Adaptive Video Streaming With Automatic Quality-of-Experience Optimization

Guanghui Zhang, Jie Zhang, *Member, IEEE*, Yan Liu, Haibo Hu, *Senior Member, IEEE*, Jack Y. B. Lee, *Senior Member, IEEE*, and Vaneet Aggarwal, *Senior Member, IEEE*

Abstract—Video streaming has grown tremendously in recent years and it is now one of the main applications on the Internet. Due to the networks' inherent bandwidth fluctuations, various rate-adaptive streaming algorithms have been developed to compensate for such fluctuations to improve Quality-of-Experience (QoE). However, in practice, the preference for QoE typically differs significantly across different viewers and there is no systematic way so far to comprehensively incorporate different sets of conflicting QoE objectives into the algorithm design. Thus, it is not surprising that the QoE performance achieved by the existing algorithms is in fact far from optimal. This work aims at attacking the heart of the problem by developing a novel framework called Post Streaming Quality Analysis (PSQA) that can maximize the QoE under *any* preference through automatically tuning the adaptation logic of the streaming algorithms. Evaluation results show that the QoE achieved by PSQA is substantially better than the existing approaches and in some scenarios even close to optimal. Moreover, PSQA can be readily implemented into real streaming platforms, offering a practical and reliable solution for high-performance streaming services.

Index Terms—Video streaming, quality-of-experience, DASH, video reliability

1 Introduction

Without a doubt, video streaming is one of the fastest-growing applications on the Internet. A report by Cisco [1] estimated that the global video streaming traffic will increase 15-fold from 2017 to 2022, accounting for 82% of all Internet traffic by 2022.

Given that the Internet does not offer any bandwidth guarantees, the primary focus of the streaming vendors nowadays is the development of adaptive streaming systems to compensate for the inherent bandwidth fluctuations. The core is to design intelligent bitrate adaptation algorithms to dynamically adjust the video quality (bitrate) in the light of past measurements (e.g., throughput, buffer occupancy, etc.), such that Quality-of-Experience (QoE) can be maximized.

In practice, however, QoE is influenced by many factors, which thus creates significant differences in QoE preference among the viewers [2]. First, different types of streaming services emphasize different QoE metrics. For example, the main focuses in on-demand streaming are video quality and playback rebuffering, while in live streaming, playback latency becomes the most essential one. Second, the properties of the last-hop network can affect the direction of the QoE optimization. For instance, in mobile networks where rapid and substantial bandwidth fluctuations are normal, smooth video playback (i.e., fewer rebuffering events) is more desirable, whereas in stable broadband networks, achieving high video quality is more likely to satisfy the viewers; Third, the playback device affects the viewer's perceptual video quality. For example, playing a video with the same bitrate, the video quality perceived on a large screen device (e.g., PC monitor) is much lower than that on a small screen one (e.g., smartphone).

In addition to the aforementioned points, there are many other potential factors that can influence QoE, such as video content, subscription plan, and so on. Therefore, this poses a fundamental problem – how to design a streaming algorithm such that the resulting QoE is satisfactory to all viewers? In recent years, although many sophisticated adaptive streaming algorithms have been proposed (e.g., [3], [4], [5], [6], [7], [8], [9], [38], [39], [40], [41], [42], [43]), most (if not all) of them only optimized the QoE towards one certain

- *Guanghui Zhang is with the Department of Computer Science, Hong Kong Baptist University, Kowloon 999077, Hong Kong, and also with the Centre for Advances in Reliability and Safety (CAiRS), Pak Shek Kok, NT 999077, Hong Kong. E-mail: ghzhang@link.cuhk.edu.hk.*
- *Jie Zhang is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei 230026, China, and also with the State Key Laboratory of Acoustics, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China. E-mail: jzhang6@ustc.edu.cn.*
- *Yan Liu is with Cloud ARCH & Platform Dept., Tencent, 518000, China. E-mail: rockyanliu@tencent.com.*
- *Haibo Hu is with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon 999077, Hong Kong, and also with the Shenzhen Research Institute, The Hong Kong Polytechnic University, Shenzhen 999077, China. E-mail: haibo.hu@polyu.edu.hk.*
- *Jack Y. B. Lee is with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, NT 999077, Hong Kong SAR. E-mail: yblee@ie.cuhk.edu.hk.*
- *Vaneet Aggarwal is with the School of Industrial Engineering, Purdue University, West Lafayette, IN 47907 USA, and also with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA. E-mail: vaneet@purdue.edu.*

Manuscript received 20 Oct. 2021; revised 24 Jan. 2022; accepted 17 Mar. 2022. Date of publication 23 Mar. 2022; date of current version 30 June 2023.
This work was supported in part by the Centre for Advances in Reliability and Safety (CAiRS) admitted under AIR@InnoHK Research Cluster. The work of Jie Zhang was supported in part by the National Natural Science Foundation of China under Grant 62101523 and in part by the Fundamental Research Funds for the Central Universities.
(Corresponding author: Jie Zhang.)
Digital Object Identifier no. 10.1109/TMC.2022.3161351

QoE preference. For example, Microsoft's Smooth Streaming exhibits very conservative behavior in its bitrate selection to keep smooth playback regardless of the video quality loss, while the Netflix player implements an aggressive rate-adaptation algorithm to favor high-quality videos but at the expense of frequent quality switches [34]. While the state-of-the-art learning-based algorithms (e.g., Pensieve [7]) have the ability to train the adaptation logics based on an arbitrary given QoE objective, due to the unawareness for QoE preference, the streaming vendor merely trains one single logic according to a certain preference and then keep the logic unmodified after its deployment [7]. As a result, these existing algorithms often lead a wrong path for the viewers' actual QoE preference, leading to the resultant QoE being far less than optimal.

To address this problem, this work develops a unified framework called Post Streaming Quality Analysis (PSQA), which incorporates different sets of conflicting QoE objectives to offer a self-adaptive system for automatic QoE optimization. PSQA provides two options: First, if a viewer is aware of their QoE preference explicitly, he/she can specify the preference before/during online streaming through an option list of the video player, then PSQA will automatically optimize/tune the adaptation logic based on the corresponding QoE objective to maximize the QoE for the given preference. Second, for cases where the viewer is clueless about the QoE preference choice, PSQA will enable a generic adaptation logic. This logic is exposed to all available QoE objectives during its training to avoid over-specialization to any particular environment, such that it can be broadly applicable to various environments without requiring the viewer to specify the preference. To the best of our knowledge, PSQA is the first streaming framework so far that can thoroughly address the QoE preference problem.

Extensive evaluations showed that PSQA has three aspects of superiority. First, PSQA breaks through the barriers to the algorithm design under different QoE objectives, so that the QoE performance achieved by PSQA is substantially better than the existing approaches, and even close to optimal in some scenarios. Second, PSQA is a general framework which is to complement rather than replace the existing streaming algorithms (i.e., only tuning their adaptation logics while keeping their streaming workflows intact). This offers an immediate and ready solution for the streaming platforms already in service. Last but not least, PSQA not only has QoE-optimization awareness, but also opens a new paradigm to network-optimized adaptive streaming. This enables PSQA to have strong robustness and achieve consistently better performance across a wide range of network environments.

The rest of the paper is organized as follows. Section 2 reviews the background and related works; Section 3 demonstrates the limitations of the existing adaptive streaming algorithms; Section 4 presents PSQA framework and applies it to optimizing the existing algorithms; Section 5 evaluates PSQA using trace-driven simulations; Section 6 reports real experimental results from a prototype implementation; and Section 7 summarizes the study and outlines some future works.

2 Background and Related Works

In this section, we first review the studies on the methodology of QoE quantification, and then review the state-of-the-art adaptive streaming algorithms.

2.1 Quality of Experience

In video streaming, Quality-of-Experience (QoE) is a notion to quantify the goodness of the viewing experience perceived by the viewers, and it is the most important metric to evaluate the efficacy of a streaming system. In general, QoE is composed of a series of QoE metrics, such as video quality [10], the frequency of video quality switches [11], the magnitude of video quality switches [11], startup delay (waiting time for playback to commence) [12], the frequency of playback rebuffering [13], the duration of playback rebuffering [13], playback latency [14], and so on.

Existing studies typically employed *QoE function*, i.e., a combination of different QoE metrics, to quantify the overall QoE performance where the coefficients (weights) in the function reflect the impact of each QoE metric. Many different formats of QoE functions have been proposed in recent years, such as weighted sum [67,14,15], exponential/logarithmic [16], [17], threshold-based table look-up [13], decision tree [18], etc. The value of the coefficients in the QoE function was usually configured by two common methods. For example, Mok *et al.* [13] and Liu *et al.* [16] employed subjective experiments carried out by real human subjects while Dobrian *et al.* [12] used a crowd-sourcing method to access the effects on user viewing engagement.

Overall, QoE function is a good tool for quantifying the QoE performance achieved by the streaming systems, so it has been widely adopted to guide the design and optimization of streaming algorithms.

2.2 Adaptive Streaming Algorithms

In recent years, much work has been done in the area of video streaming. Beginning with non-adaptive streaming, the industry soon realized that the inherent bandwidth fluctuations posed significant challenges to the bandwidth-sensitive streaming services [33]. This led to intense research in recent years on the design of adaptive streaming systems. The principle is to divide a video into a series of fixed-duration segments (e.g., each a few seconds) and then encode them with multiple target bitrate versions. The client implements a bitrate adaptation algorithm that monitors the network condition (e.g., via measured throughput) and then dynamically selects the best bitrate version for downloading future video segments. This not only enables the client to stream videos across a wide range of networks with different bandwidth limits, but also enables it to adapt to short-term bandwidth fluctuations that commonly exist in mobile networks [27]. The above led to the Dynamic Adaptive Streaming over HTTP standard (or simply known as DASH [19]) proposed by the MPEG standard committee.

To support the development of DASH, in recent years, many adaptive bitrate streaming (ABR) algorithms have been proposed. A detailed review of the existing work is beyond the scope of this paper. We refer the interested readers to the recent survey by Kua *et al.* [20] and Bentaleb *et al.*

[21] for the details. In the following, we briefly review some state-of-the-art studies.

First, many adaptation algorithms were developed based on human intuitions. For example, Jiang *et al.* [4] proposed FESTIVE which uses harmonic mean throughput over downloading past segments to decide the video bitrate. Spiteri *et al.* [5] devised an online control algorithm called BOLA which employs Lyapunov optimization to adapt the bitrate according to the buffer occupancy. Yin *et al.* [6] proposed MPC that makes bitrate decisions by solving a QoE maximization problem over a horizon of several future segments. Akhtar *et al.* [9] proposed Oboe that pre-computes the discount factor of MPC according to different network conditions, and then dynamically adjusts the discount factor at runtime to adapt to the network condition changes. Elgabli *et al.* [38] developed FastScan that optimizes QoE through solving a combinatorial optimization problem.

Second, another branch of adaptation algorithm design was built upon machine learning techniques. For instance, Mao *et al.* [7] proposed Pensieve that uses A3C (a deep reinforcement learning algorithm [22]) to train neural networks for bitrate adaptation. Zhang *et al.* [8] developed EAS-GP that employs genetic programming [23] to automatically evolve an adaptation algorithm ensemble for different network conditions. Zhao *et al.* [40] devised L2AC which also uses A3C, but in addition to bitrate adaptation, it also contains a playback rate adaptation logic specifically for achieving ultra-low playback latency in live video streaming.

In practice, however, an important fact is that the viewer's QoE preferences typically differ significantly under different streaming scenarios, e.g., some prefer high video quality while others expect less rebuffering, and the detailed QoE preference of each viewer is difficult to look into due to its diversity. Thus, the streaming vendor nowadays typically designs/trains the adaptation algorithms based on a limited set of QoE objectives, and then keeps the algorithm unmodified after deployments. As a result, it is not surprising that the QoE performance would exhibit substantial degradation and variation under different QoE preferences. To tackle this problem, this work develops the novel PSQA framework that can relate an arbitrary streaming algorithm to an arbitrary QoE objective, and then automatically optimize the algorithm's performance with respect to the given QoE objective.

2.3 Relation to an Early Version of PSQA

An early version of PSQA was reported in [24]. This study extends the earlier work in four significant aspects. First, adaptive streaming algorithms can be classified into two categories, namely heuristic-based and learning-based, and their techniques for generating the adaptation logic are different. In contrast to our earlier work only targeting optimizing the heuristic-based algorithms, we extended the scope in this work to support the learning-based ones.

Specifically, we evaluated the state-of-the-art learning-based algorithms, i.e., Pensieve [7], EAS-GP [8], and L2AC [40], and found that they exhibit far more substantial QoE variations over different streaming environments than the heuristic ones (c.f. Section 3.2). This motivates us to extend PSQA to optimize them, where PSQA utilizes the inborn offline training of the learning techniques to automatically generate a set of candidate adaptation logics based on different QoE functions, and then dynamically adjusts the logics online to cater to the diverse QoE preference (c.f. Section 4.3). Overall, this extension enables PSQA to be generalized enough to function on almost all the state-of-the-art streaming algorithms.

Second, one hard requirement of the earlier work is to require the viewers to input their QoE preference. However, in practice, it is likely that the viewer is a layperson of streaming so that he/she has no clue about the QoE preference choice. To this end, we extended PSQA to generate a generic adaptation logic. This logic is exposed to all available QoE objectives during its training to avoid over-specialization to any particular environment, such that it can be broadly applicable to various environments without specifying any preference (c.f. Sections 4.3 and 5.2). This extension effectively eliminates the obstacles to the QoE optimization under different streaming configurations.

Third, the experiments and performance evaluations have been expanded substantially in this work. While our earlier work already employed four different QoE functions for evaluations, this work further expanded the scope to include three new ones, i.e., by Mao *et al.* [7], Liu *et al.* [16], and Yi *et al.* [14], which favor high video quality and low playback latency. Furthermore, we also added five new state-of-the-art on-demand/live streaming algorithms, i.e., BOLA [5], MPC [6], Pensieve [7], EAS-GP [8], and L2AC [40], for comparison. Overall, the far broader range of evaluations enables us to obtain a better understanding for the algorithms' behaviors under different QoE objectives (c.f. Sections 3 and 5).

Last but not least, we implemented a prototype of the PSQA framework with dash.js [25] and reported the real experimental results in Section 6. The results verify the feasibility of PSQA for deployment in today's real streaming platforms and demonstrate its potential performance gains in practical operational environments.

3 Video Streaming Algorithm Re-Examined

In this section, we evaluate the state-of-the-art adaptive streaming algorithms to demonstrate the QoE preference problem in the existing streaming platform.

3.1 Experiment Setup

To emulate the diversity of the QoE preference in practice, we employed eight *existing* QoE functions from the literature to cover different QoE preferences:

QoE Function 1 (U_1): Developed by Mok *et al.* [13]. This QoE function only includes factors about playback smoothness, so we define the QoE preference as "ultra-smooth playback". Its detailed definition is:

$$U_1 = 4.23 - 0.0672 L_{ti} - 0.742 L_{fr} - 0.106 L_{tr} \tag{1}$$

where L_{ti} denotes startup delay, L_{fr} is rebuffering frequency (defined as the total number of playback suspensions in one streaming session due to client buffer underflow), and L_{tr} is rebuffering duration (defined as the total time consumed at playback suspension).

TABLE 1
The Mapping Between Video Bitrate and h_k

Bitrate (Mbps) $\rightarrow h_k$
0.2→0.78, 0.4→1.22, 0.8→2.11, 1.2→3.0, 2.2→13.4, 3.3→16.7, 5.0→22.2, 6.5→27.1, 8.6→34.0

QoE Function 2 (U_2): Developed by Joskowicz *et al.* [17]. This QoE function merely quantifies the perceptual video quality based on video bitrate, so we define the QoE preference as "ultra-high quality":

$$U_2 = avg\left[4.75 - 4.5e^{-0.77r_k}|\forall k\right] \tag{2}$$

where r_k is the video bitrate selected for video segment k.

QoE Function 3 $\sim$ QoE Function 5 ($U_3 \sim U_5$): Proposed by Yin *et al.* [6]. This QoE function considers multiple conflicting QoE metrics:

$$U_3 = \frac{1}{K}\left(\sum_{k=1}^{K} r_k - \lambda \times \sum_{k=1}^{K-1} |r_{k+1} - r_k| - \mu \times Z_p - \mu_\theta \times Z_\theta\right) \tag{3}$$

where Z_p is the total rebuffering duration in one streaming session, Z_θ is the startup delay, r_k is the bitrate selected for segment k, $|r_{k+1} - r_k|$ represents the quality variations, and K is the total number of segments in one session. There are three sets of configuration options for the component weights, namely, *Balanced*: $\lambda = 1.0$ and $\mu = \mu_\theta = 3.0$, *Penalize Rebuffering*: $\lambda = 1.0$ and $\mu = \mu_\theta = 6.0$, and *Penalize Quality Instability*: $\lambda = 3.0$ and $\mu = \mu_\theta = 3.0$. We denote the three as U_3, U_4, and U_5 respectively, and define the corresponding QoE preference as "balanced", "smooth playback", and "low quality variation".

QoE Function 6 (U_6): Proposed by Mao *et al.* [7]. This QoE function favors high-quality videos, so the QoE preference is defined as "high quality":

$$U_4 = \frac{1}{K}\left(\sum_{k=1}^{K} h_k - \sum_{k=1}^{K-1} |h_{k+1} - h_k| - 8.0 \times Z_p\right) \tag{4}$$

where h_k denotes the reward for video bitrate in segment k (the mapping between h_k and each bitrate version is listed in Table 1), Z_p is the total rebuffering duration, and K is the total number of video segments in one session.

QoE Function 7 (U_7): Proposed by Liu *et al.* [16]. This QoE function is driven by subjective evaluations with real human perception and the QoE preference is defined as "ultra-balanced":

$$U_5 = 100 - I_{ID} - I_{ST} - I_{LV} + 0.15 \times I_{ID} \times \sqrt{I_{ST} + I_{LV}} + 0.82 \times \sqrt{I_{ST} \times I_{LV}} \tag{5}$$

where I_{ID}, I_{ST}, and I_{LV} denote the penalty for startup delay, playback rebuffering, and poor video quality, respectively.

QoE Function 8 (U_8): Proposed by Yi *et al.* [14]. This QoE function incorporates the playback latency concerned in live video streaming, so the QoE preference is defined as "low latency":

TABLE 2
Evaluation Settings

Parameters	Values
Bitrate profile	{0.2, 0.4, 0.8, 1.2, 2.2, 3.3, 5.0, 6.5, 8.6} Mbps
Video duration	Empirical distribution (40s to 600s)
Segment duration	2s
Frame rate	25 fps
Initial video bitrate	0.2 Mbps
Buffer capacity	60s
Video content	700+ real-world commercial videos

$$U_6 = \frac{1}{K}\left(\sum_{k=1}^{K} (2.0 \times r_k - 8.6 \times z_k - \delta \times l_k) - 0.02 \times \sum_{k=1}^{K-1} |r_{k+1} - r_k|\right) \tag{6}$$

where l_k, r_k, and z_k represent playback latency, video bitrate, and rebuffering duration in downloading segment k, respectively. For the coefficient $\boldsymbol{\delta}$, if the latency l_k is lower than 1.1s, then $\boldsymbol{\delta}$ is 0.05, otherwise $\boldsymbol{\delta}$ is 0.1.

To evaluate the QoE performance in realistic network settings, we adopted an open trace-driven simulator developed by Mao *et al.* [7] and the source codes are available at [26]. In our evaluation, we changed a few parameters from their original settings. For example, we adopted the video bitrate profile proposed by Apple [31] where the bitrate ranges from 0.2 to 8.6 Mbps. The video duration follows an empirical distribution extracted from real streaming services [32]. The video segment size is derived from an open dataset [27] that includes a total of 700+ real-world commercial video contents. All the configurations were listed in Table 2.

Moreover, the simulator is executed over TCP throughput trace data obtained from real-world production networks. To cover different kinds of network properties, we used multiple sources of throughput trace [28], [29], which have quite different throughput features with each other due to the various network types (e.g., 3G, 4G, and Wi-Fi), collection locations (e.g., campus, subway, supermarket, etc.), and service providers (e.g., mobile operator). Their detailed statistics are summarized in Table 3. In the rest of the paper, unless stated otherwise, the data in all the trace sources will be incorporated in the evaluation. In Appendix A.1, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2022.3161351, we provide more details about the simulator as well as the TCP throughput trace data.

3.2 Result Analysis

We evaluated six state-of-the-art adaptive streaming algorithms, where three are learning-based: Pensieve [7],

TABLE 3
Statistics for Five Throughput Trace Sources

Features	#1	#2	#3	#4	#5
Mean throughput (Mbps)	5.97	1.21	10.1	3.12	4.43
Variation (CoV)	0.44	0.83	0.52	0.77	0.58
Network type	3G	3G	LTE	Wi-Fi	Wi-Fi
Collection location	L1	L2	L3	L4	L5
Service provider	S1	S2	S3	S4	S5
Session number (Kilo)	20	5	15	20	20

TABLE 4
Comparison of Normalized QoE (%) Across Different QoE Functions

	Mean	U_1 (ultra-smooth playback)	U_2 (ultra-high quality)	U_3 (balanced)	U_4 (smooth playback)	U_5 (low quality variation)	U_6 (high quality)	U_7 (ultra-balanced)	U_8 (low latency)
FESTIVE	54.9	68.6	42.5	52.1	63.9	62.5	40.4	48.6	61.3
BOLA	64.0	73.1	51.5	65.9	70.2	68.8	58.2	60.2	64.1
MPC	67.9	64.1	70.6	74.6	70.1	56.5	72.3	71.0	63.7
Pensieve	55.7	38.7	77.9	75.8	13.5	71.8	75.2	71.8	20.6
EAS-GP	68.5	52.8	72.4	87.0	61.1	80.4	75.0	80.3	39.1
L2AC	48.5	65.2	30.1	41.7	59.5	44.7	35.9	46.5	77.6

EAS-GP [8] and L2AC [40], and three are heuristic-based: FESTIVE [4], BOLA [5], and Robust-MPC [6] (henceforth called 'MPC'). In our evaluation, to emulate the existing algorithms being built upon a limited set of QoE objectives in the current commercial platform (c.f. Section 1), we trained/optimized Pensieve, EAS-GP, and MPC using only QoE function U_3 for a balanced preference. As L2AC is originally designed for live streaming, we trained it with merely U_8 to involve playback latency. The adaptation logic of BOLA and FESTIVE are pre-programmed so they do not target any QoE function.

We ran each of the algorithms over the eight QoE functions, i.e., (1), (2), (3), (4), (5), (6), respectively, to emulate the viewer's QoE preference changes. We normalized the obtained absolute QoE through the offline optimal QoE, which is the upper bound QoE computed by an omniscient policy with complete and perfect knowledge of future network throughput (according to Spiteri *et al.* [5] and the source code is offered by Mao *et al.* [30]). We summarized in Table 4 the overall mean QoE as well as the separate QoE under different QoE functions (preferences). The main observation is that the streaming algorithm's performance varies substantially across different QoE functions. Pensieve is the most obvious one. It performs best under U_2 and U_6 whereas worst under U_1 and U_4. This is due to Pensieve's inherent aggressiveness of bitrate selection which can be rewarded more under the high video quality preference (i.e., U_2 and U_6), but meanwhile incurs more rebuffering events, resulting in significant QoE degradation under the smooth playback preference (i.e., U_1 and U_4).

In addition, Pensieve's QoE under U_8 is also very bad, which is because U_8 represents the low latency preference and involves a penalty for long playback latency that has high correlations with playback rebuffering. The rationale is that during each rebuffering event where the player runs out of the video data, video playback will be suspended until sufficient data are downloaded to resume the playback. The live video source, on the other hand, continues on and thus the gap between the video playback and the rendering would be widened by the rebuffering event [39], [40], [41], [42], [43]. Therefore, the frequent rebuffering events caused by Pensieve will cause long playback latency, which in turn significantly reduces the performance under U_8.

EAS-GP exhibits similar performance to Pensieve, but to a lesser extent. This benefits from EAS-GP that generates an ensemble of adaptation logics instead of just one single logic as Pensieve does, which enables EAS-GP to have stronger robustness than Pensieve over different environments. Nevertheless, the QoE variation in EAS-GP is also significant, from a low of 39.1% to a high of 87.0%.

The live streaming algorithm L2AC is trained with U_8, so as expected, it performs best under U_8. However, due to its excessive focus on low latency, its bitrate selection is very conservative, leading to the performance under other QoE functions (especially U_2 and U_6) is extremely poor. As a result, L2AC's mean QoE is the lowest among all algorithms.

In comparison, the QoE performance of the three heuristic-based algorithms is a bit more stable, but still far from consistent. For instance, FESTIVE and BOLA achieve high QoE in U_1 and U_4 but perform much worse in U_2 and U_6. This is due to the fact that they adopt very conservative bitrate adaptation logics which can keep a low probability of rebuffering but at the expense of substantial video quality degradation. While MPC has the most stable performance among all the algorithms, it performs obviously poorly (56.5%) under U_5. This indicates that frequent bitrate switching is a distinctive feature of MPC's bitrate adaptation.

Overall, since all these existing algorithms were not designed with the diverse QoE preference in mind, but built on a limited set of QoE objectives, it is not surprising that the resulting QoE varies substantially in different scenarios. In the next section, we propose the PSQA framework to tackle this problem.

4 Post-Streaming Quality Analysis

The principle of PSQA is to exploit consistent statistical properties exhibited by network throughput over a long timescale (e.g., days) [33] to automatically optimize QoE for the streaming algorithms. PSQA begins with an *offline analysis phase* where captured throughput trace data from past streaming sessions are analyzed to optimize the adaptation logic for any given streaming algorithms according to any given QoE functions. This is done periodically (e.g., daily) to keep updating the adaptation logic for use in the second phase – *online prediction phase*, where the actual streaming occurs based on the updated logic. The system's structure is illustrated by the diagram in Fig. 1. In this section, we will first introduce the methodology of PSQA, and then demonstrate its applicability to the existing streaming algorithms.

4.1 Methodology

The offline analysis phase optimizes the adaptation logic of the target streaming algorithm according to given QoE functions. The PSQA framework does not mandate the form nor

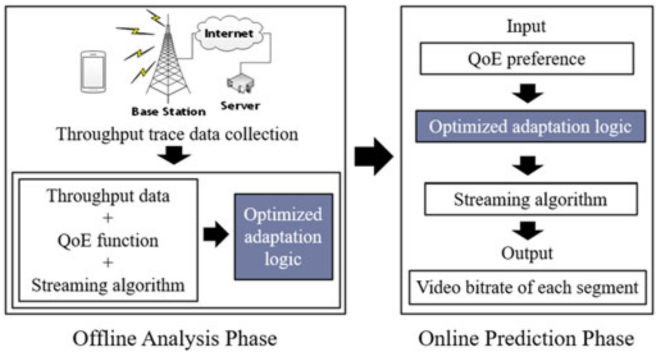

Internet
Base Station
Server
Throughput trace data collection
Throughput data + QoE function + Streaming algorithm
Optimized adaptation logic
Offline Analysis Phase
Input
QoE preference
Optimized adaptation logic
Streaming algorithm
Output
Video bitrate of each segment
Online Prediction Phase

Fig. 1. The overall structure of PSQA.

the coefficients of the QoE functions. Thus, we define a general expression $U(\mathbf{A})$ to denote the QoE function, where $\mathbf{A}$ represents a vector of QoE metrics, e.g., video bitrate, playback rebuffering, and so on. Note that the detailed QoE metrics depend on the definition of the chosen QoE function (e.g., (1), (2), (3), (4), (5), (6)).

Considering the viewer's QoE preference being diverse, we select a list of QoE functions, denoted by $\{U_s(\mathbf{A})\,|\,s = 01,\ldots,S\text{-}1\}$, to construct the objective function in PSQA's optimization where the goal is to maximize every QoE function in the list. In addition, since the QoE is experienced under changing network conditions, the objective function is calculated upon multiple streaming sessions:

$$\max\sum_{i=0}^{N-1} U_s(\mathbf{A}_i) \Big/ N\,, s= 0,1,\ldots,S-1 \qquad (7)$$

where $\mathbf{A}_i$ is the QoE metrics derived from streaming session i and N is the total session number. Other formats of objective functions (e.g., max-min) are also available and again the PSQA framework does not restrict their forms.

Given the objective function, the QoE metrics inside, i.e., $\mathbf{A}_i$, will then be determined by the streaming algorithms, so PSQA needs to optimize its adaptation logic such that the objective function can be maximized over the N streaming sessions. The structures of the adaptation logic proposed by the existing studies are various, but they can typically be formulated in the form of a combination of measurable streaming variables (e.g., past throughput, buffer occupancy, etc.) and internal parameters. We thus denote the adaptation logic with a general function $G(\cdot)$:

$$r_{i,j} = G(\theta_{i,j,0}, \theta_{i,j,1}, \ldots, \theta_{i,j,K-1}, \kappa_0, \kappa_1, \ldots, \kappa_{L-1}), j = 0,1,\ldots,J \qquad (8)$$

where $\{\theta_{i,j,0}, \theta_{i,j,1}, \ldots, \theta_{i,j,K\text{-}1}\}$ is a total of K streaming variables measured at requesting video segment j in streaming session i, $\{\kappa_0, \kappa_1,\ldots, \kappa_{L\text{-}1}\}$ is a total of L internal parameters, and $r_{i,j}$ is the output bitrate decision.

Next is to execute the adaptation logic in the real streaming environment. However, as the real network condition is not repeatable, the measurements of more detailed performance metrics such as rebuffering duration, playback latency, etc., are difficult to conduct in the real network [27]. Therefore, we propose to employ virtual streaming [7], [8], [9] to mimic the network condition by replaying TCP throughput trace data obtained as a by-product of past streaming sessions (e.g., via network capturing tools such as tcpdump).

Specifically, given the set of throughput trace data in session i, denoted by $\mathbf{C}_i$, the adaptation logic can be executed through:

$$\mathbf{A}_i = F(G(\theta_{i,j,0}, \theta_{i,j,1}, \ldots, \theta_{i,j,K-1}, \kappa_0, \kappa_1, \ldots, \kappa_{L-1}), \mathbf{C}_i) \qquad (9)$$

where the virtual streaming process is denoted by the function $F(\cdot)$ and the output is a list of QoE metrics, e.g., average bitrate, rebuffering duration, etc., collectively denoted by $\mathbf{A}_i$. Since $\mathbf{A}_i$ is also the input of the objective function (i.e., (7)), so we can then construct a link to relate the adaptation logic to (7):

$$\max_{\kappa_0,\ldots\kappa_{L-1}} \sum_{i=0}^{N-1} U_s(F(G(\theta_{i,j,0}, \theta_{i,j,1}, \ldots, \theta_{i,j,K-1}, \kappa_0, \kappa_1, \ldots, \kappa_{L-1}), \mathbf{C}_i)) \Big/ N\,,\quad s = 0,1,\ldots,S-1 \qquad (10)$$

where the set of internal parameters $\{\kappa_0, \kappa_1,\ldots, \kappa_{L\text{-}1}\}$ can be dynamically tuned by PSQA to optimize the adaptation logic.

Finally, through solving the optimization problem, i.e., (10), PSQA will find the optimized internal parameters under each QoE function $U_s(.)$ to maximize their QoE separately:

$$\prod = \left\{\kappa^*_{s,0}, \kappa^*_{s,1}, \ldots, \kappa^*_{s,L-1} | s= 0,1,\ldots,S-1\right\} \qquad (11)$$

such that the overall QoE can be maximized. The detailed method for solving the optimization problem is elaborated in Section 4.2 and 4.3.

After the offline analysis, PSQA obtains the knowledge of the optimal configurations from the past streaming sessions, which will then guide the configuration in *online prediction phase*. Specifically, the optimized parameter set, i.e., (11), will be loaded into the video player as part of the streaming metadata (e.g., MPD playlist in DASH [19]). Before streaming a new session x, the viewers will be prompted for specifying their QoE preferences (e.g., high video quality, low rebuffering, etc.) through an interface to the system (e.g., an option list in the video player). Based on the specified QoE preference, PSQA then applies the internal parameters optimized under the corresponding QoE function $U_{s'}(.)$, denoted by $\{\kappa^*_{s',0}, \kappa^*_{s',1}, \ldots, \kappa^*_{s',L-1}\}$, to configure the bitrate adaptation logic:

$$r_{x,j} = G(\theta_{x,j,0}, \theta_{x,j,1}, \ldots, \theta_{x,j,K-1}, \kappa^*_{s',0}, \kappa^*_{s',1}, \ldots, \kappa^*_{s',L-1}), j = 0,1,\ldots,J \qquad (12)$$

The rationale of PSQA is that the network throughput exhibits consistent properties over a timescale of days, so that one can analyze past streaming sessions' network conditions to achieve predictable performance (QoE) for future sessions [33]. In offline analysis, PSQA captures the statistical behavior of the underlying network to optimize the internal parameter of the adaptation logic, then the optimized parameter will likely result in good QoE for the new streaming sessions online. PSQA employs a repeated cycle of the two phases to guarantee that the value of the internal

TABLE 5
Streaming Variable and Aggressiveness Factor κ_0 in the Existing Heuristic-Based Algorithms

Algorithm	Type	Streaming variable	Range of κ_0
FESTIVE [4]	Throughput based	Harmonic mean throughput	0.1∼4.0
BOLA [5]	Buffer based	Buffer-bitrate mapping	1.0∼10.0
MPC [6]	Hybrid-throughput-buffer based	Harmonic mean throughput divided by past estimation error	0.1∼3.0

parameters can be continuously updated to maintain consistent QoE performance despite the evolution of the network conditions, system infrastructures, or QoE preferences.

4.2 Application to Heuristic-Based Algorithms

We first apply PSQA to optimizing the existing heuristic-based algorithms, and then apply it to the learning-based ones in Section 4.3. Heuristic-based algorithms are built upon human intuitions, which generally have a pre-programmed adaptation logic to dynamically select the video bitrate in the light of measured throughput, buffer occupancy, etc. To optimize them for a specific QoE preference, PSQA will directly tune their internal parameters (c.f. (8)) to control their bitrate selection behavior. In the following, we will introduce two internal parameters and demonstrate how to tune them.

Aggressiveness Factor κ_0. In general, there are four QoE metrics that are commonly considered in streaming videos, namely video quality, playback rebuffering, quality variations, and playback latency (only for live streaming). As discussed in Section 3.2, video quality and playback rebuffering are a pair of inherent conflicting metrics, and both of them are affected significantly by the bitrate selection aggressiveness. In addition, playback latency also has high correlations with the playback rebuffering. Therefore, if one can explicitly regulate the bitrate selection aggressiveness, the tradeoff among these three QoE metrics can be well controlled.

Based on this insight, we introduce the first internal parameter, named *Aggressiveness Factor κ_0*. To illustrate how κ_0 works, we take MPC [6] as an example. Specifically, in MPC, one of the streaming variables for determining video bitrate is the estimated throughput, which is reproduced below:

$$\theta_j = H_j/(1 + e_j) \tag{13}$$

where θ_j is the estimated throughput for determining the bitrate of segment j, H_j is the harmonic mean throughput in downloading the past 5 segments (i.e., segment j–6 $\sim$ j–1) and e_j is the maximum of previous estimation error. Since MPC relies on θ_j to determine the video bitrate, we can apply the Aggressiveness Factor κ_0 multiplied by θ_j to control the aggressiveness of bitrate selection:

$$\theta_j^{'} = \kappa_0 \times \theta_j \tag{14}$$

where the final output is denoted by $\theta_j^{'}$.

In PSQA, κ_0 can be applied to different streaming algorithms where the operating mechanism is similar but the specific definition of the streaming variables depends on the design of different adaptation logic. Table 5 summarizes the detailed description for the streaming variables in three heuristic-based algorithms. Specifically, FESTIVE is a throughput-based algorithm that also employs harmonic mean throughput to determine video bitrate, so κ_0 can be applied in a similar manner with MPC. BOLA is a bit different as it is a buffer-based algorithm and it has a tunable ratio in its utility function [5] to control the mapping between the selected bitrate and the buffer level, so we use the tunable ratio to denote κ_0. For the detailed definitions of the streaming variables, we refer interested readers to the algorithms' original studies, i.e., [4], [5], [6].

Bitrate Switching Factor κ_1. As the video quality variation *cannot* be effectively controlled by the Aggressiveness Factor κ_0, this motivates us to introduce the second internal parameter—*Bitrate Switching Factor* κ_1. The principle is to limit the maximum bitrate switching magnitude within κ_1 levels.

Specifically, in practice, the video segments in DASH are encoded into discrete bitrate versions $\{r_h \;|h = 01,\ldots,H\text{-}1\}$, and each bitrate version corresponds to a certain bitrate level h. On requesting each segment, the bitrate adaptation logic will first determine a candidate bitrate level for the pre-download segment j, denoted by h_j, then PSQA will compare h_j with the bitrate level of the already downloaded segment $h_{j\text{-}1}$ to limit the bitrate switching magnitude within κ_1:

$$\hat{h}_j = \begin{cases} h_{j-1} - \kappa_1 \ , \text{if } h_j < h_{j-1} - \kappa_1 \\ h_{j-1} + \kappa_1 \ , \text{if } h_j > h_{j-1} + \kappa_1 \\ \quad h_j \quad\ , \quad otherwise \end{cases} \tag{15}$$

where $\hat{h}_j$ is the final determined bitrate level.[1] Similar to κ_0, this mechanism can also be applied to different existing adaptation logics where κ_1 is dynamically tuned according to the specific QoE preference. However, it is obvious that limiting the bitrate switches would hamper the client's responsiveness to the throughput fluctuations, so that other QoE metrics, e.g., playback rebuffering, could be traded off. We will further analyze the tradeoff performance in Section 5.3.

Finally, PSQA will optimize the two internal parameters (i.e., κ_0 and κ_1) simultaneously in offline analysis, i.e., (10), and then apply their optimal value into online prediction, i.e., (12). The optimization problem (10) can be solved by brute-force search or Bayesian optimization [37] in a short time. Based on our measurement, the time-consuming is typically within 20 minutes in practice. Moreover, since PSQA is a general framework, any other internal parameters in the adaptation logic can also be applied in a similar

1. We do not claim this method is new. Some previous work, e.g., [48], likely already implemented some similar methods.

fashion for controlling the QoE metrics. Therefore, exploring other effective internal parameters could be a fruitful direction for future work.

4.3 Application to Learning-Based Algorithms

As opposed to the heuristic-based approaches, another important branch is to use machine learning techniques to generate adaptation logics. Generally, in a separate training phase, the system is exposed to the target operating environment where it trains the adaptation logics (e.g., neural network) and tunes the internal parameters (e.g., neuron weight) automatically through the observations from past experiences (e.g., resultant QoE, buffer occupancy, throughput, etc.).

Therefore, in the offline analysis phase of PSQA, we propose *specialized training* to generate/optimize the adaptation logic, which takes advantage of the inborn training process of the learning techniques to maximize the objective function, i.e., (10). Specifically, PSQA employs the virtual streaming [7], [8], [9] to speed up the training (introduced in Section 4.1), where the input is TCP throughput trace data and the output is an ensemble of adaptation logics $\{G_s \,|\, s = 01,\ldots,S\text{-}1\}$ with the goal of maximizing each QoE function in the given set $\{U_s \,|\, s = 01,\ldots,S\text{-}1\}$. To ensure that the training experiences enough real network conditions, an extensive set of throughput traces, denoted by $\mathbf{C}$, is incorporated:

$$G_s = T_x(U_s, \mathbf{C}), s = 0, 1, \ldots, S-1 \tag{16}$$

where the training process is denoted by function $T_x(.)$, which can be an arbitrary machine-learning technique, e.g., deep-reinforcement-learning in Pensieve [7] or genetic-programming adopted by EAS-GP [8].

The training (i.e., offline analysis) is executed periodically by PSQA based on newly captured trace data each day (reflecting the recent network condition). After each training, the generated adaptation logic ensemble, i.e., $\{G_s \,|\, s = 01,\ldots,S\text{-}1\}$, will be equipped with the optimal internal parameters, i.e., (11), and then applied to the online prediction phase, i.e., (12). At runtime, the viewers will be prompted for specifying their preference before/during online streaming and the system will load the adaptation logic trained by the corresponding QoE function. For instance, if the viewer prefers smooth playback, then the adaptation logic optimized by the QoE function with high penalties on rebuffering will be selected for service.

In practice, however, one can envision that the viewer can be a layperson for video streaming, so that he/she may have no clue on the QoE objective choice, resulting in the system being incapable to capture his/her QoE preference. In such a case, if the viewer's actual preference is dissimilar to the QoE function used in the training, the trained adaptation logic will suffer from extremely poor performance because the specialized training makes the trained logic over-specialized for the training context. For example, if the adaptation logic is trained with the QoE function that favors high video quality, then it would give very aggressive bitrate decisions. As a result, numerous rebuffering events would degrade the QoE significantly in the case that the viewer prefers smooth playback.

The most intuitive method to avoid over-specialization is to expose the adaptation logic to a more diverse training environment and optimize the overall performance under any potential targets. Thus, unlike maximizing every QoE function in the original objective function, i.e., (10), we define a new form to seek the maximum of the overall QoE:

$$\max_{\kappa_0,\ldots\kappa_{L-1}} \sum_{s=0}^{S-1} \sum_{i=0}^{N-1} U_s(F(G(\theta_{i,j,0}, \ldots, \theta_{i,j,K-1}, \kappa_0, \ldots, \kappa_{L-1}), \mathbf{C}_i)) \Big/ N \tag{17}$$

where the goal is to maximize the summation performance of all available QoE functions, i.e., $\{U_s(.) \,|\, s = 01,\ldots,S\text{-}1\}$, through deriving a generic adaptation logic with a set of internal parameters that are not specific to any particular QoE function, i.e., $\{\kappa_0^*, \kappa_1^*, \ldots \kappa_{L-1}^*\}$.

The problem (17) can be solved by a novel *generic training* in which the QoE of each streaming session will be quantified by all the QoE functions selected from the given list. Then, the obtained QoE under each function will be summed up to get a total reward (i.e., (17)) that will then be fed back to the adaptation logic to let the adaptation logic experience different QoE preference contexts:

$$G = T_x(\{U_s | s = 0, 1, \ldots, S-1\}, \mathbf{C}) \tag{18}$$

where $T_x(.)$ denotes the learning technique and $\mathbf{C}$ is the throughput trace data. With the generic training, the output adaptation logic G can be broadly applicable to various environments without requiring the viewer to specify the QoE preference.

Conceivably, compared to the specialized training, i.e., (16), the adaptation logic generated by the generic training would tradeoff a certain degree of QoE due to its seek of an overall optimum. In comparison, the specialized training can make the adaptation logic perform optimal in a given context, but its over-specialization would turn into obstacles when the target context becomes ambiguous. We will further compare the efficacy of the two training methods in Section 5.2.

4.4 Deployment Discussion

System Complexity. In practice, in applying a system to real streaming platforms, the computation complexity should be configured as low as possible, as the bitrate decision is performed frequently online. This can be easily done by PSQA as most of its complexities are consolidated into the offline analysis phase that can be executed on the server-side, while leaving a simple prediction phase online.

For example, for offline analysis, the CDN server can be easily extended to record the throughput trace data from past streaming sessions when it delivers the video data to the players over HTTP/TCP. The adaptation logic and the internal parameters thus can be optimized by the server and will be embedded into the meta-data of the streaming protocols (e.g., MPD in DASH) to deliver to the video player. For online prediction, the only computation requirement is to periodically update the adaptation logic, which is only performed once after each offline analysis. A deeper analysis for the system's complexity is in Appendix A.2, available in the online supplemental material.

TABLE 6
Performance Under QoE Function U_1 (Ultra-Smooth Playback)

Algorithms	Original	PSQA	Improvement	κ_0	κ_1
FESTIVE	68.6%	80.8%	17.8%	0.1	8
BOLA	73.1%	84.5%	15.6%	1.0	8
MPC	64.1%	85.7%	33.7%	0.1	8
Pensieve	38.7%	67.2%	73.6%	N/A	N/A
EAS-GP	52.8%	87.5%	65.7%	N/A	N/A
L2AC	65.2%	80.1%	22.9%	N/A	N/A

TABLE 7
Performance Under QoE Function U_2 (Ultra-High Quality)

Algorithms	Original	PSQA	Improvement	κ_0	κ_1
FESTIVE	42.5%	72.4%	70.4%	4.0	8
BOLA	51.5%	81.6%	58.4%	10.0	8
MPC	70.6%	97.6%	38.2%	3.0	8
Pensieve	77.9%	99.5%	27.7%	N/A	N/A
EAS-GP	72.4%	99.1%	36.9%	N/A	N/A
L2AC	30.1%	90.2%	199.6%	N/A	N/A

TABLE 8
Performance Under QoE Function U_3 (Balanced)

Algorithms	Original	PSQA	Improvement	κ_0	κ_1
FESTIVE	52.1%	63.4%	21.7%	2.7	8
BOLA	65.9%	72.2%	9.6%	6.7	8
MPC	74.6%	80.8%	8.3%	1.0	8
Pensieve	75.8%	82.1%	8.3%	N/A	N/A
EAS-GP	87.0%	87.3%	0.3%	N/A	N/A
L2AC	41.7%	75.4%	80.8%	N/A	N/A

TABLE 9
Performance Under QoE Function U_4 (Smooth Playback)

Algorithms	Original	PSQA	Improvement	κ_0	κ_1
FESTIVE	63.9%	72.2%	13.0%	1.6	8
BOLA	70.2%	75.1%	7.0%	4.3	8
MPC	70.1%	80.2%	14.4%	0.7	8
Pensieve	13.5%	66.3%	391.1%	N/A	N/A
EAS-GP	61.1%	83.1%	36.0%	N/A	N/A
L2AC	59.5%	80.5%	35.3%	N/A	N/A

TABLE 10
Performance Under QoE Function U_5 (Low Quality Variation)

Algorithms	Original	PSQA	Improvement	κ_0	κ_1
FESTIVE	62.5%	67.1%	7.4%	2.0	7
BOLA	68.8%	73.7%	7.1%	5.2	6
MPC	56.5%	69.6%	23.2%	0.9	4
Pensieve	71.8%	79.2%	10.3%	N/A	N/A
EAS-GP	82.4%	85.6%	6.5%	N/A	N/A
L2AC	44.7%	75.9%	69.8%	N/A	N/A

TABLE 11
Performance Under QoE Function U_6 (High Quality)

Algorithms	Original	PSQA	Improvement	κ_0	κ_1
FESTIVE	40.4%	67.9%	68.1%	3.4	8
BOLA	58.2%	73.2%	25.8%	7.9	8
MPC	72.3%	80.1%	10.8%	1.6	7
Pensieve	75.2%	87.2%	16.0%	N/A	N/A
EAS-GP	75.0%	84.9%	13.2%	N/A	N/A
L2AC	35.9%	71.3%	98.6%	N/A	N/A

QoE Preference Selection. PSQA is a general framework that does not mandate the form of the QoE objective, so any QoE functions can be included to cover the diverse QoE preferences. As introduced in Section 3.1, the eight QoE functions, i.e., (1), (2), (3), (4), (5), (6), can be incorporated into the system to offer eight QoE preferences for viewers to choose from, and the detailed mapping is: U_1 – "ultra-smooth playback", U_2 – "ultra-high quality", U_3 – "balanced", U_4 – "smooth playback", U_5 – "low quality variation", U_6 – "high quality", U_7 – "ultra-balanced", and U_8 – "low latency". All these preferences can be presented with an option list in the video player, and the viewers can specify their preference through the option list at any time before or during their viewing. Moreover, for cases where the viewer does not make any preference choice, the adaptation logic generated by the generic training (i.e., (18)) will be used for service by default.

Overall, PSQA can be readily deployed into the real streaming platforms, offering a practical solution for the streaming vendors. In Section 6, we implemented a prototype for PSQA using dash.js [25] and evaluated its performance in the real streaming settings.

5 Performance Evaluation

In this section, we apply the PSQA framework to the existing adaptive streaming algorithms and show how PSQA improves their QoE performance.

5.1 Evaluation Setting

The evaluation was conducted through trace-driven simulations with the same setup as described in Section 3. PSQA was applied to optimizing the six existing streaming algorithms, namely FESTIVE [4], BOLA [5], MPC [6], Pensieve [7], EAS-GP [8], and L2AC [40], over the eight QoE functions, i.e., (1), (2), (3), (4), (5), (6).

We used a total of 60 weeks' TCP throughput trace data ($\sim$80000 streaming sessions) in the evaluation (see Table 3). PSQA was configured to use the past one day's trace data in offline analysis phase and then the optimized adaptation logic was applied to the next day's online prediction phase. For optimizing the heuristic-based algorithms, i.e., FESTIVE, BOLA and MPC, the tuning range of κ_0 is listed in Table 5, and κ_1 is tuned in the range of $1 \sim 8$. For the learning-based algorithms, i.e., Pensieve, EAS-GP, and L2AC, the detailed training workflow follows their original studies [78,40].

5.2 QoE Performance Improvement

We first evaluate the case where the user has explicit QoE preference. Tables 6, 7, 8, 9, 10, 11, 12, 13 summarize the normalized QoE (N-QoE) performance over all the QoE

TABLE 12
Performance Under QoE Function U_7 (Ultra-Balanced)

Algorithms	Original	PSQA	Improvement	κ_0	κ_1
FESTIVE	48.6%	62.1%	27.8%	2.1	7
BOLA	60.2%	74.5%	23.8%	6.1	7
MPC	71.0%	79.9%	12.5%	0.9	6
Pensieve	71.8%	81.7%	13.8%	N/A	N/A
EAS-GP	80.3%	86.2%	7.3%	N/A	N/A
L2AC	46.5%	78.1%	68.0%	N/A	N/A

TABLE 13
Performance Under QoE Function U_8 (Low Latency)

Algorithms	Original	PSQA	Improvement	κ_0	κ_1
FESTIVE	61.3%	69.1%	12.7%	1.2	8
BOLA	64.1%	73.4%	14.5%	3.2	8
MPC	63.7%	75.2%	18.1%	0.5	7
Pensieve	20.6%	61.3%	197.6%	N/A	N/A
EAS-GP	39.1%	78.1%	99.7%	N/A	N/A
L2AC	77.6%	83.4%	7.5%	N/A	N/A

functions where each one represents one particular QoE preference specified by the viewer (c.f. Sections 3.1 and 4.4). In each table, the column labeled "Original" is the N-QoE achieved by the streaming algorithms under their original settings. Similar to Section 3, the original Pensieve, EAS-GP, and MPC were trained/pre-optimized with QoE function U_3 for a balanced preference, and the live streaming algorithm L2AC was trained with U_8 to achieve low latency. The column labeled "PSQA" is the N-QoE achieved after PSQA's optimization. We also calculated the QoE improvement proportion, labeled "improvement", which is the ratio of the improved and the original QoE. The columns labeled "κ_0" and "κ_1" are the mean values of the two internal parameters tuned by PSQA (see Section 4.2). As the internal parameter of Pensieve, EAS-GP, and L2AC are not explicit (c.f. Section 4.3), these two columns are not applicable to them. We will further analyze their internal mechanism in Section 5.3.

From the results, we can see that PSQA is able to improve the QoE of *all* the streaming algorithms under *all* the QoE preferences, and the improvement proportions are often substantial. In addition, for each algorithm, the QoE variations over different QoE preferences are significantly reduced after the optimization of PSQA.

We now elaborate on the details. First, QoE functions U_1 and U_2 are two extreme cases where U_1 only considers playback smoothness (e.g., low rebuffering) without regard to video quality, while U_2 is just the opposite. As the two QoE metrics are inherent tradeoffs, the original version of the streaming algorithms, e.g., BOLA, typically exhibits good performance in one metric, e.g., $U_1 = 73.1\%$, but performs much worse in the other one, e.g., $U_2 = 51.5\%$ (see Tables 6 and 7).

For this case, PSQA optimizes the algorithms in two opposite directions, where the principle is reflected by the internal parameter κ_0. Specifically, under U_1, the value of κ_0 is tuned to the minimum value to keep the bitrate selection conservative, so that rebuffering events can be largely avoided. On the contrary, at U_2, κ_0 is tuned to be maximum such that higher levels of bitrate can be selected to improve the video quality as much as possible. In Table 7, it is worth noting that the optimized MPC, Pensieve, and EAS-GP achieve $97.6\% \sim 99.5\%$ of N-QoE at U_2, close to the upper bound, which is because they are able to choose the highest bitrate level for almost all the video segments (while the internal parameter in Pensieve and EAS-GP is not explicit, the principle behind them is similar).

In comparison, other QoE functions are more comprehensive, incorporating both the video quality and playback rebuffering. Under U_3, the QoE of FESTIVE and BOLA is improved significantly, by 21.7% and 9.6% respectively (Table 8), which benefits from PSQA tuning the κ_0 to achieve a more appropriate bitrate selection aggressiveness, thereby obtaining a more suitable metric tradeoff. To our surprise, PSQA can still improve the QoE of MPC and Pensieve by 8.3% at U_3 despite that their original versions are already pre-optimized/trained based on U_3. According to our further investigation, we learned that it is because PSQA can improve the algorithm's robustness under the changing network conditions. We will further analyze this point in Section 5.4.

Compared to U_3, U_4 puts a larger weight on the penalty of playback rebuffering, so PSQA accordingly lowers the algorithms' bitrate selection aggressiveness (i.e., κ_0). In this way, the inappropriate aggressiveness of the original Pensieve and EAS-GP are well corrected, such that their QoE is improved from 13.5%/61.1% to 66.3%/83.1% respectively (see Table 9). On the contrary, U_6 favors high-quality videos, thus the bitrate selection aggressiveness is accordingly turned up by PSQA (see Table 11).

U_5 emphasizes the penalty of video quality fluctuations. In this case, the internal parameter κ_1 starts to exhibit its effectiveness. For example, in Table 10 the original MPC switches bitrates very frequently, so its QoE under U_5 is substantially lower than others. To tackle the issue, PSQA turns down κ_1 in MPC to limit its bitrate switching magnitude, so that a 23.2% QoE gain is achieved.

Finally, U_8 is the only QoE function incorporating the effect of playback latency. From the results in Table 13, it was observed that except for L2AC, the QoE of all other algorithms is improved by more than 10% by PSQA, which is primarily due to PSQA's optimization for the playback latency. Accordingly, we measured the playback latency achieved before and after PSQA's optimization and present the results in Fig. 2. It is clear that the latency of almost all the algorithms is reduced through the optimization of PSQA (especially Pensieve and EAS-GP) where the principle is that PSQA intelligently detects a positive correlation between playback latency and bitrate selection aggressiveness and thus turns down κ_0 to maintain low latency. Moreover, for L2AC, as it was originally trained with U_8,

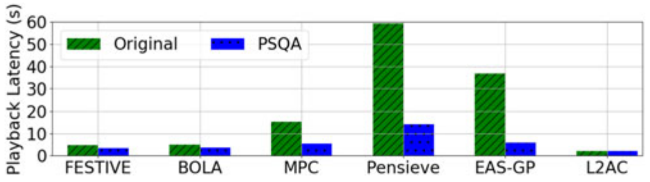


Fig. 2. Playback latency before and after PSQA's optimization.

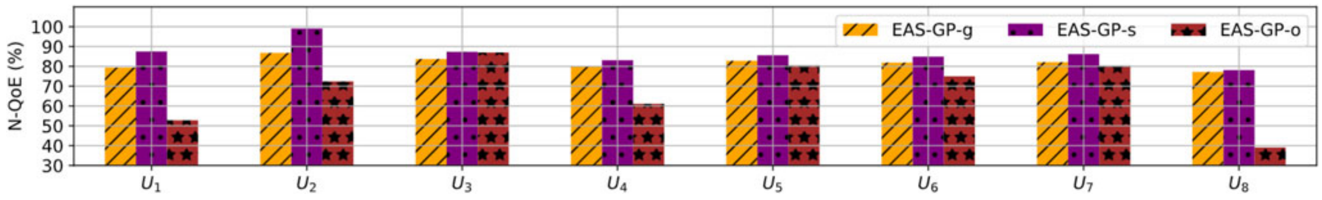


Fig. 3. The normalized QoE performance (%) under generic training and specialized training.

the QoE improvement is merely 7.8%. In contrast, its QoE is improved more significantly under $U_1 \sim U_7$, ranging from 22.9% to 98.6%.

All the above experiments only focus on the cases with explicit QoE preference, but as discussed in Section 4.3, the viewer with unknown QoE preference also existed, so we propose the generic training in Section 4.3 to fill this functional gap. In this section, we use EAS-GP to test the effectiveness of the generic training and compare it to the specialized training (i.e., require specifying preference) as well as the original version (i.e., trained only with U_3).

The resulting QoE performance is presented in Fig. 3, where EAS-GP-g, EAS-GP-s, and EAS-GP-o denote the generic training, specialized training, and the original version, respectively. Obviously, EAS-GP-g can effectively fix the extremely poor performance of EAS-GP-o at U_1, U_4, and U_8, and achieve more stable QoE over the eight QoE preferences. However, when compared to EAS-GP-s, its QoE loss shows up, ranging from 2.8% to 12.3%, which is inevitable because EAS-GP-g seeks an overall optimum without being informed of the QoE preference. Overall, the experimental results validate the efficacy of the generic training and indicate that it can effectively address the QoE preference unknown problem in practice.

5.3 Tradeoff Analysis Among QoE Metrics

The QoE improvement shown in Section 5.2 is primarily contributed by PSQA's optimization for the tradeoffs among different QoE metrics. Therefore, in this section, we take MPC (heuristic-based) and Pensieve (learning-based) as examples to further uncover the underlying mechanism.

We first investigate the PSQA optimized MPC (henceforth abbreviated as PSQA-MPC). As its mechanism is the automatic adjustment for the two internal parameters κ_0 and κ_1, we manually tuned the value of the two parameters respectively to quantify their effectiveness on different QoE metrics and N-QoE. Fig. 4 plots the results of κ_0. In the upper chart, the increasing κ_0 incurs the increment of both video bitrate and rebuffering duration, which is due to the increased bitrate selection aggressiveness that changes the tradeoff point between the two metrics. In the lower chart, as the three QoE functions have quite different weights on bitrate and rebuffering (i.e., different preferences), the peak points of QoE correspond to different values of κ_0. For example, the peak point under U_4 has the lowest κ_0 as a more conservative adaptation logic is required while under U_6 is the highest as high video quality can be rewarded more.

Fig. 5 illustrates the result for κ_1. It is observed in the upper chart that although decreasing κ_1 can significantly reduce the video quality variations, it also leads to video quality degradation and more rebuffering events. The reason is that limiting the bitrate switches weakens the adaptation logic's capability to adapt to the throughput fluctuations. The resultant QoE performance is depicted in the lower chart. With κ_1 decreasing, U_5 reaches the peak at $\kappa_1 = 4$, whereas U_3 keeps degrading.

Next is to investigate PSQA-Pensieve. Analyzing the machine-learning based algorithm is challenging because the trained adaptation logic (e.g., neural network) is too opaque to gain insights. To shed light on PSQA-Pensieve, we adopted a new method. Specifically, we freeze the less critical streaming variables in the logic, e.g., set buffer occupancy to fixed 2s and the last segment bitrate to 200kbps, and then quantify the relationship between the bitrate choice versus the measured throughput, so that the bitrate selection behavior under the training with different QoE functions can be explicitly compared.

We plotted the results in Fig. 6 to compare the performance under U_3, U_4, and U_6. Note that the calibration values of the y-axis indicate the available video bitrate versions. We observed that, as the measured throughput increases, the selected bitrate of the three cases all gradually increases, but their rising slopes are different. First, under

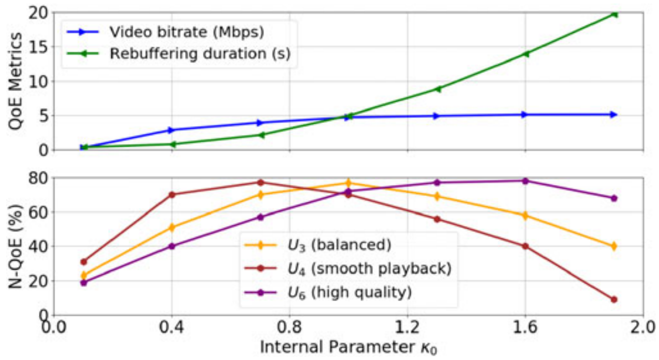


Fig. 4. The impact of the internal parameter κ_0 (PSQA-MPC).

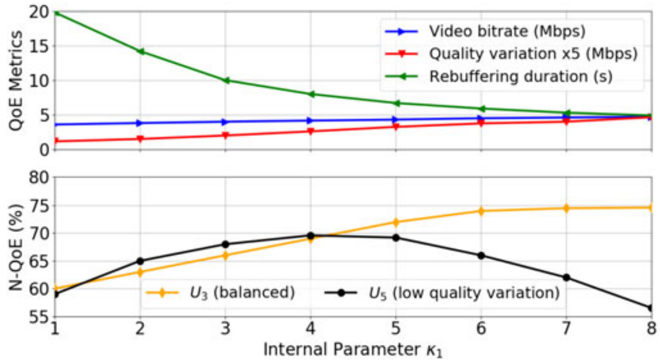


Fig. 5. The impact of the internal parameter κ_1 (PSQA-MPC).

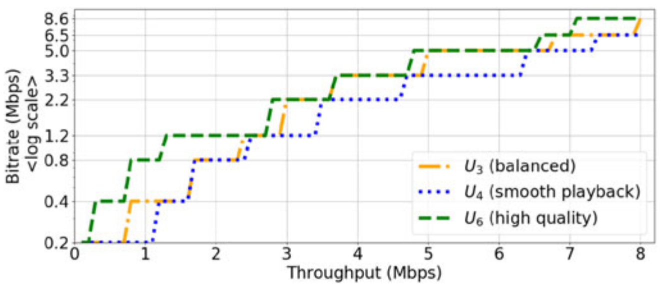


Fig. 6. Bitrate selection aggressiveness across three QoE functions (PSQA-Pensieve).

U_4, it is clear that PSQA-Pensieve intentionally selects bitrates much lower than the measured throughput to prevent rebuffering. On the contrary, the bitrate selection under U_6 is most aggressive, even occasionally selecting bitrates higher than the measured throughput, so that high video quality can be maintained to prevent unnecessary QoE degradation. By comparison, the bitrate selection at U_3 is more moderate and balanced.

5.4 Network-Optimized Adaptive Streaming

One experiment in Section 5.2 (i.e., in Table 8) shows that although the original version of MPC and Pensieve are pre-optimized/trained with U_3, their QoE can still be improved significantly after PSQA's optimization. We conjecture that this is due to PSQA's ability to improve the streaming algorithm's robustness across different network conditions. To verify our conjecture, in this section, we further investigate PSQA's performance under different network environments separately. Note that we only show MPC's results as Pensieve has similar performance patterns. In addition, the training and testing of MPC and PSQA-MPC are all under U_3.

At first, in Table 14, we rank the network condition of the five throughput trace sources #1 ~ #5 as "Good", "Medium", and "Poor", based on their mean throughput and variations, and then measure their QoE and the internal parameter κ_0 separately. From the results, it is clear that PSQA is able to improve MPC's QoE in all the five trace sources, ranging from 5.4% to 11.6%. The mechanism can be reflected by the value of κ_0 which is varied by PSQA to appropriately adjust the bitrate selection aggressiveness according to the specific network conditions of each trace source. For instance, PSQA holds the lowest κ_0 in #2 where the network condition is poor (low throughput and high variation), and conversely, gives the largest κ_0 in #3 to exploit the abundant throughput.

TABLE 14
QoE Performance and Internal Parameter κ_0 Across Five Throughput Trace Sources

Features	#1	#2	#3	#4	#5
Network condition	G	P	G	M	M
Mean throughput (Mbps)	5.97	1.21	10.1	3.12	4.43
Variation (CoV)	0.44	0.83	0.52	0.77	0.58
N-QoE of MPC (%)	78.1	70.1	76.4	72.8	73.5
N-QoE of PSQA-MPC (%)	82.3	78.2	83.7	79.2	80.4
Improvement (%)	5.4	11.6	9.6	8.8	9.4
κ_0	1.07	0.79	1.13	0.91	0.94

*In the row of Network Condition, "G" means "Good", "P" means "Poor" and "M" means "Medium".

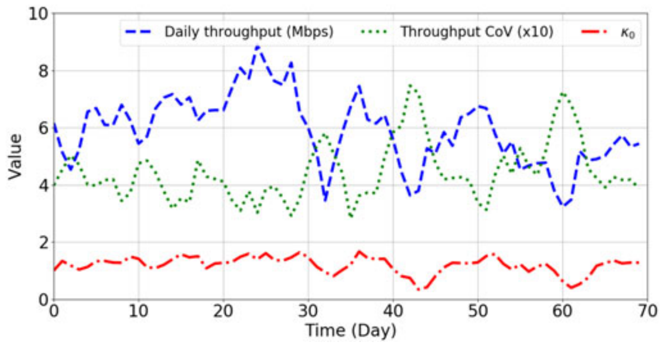


Fig. 7. The evolution of the internal parameter κ_0 over 70 days (PSQA-MPC).

Next, we further study the temporal variations of κ_0. Fig. 7 plots the daily mean value of κ_0 over a period of 70 days where the x-axis is the number of days. In addition, the daily mean throughput and CoV are also plotted to show the network condition. The major observation is that κ_0 is constantly changing over time and its trajectory clearly correlates with the daily network conditions over the 70 days. This suggests that PSQA's periodical optimization is essential to its performance, because it enables PSQA to have the ability to adapt to the long-term (e.g., day) variations in the network condition (e.g., throughput variations) and thus achieve strong robustness over the complicated streaming/network environments.

In the default setting, the adaptation logic in PSQA is updated on a daily basis, so one might be interested in the effects of offline analysis with shorter or longer intervals. To this end, we tuned the time interval from 5 minutes to 7 days to see the impacts on the QoE performance. Note that the throughput trace data used in offline analysis is merely from the last interval. Since PSQA-Pensieve requires longer processing time than PSQA-MPC, it is not appliable to the time intervals shorter than 1 hour. The result is summarized in Table 15 which is observed that both too short and too long of time intervals can result in poorer QoE performance. Longer intervals decrease the updating frequency for the adaptation logic, thereby hampering the PSQA's responsiveness to the network condition changes. On the contrary, shorter intervals introduce more noises in the throughput detection, thereby reducing the correlation of the network states. Therefore, in this work, we adopted 1 day as the default value. A deeper sensitivity analysis for the amount of training data and the processing time is shown in Appendix A.2.

6 Real Implementation

We implemented a prototype of PSQA with the well-known video player *dash.js* (version 3.11) [25] to validate PSQA's

TABLE 15
Sensitivity Analysis on the Time Interval of Offline Analysis

	5 mins	1 hour	6 hours	1 day	7 days
PSQA-MPC	67.6	74.3	78.9	80.8	76.4
PSQA-Pensieve	N/A	68.4	77.8	82.1	80.8

TABLE 16
QoE Performance in Real Experiments

	Original	PSQA	Improvement
FESTIVE	4.65	6.05	29.7%
BOLA	5.63	6.71	19.2%
MPC	6.22	7.41	18.9%
Pensieve	4.73	7.48	58.0%
EAS-GP	6.06	7.95	31.3%
L2AC	4.46	6.55	46.9%

feasibility and performance in real-world streaming setups. Specifically, we implemented the original version of the six streaming algorithms (i.e., [4], [5], [6], [7], [8], [40]) into dash.js. For Pensieve and L2AC, dash.js was configured to fetch the bitrate choice decisions from a specialized bitrate decision server where the trained neural network is deployed. Other four algorithms were directly embedded into "AbrController.js" of dash.js to execute. For PSQA, we implemented its offline analysis phase with Python script (version 2.7.18) and then ran the optimized adaptation logic online via dash.js.

In the streaming setup, the video server host ran Linux with the Apache httpd [34] serving video data over TCP CUBIC [35] and the video client was a Google Chrome browser running in a smartphone with the Android operating system. We employed an improved version of DummyNet [36] to emulate the network conditions between the client and the server, where the available throughput is constricted by the TCP throughput trace data [28], [27], [29] along with 80 ms minimal RTT to model propagation delay.

In addition, we selected several 300s-long videos from the open dataset [27] which includes a total of 700+ real-world commercial video contents. Each video was divided into 150 segments, so every segment is approximately 2s of playback. These videos were encoded into different bitrate versions via H.264 codec (using FFmpeg) [44], where the available bitrate levels are {0.2, 0.4, 0.8, 1.2, 2.2, 3.3, 5.0, 6.5, 8.6} Mbps. Other evaluation settings were identical to those described in Section 3 and a sensitivity analysis on the impact of the video codecs is shown in Appendix A.3.

We ran each streaming algorithm twice, i.e., using their original settings (i.e., without PSQA) for the first time, and then applying PSQA for a second time. Both of the runs executed 1000 streaming session traces (over 7 days) where each session was evaluated by a random QoE function from (1), (2), (3), (4), (5), (6). Note that the throughput trace data used for each run were exactly the same.

Table 16 summarizes the absolute QoE achieved by the original version (denoted by "Original") and the PSQA-optimized version (denoted by "PSQA") of each algorithm. Accordingly, we calculated the proportion of the QoE improvements. We observed that PSQA is able to improve the QoE of *all* the six streaming algorithms significantly, ranging from 18.9% to 58.0% (a bandwidth consumption analysis is shown in Appendix A.4). In summary, the real experimental results verify PSQA's efficacy in the practical operational environments and demonstrate that PSQA has the ability to offer an immediate and practical solution for real streaming platforms.

7 Summary and Future Works

The PSQA framework developed in this study introduces a novel paradigm to QoE optimization. For any existing adaptive streaming algorithms, PSQA is able to automatically optimize/tune their adaptation logic with respect to any format of QoE objectives. This enables the optimized streaming algorithms to achieve consistently better QoE performance and stronger robustness across different streaming scenarios. Moreover, PSQA offers a useful tool to systematically investigate and explore the impact of different kinds of adaptation logic on various QoE metrics. This can potentially provide more insights on the design of new adaptive streaming algorithms.

There are two directions for future work. First, this work only incorporates the cases of on-demand and live video streaming. However, PSQA is a generic framework that can be potentially extended to any other type of streaming service, such as 360-degree video streaming, short video streaming, and so on. Second, since PSQA is decoupled from the underlying streaming algorithms, we can further modify the latter to explore the use of other machine learning or heuristic paradigms to further improve QoE.

Acknowledgment

The authors wish to thank the associate editor and the anonymous reviewers for their insightful comments in improving this paper. The work presented in this article is supported by Centre for Advances in Reliability and Safety (CAiRS) admitted under AIR@InnoHK Research Cluster.

References

[1] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecase Update, 2017-2022*. San Jose, California, USA: Cisco Inc. Mar. 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] 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, Feb. 2021.

[3] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. ACM Conf. Multimedia Syst. (MMSys'11)*, San Jose, USA, Feb. 2011, pp. 157–168.

[4] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, Efficiency, and stability in HTTP-based adaptive video streaming with festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014.

[5] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1698–1711, Aug. 2020.

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

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

[8] G. Zhang and J. Y. B. Lee, "Ensemble adaptive streaming – A new paradigm to generate streaming algorithms via specializations," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1346–1358, Jun. 2020.

[9] Z. Akhtar, Y. S. Nam, and R. Govindan, "Oboe: Auto-tuning video ABR algorithms to network conditions," in *Proc. ACM SIGCOMM*, Budapest, Hungary, Aug. 2018, pp. 44–58.

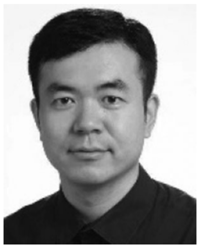
[10] Z. Li, A. Aaron, L. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," Jun. 2016. [Online]. Available: http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html

[11] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *Int. J. Hum.-Comput. Stud.*, vol. 64, no. 8, pp. 637–647, Aug. 2006.
[12] F. Dobrian *et al.*, "Understanding the impact of video quality on user engagement," in *Proc. ACM SIGCOMM*, Toronto, Canada, Aug. 2011, pp. 362–373.
[13] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the quality of experience of HTTP video streaming," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, Dublin, Ireland, May 2011, pp. 485–492.
[14] G. Yi *et al.*, "The ACM multimedia 2019 live video streaming grand challenge," in *Proc. ACM Int. Conf. Multimedia*, Nice, France, Oct. 2019, pp. 2622–2626.
[15] X. Liu *et al.*, "A case for a coordinated internet video control plane," in *Proc. ACM SIGCOMM*, Helsinki, Finland, Aug. 2012, pp. 359–370.
[16] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao, "Deriving and validating user experience model for DASH video streaming," *IEEE Trans. Broadcast.*, vol. 61, no. 4, Dec. 2015, pp. 651–665.
[17] J. Joskowicz and J. C. L. Ardao, "A parametric model for perceptual video quality estimation," *Telecommun. Syst.*, vol. 49, no. 1, pp. 49–62, 2012.
[18] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 339–350, Oct. 2013.
[19] T. Stockhammer, "Dynamic adaptive streaming over HTTP: Standards and design principles," in *Proc. ACM Multimedia Syst.*, San Jose, USA, Feb. 2011, pp. 133–144.
[20] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1842–1866, Mar. 2017.
[21] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over HTTP," *IEEE Commun. Surv. Tut.*, vol. 21, no. 1, pp. 562–585, Jan. 2019.
[22] V. Mnih, A. P. Badia, M. Mirza, and A. Graves, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 2016, pp. 1928–1937.
[23] J. R. Koza, "Genetic programming as A means for programming computers by natural selection," *Springer Statist. Comput.*, vol. 4, no. 2, Jun. 1994, pp. 87–112.
[24] Y. Liu and J. Y. B. Lee, "A unified framework for automatic quality-of-experience optimization in mobile video streaming," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.
[25] dash.js. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js/wiki
[26] Trace Driven Simulator. [Online]. Available: https://github.com/hongzimao/pensieve/tree/master/sim
[27] G. Zhang, R. K. H. Ngan, and J. Y. B. Lee, "EmuStream - An end-to-end platform for streaming video performance measurement," *IEEE Access*, vol. 8, pp. 669–80, Dec. 2019.
[28] Mobile Throughput Trace Data. [Online]. Available: https://github.com/Zhang-Guanghui-Nick/TCPtrace/releases/tag/TCP-trace
[29] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: Analysis and applications," in *Proc. ACM Multimedia Syst.*, Oslo, Norway, Feb. 2013, pp. 114–118.
[30] Offline Optimal QoE Calculated by Dynamic Programming. [Online]. Available: https://github.com/hongzimao/pensieve/blob/master/test/dp.py
[31] *Best Practices for Creating and Deploying HTTP Live Streaming Media for the Iphone and Ipad*, Cupertino, California, USA: Apple Inc, retrieved on Aug. 2016. [Online]. Available: https://developer.apple.com/library/ios/technotes/tn2224/_index.html
[32] Empirical Video Duration Distribution. [Online]. Available: https://github.com/mclab-cuhk/Early-depature-trace/blob/main/VideoDurationAll.txt
[33] Y. Liu and J. Y. B. Lee, "Post-streaming rate analysis - A new approach to mobile video streaming with predictable performance," *IEEE Trans. Mobile Comput.*, vol. 16, no. 12, pp. 3488–3501, Dec. 2017.
[34] Apache HTTP Server Project. May 2016. [Online]. Available: http://httpd.apache.org/
[35] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM Operating Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.
[36] An Improved Version of Dummynet. [Online]. Available: https://github.com/mclab-cuhk/netmap-ipfw
[37] Bayesian Optimization Library. [Online]. Available: https://github.com/fmfn/BayesianOptimization
[38] A. Elgabli and V. Aggarwal, "FastScan: Robust low-complexity rate adaptation algorithm for video streaming over HTTP," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 2240–2249, Jul. 2020.
[39] G. Zhang, J. Lee, K. Liu, H. Hu, and V. Aggarwal, "A unified framework for flexible playback latency control in live video streaming," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 12, pp. 3024–3037, Dec. 2021.
[40] Y. Zhao, Q. Shen, W. Li, T. Xu, W. Niu, and S. Xu, "Latency aware adaptive video streaming using ensemble deep reinforcement learning," in *Proc. ACM Int. Conf. Multimedia*, Nice, France, Oct. 2019, pp. 2647–2651.
[41] C. Gutterman, B. Fridman, T. Gilliland, Y. Hu, and G. Zussman, "Stallion: Video adaptation algorithm for low-latency video streaming," in *Proc. ACM Multimedia Syst. Conf.*, Istanbul, Turkey, May 2020, pp. 327–332.
[42] L. Sun, T. Zong, Y. Liu, Y. Wang, and H. Zhu, "Optimal strategies for live video streaming in the low-latency regime," in *Proc. IEEE Int. Conf. Netw. Protoc.*, Chicago, IL, USA, Oct. 2019, pp. 1–4.
[43] T. Karagkioules, R. Mekuria, D. Griffioen, and A. Wagenaar, "Online learning for low-latency adaptive streaming," in *Proc. ACM Multimedia Syst. Conf.*, Istanbul, Turkey, May 2020, pp. 315–320.
[44] Youtube encoding settings. [Online]. Available: https://support.google.com/youtube/answer/2853702?hl=zh-Hant#zippy=%2Cp%2Cp-fps
[45] G. Zhang, K. Liu, H. Hu, V. Aggarwal, and J. Lee, "Post-streaming wastage analysis - A data wastage aware framework in mobile video streaming," *IEEE Trans. Mobile Comput.*, early access, Mar. 30, 2021.
[46] A. Sundarrajan *et al.*, "Midgress-aware traffic provisioning for content delivery," in *Proc. USENIX Annu. Tech. Conf.*, Jul. 2020, pp. 543–557.
[47] Amazon CDN Pricing. [Online]. Available: https://aws.amazon.com/cloudfront/pricing/?nc1=h_ls
[48] C. Wang, A. Rizk, and M. Zink, "SQUAD: A spectrum-based quality adaptation for dynamic adaptive streaming over HTTP," in *Proc. Int. Conf. Multimedia Syst. (MMSys '16)*, May 2016, Klagenfurt am Wörthersee, Austria, pp. 1–12.
[49] M. Uhrina, J. Frnda, and L. Sevcik, "Impact of H. 264/AVC and H. 265/HEVC compression standards on the video quality for 4K resolution," *Adv. Elect. Electron. Eng.*, vol. 12, no. 4, pp. 368–376, 2014.

Guanghui Zhang received the MS degree in electronic science and technology from Peking University in 2016, and the PhD degree in information engineering from The Chinese University of Hong Kong in 2020. He is currently a research assistant professor with the Department of Computer Science, Hong Kong Baptist University. From 2020 to 2021, he was a postdoctoral fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, and with the Centre for Advances in Reliability and Safety, The Hong Kong Polytechnic University. His research interests include networking system, multimedia system, and machine learning.

Jie Zhang (Member, IEEE) was born in Anhui Province, China, in 1990. He received the BSc (with Hons.) degree in electrical engineering from Yunnan University in 2012, the MSc (with Hons.) degree in electrical engineering from Peking University, Beijing, China, in 2015, and the PhD degree in electrical engineering from the Delft University of Technology, Delft, The Netherlands, in 2020. He is currently an assistant professor with the National Engineering Laboratory for Speech and Language Information Processing, Faculty of Information Science and Technology, University of Science and Technology of China, Hefei, China. His current research interests include multi-microphone speech enhancement, sound source localization, binaural auditory, speech recognition, and speech processing over wireless (acoustic) sensor networks. He was the recipient of the Best Student Paper Award for his publication at the 10th IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM) in Sheffield, U.K.

Yan Liu received the BEng degree in communication engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2012, and the PhD degree in information engineering from the Chinese University of Hong Kong, Hong Kong, in 2016. He is currently a senior engineer with Cloud ARCH & Platform Dept., Tencent, China. His research interests include computer networks, including but not limited to Internet congestion control and video streaming.



Haibo Hu (Senior Member, IEEE) is currently an associate professor with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University and the programme leader of BSc (with Hons.) in information security. He has authored or coauthored more than 80 research papers in refereed journals, international conferences, and book chapters. His research interests include cybersecurity, data privacy, Internet of Things, and adversarial machine learning. As a principal investigator, he has received more than 20 million HK dollars of external research grants from Hong Kong and mainland China as of year 2020. He was in the organizing committee of many international conferences, such as ACM GIS 2021, 2020, IEEE ICDSC 2020, IEEE MDM 2019, DASFAA 2011, DaMEN 2011, 2013 and CloudDB 2011, and in the programme committee of dozens of international conferences and symposiums. He is the recipient of a number of titles and awards, including the IEEE MDM 2019 Best Paper Award, WAIM Distinguished Young Lecturer, ICDE 2020 Outstanding Reviewer, VLDB 2018 Distinguished Reviewer, ACM-HK Best PhD Paper, Microsoft Imagine Cup, and GS1 Internet of Things Award.

Jack Y. B. Lee (Senior Member, IEEE) received the BEng and PhD degrees in information engineering from the Chinese University of Hong Kong, Hong Kong, in 1993 and 1997, respectively. He is currently an associate professor with the Department of Information, Chinese University of Hong Kong. His research interests include multimedia communications systems, mobile communications, protocols, and applications. He specializes in tackling research challenges arising from real-world systems. He works closely with the industry to uncover new research challenges and opportunities for new services and applications. Several of the systems research from his lab have been adopted and deployed by the industry.

Vaneet Aggarwal (Senior Member, IEEE) received the BTech degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 2005, and the MA and PhD degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 2007 and 2010, respectively. From 2010 to 2014, he was a senior member of the Technical Staff Research with AT&T Labs Research, Bedminster, NJ, USA. He was an adjunct assistant professor with Columbia University, NY, USA, from 2013 to 2014, and an adjunct professor with IISc Bangalore, India, from 2018 to 2019. He is currently a faculty with Purdue University, West Lafayette, IN, USA. His research interests include communications and networking, video streaming, cloud computing, and machine learning. He was the recipient of the Princeton University's Porter Ogden Jacobus Honorific Fellowship in 2009, AT&T Vice President Excellence Award in 2012, AT&T Senior Vice President Excellence Award in 2014, 2017 Jack Neubauer Memorial Award recognizing the Best Systems Paper published in the *IEEE Transactions on Vehicular Technology*, and 2018 Infocom Workshop HotPOST Best Paper Award. He is on the editorial board of *IEEE Transactions on Communications*, *IEEE Transactions on Green Communications and Networking*, and *IEEE/ACM Transactions on Networking*.