IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 7, 2021, accepted December 5, 2021, date of publication December 8, 2021,
date of current version December 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3133851

QoE-Aware Bitrate Selection in Cooperation With In-Network Caching for Information-Centric Networking

YUSAKU HAYAMIZU[1], (Member, IEEE), KOKI GOTO[2], (Member, IEEE), MASAKI BANDAI[3], (Member, IEEE), AND MIKI YAMAMOTO[2], (Senior Member, IEEE)

[1]Network Research Institute, National Institute of Information and Communications Technology (NICT), Koganei, Tokyo 184-8795, Japan
[2]Faculty of Engineering Science, Kansai University, Suita, Osaka 564-8680, Japan
[3]Faculty of Science and Technology, Sophia University, Chiyoda, Tokyo 102-0081, Japan

Corresponding author: Yusaku Hayamizu (hayamizu@nict.go.jp)

This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 17H01740.

ABSTRACT Adaptive Bitrate (ABR) algorithms used in MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) can be applied to video streaming over Information-Centric Networks (ICNs). However, in-network cache, which is an inherent and important feature of ICN, might negatively affect the Quality of Experience (QoE) of streaming users due to misestimating download throughput when fetching from the in-network cache. A promising solution to this problem is to use a Server and Network Assisted DASH (SAND)-like approach: ICN routers in the network notify a user application of the available bandwidth. However, with a naive network-assisted approach, the user application cannot fully utilize the cached segments when high-quality video segments are accidentally stored in the router on the user side of the congestion. In this paper, we propose an intelligent QoE-aware ABR selection method that works in cooperation with in-network Caching functions, called *QoE-ABC*. It is suitable for video streaming over an ICN. In QoE-ABC, QoE-aware adaptation logic running on the user application selects a bitrate that matches the bandwidth of the bottleneck in the original server or of any intermediate router on the download path depending on the situation. Only when the user-perceived QoE is expected to be improved, the user application tries to aggressively download high-quality segments from the in-network cache. Simulation results show that QoE-ABC can obtain high-quality video segments from in-network caches compared with the conventional ABR representatives and achieve high-level QoE performance for users with various preferences.

INDEX TERMS ICN, CCN, NDN, video streaming, MPEG-DASH, adaptive bitrate.

I. INTRODUCTION

MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [1] is widely used in adaptive video streaming to select a bitrate that matches the network bandwidth. Adaptive Bitrate (ABR) algorithms executed in MPEG-DASH are classified into three types [2]: *rate-based algorithms (RBA)* [3] in which the download throughput of each segment is used for bandwidth estimation, *buffer-based algorithms (BBA)* [4] in which the playout buffer level of the user's application is used for bitrate selection, and *hybrid-based algorithms (AdapTech)* [5] in which both the download throughput and playout buffer level are considered.

Information-Centric Networking (ICN), such as Content-Centric Networking (CCN) [6] and Named-Data Networking (NDN) [7], has been widely studied for efficient content distribution including video streaming [8]. In ICN, routers are generally equipped with cache storage, i.e., *in-network cache*, and can provide cached video contents in place of the original server(s). Although users can quickly download video contents from the routers, it is difficult for them to accurately estimate the available network bandwidth because they cannot specify whether a downloaded segment is to be obtained from the original content server(s) or the intermediate routers. Rate-based and hybrid-based algorithms, which

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

use download throughput for bandwidth estimation, may be negatively affected by in-network caches [9].

Goto *et al.* [10] evaluated Quality of Experience[1] (QoE) performance of three representative ABR algorithms in MPEG-DASH with occasional cache hits over ICN. They found that throughput-based algorithms such as RBA and AdapTech try to select a bitrate higher than that with conservative BBA and basically have better performance in terms of quality of the downloaded segments. However, they potentially suffer from *stalling* and *rebuffering* due to depletion of the playout buffer and from unnecessary changes in video segment quality due to misestimating the available network bandwidth. These events severely affect users' QoE. Furthermore, the severity of the effect differs between users depending on their *tastes* and *preferences* [10], [12]. For example, some users do not mind rebuffering as they prefer video segments with high-definition (HD) quality while other users dislike rebuffering as they prefer stable uninterrupted playback. Therefore, an ABR algorithm that performs well for users with various preferences is highly desired for ICN environments with in-network cache.

In this paper, we propose an intelligent QoE-Aware Bitrate selection method that works in cooperation with an in-network Caching function called *QoE-ABC*. It is suitable for video streaming over ICN. QoE-ABC focuses on the consecutiveness of cached segments to fully utilize them. Routers perform bandwidth estimation and notify the user's application of cached segment information. The user application executes the QoE-aware ABR algorithm and accordingly selects a bitrate that matches the bandwidth of the bottleneck link in the original server or of the intermediate routers depending on the situation. This means that the user's application does not unnecessarily select a higher bitrate when downloading from the server. Only when the segments have been provided by an intermediate router and the QoE is expected to be improved, the user's application aggressively select a higher bitrate to download high-quality segments without rebuffering and unnecessary changes in segment quality. Experiments designed to evaluate the QoE performance of our proposed algorithm compared with that of existing ABR algorithms revealed that it has the best performance for users with various preferences due to effectively utilizing in-network cached segments.

The rest of the paper is organized as follows. In Section II, we explain our motivation for this study, i.e., the problem that occurs when "MPEG-DASH meets ICN." Section III describes the key ideas for solving this problem. Section IV explains the design of our proposed method in detail. In Section V, we present simulation results showing the efficiency of the proposed method. In Section VII, we introduce related studies. Section VIII concludes the paper with a summary of the key points and a mention of future work.

[1]The definition of QoE commonly used in [11], [12] is explained in Section III.

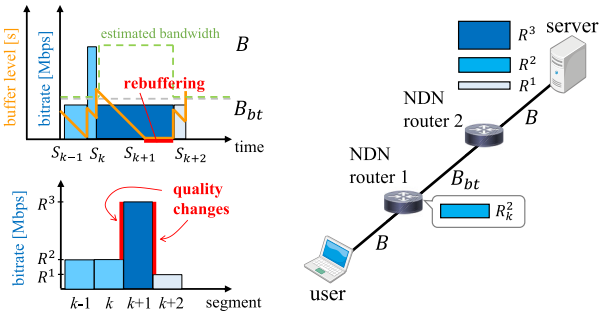

FIGURE 1. Effects of cached segments on user's QoE for DASH-like video streaming over NDN.

II. BACKGROUND AND MOTIVATION

A. NDN AND DASH

NDN is a representative platform for ICN communications. In NDN, a user transmits an Interest packet in order to obtain a Data packet from content servers. The Interest packet name indicates the desired content, e.g., "/abc.com/news/today," and the corresponding Data packet has the same name. These packets are transferred by NDN routers using name-based forwarding. An Interest packet is forwarded by looking up a name entry in Forwarding Information Base (FIB) in an NDN router, and the Data packet is transmitted in accordance with a Pending Interest Table (PIT) along the reverse path of the Interest packet. NDN routers in the path store the Data packet in their Content Store (CS) as in-network cache. When another Interest packet arrives at a router, the router checks the CS. If a Data packet whose name exactly-matches to the received Interest packet is found, a cache hit occurs and the router sends the Data packet from its CS. Otherwise, the router forwards the Interest packet toward the origin content server(s) in accordance with the FIB. Use of a CS enables users to quickly download contents and reduces the traffic load. Therefore, NDN is expected to be used for multimedia content delivery such as adaptive video streaming [13]–[15].

B. MOTIVATION

Figure 1 illustrates the effects of cached segments on the user's QoE for DASH-like video streaming over NDN. The bandwidth of the bottleneck link B_{bt} is much lower than that of the other links ($B > B_{bt}$). In the server, the video file is divided into K segments, and each segment has three-level representations of quality: R_1, R_2, and R_3 ($R_1 < R_2 < R_3$). The user's application executes the throughput-based ABR algorithm.

Herein, we assume that the k-th middle quality segment s_k with quality R_2 is accidentally cached in NDN router 1, as shown in Fig. 1. The user's application has downloaded segment s_{k-1}, which has quality $R_2 \simeq B_{bt}$ from the server, thus estimating that quality R_2 is appropriate for the next segment s_k. Since the user's application obtains s_k from the cache storage on router 1, the download throughput surprisingly increases to B, and the user's application requests the next segment s_{k+1} that have quality $R_3 (R_3 < B)$. Due to

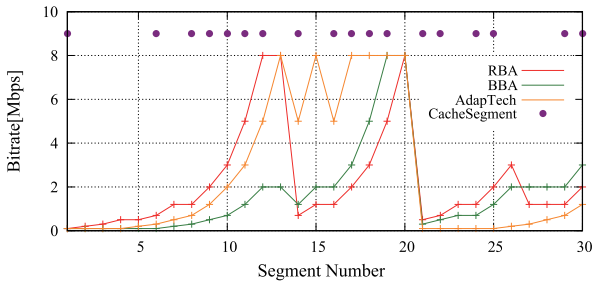

FIGURE 2. Bitrate selected by user's application.

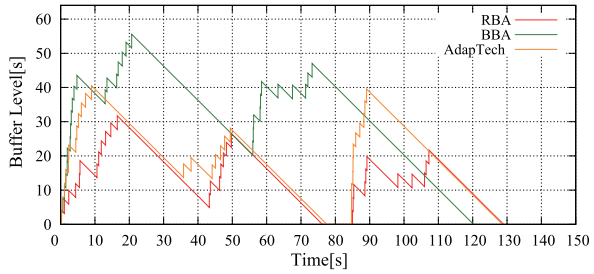

FIGURE 3. Playout buffer level.

misestimation of the throughput of the bottleneck link, the download time of segment s_{k+1} unexpectedly increases because the user's application needs to download it from the server through the bottleneck link, which has narrow bandwidth ($B_{bt} < R_3$). Meanwhile, the video content on the playout buffer in the user's device is consumed and then depleted, and playback is suspended until the next segment is newly downloaded. For the next segment s_{k+2}, the user's application selects the lowest quality R_1 to minimize the rebuffering time, resulting in a large change in quality.

To confirm our concern, we conducted a computer simulation in the above model. The bandwidth of the bottleneck link B_{bt} was set to 1 [Mbps], and B was set to 10 [Mbps]. Figure 2 shows the bitrate selected by the user's application for each ABR algorithm. The purple dots denote segments cached in router 1. As shown in the figure, when segments were consecutively downloaded from router 1, the two throughput-based algorithms, RBA and AdapTech, misestimated the bandwidth and attempted to increase the bitrate. Although sporadical cache hits may not have a great effect, and the RBA and AdapTech algorithms might absorb the effects, consecutive cache hits gradually worsen the RBA and AdapTech bitrate selections. After such worsening, at segment 13, the user's application misestimates the bandwidth and selects a bitrate (8 Mbps) that is higher than the bottleneck bandwidth (1 Mbps). The ABR algorithm running in the application then believes that the network condition has improved and the available bandwidth has increased because the video segments are continuously provided with a high throughput (10 Mbps) from in-network caches. However, segment 13 has not been stored in the router, thus the user's application needs to download it from the server with a narrow bandwidth. This leads to a large change in quality, and playback is suspended for about 10 [s] to enable the new segment to be rebuffered, as shown in Fig. 3.

One naive approach to overcoming this problem is to have an intermediate NDN router (router 2 in our example) measure the available bandwidth of the bottleneck link and notify the user's application, as is done in SAND [16], [17]. With this approach, the user's application can stably select a bitrate that matches the available bandwidth of the bottleneck link. However, usage of cached segments is highly constrained by this conservative bitrate selection. This means that one of the most important advantages of ICN, i.e., in-network caching, is negated. Badov *et al.* [18] determined that ICN routers should preferentially retain contents that have passed through congested links. However, depending on the application, especially for MPEG-DASH, it is a challenging problem to effectively and actively utilize in-network cache on the user side of the congestion. An intelligent ABR algorithm that leverages the in-network cache by inclusively considering user-perceived QoE performance is required.

III. DESIGN RATIONALE

One simplistic approach to fully utilizing in-network-cached segments is to have an intermediate router notify the user's application of information about cached segments in addition to available bandwidth information. If a requested segment is fortuitously stored in a router on the user side of the bottleneck link, the user's application can select a higher bitrate because the router can transmit the segment without it passing through the bottleneck link. If the next segment, however, is not stored in the router, the user's application needs to switch back to the previous low-quality bitrate, which involves the cost of quality switching. Therefore, whether in-network cache should be used or not should be decided with consideration of the user's QoE.

QoE is generally defined by considering four factors: (1) quality of downloaded segments, (2) quality changes between consecutive segments, (3) rebuffering time, and (4) startup delay. The higher the average bitrate of downloaded segments, the better the QoE. However, the quality changes between consecutive segments should be minimized. The startup delay, i.e., the time from clicking to select video content to the starting of the playback of the first segment, should be as short as possible, and rebuffering due to playout buffer depletion should not occur. To take account of these factors, we use an equation defined in [11], [12]:

$$QoE_{1,K} = \sum_{k=1}^{K} q(R_k) - \lambda \sum_{k=1}^{K-1} |q(R_{k+1}) - q(R_k)| - \mu \sum_{k=1}^{K} b_k - \mu_s D, \quad (1)$$

where one video is divided into K segments, R_k denotes the bitrate for segment k, and $q(R_k)$ is the utility function when the user watches a segment downloaded at a bitrate of R_k. The first term in the equation represents the total bitrate at

which the entire video was downloaded. The second term represents the sum of the quality changes between consecutive segments. The third term shows the sum of rebuffering time b_k. The fourth term D is startup delay. λ, μ, and μ_s are weighting factors reflecting user tastes.[2]

To address the trade-off between maximizing the total bitrate for the downloaded segments and minimizing the negative factors, i.e., quality changes, we focus on the *consecutiveness* of cached segments. As mentioned previously, because we assume a user application can obtain the minimum bandwidth information for the bottleneck link, rebuffering, another negative factor of QoE, never occurs.[3] If user applications know the minimum bandwidth, each user application tends to select a constant bitrate that matches the bandwidth of the bottleneck link for each segment. In the network, segments with the same bitrate are likely to be cached when the caching policy is an ICN representative one, i.e., Cache Everything Everywhere (CEE) and Least Recently Used (LRU).

When same-quality segments are continuously provided from cache, the user's application can download them at a high-quality bitrate with minimum quality changes. When segments are cached at random, the playback quality fluctuates greatly. We examined the number of consecutive cache hits that results in QoE improvement by using Eq. 1. We assumed a user application selects R^{ch} when downloading segments from in-network cache and selects $R^{sv}(< R^{ch})$ when downloading from a video server. We also assumed that the user application had already downloaded segment s_{k-1} from the server and that cache hits occur n consecutive times. The user's application then obtains segment s_{k+n} from the server again. The QoE for these segments is given by,

$$\begin{aligned} QoE^{ch}_{k-1,k+n} = R^{sv}_{k-1} + \sum_{j=k}^{k+n-1} R^{ch}_{j} + R^{sv}_{k+n} \\ - \lambda \left(|R^{ch}_{k} - R^{sv}_{k-1}| + |R^{sv}_{k+n} - R^{ch}_{k+n-1}|\right) \\ - \mu_s D. \end{aligned} \tag{2}$$

If the user's application obtains all the segments $[k-1 : k+n]$ from the server, the QoE is denoted as,

$$QoE^{sv}_{k-1,k+n} = \sum_{j=k-1}^{k+n} R^{sv}_{j} - \mu_s D. \tag{3}$$

The necessary condition for improving user QoE is,

$$QoE^{ch}_{k-1,k+n} > QoE^{sv}_{k-1,k+n}. \tag{4}$$

Therefore, when $n > 2\lambda$, QoE increases. In other words, n consecutive cache hits bring about definite QoE improvement for the user.

[2]This point is explained in detail in Section V-D.

[3]The method for notifying the user's application of the bandwidth information is explained in Section IV.

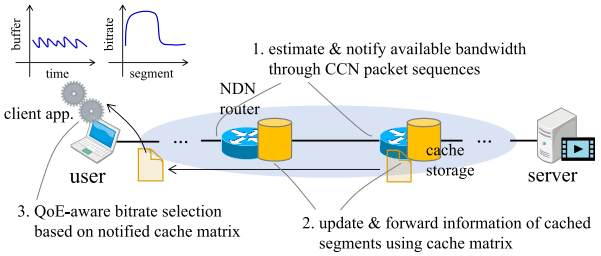

FIGURE 4. Overview of QoE-ABC.

IV. QoE-ABC: QoE-AWARE BITRATE SELECTION IN COOPERATION WITH IN-NETWORK CACHING

Since the consecutiveness of cache hits improves user QoE, we need to effectively take it into account for deciding bitrate. Our proposed QoE-ABC, i.e., QoE-Aware Bitrate selection in cooperation with in-network Caching, proactively focuses not only on the next segment to be downloaded but also several future segments for bitrate selection. As shown in Fig. 4, QoE-ABC is composed of three functions: (1) each intermediate router estimates the available bandwidth of its outgoing-link for contents and shares it with the downstream nodes, (2) each router notifies the user's application of which segments are in the router's cache by using a *cache matrix*, and (3) the user's application executes the QoE-aware ABR algorithm to select an appropriate bitrate on the basis of the received information in the cache matrix.

First, we explain how intermediate routers estimate the available bandwidth and pass the information to the downstream nodes. Badov *et al.* [18] estimated the bandwidth B_l of link l in a download flow using,

$$B_l = \frac{C_l}{F_l}, \tag{5}$$

where C_l is link capacity and F_l is the number of active flows passing through link l. A router calculates the number of concurrent flows F_l for each video segment k and adds the number to the last Data packet of segment k as B_{bt}. The downstream router that receives the transferred Data packet through link l updates the value only when $B_{l'} < B_{bt}$, where l' is the next downstream link for the router. This measurement packet is used to monitor the minimum available bandwidth of the download path, which is sent to a user application and used for executing the QoE-aware ABR algorithm, as described later. The application calculates the available bandwidth for the video streaming server on the basis of the exponentially weighted moving average of the received B_{bt}.

Alg. 1 shows how each router updates the cache matrix and shares it with the streaming user application. Router r knows whether segment k of bitrate R_j is in its cache and stores this information in a local binary matrix, $\{CS^r_{j,k}\}$. When router r receives a Data packet for segment c, it tries to copy this matrix into cache matrix $\{CM_{j,k}\}$. In NDN, the optional information field of a Data packet has limited capacity, thus the amount of information stored in the Data packet should

Algorithm 1 Update Cache matrix

Input: $\{CS^r_{j,k}\}, B_l, c, n$
Output: $\{CM_{j,k}\}$
1: When router r receives a Data packet for segment c
2: **for** $j = 1$ to $Q(B_l)$ **do**
3: **for** $k = c$ to $c + n$ **do**
4: **if** $CM_{j,k} = 1$ **then**
5: do nothing
6: **else**
7: $CM_{j,k} \leftarrow CS^r_{j,k}$
8: **end if**
9: **end for**
10: **end for**
11: Look up PIT to obtain a corresponding PIT entry p
12: **if** $p \neq null$ **then**
13: Forward a Data packet to downstream face(s) in accordance with PIT entry p
14: **else**
15: Discard the received Data packet
16: **end if**
17: **return**

be reduced as much as possible. If each router copies all elements of its cached segments in the storage, much of the space in the header field of the Data packet will be consumed. Moreover, parsing the packet header and processing/replacing the values will cause a packet processing delay at the router and degrade packet forwarding performance. Thus, in our proposed algorithm, the rows in the cache matrix are limited by the available bandwidth B_l[4] of output-link l, and the columns are limited by design parameter n to reduce the amount of stored information. When a user application requests segment c, router r notifies the user application of elements only in the range $[c, c + n]$. As a result, the computation complexity of this algorithm is $\mathcal{O}(\mathcal{J} \cdot \mathcal{N})$. After the CS procedure, the router looks up the PIT to find the PIT entry for the Data packet forwarding. If an entry is found, i.e., $p \neq null$, the router forwards the Data packet to the downstream face(s) in accordance with the PIT entry. Otherwise, the received Data packet is dropped.

Figure 5 illustrates an example of sharing information for cached segments. Each router ($r1$ and $r2$) prepares a binary cache matrix $\{CS^{r1/r2}_{j,k}\}$ in advance. Before $r1$ sends a Data packet to $r1$, it adds its matrix to the packet as $\{CM_{j,k}\}$. The available bandwidth of the output link between $r2$ and $r1$ is $B_{r1,r2}$ ($R_2 < B_{r1,r2} < R_3$), thus $r2$ partially copies $CS^{r2}_{j,k}$, for which the number of rows is less than 3. The number of columns is limited to the number of segments $[k, k+n]$. When $r2$ forwards the Data packet to $r1$, $r1$ conducts the same procedures. However, if the elements have already been set to 1, $r1$ does not update their values in order to keep options that user application u downloads the cached segments $R_{1,c}$

[4] $Q(x)$ is a function for quantizing the actual bandwidth to a representation using $\max_{\{\forall R^i \in R | R^i \leq x\}} R^i$.

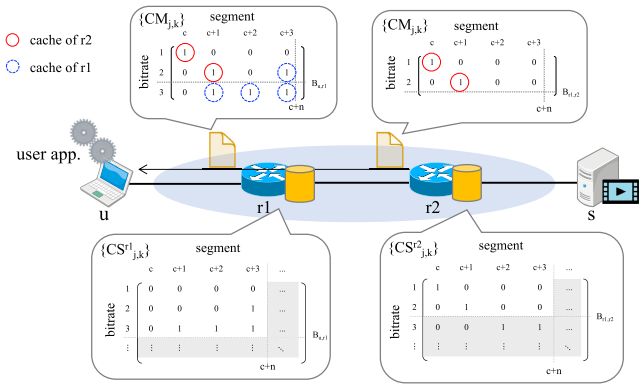

FIGURE 5. Sharing of meta-information for cached segments using cache matrix.

and $R_{2,c+1}$ from $r2$. At $r1$, the available bandwidth is $B_{u,r1}$ ($R_3 < B_{u,r1} < R_4$), thus the matrix is expanded to three rows × four columns.

Finally, the user's client application runs QoE-aware bitrate selection as shown in Alg. 2. In the initial state, *counter* has been set to 0. When the application has received all the Data packets for segment $c - 1$ (line 1), it checks the received cache matrix $\{CM_{j,k}\}$ and counts the number of consecutive segments CC_j for each bitrate j (lines 2–11). If $CC_j = n$, which means there are enough cached segments for satisfactory QoE, the representation R_{cache} is updated to R_j, and the application saves n to *counter* (lines 12–15) for use in selecting the bitrate for subsequent segments as described in lines 29–42. For all $j \in \mathcal{J}$, the bitrates are looked up in ascending order, thus the highest bitrate that gives the maximum QoE can be selected. If n consecutive segments are fortuitously found in the matrix, i.e., $R_{cache} \neq null$, the representation of the next segment R^{c+1} is set to that bitrate (lines 17–18). If such segments are not found, i.e., $R_{cache} = null$, the representation of the next segment R^{c+1} is selected on the basis of the current available bandwidth of the bottleneck link between the user and the server (lines 19–28). If the current playout buffer level $b_{current}$ is smaller than the conservative threshold b_{con}, the application decreases one representation R^{c+1}_{--} in order to avoid buffer depletion (lines 21–22). If $b_{current}$ is larger than b_{agg}, which is the aggressive threshold, the application simply increases one representation R^{c+1}_{++} in order to download a slightly higher bitrate and thereby increases QoE (lines 23–24). Otherwise, the original bitrate computed by the QoE-ABC algorithm is used without any change (lines 20, 25–26).

Once the counter value is updated to n, the adaptation mode switches to one in which the user application selects the same bitrate as that for the last segment R_{c-1} for the next segment R_c, i.e., $R^c \leftarrow R^{c-1}$ (line 30). After setting the next bitrate, the application checks the value of n in the newly received cache matrix $CM_{j,k}$ and updates it $counter \leftarrow CC_j$ whether the n consecutiveness is intermittent or not (lines 31–41). If the counter value is reduced to 0 and the application has successfully downloaded the cached video segments, the adaptation mode is switched back to the

original one, lines 2–28, and the application again waits for a chance to obtain the next consecutively cached segments. With this QoE-aware approach, a user application can obtain video segments from in-network cache at appropriate bitrates without consuming much of the playout buffer while avoiding congestion.

The computation complexity of this algorithm is $\mathcal{O}(\mathcal{J} \cdot \mathcal{N})$, where $\mathcal{J}$ is the number of representations (i.e., encoding bitrates) of streaming video content. In general, $|\mathcal{J}|$ is 12 at the most. $\mathcal{N}$ (or n) is a control parameter in the client's ABR algorithm. In our setting, $\mathcal{N} = 3$ is basically used, and $\mathcal{N} = 7$ is used only for a user whose QoE preference is "avoid instability," as described in Section V-D. Thus, even in the worst case, computation complexity is relatively low.

Algorithm 2 QoE-Aware Bitrate selection

Input: $\{CM_{j,k}\}, B_{bt}, R_{c-1}, n, counter$
Output: $R^c, counter$

When user application u receives last segment $c-1$
if $counter = 0$ **then**
  for $j = 1$ to J **do**
    $CC_j \leftarrow 0$
    for $k = c$ to $c + n - 1$ **do**
      if $CM_{j,k} = 1$ **then**
        $CC_j \leftarrow CC_j + 1$
      else
        $CC_j \leftarrow 0$
      end if
    end for
    if $CC_j = n$ **then**
      $R_{cache} \leftarrow R_j$
      $counter \leftarrow n$
    end if
  end for
  if $R_{cache} \neq null$ **then**
    $R^c \leftarrow R_{cache}$
  else
    $R^c \leftarrow Q(B_{bt})$
    if $b_{current} < b_{con}$ **then**
      $R^c \leftarrow R^{c-1}_{--}$
    else if $b_{agg} < b_{current}$ **then**
      $R^c \leftarrow R^{c-1}_{++}$
    else
      do nothing
    end if
  end if
else
  $R^c \leftarrow R^{c-1}$
  for $j = R^c$ **do**
    $CC_j \leftarrow 0$
    for $k = c + 1$ to $c + n - 1$ **do**
      if $CM_{j,k} = 1$ **then**
        $CC_j \leftarrow CC_j + 1$
      else
        break
      end if
    end for
    $counter \leftarrow CC_j$
  end for
end if

V. EVALUATION

A. MODEL

We evaluated our proposed QoE-ABC by modifying NS-3 [19] based ndnSIM [20] to realize DASH-like communications over ICN. Specifically, we used ndnSIM 2.1 with the NDN Forwarding Daemon (NFD) [21] as the packet forwarding engine. For routing, we configured static routes for the data retrieval paths rather than dynamic routes as in the named-data link state routing protocol [22]. We used the dumbbell model for the evaluation topology with a bottleneck link bandwidth of 1.2 [Mbps]. To observe the effects of segment-level cache hits, segments were randomly stored in router 1 on the user side of the bottleneck link. The duration of the video content was 120 [s], and that for each segment was 4 [s]. We prepared ten representation levels: $\mathcal{R} = \{0.1, 0.2, 0.3, 0.5, 0.7, 1.2, 2.0, 3.0, 5.0, 8.0[\text{Mbps}]\}$. The playout buffer capacity was set to 60 [s]. The buffer thresholds, b_{con} and b_{agg}, were set to 12 [s] and 20[s], respectively. The video downloaded by the user contained actual multimedia contents.[5] The performance of the proposed algorithm was compared against those of RBA, BBA, and AdapTech, which are described in Section II-B. The QoE parameters $\{\lambda, \mu, \mu_s\}$ were set to $\{1.0, 4.3, 4.3\}$, respectively, thus we set the control parameter n in QoE-ABC to 3.

B. QoE-ABC IN ACTION

We investigated the bitrate selection behavior of our proposed QoE-ABC algorithm. Figures 6 and 7 show the bitrate selected by each ABR algorithm and the characteristics of the playout buffer level, respectively. In this example, the segments for the first half of the video were consecutively cached while those in the second half were not. As shown in Fig. 6, the proposed algorithm proactively downloaded the segments with high representations for segments [2, 6]. Segments 7 and 8 were not stored in router 1, thus the algorithm immediately reduced the bitrate used to match the bandwidth of the bottleneck link in order to avoid buffer depletion. Then, for segments [9, 13], the algorithm selected the high bitrate again because these segments were stably stored in cache.

In the second half of the video, the segments were sporadically cached, thus the QoE-ABC algorithm did not unnecessarily increase the bitrate and instead selected the appropriate bitrate in a stable manner. As a result, as shown in Fig. 6, the playout buffer level with the QoE-ABC algorithm was lower than with the other three algorithms. The QoE-ABC

[5] https://peach.blender.org

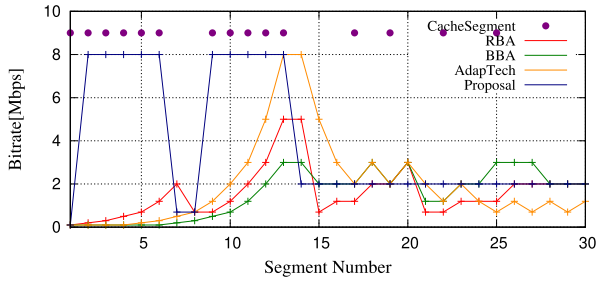

FIGURE 6. Bitrate by selected the user application.

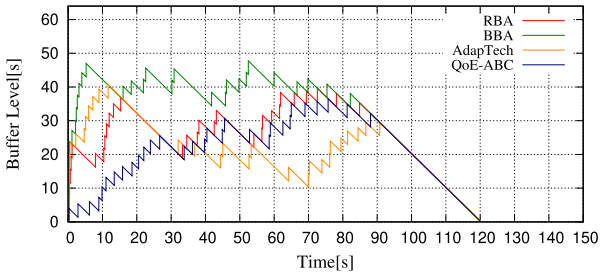

FIGURE 7. Playout buffer level.

algorithm knew the available bandwidth for router 1 or the server, thus it could select the bitrate aggressively while keeping the buffer level low and also avoiding stalling as shown in Fig. 7. As a result, there was no stalling or redundant quality changes, which resulted in a high QoE. In contrast, the other three algorithms selected bitrates lower than those selected by the QoE-ABC algorithm to avoid buffer depletion from the beginning. Since these algorithm use end-host-based (not network-assisted) approaches, they increase the bitrate carefully. The throughput-based approaches, RBA and AdapTech, try to actively increase the bitrate compared with the more conservative BBA. However, their QoE improvement by bitrate selection was limited compared with that of QoE-ABC because, in the second half of the video, throughput fluctuations due to the sporadical cache hits caused many quality changes. Although BBA is unlikely to produce negative influences due to cached segments, it cannot select high-quality segments due to its conservative approach.

C. QoE PERFORMANCE

In this section, we evaluated the total QoE performance for a random segment cache hit by changing the number of cached segments in the network from 0 to 30, i.e., {0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30}. For 0 and 30 segments, we conducted only one simulation because cache hits never occur or always occur in these two cases. For the remaining 9 cases, we conducted 100 simulations each to evaluate performance for 100 random video segment placement patterns. We therefore conducted 902 simulations in total. As shown in Eq. 1, the total QoE is based on the download bitrate, quality changes, rebuffering time, and startup delay. Figure 8 shows the total QoE performance of each algorithm. Our proposed QoE-ABC performed well compared with the other three.

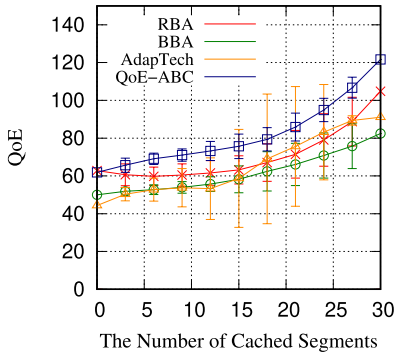

FIGURE 8. Total QoE performance.

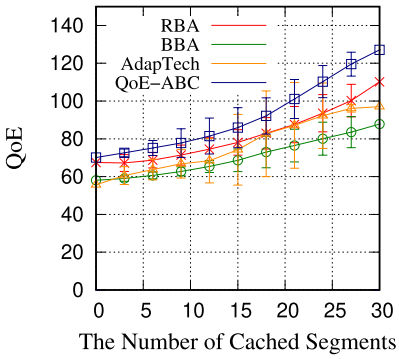

FIGURE 9. QoE component: download bitrate.

As shown in Fig. 9, it selected a higher download bitrate than the other algorithms. Since the opportunities for consecutive cached segments will increase proportionally with the number of cached segments, our proposed QoE-ABC will be able to download at a higher bitrate from in-network cache when there is a higher cache hit rate.

With the throughput-based AdapTech, rebuffering occurs due to misestimating the download throughput and inappropriate bitrate selection, as shown in Fig. 11. Playback is suspended until the next segment is downloaded, thus the user has to wait for the segment to download. Rebuffering never occurs with QoE-ABC because it can measure the minimum bandwidth of the bottleneck link and select an appropriate bitrate. This enables the user to watch the video without any stress due to the absence of video playback suspension.

RBA and AdapTech cause unnecessary fluctuations in segment quality because they are throughput-oriented and thus cannot take into account the effect of cache hits when downloading segments. QoE-ABC can change the quality because it drastically increases the bitrate at the beginning of consecutive segment cache hits and decreases at the end. Nevertheless, the quality changes with QoE-ABC are comparable to those of RBA and AdapTech, as shown in Fig. 10. This means that QoE-ABC simply requires necessary quality changes for improving user QoE.

QoE-ABC shows QoE performance for startup delay comparable to that of RBA and BBA, as shown in Fig. 12. Hence, there is no great effect on total QoE performance. The startup

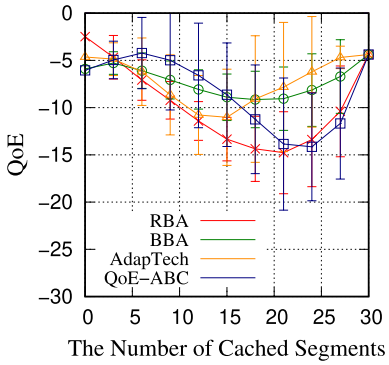

FIGURE 10. QoE component: quality change.

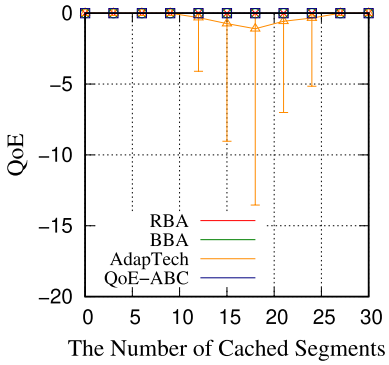

FIGURE 11. QoE component: rebuffering time.

delay of AdapTech is longer than for the other three algorithms because the user application continues to select the lowest bitrate and does not start playback until finishing the download of the initial three segments in order to avoid panic mode.

D. CONSIDERING USER DIVERSITY

The tastes and preferences of streaming users, i.e., different perception for watching a video segment with the same quality and bitrate, is an important factor in evaluating QoE performance. We investigated user diversity by using weighting parameters as studied in research [10], [12]. In [12], $q(R_k)$ in Eq. 1 mapped the downloaded bitrate to the quality perceived by a user, and there are three types of quality perception: $QoE_{lin.}$, QoE_{log}, and QoE_{HD}. As shown in Table 1, with $QoE_{lin.}$ and QoE_{log}, $q(R_k)$ gradually increases with the download bitrate. QoE_{log} considers only the saturation of QoE improvement for downloading at a higher bitrate [12], [23]. In contract, with QoE_{HD}, user-perceived QoE increases in a stepwise manner, meaning that the QoE improvement for a streaming user depends on whether the downloaded segment has HD quality.

Furthermore, we evaluated QoE from the viewpoint of user preferences. There are three basic user types: *avoid instability, balanced,* and *avoid rebuffering* [11]. A user whose type is "avoid instability" prefers to watch videos with stable quality. One whose type is "avoid rebuffering" dislikes

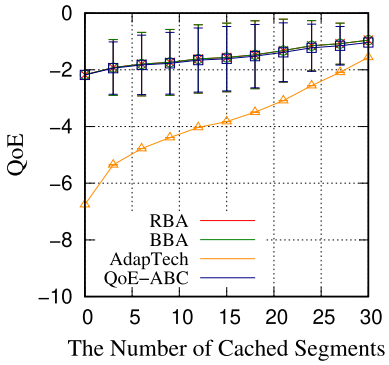

FIGURE 12. QoE component: startup delay.

TABLE 1. Parameters for considering differences in quality perception.

	$q(R_k)$
$QoE_{lin.}$	R
QoE_{log}	$log(R/R_{min})$
QoE_{hd}	$0.1 \rightarrow 0.6, 0.2 \rightarrow 0.8, 0.3 \rightarrow 1, 0.5 \rightarrow 1.4,$ $0.7 \rightarrow 1.9, 1.2 \rightarrow 3, 2.0 \rightarrow 12, 3.0 \rightarrow 16,$ $5.0 \rightarrow 22, 8.0 \rightarrow 33$

suspension of playback due to rebuffering. Table 2 shows the nine combinations of QoE parameter setting used in this evaluation.

Table 3 shows the QoE performance for each user type. Each element in the table is the average value of the total-QoE/QoE component. The QoE component for "startup delay" is omitted because there was no large difference. The number of cached segments in router 1 was 15, which means it had a cache hit rate of 50%.

QoE-ABC had the best performance for all user types and quality perceptions. This is because it can select a bitrate that matches the available link bandwidth in a stable manner and does not cause unnecessary changes in segment quality. Therefore, it performed well, especially for the "avoid instability" user type. For QoE_{HD}, its QoE performance was much better than that of RBA. Although RBA can download at higher quality bitrate (264.8) than QoE-ABC (240.1), it frequently switches segment quality depending on the measured throughput and causes considerable QoE degradation (−398.5) due to cache hits. The QoE degradation of QoE-ABC is much low (−40.4), because it does not unnecessarily increase the bitrate and selects a stable bitrate even if there are cached segments. The performances of BBA and AdapTech were not much lower than that of RBA, but they had characteristics similar to those of the lowest RBA.

For the "balanced" user type, the performance of QoE-ABC was significantly higher than those of RBA and BBA, especially for QoE_{HD}. The total QoE of AdapTech was slightly less than that of QoE-ABC because rebuffering did not occur in QoE-ABC while AdapTech suffered from long rebuffering times (6.2 [s] at the maximum). For QoE_{lin} and QoE_{log}, QoE-ABC performed better than

TABLE 2. Parameters for considering user diversity.

	Avoid instability	*Balanced*	*Avoid rebuffering*
$QoE_{lin.}$	{3.0, 8.0, 8.0}	{1.0, 8.0, 8.0}	{1.0, 16.0, 16.0}
QoE_{log}	{3.0, 4.3, 4.3}	{1.0, 4.3, 4.3}	{1.0, 8.6, 8.6}
QoE_{HD}	{3.0, 8.0, 8.0}	{1.0, 8.0, 8.0}	{1.0, 16.0, 16.0}

AdapTech. For the "avoid rebuffering" user type, QoE-ABC had the best performance for QoE_{in}, QoE_{log}, and QoE_{HD}. These results demonstrate that QoE-ABC inclusively has stable and better QoE performance for various user types than the existing end-to-end based ABR algorithms due to utilizing in-network computing functionalities, i.e., notifications of available throughput and cached segment information.

E. DATASET-DRIVEN EVALUATION

Additionally, we evaluated our proposed algorithm using the real bandwidth traces dataset published by the U.S. Federal Communications Commission[6] in order to validate its effectiveness and practicality. We changed the available bandwidth of the bottleneck link in accordance with the trace data and evaluated the QoE of streaming users. As the user type, we used the combination of QoE_{log} and *Balanced*, which was also used in the evaluation described in Section V-C.

Figure 13 shows the total QoE performance for each algorithm under bandwidth fluctuation conditions. This figure corresponds to Fig. 8 in which the available bandwidth of the bottleneck link is constant bitrate (CBR). Compared with the results in Fig. 8, the QoE performance of all algorithms was slightly higher because the average bandwidth of the trace data was about twice as wide. However, there was no large difference in the rankings of performance. Our proposed QoE-ABC stably provided high QoE performance as it did in the previous evaluation. BBA performed buffer-driven conservative bitrate selection, thus the user QoE improvement was slightly limited compared with that of RBA and AdapTech, as shown in Fig. 14.

In terms of video quality variation (Fig. 15), BBA and AdapTech mitigated the effects of bandwidth fluctuation and reduced the quality changes because these buffer-oriented methods absorb fluctuations in throughput in the bottleneck link by using playout buffers. Figure 16 shows the QoE performance of rebuffering. We did not observe any stalling of video segments in this trace-driven evaluation. The average throughput of the trace data was higher than for the case shown in Fig. 11. This may be because the playout buffer was fortunately able to absorb the bandwidth fluctuation. The video utilized in this evaluation was encoded with ten representations; however, the actual number of bits may have been low, especially for the higher representations, due to the poorness of the original video's resolution. The startup delay was shorter with each algorithm because the average

[6]https://www.fcc.gov/oet/mba/raw-data-releases

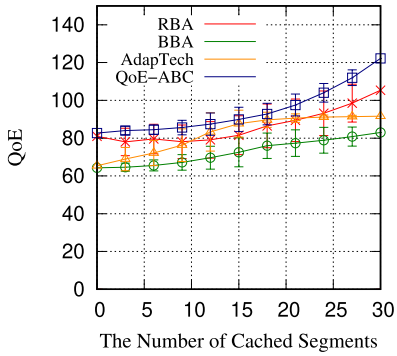

FIGURE 13. Total QoE performance under bandwidth fluctuation conditions.

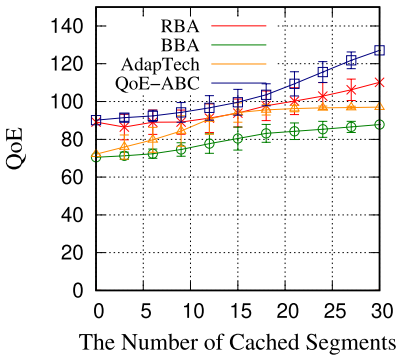

FIGURE 14. QoE component: downloaded bitrate under bandwidth fluctuation conditions.

throughput was higher than in the CBR scenario. There were slight differences compared with the CBR scenario due to the throughput fluctuation. However, QoE-ABC downloaded at a higher bitrate from the in-network caches only when a sufficient number of segments were consecutively stored and the user QoE was expected to be improved due to necessary quality changes.

These results indicate that our proposed QoE-ABC performs more effectively in a real Internet environment than existing end-host-based ABR algorithms due to proactively using in-network functions, i.e., caching and notifications of bandwidth information.

VI. DISCUSSION

QoE Optimization: The QoE function can vary across users, contents, service types, and so on, and it is often not known or commonalized. Thus, the optimization of the QoE function for a particular user is a challenging problem. One possible approach to improving the QoE for various users is to customize the QoE function for each user by using existing HTTP tracking technologies such as Cookie. To this end, a feedback-loop from user applications to the video streaming server is necessary, as investigated in SENSEI [24]. SENSEI utilizes Amazon MTurk to conduct MOS evaluations for improving the QoE of streaming users. The idea of this kind of feedback-style QoE control can be applied to our assumed QoE control. Streaming user applications feed back perceived

TABLE 3. QoE performance considering user diversity.

		Avoid instability				*Balanced*				*Avoid rebuffering*			
		total	bitrate	change	rebuff.	total	bitrate	change	rebuff.	total	bitrate	change	rebuff.
RBA	$QoE_{lin.}$	-22.2	58.2	-77.4	0	29.4	58.2	-25.8	0	26.53	58.2	25.8	0
	QoE_{log}	36.6	78.1	-40.0	0	63.2	78.1	-13.3	0	61.7	78.1	-13.3	0
	QoE_{HD}	-136.6	264.8	-398.5	0	129.1	264.8	-132.8	0	126.2	264.8	-132.8	0
BBA	$QoE_{lin.}$	-3.3	58.1	-58.5	0	35.7	58.1	-19.5	0	32.8	58.1	-19.5	0
	QoE_{log}	40.4	68.7	-26.7	0	58.2	68.7	-8.9	0	56.6	68.7	-8.9	0
	QoE_{HD}	8.0	278.0	-267.1	0	186.1	278.0	-89.0	0	183.2	278.0	- 89.0	0
AdapTech	$QoE_{lin.}$	0.8	77.9	-68.6	-1.4	46.54	77.9	-22.9	-1.4	38.1	77.9	-22.9	-2.7
	QoE_{log}	36.6	74.2	-33.1	-0.7	58.7	74.2	-11.0	-0.7	54.1	74.2	-11.0	-1.5
	QoE_{HD}	47.5	345.2	-289.2	-1.4	240.33	345.2	-96.4	-1.4	231.9	345.2	-96.4	-2.7
QoE-ABC	$QoE_{lin.}$	36.3	46.6	-7.3	0	57.6	83.5	-22.9	0	54.6	83.5	-22.9	0
	QoE_{log}	64.7	75.8	-9.4	0	75.7	85.9	-8.6	0	74.1	85.9	-8.6	0
	QoE_{HD}	196.7	240.1	-40.4	0	290.9	386.8	-92.9	0	287.9	386.8	-92.9	0

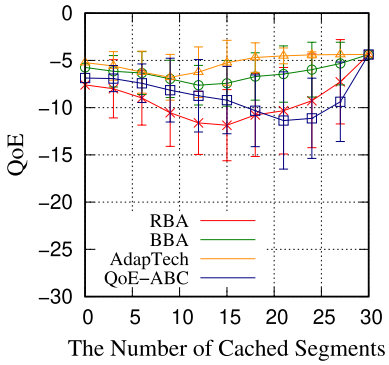

FIGURE 15. QoE component: quality change under bandwidth fluctuation conditions.

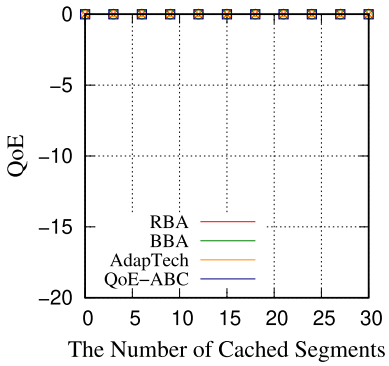

FIGURE 16. QoE component: rebuffering time under bandwidth fluctuation conditions.

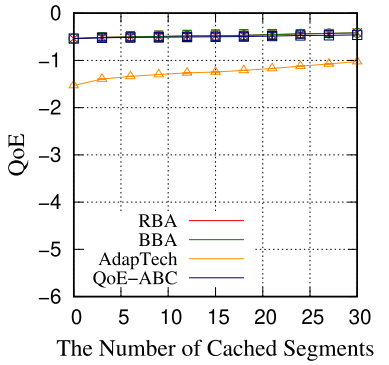

FIGURE 17. QoE component: startup delay under bandwidth fluctuation conditions.

QoE assessment and map the reported results for use as weighting parameters to a common QoE function.

Bandwidth Estimation: In our bandwidth estimation method, a router calculates the number of concurrent flows F_l for each video segment k and sends them to the client application by adding the information to the last Data packet comprising video segment k. After receiving the last packet, the client immediately calculates the next segment bitrate and requests the next segment $k+1$, thus the possibility of mis-estimation of B_l should be low. However, when the number of flows in a bottleneck link fluctuates greatly relative to the above time scale, we need to take it into account by shortening the time period for bandwidth estimation, e.g., per 10 or 100 packets for each segment. Also, in our bandwidth estimation method, we use the number of concurrent flows F_l in the bottleneck link l. It is difficult to estimate the precise value of available bandwidth, when the number of concurrent flows on the link increases because probability that a flow come in or out on the link increases. Some modifications that estimate the available bandwidth in a stable manner are required for large scale scenarios.

Cache Matrix Limitation: In our algorithm, router r checks the cached video segments and creates cache matrix $\{CS^r_{j,k}\}$ in advance to enable it to proactively share the cached information. When the number of cached video segments increases, router r creates a larger matrix for each content and maintains them. This requires a large amount of memory usage. If our proposed protocols are applied to all the video contents, the router's processing might not scale. Thus, we need to partially apply our proposed protocols to specific contents, such as popular contents requested by many users.

ICN Deployment & Implementation: We assume that our proposed ICN-based video streaming applications run over the IP as defined in a deployment scheme called "ICN-as-a-Overlay" in RFC8763 [25]. We are currently developing an open source software router called "Cefore" that will enable ICN communications [26], [27]. Cefore is compliant with CCNx defined in RFC8569 [28] and RFC8609 [29] and works over the Internet as a software

router that supports TCP/UDP applications by using the IP tunnel as recommended in RFC8763. Our assumed deployment model is not limited to the ICN-as-a-Overlay scheme; it can also be used in the ICN-as-a-Slice scheme [30]. As studied in this approach, we believe that the proposed ICN-based video streaming service can be implemented in a virtualized network service function as a 5G service slice at low cost by using modern virtualization technologies. The in-network caching function of ICN would be provided by gateway nodes, i.e., ICN service routers, between the edge and the Internet core as a video streaming service slice. There are a variety of video contents, and slice capacity is limited, thus which video contents to be stored on each slice should be carefully decided on the basis of popularity, efficiency, and/or other appropriate metrics.

VII. RELATED WORK

A. VIDEO STREAMING OVER HTTP

Tian and Liu [31] examined the trade-off between responsiveness to network congestion and smoothness of TCP throughput in DASH with multiple Content Delivery Network (CDN) servers. To balance this trade-off, they proposed a video adaptation algorithm that a user sends feedback signals to select an appropriate bitrate on the basis of the amount of playout buffer. FESTIVE [32] and PANDA [33] are hybrid-based ABR algorithms that focus on fairness among streaming users. FESTIVE conducts randomized scheduling of downloading video segments to avoid synchronization and stateful bitrate selection to efficiently stabilize performance. PANDA applies the principle of TCP's Additive Increase Multiplicative Decrease (AIMD) to an application-level bitrate selection scheme to improve fairness among users. Qin *et al.* [34] analyzed the characteristics of variable bitrate (VBR)-encoding by using video datasets with diverse genre content, encoding technologies, and platforms and then identified the characteristics of each VBR encoding type to investigate the effects on user QoE. Their investigation showed that high-quality video segments with complex scenes bring higher QoE improvement compared with those with simple scenes. Their proposed algorithm utilizes a proactive approach similar to that of our proposed QoE-ABC, which considers future several segments as well as the next segment. However, all of these approaches assume an HTTP environment over TCP/IP, not over ICN, thus the proposed solutions cannot be simply applied to adaptive streaming over ICN due to the architectural gap.

B. VIDEO STREAMING OVER ICN

Adaptive video streaming over ICN is a promising solution for reducing video traffic by leveraging in-network caching. The content name-based real-time streaming [35], [36] framework uses "symbolic interest," which eliminates sequence number identification and enables content-based streaming and multicasting using a network-condition-oriented rate control scheme. However, this approach is limited to "real-time" video streaming using scalable video coding, not DASH-like VoD streaming using adaptive video coding.

Grandl *et al.* [9] studied the problem of download throughput fluctuation similar to our concern, as described in Section II-B. They experimentally clarified the trajectory of rate selection by streaming users and then analyzed the cache-server oscillations that occur due to misestimating throughput. To overcome this problem, they introduced DASH-INC, a rate adaptation mechanism to select an appropriate bitrate in accordance with the in-network cache rate. However, their paper only introduced the concept of an ABR algorithm, not a specific algorithm. Lederer *et al.* [37] conducted trace-driven experiments using real-world mobile bandwidth traces that demonstrated the effectiveness of multi-path/-link in-parallel data transmission. Their paper evaluated performance only for download throughput and buffer level, not total QoE performance. Furthermore, specific adaptation logic has not been reported.

Samain *et al.* [13] presented a network-assisted bitrate selection mechanism for avoiding bitrate oscillations induced by in-network cache while maintaining a comparable average quality and increasing cache hit ratio. Nguyen *et al.* [14] and Ueda *et al.* [15] improved the adaptation logic run on the user applications by using congestion feedback to optimize utility fairness among the users. Fairness and stability in video streaming were improved and the cache hit ratio was increased by using this adaptation logic. However, these approaches passively utilize the in-network cache functionality while ours actively tries to leverage cached segments by sharing their meta-information to enhance user QoE. Furthermore, these approaches focus on performance for average quality, quality switching, or rebuffering, not total QoE performance. Additionally, the diversity of QoE requirements stemming from differences in user preferences has not been evaluated.

Li *et al.* [38] investigated a caching scheme in which the high-bitrate segments are stored at the network edge while low-bitrate contents are pushed into the network core. With this cache partitioning, bitrate oscillation due to in-network cache is noticeably reduced while ensuring high-quality video playback. This caching scheme, however, is executed in routers along the download path while our proposed ABR algorithm is executed on a user device.

Lederer *et al.* [39] examined ways to implement DASH over ICN. In their groundbreaking implementation, ISO/IEC MPEG standard DASH and ICN representative CCN [6] were used. They evaluated one of the most important advantages of ICN, i.e., leveraging multiple wireless links and interfaces such as WiFi and 4G/LTE in parallel. Their proposed DASH over ICN approach can outperform the classical DASH with single-path communications in terms of throughput, playback stability, and communications resiliency. This implementation is helpful for implementing our proposed QoE-ABC in real software programs.

VIII. CONCLUSION

Application of ICN to MPEG-DASH-like adaptive video streaming has been widely studied to leverage an attractive feature of ICN, i.e., in-network cache. However, when ICN has been simply applied to video streaming, the in-network cache negatively affects user QoE. In this paper, we investigated this problem and confirmed that it is real. We presented QoE-ABC to overcome this problem. With this method, routers share information about the available bandwidth and cached segments to users. A user runs a QoE-aware bitrate selection algorithm that focuses on the consecutiveness of cached segments and uses a higher download bitrate only when the total QoE is expected to be improved. Computer simulation demonstrated that proposed QoE-ABC achieves stable and better QoE performance for various user types with minimum quality changes and no rebuffering. Future work includes developing and implementing our algorithm in open-source ICN software such as Cefore and comparing its QoE performance with that of representative DASH/HLS algorithms in a visible manner. Furthermore, a cooperative design that takes into account both the ABR algorithm and the lower transport layer technologies should be addressed.

REFERENCES

[1] *Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats*, Standard ISO/IEC 23009-1:2014, May 2014. [Online]. Available: https://www.iso.org/standard/65274.html

[2] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1842–1862, 3rd Quart., 2017.

[3] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, Feb. 2011, pp. 169–174.

[4] 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.

[5] S. Akhshabi, S. Narayanaswamy, A. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," *Signal Process., Image Commun.*, vol. 27, no. 4, pp. 271–287, 2012.

[6] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. ACM CoNEXT*, Sep. 2019, vol. E102-B, no. 9, pp. 1–12.

[7] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.

[8] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, and D. Rossi, "Dynamic adaptive video streaming: Towards a systematic comparison of ICN and TCP/IP," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2166–2181, Oct. 2017.

[9] R. Grandl, K. Su, and C. Westphal, "On the interaction of adaptive video streaming with content-centric networking," in *Proc. 20th IEEE Int. Packet Video Workshop*, Dec. 2013, pp. 1–8.

[10] K. Goto, Y. Hayamizu, M. Bandai, and M. Yamamoto, "QoE performance of adaptive video streaming in information centric networks," in *Proc. IEEE LANMAN*, Jul. 2019, pp. 1–2.

[11] 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.

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

[13] J. Samain, G. Carofiglio, M. Tortelli, and D. Rossi, "A simple yet effective network-assisted signal for enhanced DASH quality of experience," in *Proc. ACM NOSSDAV*, Jun. 2018, pp. 55–60.

[14] D. Nguyen, J. Jin, and A. Tagami, "Cache-friendly streaming bitrate adaptation by congestion feedback in ICN," in *Proc. ACM ICN*, Sep. 2016, pp. 71–76.

[15] K. Ueda, D. Nguyen, M. Miyamoto, S. Aikawa, Y. Yoshida, and A. Tagami, "Demo: Dynamic adaptive streaming over NDN using explicit congestion feedback," in *Proc. IEEE LANMAN*, Jul. 2017, pp. 1–2.

[16] *Information Technology—Dynamic Adaptive Streaming Over 3 HTTP (DASH)—Part 5: Server and Network Assisted DASH (SAND)*, Standard ISO/IEC 23009-5:2017, Feb. 2017. [Online]. Available: https://www.iso.org/standard/69079.html

[17] E. Thomas, M. van Deventer, T. Stockhammer, A. Begen, and J. Famaey, "Enhancing MPEG DASH performance via server and network assistance," *SMPTE Motion Imag. J.*, vol. 126, no. 1, pp. 22–27, Jan./Feb. 2017.

[18] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, "Congestion-aware caching and search in information-centric networks," in *Proc. ACM ICN*, Sep. 2015, pp. 37–46.

[19] *Network Simulator NS-3*. Accessed: Dec. 10, 2021. [Online]. Available: https://www.nsnam.org/

[20] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "NDNSIM 2.0: A new version of the NDN simulator for NS-3," NDN, USA, Tech. Rep. NDN-0028, Revision 2, 2016. Accessed: Dec. 10, 2021. [Online]. Available: http://ndnsim.net/2.1/

[21] *NFD: Named Data Networking Forwarding Daemon*. Accessed: Dec. 10, 2021. [Online]. Available: https://named-data.net/doc/NFD/

[22] A. Hoque, S. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: Named-data link state routing protocol," in *Proc. ACM ICN Workshop*, Aug. 2013, pp. 15–20.

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

[24] X. Zhang, Y. Ou, S. Sen, and J. Jiang, "SENSEI: Aligning video streaming quality with dynamic user sensitivity," in *Proc. USENIX NSDI*, Apr. 2021, pp. 303–320.

[25] A. Rahman, D. Trossen, D. Kutscher, and R. Ravindran, *Deployment Considerations for Information-Centric Networking (ICN)*, Internet Res. Task Force, document RFC 8763, 2018. Accessed: Dec. 10, 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8763

[26] *Cefore*. Accessed: Dec. 10, 2021. [Online]. Available: https://cefore.net

[27] H. Asaeda, A. Ooka, K. Matsuzono, and R. Li, "Cefore: Software platform enabling content-centric networking and beyond," *IEICE Trans. Commun.*, vol. E102-B, no. 9, pp. 1792–1803, Sep. 2019.

[28] M. Mosko, I. Solis, and C. Wood, *Content-Centric Networking (CCNx) Semantics*, Internet Res. Task Force, document RFC 8569, 2019. Accessed: Dec. 10, 2021. [Online]. Available: https://tools.ietf.org/html/rfc8569

[29] M. Mosko, I. Solis, and C. Wood, *Content-Centric Networking (CCNx) Messages in TLV Format*, Internet Res. Task Force, document RFC 8609, 2019. Accessed: Dec. 10, 2021. [Online]. Available: https://tools.ietf.org/html/rfc8609

[30] R. Ravindran, A. Chakraborti, S. O. Amin, A. Azgin, and G. Wang, "5G-ICN: Delivering ICN services over 5G using network slicing," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 101–107, May 2017.

[31] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proc. ACM CoNEXT*, Dec. 2012, pp. 109–120.

[32] 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.

[33] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. 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.

[34] Y. Qin, S. Hao, K. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue, "ABR streaming of VBR-encoded videos: Characterization, challenges, and solutions," in *Proc. ACM CoNEXT*, Dec. 2018, pp. 366–378.

[35] K. Matsuzono and H. Asaeda, "NRTS: Content name-based real-time streaming," in *Proc. IEEE CCNC*, Jan. 2016, pp. 537–543.

[36] K. Matsuzono, H. Asaeda, and T. Turletti, "Low latency low loss streaming using in-network coding and caching," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[37] S. Lederer, C. Mueller, B. Rainer, C. Timmerer, and H. Hellwagner, "Adaptive streaming over content centric networks in mobile networks using multiple links," in *Proc. IEEE ICC Workshops*, Jun. 2013, pp. 677–681.

[38] W. Li, S. M. A. Oteafy, M. Fayed, and H. S. Hassanein, "Bitrate adaptation-aware cache partitioning for video streaming over information-centric networks," in *Proc. IEEE LCN*, Oct. 2018, pp. 401–408.

[39] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks," *IEEE Netw.*, vol. 28, no. 6, pp. 91–96, Nov. 2014.

YUSAKU HAYAMIZU (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in engineering from Kansai University, in 2014, 2016, and 2019, respectively. He is currently a Researcher with the National Institute of Information and Communications Technology (NICT). His research interests include computer networks, information centric networks, traffic control, in-network computing, and network softwarelization. He is a member of the ACM and the IEICE. He was a recipient of Best Paper Awards from the 2017 IEEE CQR Workshop and the 2018 IEEE LANMAN Symposium.

KOKI GOTO (Member, IEEE) received the B.E. and M.E. degrees in engineering from Kansai University, in 2018 and 2020, respectively. He is currently working at NTT DATA INTELLILINK Corporation. His research interests include computer networks, information centric networks, and video streaming. He is a member of the IEICE.

MASAKI BANDAI (Member, IEEE) received the B.E., M.E., and Ph.D. degrees from Keio University, in 1996, 1998, and 2004, respectively. From 2004 to 2010, he was an Assistant Professor with Shizuoka University. From 2006 to 2007, he was a Visiting Assistant Professor with The University of British Columbia. He is currently a Professor with Sophia University. His current research interests include computer networks, network computing, and network applications. He is a member of ACM, IEICE, and IPSJ.

MIKI YAMAMOTO (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in communications engineering from Osaka University, in 1983, 1985, and 1988, respectively. He joined the Department of Communications Engineering, Osaka University, in 1988. He moved to the Department of Electrical Engineering and Computer Science, Kansai University, in 2005, where he is currently a Professor. He visited the University of Massachusetts at Amherst, in 1995, and a Visiting Professor, in 1996. His research interests include content oriented networks, high speed networks, wireless networks, and the evaluation of performance of these systems. He is a member of ACM and IPSJ, and a fellow of IEICE. He has received the Best Paper Award of IEEE CQR Workshop 2017 and IEEE LANMAN Workshop 2018. He has served many technical and geographical activities in IEEE, including the Kansai Section Chair, an Executive Committee Member of Region 10, and the Technical Program Co-Chair of ICC 2019 CQRM Symposium.