IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 24, 2020, accepted January 1, 2021, date of publication January 18, 2021, date of current version January 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3052552

Balancing Quality of Experience and Traffic Volume in Adaptive Bitrate Streaming

TAKUTO KIMURA[1], TATSUAKI KIMURA[2], (Member, IEEE), ARIFUMI MATSUMOTO[3], AND KAZUHISA YAMAGISHI[1]
[1]NTT Network Technology Laboratories, NTT Corporation, Musashino, Tokyo 180-8585, Japan
[2]Department of Information and Communications Technology, Osaka University, Osaka 565-0871, Japan
[3]Japan Infra Waymark Corporation, Tokyo 104-0061, Japan

Corresponding author: Takuto Kimura (takuto.kimura.mx@hco.ntt.co.jp)

ABSTRACT Adaptive bitrate (ABR) streaming services have spread with advances in the codec, video streaming, and network technologies. For smooth video playback in ABR streaming services, a video player runs an ABR algorithm, which dynamically adjusts the bitrate of video data on the basis of the statuses of the network and player. Existing ABR algorithms calculate a suitable bitrate to maximize the quality of experience (QoE). However, providing a high-QoE video increases network investment costs and content delivery network (CDN) usage fees. According to a survey conducted by the Streaming Video Alliance, mobile users prefer low-traffic videos to high-QoE videos. To reduce traffic volume, commercial video-streaming services enable users to set an upper limit of the bitrate. However, this cannot always achieve the required QoE because they cannot select a high bitrate even when the communication environment improves during viewing. In this paper, we propose BANQUET, a novel ABR algorithm that can reduce the traffic volume while maintaining QoE above the *target QoE*. The target QoE can be set by users or streaming providers considering user's preferences or CDN budget. BANQUET selects a suitable bitrate by estimating QoE and traffic volume that will be experienced by all the bitrate patterns for the next several chunks on the basis of future throughput and a buffer transition calculation. The trace-based simulation showed that BANQUET reduces traffic volume 18.3%–51.2% on average in the mobile environment and 1.2%–38.3% in the broadband environment while maintaining QoE the same as or better than existing algorithms.

INDEX TERMS Adaptive bitrate streaming, quality of experience, traffic volume reduction, video streaming.

I. INTRODUCTION

Video traffic volume is expected to grow four-fold from 2017 to 2022 and account for 82% of all traffic in 2022 [1]. However, this increase in video traffic may cause network congestion and thus degrade the video Quality of Experience (QoE). Studies have shown that users tend to abandon watching videos when the QoE degrades [2], [3]. Thus, video-streaming providers need to provide high-QoE videos to prevent service cancellations. On the other hand, such high-QoE videos increase network investment costs and Content Delivery Network (CDN) usage fees for network providers and streaming providers. Also, a survey conducted by the Streaming Video Alliance in North America [4] found that mobile users prefer low-traffic videos to high-QoE videos because they want to watch more videos with the pay-per-use (e.g., US $10 per 1 GB) or data capped (e.g., 1 or 3 GB per month) plans. Therefore, QoE and traffic volume need to be balanced in video streaming.

The associate editor coordinating the review of this manuscript and approving it for publication was Martin Reisslein.

Video-streaming technologies can be classified into Real-time Transport Protocol (RTP)-based streaming [5] and Adaptive Bitrate Streaming (ABR). The RTP-based streaming can be used in combination with H.264/AVC [6] or H.264/SVC [7]. RTP-based streaming with H.264/AVC is generally used in Internet Protocol Television (IPTV) since IPTV must satisfy the strict quality requirements. The video data is streamed in real-time, but the video may be distorted since the packets may be dropped since a User Datagram Protocol (UDP) is used as the transport layer. Thus, IPTV conceals the lost packet with Forward Error Correction (FEC). In RTP-streaming with H.264/SVC, the video streaming server sends a base layer with lower quality and then sends an enhancement layer with higher quality. If the enhancement layer packets are dropped, viewers perceive degraded quality since the base layer is displayed, but if base layer packets are dropped, quality is seriously distorted due to packet loss. In contrast, ABR streams video data with HTTP Live Streaming (HLS) [8] or Dynamic Adaptive Streaming over HTTP (DASH) [9] on a reliable transport layer such as Transmission Control Protocol (TCP) or QUIC/UDP [10]. The ABR system changes the bitrate of streamed video depending on the statuses of the network and the player. Thus, the video is not distorted, but the bitrate of the streamed videos may be changed and rebuffering events may occur. To deal with network quality degradation, ABR is more commonly used than RTP-based streaming with H.264/SVC. In fact, commercial video-streaming services such as Hulu [11], YouTube [12], and Netflix [13] stream videos with the ABR system.

The simplest ABR system consists of a video-streaming server and a client. The streaming server stores video data, which is encoded with different bitrate settings (e.g., 200, 500, and 1000 kbps). Each video data is divided into chunks, which are a few seconds of video data (e.g., 3 sec). When the user plays a video, the client downloads the chunk with a suitable bitrate selected by the ABR algorithm and plays it sequentially. Thus, the client can switch to the chunk with a lower bitrate when the throughput becomes low or the remaining player buffer length becomes short. Downloading chunks with high bitrate improves QoE but also leads to an increase in the traffic volume. Therefore, the ABR algorithm is the key to controlling QoE and traffic volume in video streaming.

Because of its importance, many ABR algorithms have been proposed that aim to achieve high-QoE. PANDA [14] selects the bitrate on the basis of the measured throughput. In contrast, Buffer-based Algorithm (BBA) [15] and Buffer Occupancy-based Lyapunov Algorithm (BOLA) [16] select the bitrate on the basis of the player's buffer length. Model Prediction Control (MPC) [17], Cross Session Stateful Predictor (CS2P) [18], and Pensieve [19] use both throughput and buffer length to select the bitrate. As a meta-algorithm, Oboe [20] tunes the parameters of these methods by solving the optimization problem. Since such existing algorithms only aim to achieve high-QoE, the traffic volume is not taken into account.

On the other hand, several efforts have been made in commercial video-streaming services to reduce traffic volume. For example, DAZN [21], Abema TV [22], and Hulu [11] have implemented a traffic volume reduction mode that allows users to reduce the traffic volume by not requesting the chunk with a high bitrate. Netflix [13] allows users to adjust the data usage settings by selecting the quality level from a list of high, medium, and low options [23]. YouTube [12] users can set the maximum bitrate by setting the corresponding resolution. Although the setting manners are different, all of these methods reduce the traffic volume by not requesting the chunk with a high bitrate. Even if we use these methods, they do not always achieve the QoE required by users or streaming providers. This is because these methods cannot select a high bitrate when the communication environment improves during the viewing. For example, if a user starts watching a video while on a train stopping at a crowded station, the existing methods select a low bitrate because of the low throughput. Then, if the train leaves the station and the throughput changes from low to high, existing methods cannot select the high bitrate due to the maximum bitrate limitation. Thus, the QoE may be below the required QoE level. To achieve the required QoE level, the ABR algorithm needs to select the bitrate higher than the upper limit in this case. Therefore, we need to construct an ABR algorithm that reduces the traffic volume while maintaining the required QoE level.

We propose a BAlaNcing QUality of Experience and Traffic volume algorithm, named BANQUET. BANQUET is a novel ABR algorithm that enables users or streaming providers to control the balance of the QoE and traffic volume. Assuming a *target QoE*, BANQUET adaptively selects the bitrate that minimizes the traffic volume while maintaining the QoE above the target QoE. The target QoE means the quality that needs to be achieved. The target QoE can be defined as various kinds of QoE metrics such as a Mean Opinion Score (MOS) on a 5-point Absolute Category Rating (ACR) scale [24], [25] or utility score used in the previous research [16]–[20]. Users or streaming providers can preset the target QoE on the basis of the user's contracted mobile plan, user's attributes (e.g., premium user or not), the tariff setting of the CDN, and so on. Before receiving chunks, BANQUET estimates the QoE and traffic volume of all the possible bitrate patterns for the next several chunks by estimating the future throughput series and the player's buffer transition. It then selects the bitrate of the next chunk from the bitrate pattern with the smallest traffic data while satisfying the target QoE.

We evaluate BANQUET through trace-based simulation using throughput traces collected in the actual mobile and broadband environments. In the simulation, we evaluate how BANQUET reduces the traffic volume while maintaining the same QoE as existing ABR algorithms. Furthermore, we evaluate the applicability of BANQUET to several QoE metrics, bitrate settings, and optimized hyper-parameters and evaluate the computational cost on an actual smartphone.

Compared with the earlier paper [26], this paper extends the evaluation significantly. This extension includes the evaluation results for the various network environments, QoE metrics, bitrate settings, and the hyper-parameter optimization.

The remainder of this paper is constructed as follows. First, we describe related work and its limitations in Section II. We then give details on BANQUET in Section III. We evaluate BANQUET through a trace-based simulation and discuss it in Section IV. Finally, we conclude the paper and describe future work in Section V.

II. RELATED WORK

A. QoE METRICS IN ABR

Many QoE-estimation methods have been proposed and standardized [24], [25], [27]–[33]. As described in Section I, the video image streamed with the ABR system is not distorted, but the bitrate of the streamed videos may change and the rebuffering events may occur. Thus, the QoE-estimation method for the ABR system should consider the effect of selected bitrates, bitrate switching, and rebuffering events.

There are two main categories of the QoE-estimation methods: utility score and MOS. The utility score is mainly used to evaluate existing ABR algorithms [16]–[20], and the MOS is used to monitor the service quality. The equation's form is different in the MOS-estimation models and utility functions, but the input variables are the same: the audio and bitrate series, resolution series, framerate series, and rebuffering-related information.

The utility score is calculated as a weighted sum of the bitrate utility, the rebuffering penalty, and the bitrate switching penalty. The form of the utility function is expressed as

$$Utility = \frac{1}{M}\sum_{n=1}^{M} Q(R_n) - \frac{1}{M}\mu T - \frac{1}{M}\sum_{n=1}^{M-1} |\Delta Q(R_n)|, \quad (1)$$

where R_n is the bitrate of the n-th chunk and $Q(\cdot)$ is the bitrate utility function, which outputs the per-chunk quality. $\Delta Q(R_n)$ is defined as $Q(R_{n+1}) - Q(R_n)$, μ is the weight parameter for the rebuffering penalty, T is the sum of the rebuffering time, and M is the number of chunks included in the whole video. In this function, the first term indicates the bitrate utility, the second term indicates the rebuffering penalty, and the third term indicates the smoothness penalty. The bitrate utility and bitrate switching penalty are calculated through the bitrate utility function $Q(\cdot)$. The utility function is widely used to evaluate the existing ABR algorithms [16]–[20] since we can obtain variants of this model easily by changing the bitrate utility function and the weight for penalties.

The MOS is calculated with MOS-estimation models constructed through subjective quality assessment tests. Since MOS directly reflects the perceptual characteristics of humans, many MOS-estimation models have been studied. These models differ in inputs, estimation accuracy, and computational costs. In the ABR system, the player cannot compare the encoded video with the original. Thus, a no-reference model needs to be used that estimates MOS only from the encoded video. No-reference models can be categorized into metadata-based models [24], [25], [29], bitstream-based models [29]–[31], and pixel-based models [32], [33] depending on the input information. Since the ABR algorithm runs in the client terminal such as smartphones, the cost of collecting the input should be low. Thus, metadata-based models are suitable to estimate MOS in the ABR system. There are metadata-based models such as ITU-T P.1203 mode 0 [29] and a model proposed by Yamagishi and Hayashi [24], Yamagishi [25]. According to Lebreton and Yamagishi [34], there is no statistical difference in their estimation accuracy. However, the standardized model has a higher computational cost than the model by Yamagishi and Hayashi because the standardized model uses a machine learning-based method. Therefore, the model proposed by Yamagishi and Hayashi is the most suitable to estimate the MOS in the ABR system. This model is formulated as

$$\begin{aligned}
MOS &= 1 + (O.35 - 1)\cdot S,\\
S &= \exp\left(-\frac{N_r}{s_1}\right)\cdot \exp\left(-\frac{L}{s_2\cdot D}\right)\cdot \exp\left(-\frac{A}{s_3\cdot D}\right),\\
D &= \sum_{i}^{M} l_i\\
O.35 &= \frac{\sum_{t=1}^{D} w_1(t)\cdot w_2(t)\cdot O.34(t)}{\sum_{t=1}^{D} w_1(t)\cdot w_2(t)},\\
w_1(t) &= t_1 + t_2\cdot \exp\left(\frac{u(t)}{t_3}\right),\\
u(t) &= \frac{t}{D},\\
w_2(t) &= t_4 - t_5\cdot O.34(t),\\
O.34(t) &= f(av_1 + av_2\cdot O.21(t) + av_3\cdot O.22(t)\\
&\quad + av_4\cdot O.21(t)\cdot O.22(t), 1, 5),\\
O.21(t) &= f\left(a_1 + \frac{1-a_1}{1+\left(\frac{ar(t)}{a_2}\right)^{a_3}}, 1, 5\right),\\
O.22(t) &= X(t) + \frac{1-X(t)}{1+\left(\frac{vr(t)}{Y(t)}\right)^{v_1}},\\
X(t) &= f\left(4\cdot\frac{1-\exp(-v_3\cdot rs(t))\cdot fr(t)}{v_2 + rs(t)} + 1, 1, 5\right),\\
Y(t) &= \frac{v_4\cdot rs(t) + v_6\cdot \log_{10}(v_7\cdot fr(t) + 1)}{1 - \exp(v_5\cdot rs(t))},\\
f(x,a,b) &= \begin{cases} a & x < a \\ x & a \le x < b \\ b & b \le x, \end{cases}
\end{aligned} \quad (2)$$

where $O.35$ is the audiovisual coding quality, S is the quality with rebuffering effect, D is the video duration, l_i is the length of i-th chunk, N is the number of chunks, N_r is the number of rebuffering events, L is the total length of the rebuffering, A is the average rebuffering interval, T is the content length, $O.34(t)$ is the audiovisual quality at t, $w_1(t)$ and $w_2(t)$ are weights dependent on t, $O.21(t)$ is the audio quality at t, $ar(t)$ is the audio bitrate at t, $O.22(t)$ is the video quality at t, $vr(t)$ is the video bitrate at t, $X(t)$ is the maximum video quality at t, $rs(t)$ is the number of pixels in the video at t, $fr(t)$ is the framerate at t, and s_1 to s_3, t_1 to t_5, av_1 to av_4, a_1 to a_3, and v_1 to v_7 are coefficient values.

B. ABR ALGORITHM

Many ABR algorithms have been proposed and can be classified into three categories: rate-based, buffer-based, and hybrid-based algorithms.

Rate-based algorithms select the bitrate by using only the throughput information measured during the chunk download. An algorithm adopted in dash.js [35] selects the maximum bitrate that does not exceed the estimated throughput. This algorithm selects a high bitrate when the throughput is high and vice versa. Thus, if the throughput fluctuates largely, the selected bitrate also switches frequently. Because such a bitrate switching degrades QoE [36], rate-based algorithms may degrade QoE.

Buffer-based algorithms select the bitrate only on the basis of the player's remaining buffer length. These algorithms create a map, which represents a correspondence between the remaining buffer length and the bitrate. Various algorithms for making maps have been proposed. BBA [15], which was used by Netflix [13] in the past [15], makes the map by assuming a linear relationship between the buffer length and the appropriate bitrate. BOLA [16], which is used as the default algorithm for dash.js [35], makes the map by using the Lyapunov optimization. Since these algorithms depend on the buffer length, they are less affected by short-term throughput fluctuations than rate-based algorithms. However, if the buffer is long, they try to select a high bitrate even when the throughput is low. Selecting a bitrate higher than the throughput leads to a shorter buffer, and the algorithm may have to select the lower bitrate at the next selection. This behavior results in an increase in bitrate switching and rebuffering. Thus, buffer-based algorithms may also degrade QoE due to the bitrate switching and the rebuffering.

As a unified approach, hybrid-based algorithms select the bitrate using both the throughput information and the remaining buffer length. MPC [17] and CS2P [18] formulate the bitrate-selection problem by using model prediction control. Although these methods can suppress bitrate fluctuations, they tend to select a low bitrate because they predict the throughput series conservatively to avoid rebuffering. On the other hand, Pensieve [19] uses deep reinforcement learning instead of predicting the throughput series to select the bitrate. Although Pensieve improves the QoE without the throughput prediction, it does not select a bitrate on the basis of the traffic volume. In addition, Oboe [20] optimizes the parameters of these algorithms as a meta-algorithm.

As described above, these algorithms do not consider the traffic volume aspect. Since QoE is determined by multiple factors such as selected bitrates, rebuffering, and bitrate switching [36], there may be room to reduce the traffic volume while maintaining the same QoE. In addition, existing algorithms may provide a higher QoE than that required by the users or streaming providers since they do not always require the high-QoE video as described in Section I.

C. DATA SAVING IN ABR

There are several methods to reduce traffic volume in the ABR system. They can be categorized into analysis-based and non-analysis-based approaches.

Analysis-based approaches try to reduce the traffic volume by analyzing the video content [37], [38]. Content-aware encoding [37] adopted in Netflix [13] optimizes the video encoding settings by analyzing the characteristics of the videos. Although this method contributes to avoiding excessive traffic volume, huge computational costs will be incurred to re-encode the encoded videos. Statistically Indifferent Quality Variation (SIQV) [38] can avoid the re-encoding process by removing the selectable bitrates on the basis of the statistical quality difference of each video chunk. If there is no statistical difference in quality between adjacent selectable bitrates, SIQV removes the higher bitrate from the candidates. SIQV can reduce the traffic volume while maintaining the QoE, but the computational cost is still high since SIQV must analyze the encoded video content. Furthermore, if the ABR algorithm chooses a higher rate when we use these analysis-based methods, it may achieve a higher QoE than the users or streaming providers need and result in an increase in traffic volume.

Non-analysis-based approaches set the upper limit of the selectable bitrates without analyzing the video content. These approaches are adopted in commercial video streaming services such as DAZN [21], Abema TV [22], Hulu [11], Netflix [13], and YouTube [12]. DAZN, Abema TV and Hulu have a "data saving mode" or "data saver" to reduce the traffic volume. Users can turn this mode on to suppress the traffic volume. Netflix allows users to adjust the data usage settings by selecting the quality from a list of high, medium, and low options [23]. YouTube users can select the maximum selectable bitrate by setting the corresponding resolution. Although the setting manners are different, these services reduce the traffic volume by setting the maximum bitrate that ABR algorithms can select. However, these methods may not always achieve the user's required QoE. This is because they cannot select a high bitrate even if the communication environment improves during viewing as described in Section I.

III. BANQUET

To reduce the traffic volume while maintaining the QoE above the target QoE, we propose BANQUET. We first explain how BANQUET works in the video-streaming system and explain its calculation step in Subsection III-A. Then, we describe the details of the bitrate-selection step in Subsection III-B.

A. OVERVIEW OF BANQUET

In this subsection, we describe how BANQUET works in the video-streaming system.

Figure 1 shows a video-streaming system using BANQUET. In this system, the client and the streaming server communicate with the HTTP protocol on reliable transport, e.g., TCP or QUIC/UDP [10], the audio is encoded by AAC-LC [39], and the videos are encoded by H.264/AVC [40]. BANQUET is implemented inside the video player application. When the player requests the next chunk, the player sends a bitrate-calculation request to BANQUET together with the selectable bitrates, buffer information, and measured throughput information (P1).

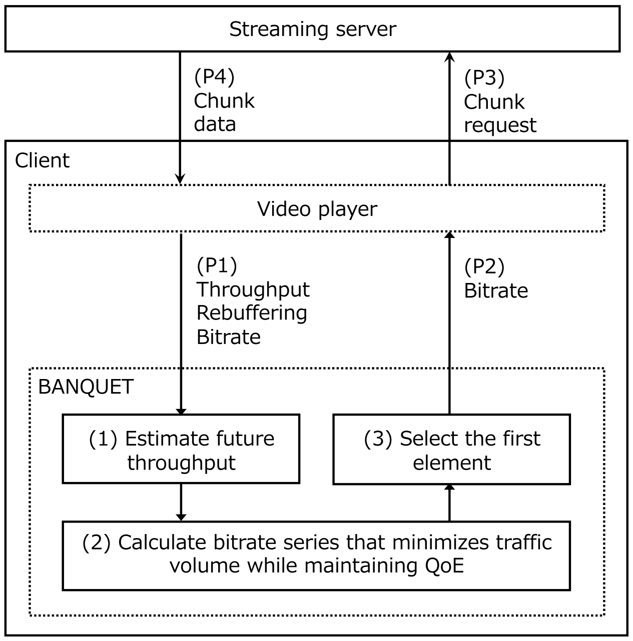

Streaming server
(P4) Chunk data
(P3) Chunk request
Client
Video player
(P1) Throughput Rebuffering Bitrate
(P2) Bitrate
BANQUET
(1) Estimate future throughput
(3) Select the first element
(2) Calculate bitrate series that minimizes traffic volume while maintaining QoE

FIGURE 1. **Overview of video-streaming system using BANQUET.**

On the basis of this information, BANQUET calculates a suitable bitrate and notifies the player (P2). Then, the player requests the chunk with the notified bitrate (P3) and receives the chunk (P4). This process is performed every time the player requests a chunk. The selectable bitrates can be obtained from the mpd file or the m3u8 file in the case of DASH [9] or HLS [8], respectively. Since the audio bitrate generally corresponds to the video bitrate or resolution, BANQUET assumes that each bitrate value in the selectable bitrates indicates the sum of the video and audio bitrate.

TABLE 1. **Notations.**

Notation	Meaning
t	Time (sec)
t_0	Calculation start time (sec)
b_t	Buffer length at t (sec)
$\boldsymbol{r}$	Candidate bitrate series
$\mathcal{R}$	Set of candidate bitrate series
r_{max}	The highest bitrate among the selectable bitrates (bps)
v_i	Average audiovisual bitrate of the i-th chunk (bps)
l_i	Length of the i-th chunk (sec)
$\boldsymbol{c}$	Estimated throughput series
y_t	Bits that cannot be downloaded at t
n_t	Number of chunks already started downloading
$g_{i,t}$	Sum of downloaded bits by t at the i-th chunk
$h_{i,t}$	Buffer length at t when the i-th chunk is downloaded not considering buffer consumption (sec)
h	Throughput prediction horizon (sec)
M	The number of chunks already received
q_t	QoE at t
T_{high}	Upper limit of buffer (sec)
T_q	Target QoE
T_d	Threshold of length of bitrate-series candidates
N_c	The number of chunks to estimate the future throughput

We next describe the calculation procedure of BANQUET (P2). The notations are summarized in Table 1. We use t as the index of time and i as the index of the chunk. In this procedure, BANQUET calculates the bitrate in three steps.

First, BANQUET estimates a future throughput series $\boldsymbol{c}_{t_0} = (c_{t_0+1}, \cdots, c_{t_0+h})$ where t_0 and h denote the calculation start time and throughput prediction horizon. If the latest N_c measured throughput values are denoted as $\bar{c}_{t_0-N_c+1}, \cdots, \bar{c}_{t_0-N_c+1}$ from the oldest to latest, c_{t_0+1} is estimated by calculating the harmonic mean of $\bar{c}_{t_0-N_c+1}, \cdots, \bar{c}_{t_0}$. Then, c_{t_0+2} is calculated as the harmonic mean of $\bar{c}_{t_0-N_c+2}, \cdots, \bar{c}_{t_0}$ and c_{t_0+1}. As described, the future throughput series are estimated using the harmonic mean of the latest N_c measured or estimated throughput values sequentially. We use this method because it is robust against outliers [17], [41].

Second, BANQUET calculates the suitable bitrate series that minimizes the traffic volume while maintaining QoE higher than the target QoE, T_q by using information from the video player. BANQUET calculates the QoE and the buffer's transition considering chunks already received by using this information. The QoE can be calculated with any models or functions such as MOS-estimation models [24], [29] or utility functions [16]–[19] described in Subsection II-A. The bitrate series is expressed as the vector of bitrate (e.g., (500, 1000, 1000, 500)). If no bitrate series satisfies the QoE constraint, BANQUET selects the bitrate series that achieves the highest QoE. We chose this policy to ensure that BANQUET does not select the lowest bitrate to reduce traffic volume if no bitrate series satisfies the target QoE. The details of this step are given in Subsection III-B.

Third, BANQUET selects the bitrate for the next chunk from the most suitable bitrate series.

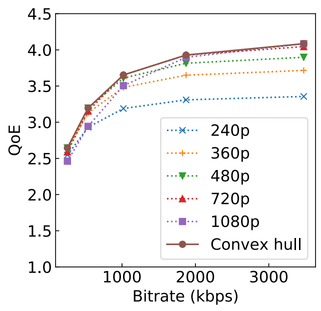


FIGURE 2. **Example of the correlation between the bitrate and QoE for each resolution and convex hull.**

The target QoE T_q can be set by users or streaming providers with any policies. For example, the service providers can set the target QoE in accordance with the budget available for CDN. Figure 2 shows the sample correlation between the bitrate and QoE for each resolution and *convex hull* [37], which is the maximum QoE for each bitrate. The QoE is calculated by using the MOS-estimation model [34] introduced in Subsection II-A with the assumption that the ABR algorithm selects the same bitrate for all chunks from five available bitrates (e.g., 256, 538, 1019, 1873, 3476 kbps), and there is no rebuffering. By multiplying the estimated total viewing time in a month and the unit price of the CDN cost to the bitate, we can obtain the correlation between the CDN cost and QoE. Hence, the streaming providers can set the target QoE T_q depending on the CDN cost budget by transferring the budget to QoE by using the convex hull. This policy is an example, and the streaming providers can set the target QoE depending on other policies that consider the user's preferences, contracted plans (e.g., premium or not), and so on. The appropriate setting policy of the target QoE T_q is out of this paper's scope.

B. DETAILS OF BANQUET

In this subsection, we describe the second step of the procedure (P2) described in Subsection III-A. To explain how to calculate the most suitable bitrate series, we first give an example of the calculation.

In this example, BANQUET can select the bitrate from 100, 500, and 1000 kbps, the first chunk with 500 kbps is already received, and the chunk length is fixed as 3 seconds. The goal of this example is to select a bitrate series that minimizes the traffic volume while maintaining QoE higher than 3.5 ($T_q = 3.5$). First, BANQUET makes a set consisting of all the candidate bitrate series including received chunks with the length up to T_d, which is the threshold of length of the candidate bitrate series. In this example, $T_d = 4$ and the first element of all the candidates is fixed as 500 kbps. Since there are three candidate bitrates for each future chunk, the size of this set is $3^4 = 81$. Especially, we focus on three sample bitrate series $(500, 100, 500, 500, 1000)$ (Series A), $(500, 500, 500, 1000, 500)$ (Series B), and $(500, 100, 100, 100, 100)$ (Series C). For each series, BANQUET calculates the QoE and the traffic volume when all the chunks with the candidate bitrate series are received. To calculate the QoE, BANQUET estimates the transition of the buffer length. The buffer length for each time is calculated by using the bitrate and estimated throughput series. By counting the time when the buffer length reaches zero, BANQUET can calculate the number of rebuffering events and the rebuffering time. This rebuffering information and the selected bitrate series are used to calculate the QoE. For example, QoE for Series A, B, and C can be calculated as 3.6, 3.8, and 2.5, respectively. The traffic volume can be calculated as the sum of the products of each element of the bitrate series and chunk length. For example, the traffic volume for Series A, B, and C can be calculated as 975, 1125, and 337.5 kB, respectively. After the calculation, BANQUET removes the bitrate series that does not achieve a QoE above 3.5. In this case, Series C is excluded from the candidates. Finally, BANQUET selects the series that has less traffic volume. Though Series B achieves the highest QoE, BANQUET selects Series A as the suitable bitrate series since it has less traffic volume than Series B. Note that the calculation time may be longer when the h or T_d is large since BANQUET searches for the suitable bitrate series on the basis of a brute-force search. Thus, we evaluate the calculation time and analyze the decision policy of h and T_d to reduce the calculation time in Subsection IV-D.

Algorithm 1 Bitrate-Series Calculation Algorithm

```
Input: $\boldsymbol{c}, T_q, T_d, h$
Output: $\hat{\boldsymbol{r}}$
 1: Initialize $\hat{\boldsymbol{r}}$
 2: $t \leftarrow t_0$
 3: $n_{t_0} \leftarrow M, y_{t_0} \leftarrow 0$
 4: $\mathcal{R} \leftarrow$ all the bitrate patterns up to the next $T_d$ chunks
 5: for each $\boldsymbol{r} \in \mathcal{R}$ do
 6:     while $t \leq t_0 + h$ do
 7:         if $b_{t-1} < T_{high}$ then
 8:             Update $b_t, n_t, y_t$
 9:         else
10:             $b_t \leftarrow b_{t-1} - 1$
11:         end if
12:         $t \leftarrow t + 1$
13:     end while
14:     Calculate $q_t$
15:     if $(T_q \leq q_t) \wedge (\sum r_i l_i \leq \sum \hat{r}_i l_i)$ then
16:         $\hat{\boldsymbol{r}} \leftarrow \boldsymbol{r}$
17:     end if
18: end for
```

Algorithm 1 is a pseudo code that calculates the suitable bitrate series. The input of this algorithm is the estimated throughput series $\boldsymbol{c}_{t_0}$, T_q, T_d, and h. Threshold T_d is a parameter to limit the length of the candidate bitrate series. Lines 1 to 4 correspond to initialization steps. In line 1, candidate solution $\hat{r} = (r_1, \cdots, r_i, \cdots r_{M+T_d})$ is initialized with the bitrate series already selected for $1 \leq i \leq M$ and $r_i = r_{max}$ for $M+1 \leq i \leq M+T_d$. Here, M is the number of chunks already received and r_{max} is the highest bitrate among the selectable bitrates. Then, the set of candidate bitrate series $\mathcal{R}$ is set with all the patterns of the candidate bitrate series up to the next T_d chunks. The bitrate series of these candidates for $1 < i \leq M$ is the bitrate series already selected.

In lines 6 to 13, the buffer transition for each $\boldsymbol{r} \in \mathcal{R}$ is estimated. In this procedure, BANQUET calculates buffer length b_t by updating n_t and y_t. n_t is the number of chunks that have already started downloading, and y_t is the number of bits downloaded at time t. We give the details on update procedures (line 8) later. In line 14, BANQUET calculates QoE q_t on the basis of the candidate bitrate series $\boldsymbol{r}$ and the buffer length b_t. From lines 15 to 17, BANQUET updates $\hat{\boldsymbol{r}}$ if $\boldsymbol{r}$ produces lower traffic volume than $\hat{\boldsymbol{r}}$ while achieving the target QoE. In line 15, BANQUET calculates the traffic volume by multiplying the selected bitrate and the chunk length for each chunk and summing them. The traffic volume is the sum of the products of the selected bitrate r_i and the chunk length l_i. If not all the chunks can be received with the throughput series $\boldsymbol{c}_{t_0}$, BANQUET calculates the QoE and the traffic volume when receiving as many chunks as possible.

We now describe how to update b_t, n_t, and y_t in line 8. First, we calculate n_t. By definition, if the chunk that was being received from the previous time $t-1$ was not completely received at time t, $n_t = n_{t-1}$. Otherwise, the player receives as many chunks as possible unless the buffer length exceeds T_{high}, where T_{high} is the upper limit of the buffer length. Thus, n_t is calculated as

$$n_t = \begin{cases} n_{t-1} & y_{t-1} \geq c_t \\ \max\left\{m \mid g_{m-1,t} < c_t, h_{m-1,t} \leq T_{high}\right\} & y_{t-1} < c_t. \end{cases} \tag{3}$$

where $g_{i,t}$ and $h_{i,t}$ represent the number of received bits and buffer length if the player receives the i-th chunk in this time t, respectively. $g_{i,t}$ and $h_{i,t}$ can be calculated as

$$g_{i,t} = y_{t-1} + \sum_{k=n_{t-1}+1}^{i} r_k l_k \tag{4}$$

$$h_{i,t} = b_{t-1} + \sum_{k=n_{t-1}+1}^{i} l_k, \tag{5}$$

where l_k denotes the length of the k-th chunk. l_k can be obtained from mpd file or m3u8 file in the case of DASH [9] and HLS [8], respectively.

Next, we calculate y_t, i.e., the number of bits that cannot be downloaded at time t. y_t is calculated to estimate how bits should be received in the next time slot. If the buffer length reaches T_{high} while receiving chunks, $y_t = 0$. Otherwise, y_t is calculated as the difference between the received bits in this time slot and the network bandwidth. Thus, y_t is formulated as

$$y_t = \max(g_{n_t,t} - \tilde{c}_t, 0). \tag{6}$$

Note that $y_t = 0$ if $t = t_0$ since the calculation of BANQUET starts only if the latest chunk is completely downloaded.

By using the $h_{n_t,i}$, the buffer length b_t is calculated by consuming the buffer if the playback is started. Thus, b_t is expressed as

$$b_t = \begin{cases} h_{n_t,t} & \text{playback is not started} \\ h_{n_t,t} - 1 & \text{playback is started.} \end{cases} \tag{7}$$

Finally, BANQUET calculates q_t in line 14. Streaming providers can use any QoE-estimation model to calculate q_t. As described in Subsection III-A, streaming providers can use any QoE-estimation models such as the MOS-estimation model or utility function introduced in Subsection II-A. The MOS and utility can be calculated from the chunk information and the rebuffering information. The chunk information contains the video bitrate, video resolution, video framerate, and audio bitrate for each second. This information can be collected from the selected bitrate and mpd file for DASH [9] or m3u8 file for HLS [8], respectively. When the streaming provider streams the video with HLS, we may obtain only the audiovisual bitrate. In that case, we can calculate the video bitrate by subtracting the predefined audio bitrate (e.g., 128 kbps) from the audiovisual bitrate. The rebuffering information contains the number of rebuffering events, total rebuffering time, and the average rebuffering interval. These are calculated from the transition of the buffer length $\{b_t\}$. The number of rebuffering events is calculated by counting the number of times the buffer is depleted, the total rebuffering time is calculated by counting the times the buffer length is lower than T_{start} and the playback does not start, and the average rebuffering interval is calculated by averaging the rebuffering interval. By using this information, BANQUET calculates the QoE when the user views all the chunks with candidate bitrate series. The sample formulas of the QoE metric and the sample calculation results are given in Subsections II-A and IV-A, respectively.

IV. EVALUATION

We evaluated BANQUET through trace-based simulation. Since this evaluation is conducted on various combinations of settings as described in Sec. IV-A, the real-time evaluation takes too long (e.g., 3,888 years). Thus, BANQUET is evaluated on a chunk-level simulator written in C. This simulator does not simulate the low-level protocols such as TCP or UDP/QUIC and relies on the traces collected in the actual environment [42], [43]. This evaluation method is similar to Sabre [44] or Pensieve [19] and has been demonstrated that this approach evaluates ABR algorithms with the same accuracy as in a real environment [44].

A. SIMULATION SETTINGS

1) NETWORK TRACES

To show that BANQUET can reduce the traffic volume while maintaining the QoE regardless of the network environment, we conduct an extensive simulation with various network traces. These traces were measured by downloading sample files or accessing web sites using smartphones or PCs. For maintaining the variety of traces, two types of actual throughput traces were used in this simulation: one was collected in mobile networks [42] and the other in broadband networks [43].

The mobile traces were collected in Norway [42] by two mobile operators and for different mobility patterns, i.e., static, pedestrian, car, tram, and train. These traces include the average throughput value per second. We concatenated all the traces and divided the data into short traces lasting 300 seconds. In this simulation, traces whose average throughput is less than 0.2 Mbps were excluded, and 473 throughput traces were obtained. This preprocessing is the same as that in previous work [19] and is done to exclude trivial results; if the average throughput is under the lowest bitrate, all the ABR algorithms select the lowest bitrate. The preprocessed traces total roughly 40 hours.

The broadband traces were published by the Federal Communications Commission (FCC) in the United States [43]. These traces include the hourly throughput data when measurement devices accessed nine major websites. The measurements were conducted from 477 locations in the United States between February 1 and 28, 2019. After the same preprocessing as for the mobile one, we picked the first

473 traces, the same number of mobile traces, to equalize the statistical error.

Figure 3 shows the Cumulative Distribution Functions (CDFs) of the average and the standard deviation for both traces. These figures show that the throughputs of broadband traces are relatively small and stable, whereas those of mobile traces are relatively high and widely distributed.

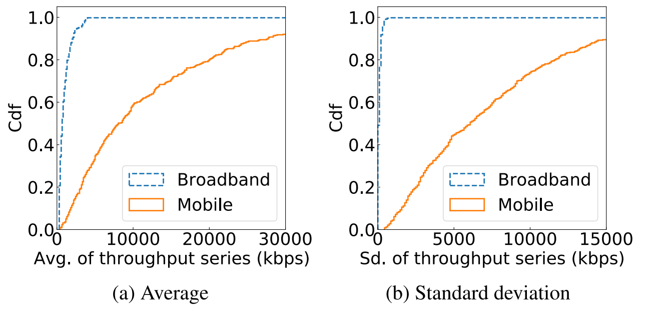


FIGURE 3. Statistics distribution of throughput traces used in the simulation.

2) ABR ALGORITHMS

As we described in Subsection II-B, there are many existing ABR algorithms, but none considers the traffic volume aspect. To be able to analyze why BANQUET works well, we selected representative algorithms from the rate-based and buffer-based algorithms. In addition, the default algorithm of dash.js [35] is selected because this algorithm is used in production environments [44]. The following briefly describes each algorithm.

1) Rate-based algorithm (RB): RB selects the maximum bitrate that is no higher than 0.9 of the harmonic mean of the throughputs of the last five chunks. In contrast to the other algorithms, the buffer length does not affect the bitrate selection by RB. Figure 4(a) shows the relationship between the estimated throughput and the bitrate selected by RB.
2) Buffer-based algorithm (BBA) [15]: BBA selects the bitrate by using the buffer length. We used BBA-2 [15] and set default parameters as recommended in the original paper [15]. In contrast to RB, BBA does not use throughput information. The relationship between the buffer length and the bitrate selected by BBA is shown in Fig. 4(b).
3) dash.js algorithm (BOLA) [16]: BOLA is the default bitrate-selection algorithm adopted in dash.js [35]. BOLA is a buffer-based algorithm constructed on the basis of Lyapunov optimization. We used default parameters for BOLA recommended in the original paper [16]. Similar to BBA, throughput information is not used in BOLA. The correlation between the buffer length and the bitrate selected by BOLA is shown in Fig. 4(b). BOLA tends to select the intermediate bitrate more often than BBA does.

The settings of BANQUET are summarized in Table 2. We analyze the sensitivity of the parameters of BANQUET T_c, h, and T_d in Subsection IV-D.

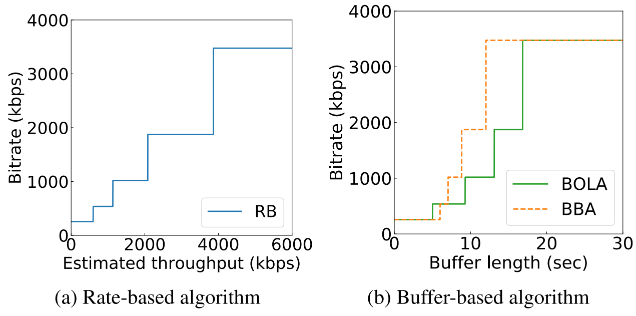


FIGURE 4. Bitrate-selection policy of existing algorithms.

TABLE 2. Settings of BANQUET, chunk, and video player.

Category	Description	Value
BANQUET	T_c	$1 \cdots 10$
	h	$1 \cdots 30$ (sec)
	T_d	$1 \cdots 5$
Chunk	Chunk length	3 (sec)
	Video duration	180 (sec)
	Audio bitrate	128 (kbps)
Video player	Threshold to start the playback	5 (sec)
	Upper limit of buffer	30 (sec)

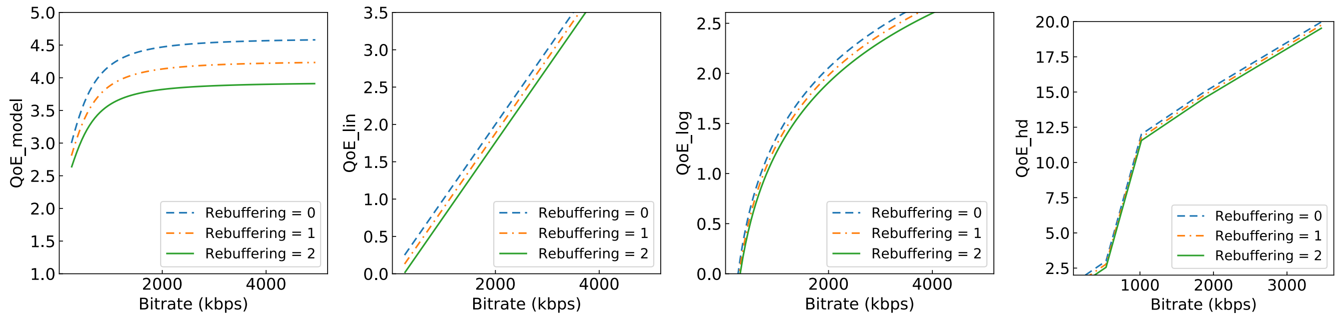
3) VIDEO BITRATE, CHUNK, AND VIDEO PLAYER SETTINGS

To evaluate the applicability for different video bitrate settings, we prepared two sets of the bitrate settings: A and B. The resolution and framerate are the same for Sets A and B, but the bitrate settings differ. Set A is determined on the basis of commercial video-streaming services for a realistic setting. We measured the bitrates of approximately 100 videos on YouTube [12] for each resolution (240p, 360p, 480p, 720p, and 1080p). We then calculated the average bitrate for each resolution and set the results as the video settings. Set B was set to the same value as in a previous study [19]. The video and audio encoding schemes are assumed to be H.264/AVC and AAC-LC, respectively. Set B has a higher maximum bitrate and more constant bitrate interval than Set A. The details of the video parameters are summarized in Table 3.

TABLE 3. Set of the video encoding parameter.

Name	Resolution	Bitrate (kbps)	Framerate (fps)
Set A	240p (426×240)	256	30
	360p (640×360)	538	30
	480p (854×480)	1019	30
	720p (1280×720)	1873	30
	1080p (1980×1080)	3476	30
Set B	240p (426×240)	300	30
	360p (640×360)	750	30
	480p (854×480)	1200	30
	720p (1280×720)	1850	30
	1080p (1980×1080)	2850	30

The chunk length and the video duration were fixed as 3 and 180 seconds, respectively. The audio bitrate and



(a) QoE_{model} (b) QoE_{lin} (c) QoE_{log} (d) QoE_{hd}

FIGURE 5. The characteristics of the QoE metrics used in the simulation.

framerate were fixed as 128 kbps and 30 fps regardless of the video bitrate. The threshold to start the playback and the upper limit of the buffer were set as 5 and 30 seconds, respectively. These parameters are summarized in Table 2.

4) QoE METRICS

BANQUET can be used in combination with any QoE metrics. To evaluate whether BANQUET can maintain the QoE over the target QoE for different QoE settings, four QoE metrics were used in this simulation. The QoE_{model} indicates MOS calculated with the MOS-estimation model, and QoE_{lin}, QoE_{log}, and QoE_{hd} indicate utility scores calculated with the different settings of a utility function.

QoE_{model} is calculated using the MOS-estimation model proposed by Yamagishi and Hayashi [24], Yamagishi [25]. QoE_{model} outputs the estimated MOS on a 1 to 5 quality scale on the basis of audio and video coding quality per second, rebuffering information (i.e., the number of rebuffering events, the total length of rebuffering events, the average of the interval between rebuffering events), and content length. The exact formulation is given in Subsection II-A. The coefficients in this model are set in accordance with the previous paper [34].

QoE_{lin}, QoE_{log}, and QoE_{hd} are calculated with the utility function. The utility function is widely used in existing papers [17]–[20] to evaluate the performance of existing ABR algorithms. The function used in this simulation is the modified version of Eq. (1) expressed as

$$Utility = \frac{1}{N}\sum_{n=1}^{N} Q(R_n) - \frac{1}{M}\mu T - \frac{1}{N}\sum_{n=1}^{N-1} |\Delta Q(R_n)|, \quad (8)$$

where N is the number of chunks received up to the calculation timing and the definition of other variables are the same as in Eq. 1. In this function, the first, second, and third terms indicate the bitrate utility, rebuffering penalty, and smoothness penalty, respectively. We modified the original function by replacing the denominator of the bitrate utility and the smoothness penalty from N to M. This is because dividing the rebuffering penalty by N instead of M leads to an overestimation of the rebuffering penalty during playback. Since $N = M$ when the playback is completed, this is consistent with the utility function used in the existing research after the playback. We consider several types of utility functions by setting different expressions of $Q(R_n)$ and μ. These settings are similar to those in previous studies [16]–[20]. The following briefly describes each QoE metric.

1) QoE_{lin}: $Q(R_n) = R$
This bitrate utility evaluates the image quality score linearly with respect to the bitrate.
2) QoE_{log}: $Q(R_n) = \log(R/R_{min})$
This bitrate utility reflects the Weber-Fechner law [45], which states that the subjective quality is proportional to the logarithm of the intensity of the stimulus received.
3) QoE_{hd}: $Q(R_n)$ is defined in accordance with Table 4.
This bitrate utility evaluates the rate of the high-definition (HD) resolution higher and the rate of the low resolution lower.

The exact settings of $Q(R_n)$ and μ are shown in Table 4.

Figure 5 shows how the QoE metrics change as the bitrate and the number of rebuffering changes in these simulation settings on basis of Eq. (2) and (8). In these figures, the video resolution is fixed as 480p, and each rebuffering event lasts 3 seconds. As the bitrate increases, QoE gradually increases, but the rate of increasing differs depending on the QoE metrics. If a rebuffering event occurs, QoE_{model} is roughly 0.9 times, while the utility scores slightly decrease.

TABLE 4. Settings of a part of QoE metrics in our evaluation.

Name	bitrate utility ($Q(R_n)$)	rebuffer penalty (μ)
QoE_{lin}	R	4.3
QoE_{log}	$\log(R/R_{min})$	2.66
QoE_{hd}	253→2, 538→3 961→12, 1771→15 3352→20	8

B. BANQUET VS. EXISTING ALGORITHMS

In this subsection, we demonstrate BANQUET can reduce the traffic volume while maintaining the same QoE as the existing algorithms. Since the resulting QoE and traffic volume of

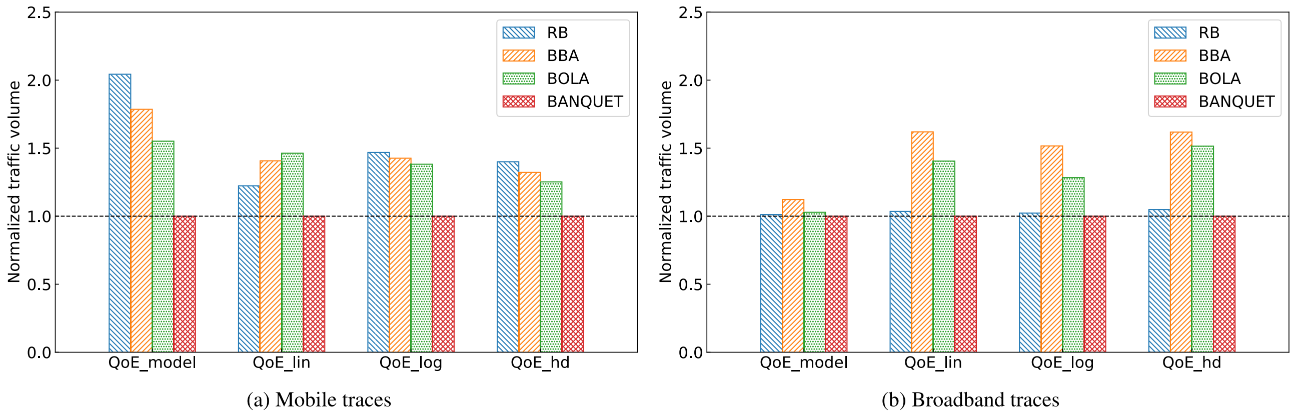


FIGURE 6. Normalized average traffic volume when the target QoE is set to that of the existing algorithm for Set A.

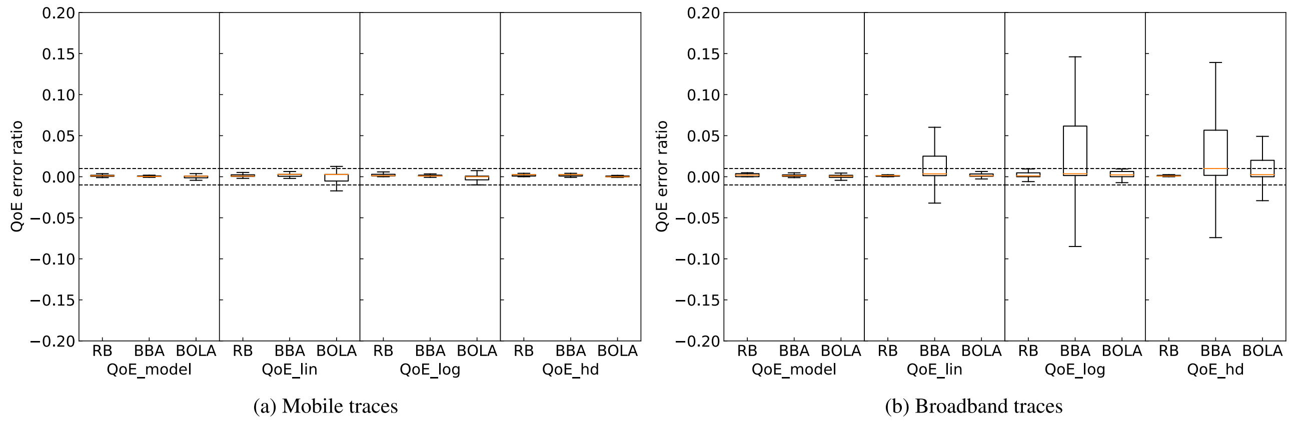


FIGURE 7. QoE error ratio between BANQUET and existing algorithms for Set A. Positive ratio means the QoE metric of BANQUET is larger than those of existing algorithms. Target QoE is set to that of existing algorithms. The dashed line means the ±1% error.

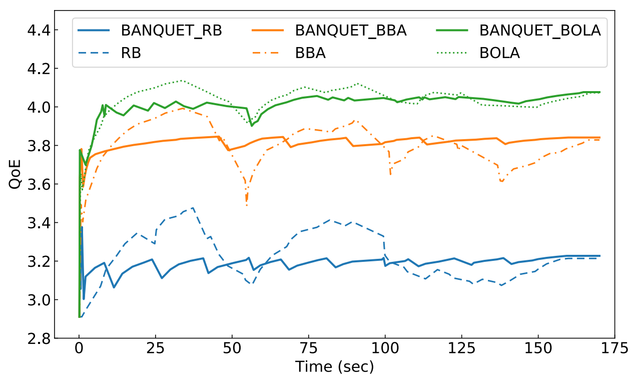


FIGURE 8. Sample QoE transitions for a specific network trace. BANQUET_RB, BANQUET_BBA, and BANQUET_BOLA indicate the QoE transition of BANQUET when the target QoE is the QoE of RB, BBA, and BOLA, respectively.

ABR algorithms differ for each trace, we need to adjust the target QoE of BANQUET for each trace to fairly compare the performance of BANQUET and the other algorithms. To do this, we first obtained the QoE of each existing algorithm for each throughput trace and then set the obtained QoE as the target QoE of BANQUET. Then, we calculated the difference in the obtained traffic volume of each algorithm. The following results were obtained when we set $h = 30$, $T_d = 5$, and $T_c = 4$, which resulted in the best performance. The results of other settings will be given in Subsection IV-D.

1) OVERALL RESULTS

Figure 6 shows the average traffic volume of each existing algorithm normalized by that of BANQUET for each QoE metric, and Fig. 7 shows the boxplots that represent the distributions of the QoE error ratio between BANQUET and existing algorithms for Set A. The QoE error ratio is calculated by dividing the difference in the QoE of an existing algorithm and that of BANQUET by the QoE of BANQUET. The QoE error ratios with positive and negative values represent cases where the QoE of BANQUET was higher and lower, respectively. Thus, the result is better if the QoE error ratio is close to zero since it means BANQUET controlled the QoE to the target QoE precisely. In Fig. 7, the dotted line indicates the $\pm 1\%$ error, the lower and upper ends of the box represent the first and third quartiles, respectively, and the

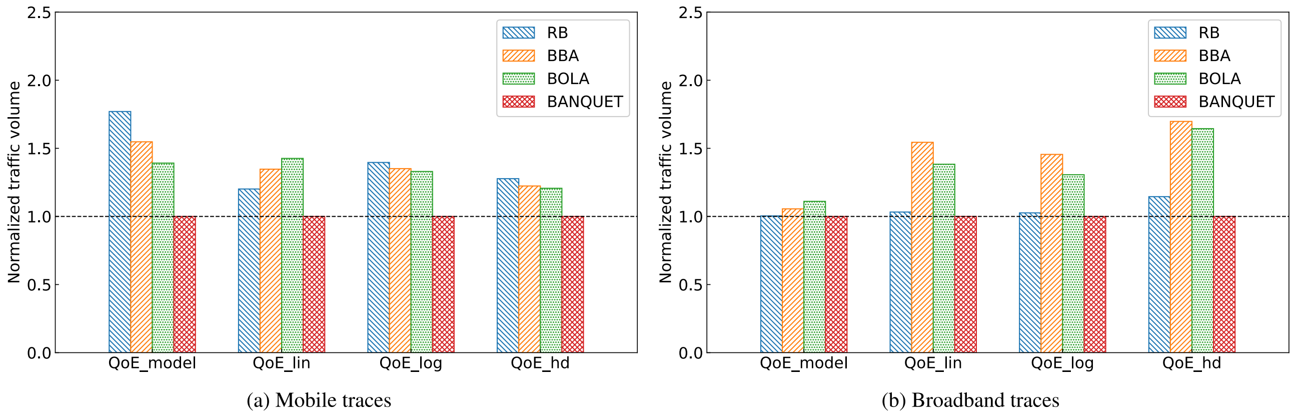


FIGURE 9. **Normalized average traffic volume when the target QoE is set to that of the existing algorithm for Set B.**

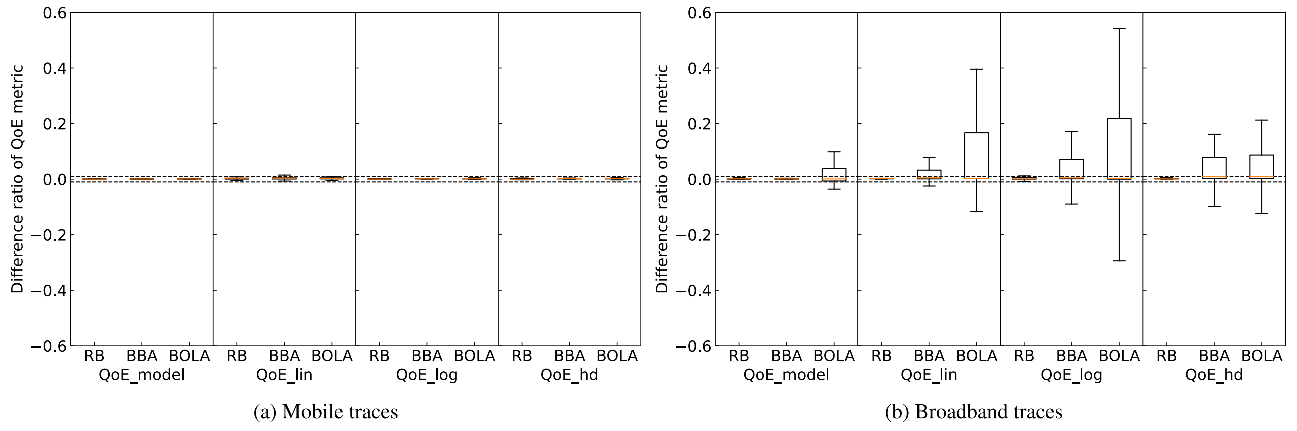
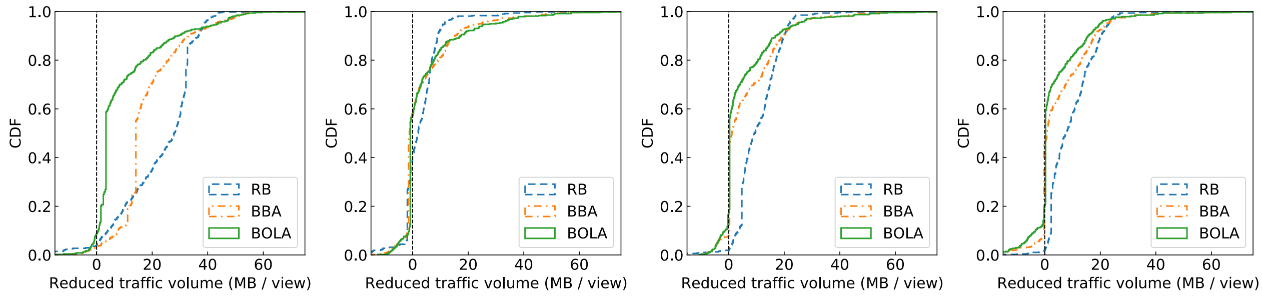


FIGURE 10. **QoE error ratio between BANQUET and existing algorithms for Set B. Positive ratio means the QoE metric of BANQUET is larger than those of existing algorithms. Target QoE is set to that of existing algorithms. The dashed line means the ±1% error.**

length of the whiskers represents a length 1.5 times the size of the box.

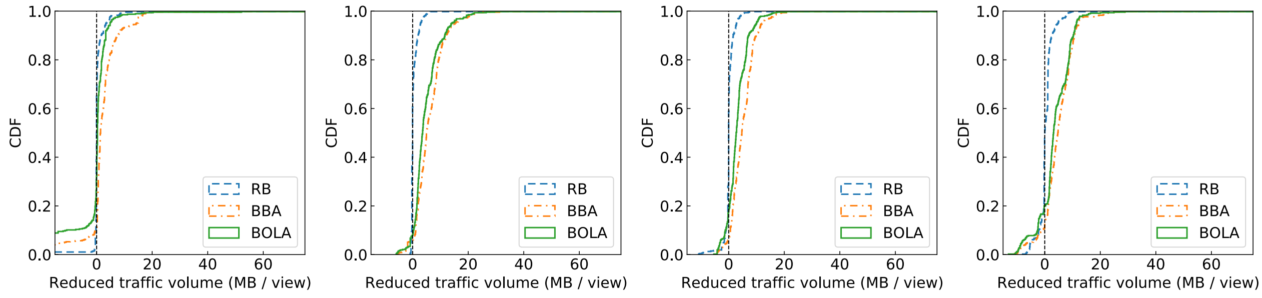
From Fig. 6, the normalized traffic volume of the existing algorithms is more than one for all the simulation settings. In addition, Fig. 7 indicates that BANQUET maintained QoE almost equal to or higher than those of existing algorithms for all the simulation settings. Specifically, traffic volume is reduced by 18.3%–51.2% in the mobile environment and 1.2%–38.3% in the broadband environment on average, while maintaining the same or better QoE. Note that BANQUET is outperformed in terms of QoE in some conditions by the existing algorithms in the broadband environment as shown in Fig. 7(b). This is because the QoE of the existing algorithm is sometimes lower than that when the lowest bitrate is always selected. In such cases, the existing algorithm selected unnecessarily high bitrates, which resulted in undesirable rebuffering. However, BANQUET reduces the traffic volume in such a network environment while maintaining a higher QoE than the existing algorithms. Figure 8 indicates QoE transition for a specific mobile throughput trace. Broken and solid lines indicate the QoE transitions of existing algorithms and BANQUET, respectively. The target QoE is set in accordance with the QoE of existing algorithms, but other parameters of BANQUET are not changed. These plots show that BANQUET precisely controls QoE for the target QoE.

Figures 9 and 10 show the same results for Set B. Traffic volume is reduced by 16.8%–43.5% in the mobile environment and 0.4%–41.1% in the broadband environment on average, while maintaining the same as or better QoE than existing algorithms. In the mobile environment, the reduced traffic volume of Set B is less than that of Set A. This is because the maximum selectable bitrate of Set B was lower than that of Set A. If existing algorithms select a high bitrate, rebuffering and bitrate switching are more likely to occur, which makes room for BANQUET to reduce traffic volume. In the fixed environment, there was no significant difference between Sets A and B since the throughput is too low to select the highest bitrate. In addition, the QoE of BANQUET is higher than those of the buffer-based algorithms (BBA, BOLA). This is because the difference between the minimum and second minimum bitrate is larger for Set B than for Set A. Since buffer-based algorithms select a bitrate on the basis of buffer length, they sometimes select a bitrate higher than the throughput. Thus, the rebuffering risk of Set B is higher than



(a) QoE_{model} (b) QoE_{lin} (c) QoE_{log} (d) QoE_{hd}

FIGURE 11. Cumulative distribution function of reduced traffic volume in mobile traces.



(a) QoE_{model} (b) QoE_{lin} (c) QoE_{log} (d) QoE_{hd}

FIGURE 12. Cumulative distribution function of reduced traffic volume in broadband traces.

that of Set A, and the QoE of the existing algorithm is lower than that when the lowest bitrate is always selected.

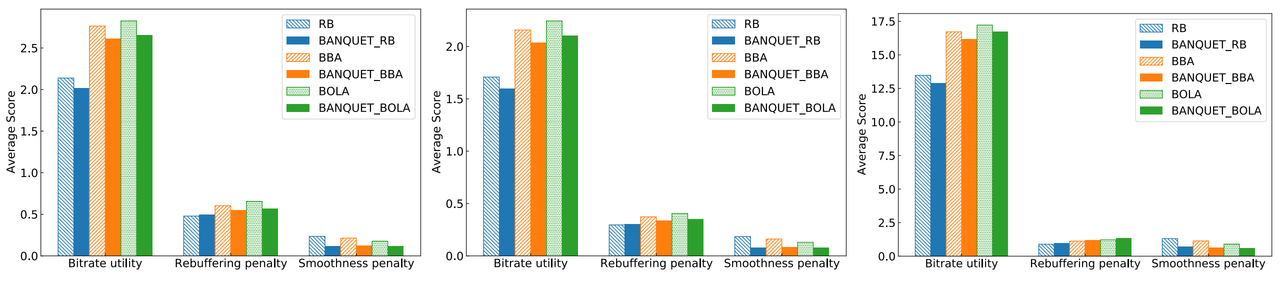
Compared with RB in the broadband environment, the traffic volume reduction ratio is limited, although BANQUET can reduce the traffic volume considerably in the mobile environment as shown in Fig. 6. This is because the broadband throughput traces are stable, as shown in Fig. 3(b). In such a stable network environment, the bitrate selected by BANQUET is almost the same as that of RB. Thus, there is no room to decrease the rebuffering penalty and smoothness penalty to maintain the same QoE as that of RB. On the other hand, BANQUET can reduce the traffic volume more than BBA and BOLA even in the broadband environment. This is because BBA and BOLA are buffer-based algorithms, i.e., they tend to select a high bitrate when the buffer length increases, even if the throughput is low. Thus, these algorithms increase traffic volume and incur a high risk of rebuffering. BANQUET, on the other hand, selects a stable bitrate in a stable network environment. Thus, BANQUET can reduce the traffic volume while maintaining the same QoE as buffer-based algorithms.

In summary, BANQUET reduced the traffic volume while maintaining QoE the same as or better than that of the existing algorithms, regardless of any combination of the QoE metric, the ABR algorithm, the network environment, and the bitrate settting. These results mean that if a user subscribes to a mobile communication plan with a data cap, the user can view videos up to 1.95 times more while the same QoE as the existing algorithms is maintained. Also, this results indicate streaming providers can reduce the streaming cost (e.g., CDN cost) using BANQUET. For example, if 1% of video-streaming traffic [1] shifts from BOLA to BANQUET and the streaming provider uses Amazon CloudFront with the most effective plan in the US [46], the CDN cost decreases by 4.3 million USD per month while the same QoE (QoE_{model}) as BOLA is maintained.

2) DETAILED RESULTS

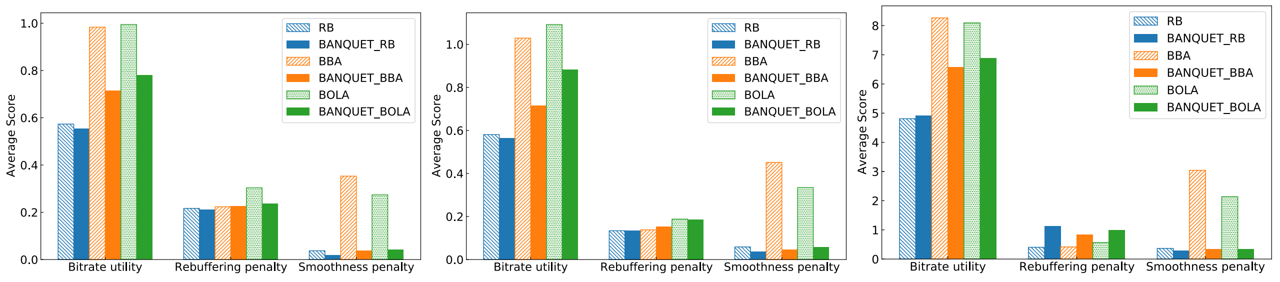
To further understand BANQUET's performance, we analyzed the characteristics of the reduced traffic volume distribution, the contribution of each component of the QoE metric, and the performance in accordance with the network condition. The following results are for Set A since the results for Set B were almost the same.

Figures 11 and 12 show the CDFs of reduced traffic volume for mobile and broadband environments, respectively. The x-axis represents the traffic volume reduced by BANQUET compared with the existing algorithms for each throughput trace. Positive and negative values indicate the reduced and increased traffic volume by BANQUET, respectively. The results show that the traffic volume is almost the same as that of the existing algorithms in the broadband environment, while the traffic volume is significantly reduced in the mobile environment. Particularly, when BANQUET used QoE_{model} as the QoE metric in the mobile environment, BANQUET reduced the traffic volume in more than 90% of



(a) QoE_{lin} (b) QoE_{log} (c) QoE_{hd}

FIGURE 13. **Comparing BANQUET with existing ABR algorithms by analyzing their individual components for mobile traces.**



(a) QoE_{lin} (b) QoE_{log} (c) QoE_{hd}

FIGURE 14. **Comparing BANQUET with existing ABR algorithms by analyzing their individual components for broadband traces.**

the cases compared with any existing algorithm. In contrast, BANQUET reduced the traffic volume for utility-based QoE metrics QoE_{lin}, QoE_{log}, and QoE_{hd} in up to 40% of cases. This is because the QoE_{model} is more affected by rebuffering than the utility score. More specifically, as we described in Subsection IV-A, QoE is calculated by subtracting the rebuffering effect from the bitrate utility in utility scores QoE_{lin}, QoE_{log}, QoE_{hd}, whereas MOS in QoE_{model} is calculated as a ratio (e.g., 0.9) to audiovisual score in accordance with the rebuffering information. Therefore, when using QoE_{model}, the QoE of the existing algorithm is significantly degraded by rebuffering and thus BANQUET can reduce the traffic volume while maintaining the same QoE.

To analyze in detail why BANQUET reduced the traffic volume while maintaining the QoE at the same level as the existing algorithms, we decompose the results of QoE in the bitrate utility, rebuffering penalty, and smoothness penalty when we used QoE_{lin}, QoE_{log}, or QoE_{hd} as the QoE metric. Figures 13 and 14 show the average of each term in the results of the QoE of the existing algorithms and BANQUET. In these figures, the target QoE of BANQUET differs depending on the QoE of existing algorithms. Thus, we plotted the results of BANQUET for each existing method. For example, BANQUET_RB is the result of BANQUET when the target QoE is set to the QoE of RB. All the figures exhibit a similar tendency: BANQUET decreases the bitrate utility but suppresses the rebuffering and smoothness penalties. The low bitrate utility indicates that BANQUET selected the lower bitrate than existing algorithms, and the low rebuffering, and smoothness penalties indicate that BANQUET selects a more stable bitrate than existing algorithms. Therefore, selecting a stable bitrate contributed to reducing the traffic volume while maintaining the QoE.

To clarify under what network conditions BANQUET reduces the traffic volume, we analyzed the impact of the average throughput of each trace on the traffic volume reduction. The results are shown in Fig. 15. In this analysis, we focused on the mobile environment because the broadband environment has less variability in average throughput. The y-axis of this graph indicates the average of the normalized traffic volume compared with BANQUET. The results indicate that the traffic volume reduction ratio tends to decrease as the average throughput increases for BBA and BOLA. This is because such buffer-based algorithms select a high bitrate when the buffer length is longer, even if the average throughput is lower. Selecting such a high bitrate causes rebuffering, degrades QoE, and increases the traffic volume. Compared with RB, BANQUET reduced the traffic volume constantly regardless of the average throughput. This is because the bitrate selected by RB fluctuates highly if the throughput fluctuates highly. These results indicate that BANQUET reduces the traffic volume especially when the average throughput is small.

C. BANQUET VS. DATA SAVING METHOD

In this subsection, we compare BANQUET with a data saving method adopted in actual commercial video-streaming services. As described in Section II, commercial

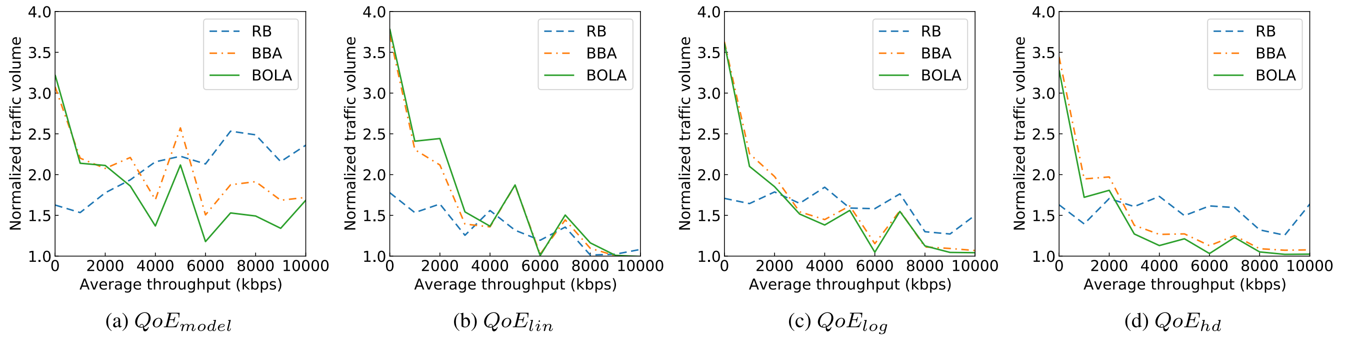


FIGURE 15. Normalized traffic volume compared with BANQUET for various throughput conditions.

video-streaming providers implement a function to set the upper limit of a selectable bitrate [11], [12], [21]–[23]. This method reduces the traffic volume, but streaming providers may not achieve the user's required QoE due to the throughput fluctuation. Thus, we assume the BOLA with the upper limit of selectable bitrate (capped BOLA) to be the existing data saving method of commercial video-streaming providers and compare it with BANQUET. In this simulation, we set 538 kbps (corresponding to 360p) as the upper limit of capped BOLA, 3.5 as the target QoE of BANQUET, QoE_{model} as the QoE metric, Set A as the bitrate setting, and mobile traces as the network traces. The other settings are the same as those in Subsection IV-A.

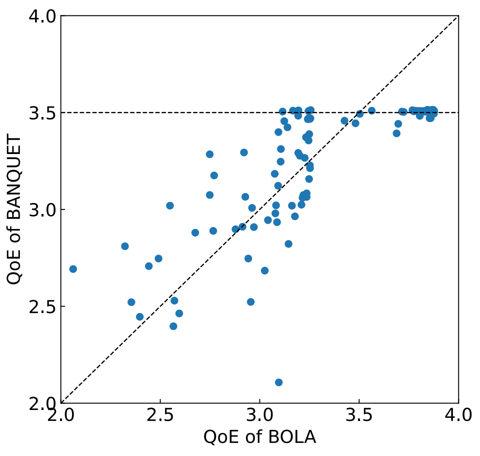


FIGURE 16. Scatter plot of the QoE of BANQUET and that of capped BOLA.

Figure 16 shows the QoE for each throughput trace. The x- and y-axes indicate the QoE of capped BOLA and BANQUET, respectively. Regardless of the QoE of capped BOLA, the plots where the value of the y-axis is close to 3.5 are expected results since the target QoE is set as 3.5. The other plots in the upper left region indicate BANQUET achieved higher QoE than BOLA, and the plots on the other side represent the opposite. There are several cases where the QoE of BANQUET is higher than that of capped BOLA, if the QoE of BANQUET is lower than the target QoE. To analyze

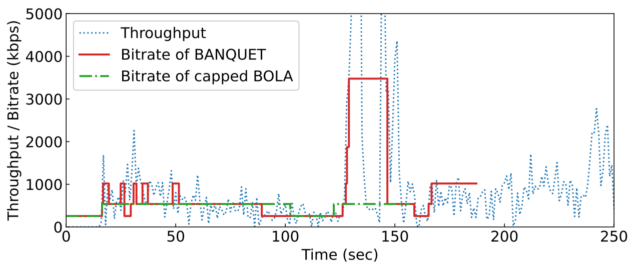


FIGURE 17. Selected bitrate when the QoE of BANQUET is larger than that of capped BOLA.

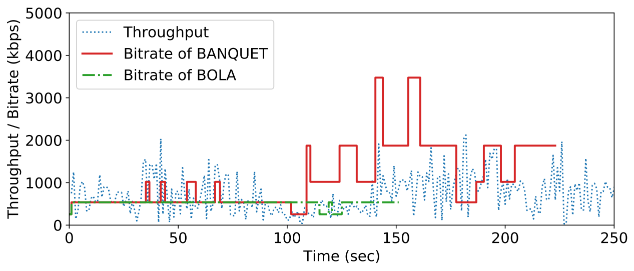


FIGURE 18. Selected bitrate when the QoE of BANQUET is less than that of capped BOLA.

the reason for this, we plotted the sample bitrate series when the QoE of BANQUET is higher than that of capped BOLA in Fig. 17. The x-axis indicates time, and the y-axis indicates the selected bitrate or the throughput. In this case, the QoE is 3.28 for BANQUET and 2.75 for capped BOLA. Up to 120 seconds, both algorithms selected almost the same bitrate because the throughput is low. From 120 to 150 seconds, BANQUET selected the highest bitrate to achieve the target QoE whereas capped BOLA could not select a bitrate higher than 538 kbps that corresponds to 360p because of the cap. Therefore, BANQUET achieved a higher QoE than capped BOLA by selecting a higher bitrate than capped BOLA when the throughput changed from low to high during the viewing.

On the other hand, there are cases where the QoE of BANQUET is lower than that of capped BOLA. To analyze the reason for this, we plotted a selected bitrate series with such a condition in Fig. 18. In this case, the QoE is 2.52 for

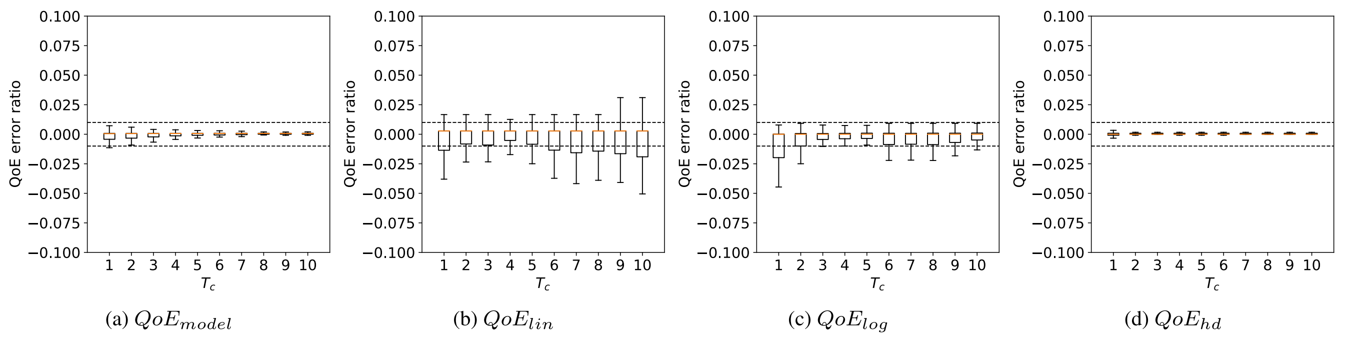


FIGURE 19. Performance evaluation for various number of chunks to calculate the future throughput series T_c.

BANQUET and 2.95 for capped BOLA. This figure shows that BANQUET selects a higher bitrate than the throughput from 110 seconds. This is because BANQUET tried to achieve the target QoE by incurring the risk of rebuffering. However, selecting such a higher bitrate caused the rebuffering and decreased the QoE. The root cause of such undesired rebuffering is the overestimation of the throughput. If the throughput is overestimated, BANQUET tends to select a higher bitrate to achieve the target QoE, and rebuffering occurs. To deal with such an overestimation of throughput, there are three approaches to improve BANQUET. The first is to improve the accuracy of throughput estimation. The second is to calculate the risk of rebuffering and select a conservative bitrate. The third is to abandon the receiving chunk and request another chunk with a lower bitrate when the receiving chunk takes a long time to download. These improvements of BANQUET are future works.

D. HYPER-PARAMETER OPTIMIZATION

In this subsection, we optimize the hyper-parameters of BANQUET: the throughput prediction horizon h, the threshold of the length of bitrate-series candidates T_d, and the number of chunks to calculate the future throughput series T_c. BANQUET predicts the throughput series up to h seconds by using the latest T_c chunks and computes the QoE for all possible bitrate series up to the next T_d chunks. If T_c is too small or large, the throughput estimation accuracy may decrease and the QoE error may increase. If h, T_d are set excessively large, the calculation time increases exponentially. Such a long calculation time may delay the timing of the chunk request and may cause rebuffering. In contrast, if h, T_d are set excessively low, and BANQUET may not select an appropriate bitrate because the number of candidate bitrate series is too limited. Thus, we need to optimize them to reduce computation time without degrading the performance of BANQUET. These analyses are conducted for the mobile traces with Set A as the bitrate settings.

Figure 19 shows the QoE error ratio when T_c is changed from 1 to 10 in the mobile environment compared with BOLA. These results show that the QoE error ratio is minimized for all the QoE metrics when $T_c = 4$. This is because there is a trade-off between the number of chunks and the freshness of the measured throughput. If the number of chunks is small, the estimation accuracy decreases because the throughput fluctuates, especially in the mobile environment. Also, the estimation accuracy decreases if the number of chunks is large since the old throughput values are used to estimate the future throughput. As a result, $T_c = 4$ is the best setting to control the QoE appropriately in this case.

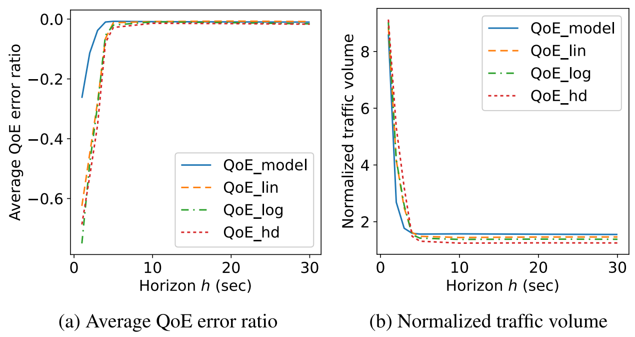


FIGURE 20. Performance evaluation for various horizon h.

Figure 20 shows results when h is changed from 1 to 30 seconds in the mobile environment compared with BOLA. Figure 20(a) shows the average QoE error ratio, and Fig. 20(b) shows the normalized traffic volume compared with BANQUET. From Fig. 20(a), all the results show the same tendency that the average QoE error ratio gradually increases as h increases and converges to 0 when h is equal to 10 seconds. If h is 10 seconds, the normalized traffic volume also converges as shown in Fig. 20(b). This is because h that is too large does not contribute to select a more appropriate bitrate for all QoE metrics. When h is too small, BANQUET selects a low bitrate and cannot achieve the target QoE since BANQUET estimates that the player cannot receive a chunk with a high bitrate within the predicted throughput horizon. As h increases, BANQUET selects a high enough bitrate to achieve the target QoE since the number of candidate bitrates that can be received in the predicted horizon increases. However, if h is large enough, the selected bitrate does not change since the first few bitrates of the candidate bitrate series do not change. This behavior occurs regardless of the QoE metric setting since the QoE does not change significantly enough to change the bitrate selection if h is large enough. Therefore, these results show that setting h as 10 seconds is sufficient.

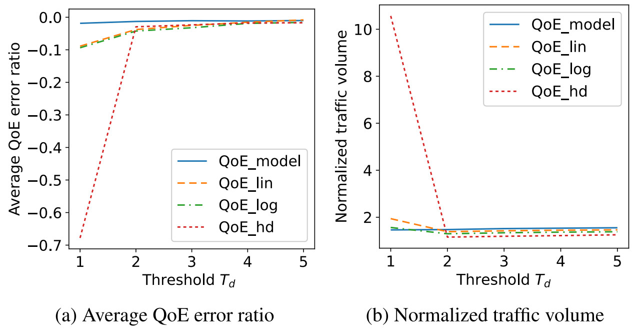


FIGURE 21. **Performance evaluation for the threshold of the length of bitrate-series T_d.**

Figure 21 shows results when T_d is changed from 1 to 5 in the mobile environment compared with BOLA. Figure 21(a) shows the average QoE error ratio, and Fig. 21(b) shows the normalized traffic volume. Similar to Fig. 20, Fig. 21(a) shows that the average QoE error ratio gradually increases as the threshold T_d increases and converges to 0 if T_d is 4 for all the QoE metrics. As shown in Fig. 21(b), the normalized traffic volume is also converged if T_d is 4. This is because the selected bitrate is not changed if T_d is sufficiently large. When T_d is small, the number of candidate bitrate series is small, but when T_d is large, BANQUET can consider the various bitrate series and select the bitrate from the most suitable series. However, T_d that is too large does not affect the selected bitrate since BANQUET outputs only the next bitrate of the candidate bitrate series. This behavior occurs regardless of the QoE metric settings. Therefore, these results show that setting T_d as 4 is sufficient.

TABLE 5. **Calculation time of BANQUET for optimized and non-optimized (h, T_d).**

	Non-Optimized calculation time (ms)	Optimized calculation time (ms)
Mean	97.3	17.6
Standard deviation	55.3	10.3

On the basis of the results in Figs. 20 and 21, we measured the calculation time when (h, T_d) is set to (10, 4) (optimized) and (30, 5) (non-optimized) on an actual smartphone, a Sony Xperia XZ. In this evaluation, we set $T_c = 4$ for both settings since T_c does not affect the calculation time unlike other parameters. The evaluation settings are the same as those described in Subsection IV-A except for the settings of h and T_d. We also used mobile throughput traces for this evaluation and set BOLA as the existing ABR algorithm. The results are summarized in Table 5, which contains the average and standard deviation of the calculation time. Table 5 shows that by optimizing parameters (h, T_d), BANQUET calculates the suitable bitrate in 17.6 ms on average, which means that BANQUET with optimized parameters reduces the calculation time by 81.9% compared with BANQUET without non-optimized parameters. In addition, the standard deviation of the calculation time is 10.3 ms, which is also sufficiently lower than the chunk download time. These results clarified that by optimizing the hyper-parameters, BANQUET can work efficiently and stably even in a limited computing environment such as smartphones.

V. CONCLUSION AND FUTURE WORK

We proposed BANQUET, a novel Adaptive Bitrate (ABR) algorithm that aims to reduce traffic volume while maintaining Quality of Experience (QoE) above the target QoE. To achieve better QoE with lower traffic volume, BANQUET estimates the buffer transition for all bitrate-selection patterns for the next several chunks and selects a suitable bitrate. We evaluated BANQUET through a trace-based simulation using actual throughput data collected in mobile and broadband environments. The simulation results clarified that BANQUET reduced the traffic volume 18.3%–51.2% on average in the mobile environment and 1.2%–38.3% in the broadband environment while maintaining QoE the same as or better than that of existing algorithms. Furthermore, by optimizing a throughput prediction horizon parameter and a search space parameter, BANQUET calculated the bitrate on an actual smartphone in 17.6 ms on average.

For future work, we will improve the performance of BANQUET by reducing the overestimation of throughput. To cope with the overestimation, we plan to improve the throughput estimation accuracy and expand BANQUET to consider the abandonment of the receiving chunk. In addition, evaluating BANQUET in the wild is also future work.

REFERENCES

[1] *Cisco Visual Networking Index: Forecast and Methodology, 2017–2022–Cisco*. Accessed: Dec. 24, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *Proc. ACM SIGCOMM Conf. SIGCOMM (SIGCOMM)*, 2011, pp. 362–373.

[3] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 2001–2014, Dec. 2013.

[4] *Mobile Video: Exposed | Streaming Video Alliance*. [Online]. Available: https://www.streamingvideoalliance.org/document/mobile-video-exposed/

[5] R. Zopf, *Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)*, RFC document 3389, Oct. 2002.

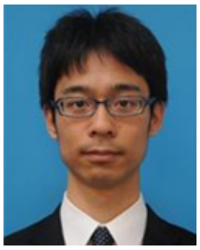
[6] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[7] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[8] R. Pantos. (Nov. 2015). *HTTP Live Streaming*. [Online]. Available: https://tools.ietf.org/html/draft-pantos-http-live-streaming-18

[9] T. Stockhammer, "Dynamic adaptive streaming over HTTP–: Standards and design principles," in *Proc. 2nd Annu. ACM Conf. Multimedia systems*, Feb. 2011, pp. 133–144.

[10] J. Iyengar and M. Thomson, ''QUIC: A UDP-based multiplexed and secure transport,'' in *Internet Engineering Task Force, Internet-Draft Draft-Ietf-Quic-Transport-32*. Reston, VA, USA: IETF, Oct. 2020.
[11] *Hulu*. Accessed: Jan. 15, 2021. [Online]. Available: https://www.hulu.com/
[12] *YouTube*. Accessed: Dec. 24, 2020. [Online]. Available: https://www.youtube.com
[13] *Netflix–Watch TV Shows Online, Watch Movies Online*. Accessed: Dec. 24, 2020. [Online]. Available: https://www.netflix.com.
[14] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, ''Probe and adapt: Rate adaptation for HTTP video streaming at scale,'' *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 719–733, Apr. 2014.
[15] T. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, ''A buffer-based approach to rate adaptation: Evidence from a large video streaming service,'' in *Proc. ACM SIGCOMM*, Aug. 2014, pp. 187–198.
[16] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, ''BOLA: Near-optimal bitrate adaptation for online videos,'' in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.
[17] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, ''A control-theoretic approach for dynamic adaptive video streaming over HTTP,'' in *Proc. ACM SIGCOMM*, Aug. 2015, pp. 325–338.
[18] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, and N. Wang, ''CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction,'' in *Proc. ACM SIGCOMM*, Aug. 2016, pp. 272–285.
[19] H. Mao, R. Netravali, and M. Alizadeh, ''Neural adaptive video streaming with pensieve,'' in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 197–210.
[20] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, ''Oboe: Auto-tuning video ABR algorithms to network conditions,'' in *Proc. ACM SIGCOMM*, Aug. 2018, pp. 44–58.
[21] *DAZN*. Accessed: Dec. 24, 2020. [Online]. Available: https://www.dazn.com/
[22] *Abema TV*. Accessed: Dec. 24, 2020. [Online]. Available: https://abema.tv/
[23] Netflix. *How Can I Control How Much Data Netflix Uses?* Accessed: Dec. 24, 2020. [Online]. Available: https://help.netflix.com/en/node/87
[24] K. Yamagishi and T. Hayashi, ''Parametric quality-estimation model for adaptive-bitrate-streaming services,'' *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1545–1557, Jul. 2017.
[25] K. Yamagishi, ''Audio-visual quality estimation device, method for estimating audio-visual quality, and program,'' WO Patent 2 017 104 416 A1, Jun. 22, 2017.
[26] T. Kimura, T. Kimura, A. Matsumoto, and J. Okamoto, ''BANQUET: Balancing quality of experience and traffic volume in adaptive video streaming,'' in *Proc. 15th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2019, pp. 1–7.
[27] M. Li and C.-Y. Lee, ''A cost-effective and real-time QoE evaluation method for multimedia streaming services,'' *Telecommun. Syst.*, vol. 59, no. 3, pp. 317–327, Jul. 2015.
[28] M. Li, C.-L. Yeh, and S.-Y. Lu, ''Real-time QoE monitoring system for video streaming services with adaptive media playout,'' *Int. J. Digit. Multimedia Broadcast.*, vol. 2018, pp. 1–11, Feb. 2018.
[29] *Parametric Bitstream-Based Quality Assessment of Progressive Download and Adaptive Audiovisual Streaming Services Over Reliable Transport*, ITU-T Recommendation document P.1203, Nov. 2016.
[30] H. T. T. Tran, T. Vu, N. P. Ngoc, and T. C. Thang, ''A novel quality model for HTTP adaptive streaming,'' in *Proc. IEEE 6th Int. Conf. Commun. Electron. (ICCE)*, Jul. 2016, pp. 423–428.
[31] 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, pp. 651–665, Dec. 2015.
[32] A. Mittal, A. K. Moorthy, and A. C. Bovik, ''No-reference image quality assessment in the spatial domain,'' *IEEE Trans. Image Process.*, vol. 21, no. 12, pp. 4695–4708, Dec. 2012.
[33] A. Mittal, R. Soundararajan, and A. C. Bovik, ''Making a 'completely blind' image quality analyzer,'' *IEEE Signal Process. Lett.*, vol. 20, no. 3, pp. 209–212, Mar. 2013.
[34] P. Lebreton and K. Yamagishi, ''Transferring adaptive bit rate streaming quality models from H.264/HD to H.265/4K-UHD,'' *IEICE Trans. Commun.*, vol. 102, no. 12, pp. 2226–2242, 2019.
[35] D. I. Forum. *GitHub–Dash-Industry-Forum/Dash.js: A Reference Client Implementation for the Playback of MPEG DASH via Javascript and Compliant Browsers*. Accessed: Dec. 24, 2020. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js
[36] N. Barman and M. G. Martini, ''QoE modeling for HTTP adaptive video streaming–a survey and open challenges,'' *IEEE Access*, vol. 7, pp. 30831–30859, 2019.
[37] Netflix. *Per-Title Encode Optimization*. [Online]. Available: https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2
[38] B. Rainer, S. Petscharnig, C. Timmerer, and H. Hellwagner, ''Statistically indifferent quality variation: An approach for reducing multimedia distribution cost for adaptive video streaming services,'' *IEEE Trans. Multimedia*, vol. 19, no. 4, pp. 849–860, Apr. 2017.
[39] *Information Technology–Generic Codingof Moving Pictures and Associated Audio Information—Part 7: Advanced Audio Coding (AAC)*, document ISO/IEC 13818-7:2006, Jan. 2006.
[40] *Advanced Video Coding for Generic Audiovisual Services*, document ITU-T Recommendation H.264Sep. 2019.
[41] Y. Qin, R. Jin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue, ''A control theoretic approach to ABR video streaming: A fresh look at PID-based rate adaptation,'' in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.
[42] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, ''Beyond throughput: A 4G LTE dataset with channel and context metrics,'' in *Proc. ACM MMSys*, Jun. 2018, pp. 460–465.
[43] F. C. Commission. *Raw Data–Measuring Broadband America–Eighth Report*. Accessed: Dec. 24, 2020. [Online]. Available: https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-eighth
[44] K. Spiteri, R. Sitaraman, and D. Sparacio, ''From theory to practice: Improving bitrate adaptation in the DASH reference player,'' *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 15, no. 2, p. 67, Jul. 2019.
[45] G. T. Fechner, H. E. Adler, D. H. Howes, and E. G. Boring, *Elements of Psychophysics* (Henry Holt editions in Psychology). New York, NY, USA: Holt, Rinehart and Winston, 1966.
[46] Amazon. *CDN Pricing | Free Tier Eligible, Pay-as-You-go | Amazon CloudFront*. Accessed: Dec. 24, 2020. [Online]. Available: https://aws.amazon.com/jp/cloudfront/pricing



TAKUTO KIMURA received the B.S. and M.S. degrees in information engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 2011 and 2013, respectively. He joined NTT Laboratories, in 2013, where has been engaged in the research of video QoE control and network management. He received the Young Researcher's Award (IEICE), the Research Encouragement Award (IEICE Technical Committee on Communication Quality) in Japan, in 2018, and the Best Research Award (IEICE Technical Committee on Communication Quality) in Japan, in 2019.

TATSUAKI KIMURA (Member, IEEE) received the B.E. degree in informatics and mathematical science, the M.I. degree in system science, and the Ph.D. degree from Kyoto University, Kyoto, Japan, in 2006, 2010, and 2017, respectively. He joined NTT Network Technology Laboratories, in 2010, where he has been researching the management of large-scale networks. He is currently an Associate Professor with Osaka University, Japan. His current research interests include stochastic analysis of wireless communication systems and their optimization. He is a member of IEICE and the Operations Research Society of Japan (ORSJ). He received Best Student Paper Awards from ORSJ, in 2010, and the Best Paper Awards from Special Interest Group of Queueing Theory of ORSJ and the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '16), in 2014 and 2016, respectively.

ARIFUMI MATSUMOTO received the B.S. and M.S. degrees in information and computer science from Kyoto University, Kyoto, in 2002 and 2004, respectively. He joined NTT, in 2004, where he has been working on the designing architecture, engineering, and standardization of IP networks, and researching QoE control technologies for video-streaming services. He is currently developing machine learning-based infrastructure inspection system with Japan Infra Waymark Corporation.

KAZUHISA YAMAGISHI received the B.E. degree in electrical engineering from the Tokyo University of Science, Chiba, Japan, in 2001, and the M.E. and Ph.D. degrees in electronics, information, and communication engineering from Waseda University, Tokyo, Japan, in 2003 and 2013, respectively. He joined NTT Laboratories, Musashino, Japan, in 2003. From 2010 to 2011, he was a Visiting Researcher with Arizona State University, Tempe, AZ, USA. His research interest includes development of objective quality-estimation models for multimedia telecommunications. He was a recipient of the Young Investigators' Award (IEICE) in Japan, in 2007, and the Telecommunication Advancement Foundation Award in Japan, in 2008, the ITU-AJ Encouragement Awards in Japan, in 2017, and the TTC Award for Distinguished Service in Japan, in 2018.