0.8, 1)	0.764	0.675	0.129	0.517
6	(1.6, 2)	0.585	0.508	0.166	0.425
7	(4, 5)	0.582	0.501	0.171	0.395
8	(8, 10)	-0.36	-0.40	0.628	-1.47
9	(1, 0)	0.622	0.519	0.153	0.444
10	(1, 0.4)	0.610	0.531	0.153	0.446
11	(1, 0.8)	0.730	0.628	0.153	0.446

test video. Additionally, we measure the similarity between the two frames before and after the skipped chunks by Structural SIMilarity (SSIM) [53]. Next, we also measure the correlation between MOS and frame similarity to assess the influence of chunk-skip events' influence on video contents with different characteristics. Fig. 6 shows a correlation pair plot among chunk skip numbers, frame similarity, and MOS. The scenes in video 1 and video 3 are mostly dynamic, while most of scenes in video 2 are static. We use the Pearson correlation coefficient to measure the correlation of two variables. We find that the correlation between MOS and the chunk skip numbers is -0.62, while the correlation between MOS and the frames similarity is only 0.18 in our experiment. Besides, the MOS has a similar linear relationship with the number of skipped chunks among different videos, according to Fig. 6. Thus, the result shows that the number of skipped chunks plays a dominant role in client perceptions so that we use the number of skipped chunks to represent the impact of chunk-skip events on QoE in this paper. On the other hand, we believe that adding the video characteristic to the chunk-skip factor can further improve the performance of QAVA.

E. Ablation Study

1) (η_1, η_2) combination: To find the (η_1, η_2) combination that can achieve good performance, we have done an ablation study of η_1 and η_2. The result is shown in Table IV. Comparing the result with the ID ranging from 1 to 7, we find that when η_1 and η_2 change between 0 and 1, clients experience good QoE, and QoEs among clients are relative fair. When η_1 and η_2 are higher than 1, DRL is more inclined to let the clients have fair quality according to the definition in Eq.(8) and results in Table IV, which makes clients suffer from poor QoE. Besides, comparing the results of ID 3~5 and ID 9~11, we find that when $\eta_1 < \eta_2$, QAVA can provide higher QoE and maintain fair QoE allocation for clients. That is consistent with the conclusion in [43]: the clients care more about inequity when their QoE is lower than others so that $\eta_1 < \eta_2$ can make QAVA tend to provide clients with high QoE. Moreover, we find that QAVA has relatively poor QoE fairness when $\eta_1 = \eta_2 = 0$ according to the result with ID 1. Meanwhile, the second and third terms in Eq.(8) make QAVA request for chunks with similar quality instead of similar bitrate, which further improves the effective utilization of bandwidth, thus making the client QoE promoted. Through our test, we find that using the (η_1, η_2) combination with ID 2~5 can make QAVA have good performance. In the future section, if not specified, we set η_1 and η_2 to 0.8 and 1 respectively.

2) Weights of metrics in the QoE model: As shown in Eq.(1), the value of QoE is dependent on the influence of multiple components. The weight of each component affects the evaluation of the overall client QoE. Multiple weight combinations were tried indicating that QAVA can be tuned to suit various scenarios. To explore the influence of tuning the weights on QoE, we replace the weight of the first component in Eq.(1) from 1 to α_0. Referring to the configuration mentioned in Section VII-C, we set the benchmark values of α_0, α_1, α_2, α_3, and α_4 to 1, 1, 0.5, 0.3, and 0.03, respectively. We regard the experiment using the QoE model with benchmark weights as the benchmark experiment. The results and analysis are included in Section VII-F.

We conduct a total of five rounds of experiments. In each round, we choose one of the benchmark values from α_0 to α_4 and multiply it by the scale of 2^{-3}, 2^{-2}, 2^{-1}, 2, 2^2, and 2^3, respectively, while fixing other weights. We get six combinations of weights through the above method. We use QoE models with these weights combinations to train the DRL on the same dataset as the benchmark experiment and test it in the same environment as the benchmark experiment.

Fig.7 shows the test results of each round. We choose the typical QoE metrics to illustrate the effect of tuning a particular α. As shown in Fig.7(a), when the α_0 is too large or too small, the perceptual quality of clients is improved, but the re-buffering ratio also increases significantly. When the α_0 is too large, the QoE model indicates QAVA to pursue high perceptual quality, resulting in lots of re-buffering, latency and chunk-skip because of downloading too large chunks. While the α_0 is too small, the first term in the reward function (i.e., Eq.(8)) is small so that the last two terms dominate the reward function, resulting in pursuing high perceptual quality which also causes lots of re-buffering, latency and chunk-skip. Besides, Fig.7(b)(c)(d)(e) implies that increasing a particular α can improve the performance of QAVA in the corresponding metric. However, it also causes performance degradation in some metrics. For instance, as shown in Fig.7(b), increasing α_1

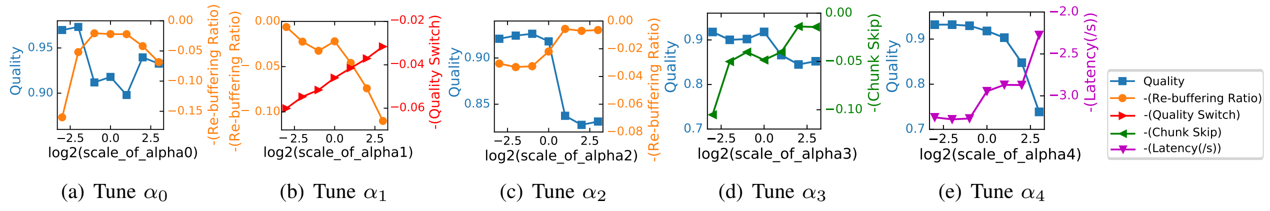


Fig. 7. Ablation study for α_0, α_1, α_2, α_3 and α_4 in the QoE model.

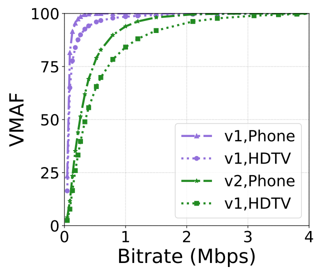


Fig. 8. Relationship between Bitrates and Perceptual Quality (VMAF) for Different Videos and Client Devices

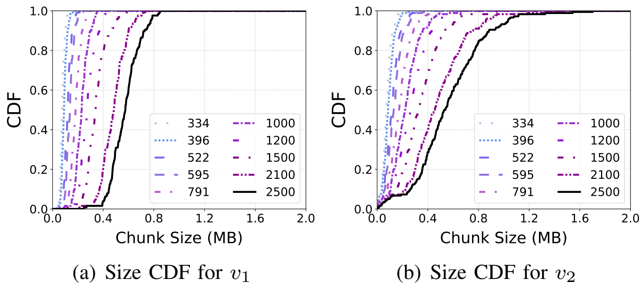


Fig. 9. The Chunk Size CDF of Video Type v_1 and Video Type v_2

TABLE V
AVERAGE QOE AND UNFAIRNESS OF DIFFERENT INPUT VERSIONS

Input version	Average QoE		Unfairness
	Phone	HDTV	
QAVA_K	0.728	0.599	0.132
QAVA_O	**0.764**	**0.676**	**0.129**

effectively decreases the quality switch while increasing the re-buffering ratio. In this paper, we choose a weight combination that has a balanced performance on each QoE metrics. Clients and CPs could tune the weights according to their preferences to suit various scenarios.

F. Experimental Results and Analysis

In this section, we compare the overall performance of all the considered approaches. Our test video dataset mainly contains two types of videos: v_1 with mostly static scenes and v_2 with mostly dynamic scenes. Using the same encoding method (H.264) and specifying the same bitrate, the perceptual quality and size distribution of the two types of videos are quite different, which is illustrated in Fig. 8 and Fig. 9. As is shown in Fig. 8, v_1 generally has higher perceptual quality comparing with v_2 if they are encoded at the same bitrate. Meanwhile, the size distribution of chunks encoding into a certain bitrate in video type v_2 has a higher variance than v_1. In the future sections, if not specified, we use the FCC trace

TABLE VI
PERFORMANCE UNDER DIFFERENT ACTIVATION FUNCTIONS

Activation function	Average QoE		Unfairness
	Phone	HDTV	
ALL ELU	0.666	0.588	0.153
ALL RELU	0.698	0.598	0.161
RELU+ELU	**0.764**	**0.676**	**0.129**

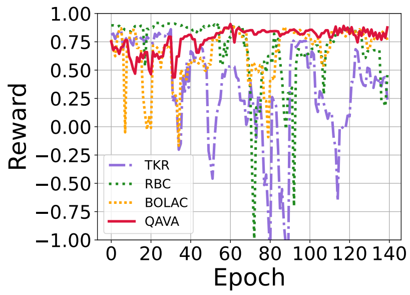


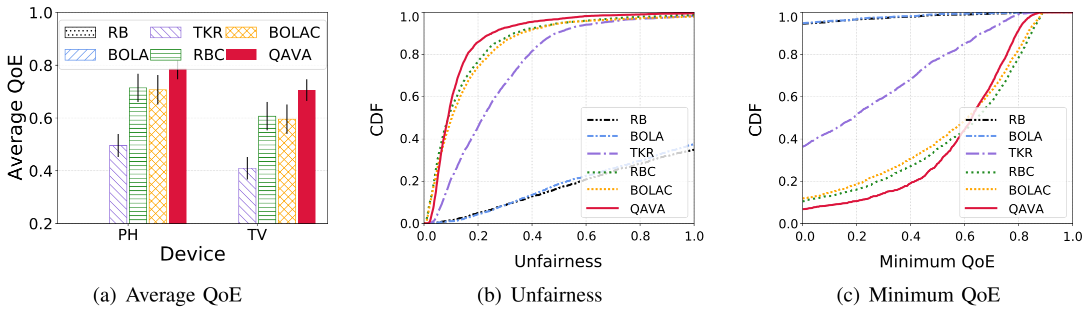
Fig. 10. Comparison between QAVA and Other Methods in Terms of Reward.

to simulate the bottleneck bandwidth and set the chunk skip tolerant factor P to 0. To facilitate the comprehension of the experiments, Table.III summarizes the default values of key parameters in the experiment implementation.

1) Exploring the DRL model: To improve the utilization of bottleneck bandwidth, QAVA makes bitrate adaptive decisions based on predicted perceptual quality rather than bitrate.

Different versions for predicted quality are employed and the best two are selected. The first version uses the K-means method to convert the 10-dimensional predicted values into three categories based on their minimum, maximum, average and standard deviation values, and then inputs them into a fully connected layer. The second version inputs all predicted quality values into the same fully connected layer. We call the DRL model trained through the two input versions as QAVA_K and QAVA_O, respectively. V shows the average QoE and unfairness values generated by the two models. The results show that although the average unfairness values of the two versions are almost the same, the resulting QoE of QAVA_O is higher than that of QAVA_K by 4.95% and 12.85%, respectively, which implies that using the original 10-dimensional predicted values can better mine video characteristics. Therefore, the second version is used.

Additionally, to further increase the classification accuracy for video characteristics, ELU [45] is employed as the activation function for the neurons used to classify the predicted quality values and this model is called as RELU+ELU. Generally, all activation functions of a NN are RELU, and we call this model as ALL RELU. Another model that employs

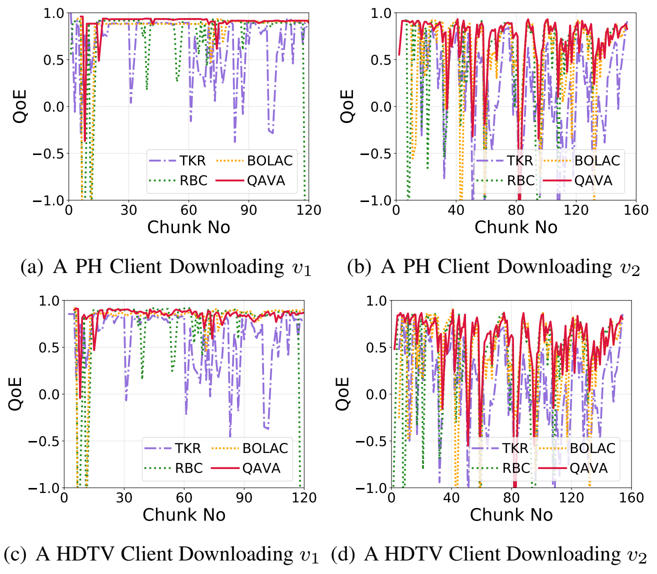


(a) Average QoE (b) Unfairness (c) Minimum QoE

Fig. 11. Comparing QAVA with Existing Methods on Average QoE, Unfairness and Minimum QoE

TABLE VII
COMPARING QAVA WITH EXISTING ALGORITHMS ON DETAILED QOE METRICS WHEN USING PHONE (PH) AND HDTV (TV).

Metric / Method	Quality		Quality Switch		Re-buffering Ratio		Chunk Skip		Latency(/s)	
	PH	TV	PH	TV	PH	TV	PH	TV	PH	TV
RB	0.778	0.652	0.133	0.129	0.264	0.267	1.771	1.688	6.542	6.378
BOLA	0.774	0.691	0.127	0.114	0.262	0.267	1.656	1.757	6.350	6.526
TKR	0.785	0.702	0.082	0.080	0.037	0.036	0.085	0.088	3.323	3.314
RBC	0.948	0.866	**0.032**	**0.055**	0.036	0.035	0.081	0.084	3.289	3.274
BOLAC	0.933	0.844	0.040	0.064	0.028	0.029	0.062	0.062	3.451	3.425
QAVA	**0.957**	**0.879**	0.034	0.058	**0.020**	**0.024**	**0.054**	**0.043**	**2.950**	**2.941**



(a) A PH Client Downloading v_1 (b) A PH Client Downloading v_2

(c) A HDTV Client Downloading v_1 (d) A HDTV Client Downloading v_2

Fig. 12. Dynamic QoE for A Single Client When Downloading A Video

ELU as all activation functions is called ALL ELU. We make a comparison among DRL models with RELU+ELU, ALL RELU and ALL ELU activation functions, and the results are shown in Table VI. The model with ALL RELU function classifies video characteristics with lower accuracy, which wastes the shared bandwidth and results in low average QoE and fairness. Meanwhile, the model with ALL ELU is not stable, so it often requests too high bitrates and results in frequent re-buffering events. Therefore ELU only is used as the activation function for the predicted quality neurons here.

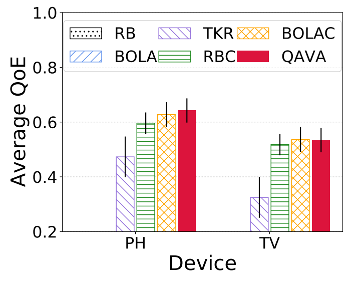
In order to better understand the learning process of the DRL agent, we visualize the reward curves of four considered methods in Fig.10. Note that as the reward values for the RB and BOLA methods are always at a very low level, we show the reward curves for RBC and BOLAC instead and allow the QAVA reward value curves to be seen more clearly. As is mentioned in section VII-A, we pre-train the DRL model based on 40-hours of data so that the reward of QAVA has a high value at the beginning of the deployment. Then, as the interaction with the real environment continues, the reward value of QAVA gradually increases until it plateaus. As shown in Fig.10, QAVA has the most stable trend compared to the other methods. It indicates that QAVA can adapt well to the dynamic changes in clients, videos, and networks.

2) General Performance: Fig. 11(a) shows the average QoE of each method. The results show two key points.

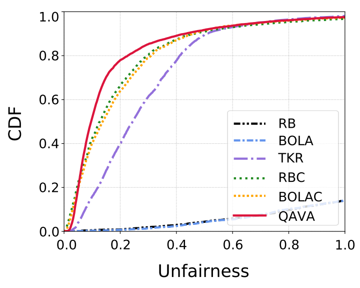
On the one hand, we find that the existence of the edge node with cache does bring huge benefits. Since RB and BOLA fetch chunks directly from the IDC/CDN servers, the client is unaware of other client requests for the same chunk, which results in large redundant transmissions and a sharp drop in QoE. In contrast, strategies with the edge node greatly reduce redundant transmissions and improve client QoE when the bottleneck bandwidth resources are insufficient.

On the other hand, we note that clients using QAVA achieve higher QoE than those of other methods. On average, QAVA outperforms TKR, RBC and BOLAC by 54.03%, 7.00% and 8.22% respectively when using a phone. QAVA also outperforms TKR, RBC and BOLAC by 64.88%, 11.37% and 13.42% respectively when using HDTV. The results show that QAVA can exploit the difference in perceptual quality among online videos to make appropriate adaptive bitrate aggregation decisions, which makes more rational use of available bandwidth resources.

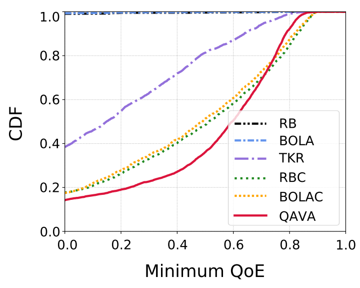
3) Detailed QoE Metrics Analysis: To deeply explore the reason why QAVA improves client QoE, the performance analysis of each strategy is presented in Table VII with detailed QoE metrics. When multiple clients compete for limited bandwidth resources, RB and BOLA send a large number of redundant requests due to the absence of the edge



(a) Average QoE

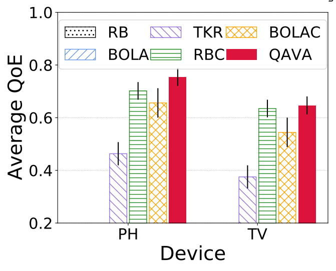


(b) Unfairness

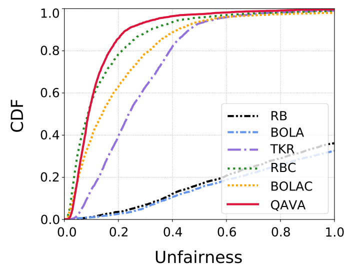


(c) Minimum QoE

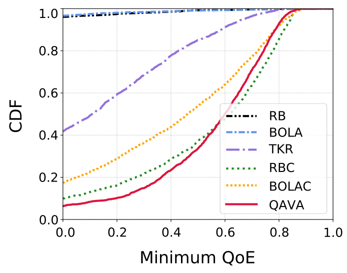
Fig. 13. Comparison of the Performance Achieved by QAVA and Existing Algorithms on Average QoE, Unfairness and Min QoE Using the HSDPA Dataset



(a) Average QoE



(b) Unfairness



(c) Minimum QoE

Fig. 14. Comparison of the Performance Achieved by QAVA and Existing Algorithms on Average QoE, Unfairness and Min QoE when $P = 1$

node, leading to mass re-buffering and chunk skip events and then resulting in a significant drop in QoE. Since TKR does not consider the simultaneous presence of multiple videos and the dynamic change of available bandwidth, its probing mechanism often fails to download high bitrate chunks due to bandwidth competition, resulting in relatively low average quality compared with other strategies with the edge node. Thanks to the advantages of the DRL framework, QAVA makes appropriate bitrate decisions according to the predicted perceptual quality, which increases the average QoE and QoE fairness among online clients, compared with other strategies. As shown in Table VII, QAVA not only increases the average perceptual quality, but also results in fewer negative effects on client QoE (i.e., re-buffering events, chunk skip events, and latency). The quality switch of QAVA is slightly higher than that of RBC, but it is still acceptable.

4) Unfairness and Minimum QoE Analysis: As shown in Fig. 11(b), QAVA has the best fairness compared with other algorithms. The average unfairness of QAVA over RB, BOLA, TKR, RBC and BOLAC has decreased by 99.19%, 98.66%, 52.57%, 19.37% and 31.38%, respectively. Fig. 11(c) shows that QAVA reduces the risk of clients being affected by low QoE. Clients have 28.36%, 35.42% and 38.97% possibility of having QoE level below 0.5 when using QAVA, RBC and BOLAC, respectively. It can be said that by making efficient use of limited bandwidth resources, QAVA guarantees fairness and maintains QoE for all online clients at a relatively high level, reducing the probability of clients stop watching videos.

5) Single Client Performance: Fig. 12 shows the dynamic changes of QoE for a single client when downloading a video. QAVA outperforms TKR, RBC and BOLAC by 27%, 8% and 5%, respectively on average QoE. The QoE standard deviation of QAVA is lower than for TKR, RBC and BOLAC by 29%, 38% and 23%, respectively. These results show that the QAVA clients benefit from high bandwidth utilization and have a higher and more stable QoE compared with other strategies.

6) Performance on the HSDPA Trace: To test the generality of the pre-trained DRL model, we use the HSDPA trace to simulate the bottleneck bandwidth, and the results are shown in Fig. 13. Compared with the FCC trace, network throughput in the HSDPA trace is smaller and changes more dramatically. We find that QAVA still maintains good performance when using the HSDPA trace. Compared with other methods, QAVA still improves QoE fairness and minimum QoE without decreasing QoE.

7) Performance when Changing the Value of P: As mentioned in Section III-C, QAVA can adjust the chunk skip tolerant factor P to a higher value to improve the smoothness of video playback. An experiment with $P = 1$ is run and the results are shown in Fig. 14. Fig. 14 shows that QoE using QAVA method is superior to those obtained when using other methods, and QAVA also greatly improves fairness and minimum QoE, which is similar to the results of $P = 0$.

8) Testing the overhead: Compared to other solutions, QAVA introduces QPM and ADM, potentially introducing additional time overhead. Therefore, to demonstrate that the time overhead introduced by these two modules can meet the time constraints required by the system, we measure the decision time of these two modules. According to the results, 98% of bitrate aggregation decisions are under 2ms, while 95% of the decisions of quality prediction are under 25ms. Therefore, in general, the use of QAVA introduces a 20-30ms delay, which is much smaller than the duration of the video chunk. Meanwhile, as shown in Table VII, the chunk skip and latency of QAVA are smaller than those of other methods. It implies that the time overhead saved by QAVA is greater than the time overhead it introduces.

At the same time, as described in section III-B, the smart edge collects information from the online clients in real-time. The frequent information interaction between the clients and the edge may cause additional bandwidth overhead. To verify whether the bandwidth overhead caused by the additional information interactions affects the performance of QAVA, we let 20 clients request a video with the minimum bitrate (i.e., 334 Kbps) simultaneously. We find that the video size is nearly one thousand times larger than the sum of all client information interaction data sizes. Therefore, the additional bandwidth overhead caused by the interaction between the clients and the smart edge does not impact the performance of QAVA.

VIII. Discussions

The design and results presented in this paper demonstrate that QAVA can intelligently realize QoE-aware video bitrate aggregation by integrating network conditions, clients' state, and video characteristics at the smart edge. It effectively reduces the congestion on the backhaul network and thus improves average QoE, QoE stability, and QoE fairness among multiple clients. However, the current design still has some limitations.

First, QAVA cannot offer differentiated services for clients with different priorities. For some CPs, clients can be subscribers on different price tiers of a streaming service. Under this circumstance, CPs should perform the aggregation on a per-tier basis. The bitrate aggregation agent should make decisions for each tier, respectively. From the perspective of a business strategy, to obtain increased benefits, different QoE weights can be set to clients of different tiers to achieve flexible QoE control between diverse tier clients. Solving this problem involves complex network economic models and client behavior pattern analysis, and it is a challenging task.

Secondly, the QoE model used by QAVA may benefit from further improvement. In this paper, the number of skipped chunks is taken as an essential dimension of QoE. However, for different video content, the impact of skipping the same number of chunks on QoE is different. For instance, for low-dynamic videos, clients are more tolerant of the number of skipped chunks than for high-dynamic videos. Therefore, in the future, QAVA could use more fine-grained video features to predict the impact of different skipped chunks on clients' QoE. This may further enhance the performance of the QoE-aware video bitrate aggregation.

Thirdly, unfortunately, as it is designed, QAVA cannot be applied directly to some emerging streaming applications, such as Augmented Reality (AR) and Virtual Reality (VR). For instance, in VR applications there are multiple tiles of video content streamed at a time. Different tiles have different quality and latency requirements. In order to realize smart QoE-aware adaptive bitrate aggregation strategies for these emerging applications, we need to fine-tune the aggregation mechanism and consider aspects relevant to specific applications. However, QAVA provides a reference solution for designing other smart bitrate delivery mechanisms for distribution of multimedia and diverse other rich media content in edge-enhanced networks.

IX. Conclusions and Future Work

This paper proposes QAVA, a novel QoE-aware Adaptive Video bitrate Aggregation solution based on Deep Reinforcement Learning (DRL) for improving the efficiency of live video streaming. QAVA utilizes the network perception, storage and computing power of the edge nodes and intelligence of DRL to aggregate client requests and adapt the bitrates based on network conditions and client states as well as video characteristics. When comparing QAVA with several state-of-the-art bitrate adaption algorithms based on the edge node, QAVA has improved with 19%-52% QoE fairness among online clients and has achieved high average client QoE levels. In the future, we expect that QAVA can be applied to emerging applications in an innovative 5G scenario, including the delivery of Augmented Reality (AR) and/or Virtual Reality (VR) content. The size of this data is much larger than that of traditional video, and there are multiple tiles of video content delivered at the same time. Different tiles have different quality and latency requirements. In our future work, we will personalize QoE-aware adaptive aggregation strategies for these emerging applications, taking advantage of the large number of edge nodes expected to be available in 5G network environments.

Acknowledgment

This work is supported by the Key-Area Research and Development Program of Guangdong Province under grant no. 2020B0101130006, the National Natural Science Foundation of China under grant no. 61972189, the Shenzhen Key Lab of Software Defined Networking under grant no. ZDSYS20140509172959989 and the Social Science Fund of Shenzhen under grant no. SZ2020D009. G.-M. Muntean acknowledges the Science Foundation Ireland (SFI)'s support via grant no. 12/RC/2289_P2 for the Insight SFI Centre for Data Analytics.

References

[1] "Top 115 live streaming statistics every broadcaster should know," https://www.dacast.com/blog/66-must-know-live-streaming-statistics/, 2022.

[2] J. Song, F. Yang *et al.*, "QoE evaluation of multimedia services based on audiovisual quality and user interest," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 444–457, Jan. 2016.

[3] Cisco, "Cisco Visual Networking Index: Forecast and methodology 2017–2022," 2019.

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

[5] 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, Jan. 2016.

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

[7] L. Zou, R. Trestian, and G.-M. Muntean, "E^{3}doas: Balancing QoE and energy-saving for multi-device adaptation in future mobile wireless video delivery," *IEEE Transactions on Broadcasting*, vol. 64, no. 1, Mar. 2018.

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

[9] D. Mi, J. Eyles *et al.*, "Demonstrating immersive media delivery on 5g broadcast and multicast testing networks," *IEEE Transactions on Broadcasting*, vol. 66, no. 2, pp. 555–570, 2020.

[10] A. Yaqoob and G.-M. Muntean, "A combined field-of-view prediction-assisted viewport adaptive delivery scheme for 360° videos," *IEEE Transactions on Broadcasting*, vol. 67, no. 3, pp. 746–760, 2021.

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

[12] I.-S. Comşa, G.-M. Muntean, and R. Trestian, "An innovative machine-learning-based scheduling solution for improving live uhd video streaming quality in highly dynamic network environments," *IEEE Transactions on Broadcasting*, vol. 67, no. 1, pp. 212–224, 2021.

[13] W. Shi, J. Cao *et al.*, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

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

[15] Qwilt, "Open edge cloud," 2019. [Online]. Available: https://qwilt.com/

[16] S. Sundaresan, W. de Donato *et al.*, "Broadband internet performance: a view from the gateway," in *Proc. ACM SIGCOMM*, 2011, pp. 134–145.

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

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

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

[20] 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.

[21] G.-M. Muntean and N. Cranley, "Resource efficient quality-oriented wireless broadcasting of adaptive multimedia content," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 362–368, 2007.

[22] 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.

[23] A.-N. Moldovan and C. H. Muntean, "Dqamlearn: Device and qoe-aware adaptive multimedia mobile learning framework," *IEEE Transactions on Broadcasting*, vol. 67, no. 1, pp. 185–200, 2020.

[24] G. Cofano, L. D. Cicco *et al.*, "Design and experimental evaluation of network-assisted strategies for HTTP adaptive streaming," in *Proc. ACM MMSys*, 2016, pp. 3:1–3:12.

[25] A. Bentaleb, A. C. Begen *et al.*, "SDNHAS: An sdn-enabled architecture to optimize QoE in HTTP adaptive streaming," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2136–2151, Jul. 2017.

[26] 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, Nov. 2018.

[27] A. Detti, B. Ricci, and N. Blefari-Melazzi, "Tracker-assisted rate adaptation for MPEG DASH live streaming," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

[28] K. J. Ma and R. Bartos, "HTTP live streaming bandwidth management using intelligent segment selection," in *Proc. IEEE GLOBECOM*, 2011, pp. 1–5.

[29] X. Ma, Q. Li *et al.*, "Steward: Smart edge based joint QoE optimization for adaptive video streaming," in *Proc. ACM NOSSDAV*, 2019, pp. 31–36.

[30] A. Zhang, Q. Li *et al.*, "Video super-resolution and caching—an edge-assisted adaptive video streaming solution," *IEEE Transactions on Broadcasting*, vol. 67, no. 4, pp. 799–812, 2021.

[31] W. Shi, C. Wang *et al.*, "Coleap: Cooperative learning-based edge scheme with caching and prefetching for dash video delivery," *IEEE Transactions on Multimedia*, vol. 23, pp. 3631–3645, 2020.

[32] A. Al-Jawad, I.-S. Comşa *et al.*, "An innovative reinforcement learning-based framework for quality of service provisioning over multimedia-based sdn environments," *IEEE Transactions on Broadcasting*, vol. 67, no. 4, pp. 851–867, 2021.

[33] I.-S. Comşa, A. Molnar *et al.*, "A machine learning resource allocation solution to improve video quality in remote education," *IEEE Transactions on Broadcasting*, vol. 67, no. 3, pp. 664–684, 2021.

[34] Z. Zhang, Y. Yang *et al.*, "Proactive caching for vehicular multi-view 3d video streaming via deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 18, no. 5, pp. 2693–2706, May. 2019.

[35] R. Zhang, T. Huang *et al.*, "Enhancing the crowdsourced live streaming: a deep reinforcement learning approach," in *Proc. ACM NOSSDAV*, 2019, pp. 55–60.

[36] H. Yeo, Y. Jung *et al.*, "Neural adaptive content-aware internet video delivery," in *Proc. USENIX OSDI*, 2018, pp. 645–661.

[37] S. Chinchali, P. Hu *et al.*, "Cellular network traffic scheduling with deep reinforcement learning," in *Proc. AAAI*, 2018, pp. 766–774.

[38] H. Mao, M. Alizadeh *et al.*, "Resource management with deep reinforcement learning," in *Proc. ACM HotNets*, 2016, pp. 50–56.

[39] L. Chen, J. Lingys *et al.*, "Auto: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proc. ACM SIGCOMM*, 2018, pp. 191–205.

[40] X. Yin, A. Jindal *et al.*, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM SIGCOMM*, 2015, pp. 325–338.

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

[42] AITrans, "Global intelligent network transmission competition," 2018.

[43] S. de Jong, K. Tuyls, and K. Verbeeck, "Artificial agents learning human fairness," in *Proc. IFAAMAS AAMAS*, 2008, pp. 863–870.

[44] V. Mnih, A. P. Badia *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. JMLR ICML*, 2016, pp. 1928–1937.

[45] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *Proc. ICLR*, 2016, p. arXiv preprint arXiv:1511.07289.

[46] 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/

[47] M. Abadi, P. Barham *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. USENIX OSDI*, 2016, pp. 265–283.

[48] ITEC, "DASHDataset2014," 2014. [Online]. Available: http://ftp.itec.aau.at/datasets/DASHDataset2014/

[49] TKN, "Yuv video sequences," 2015. [Online]. Available: http://www2.tkn.tu-berlin.de/research/evalvid/cif.html/

[50] "Tencent qq music," 2020. [Online]. Available: https://y.qq.com/

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

[52] HSDPA, "DATASET: HSDPA-bandwidth logs for mobile HTTP streaming scenarios," 2012. [Online]. Available: http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/

[53] Z. Wang, A. C. Bovik *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

Xiaoteng Ma received the B.Eng. degree in electrical engineering from the Huazhong University of Science and Technology-China 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 with Peng Cheng Laboratory, Shenzhen, China. His research interests include reliable and scalable routing of the Internet, in-network caching/computing, intelligent self-running network, edge computing, etc.

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.

Junkun Peng received the B.S. degree (2015) from Shanghai University, Shanghai, China, in information management and information system. He is currently a Master student with Tsinghua University in computer technology. His research interests include in-network caching/computing, swarm artificial intelligence, deep reinforcement learning, smart drones/robots.

Jianer Zhou is now a research associate professor at Southern University of Science and Technology, Shenzhen China. He received his PhD degree from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS) in 2016. His research interests lie in network performance, future Internet architecture and Internet measurement.

Jimeng Chai received the B.Eng. degree (2017) and Master degree (2020) from Beijing University of Post and Telecommunications, Beijing, China and Tsinghua University, Beijing, China, respectively. Her research interests include edge-assisted multimedia delivery and adaptive multimedia.

Yong Jiang is currently a Full Professor with Tsinghua Shenzhen International Graduate School, China. 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 (S'02-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 received the PhD degree from the DCU School of Electronic Engineering-Ireland for his research on quality-oriented adaptive multimedia streaming in 2003. Prof. Muntean has published over 450 papers in prestigious international journals and conferences, has authored four books and 25 book chapters, and has edited 7 other books. His research interests include quality-oriented and performance related issues of adaptive multimedia delivery, performance of wired and wireless communications, energy-aware networking, and personalized technology-enhanced learning. Prof. Muntean is an Associate Editor of the IEEE Transactions on Broadcasting, the Multimedia Communications Area Editor of the IEEE Communication Surveys and Tutorials, and a reviewer for other important international journals, conferences, and funding agencies. He is a senior member of IEEE and IEEE Broadcast Technology Society.