IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 28, 2019, accepted January 20, 2020, date of publication January 27, 2020, date of current version February 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2969696

Design and Implementation of Bitrate Adaptation Schemes for Power Capping in Wi-Fi Video Streaming

GYUWHAN KIM[1], DAYOUNG LEE[2], AND MINSEOK SONG[2], (Member, IEEE)
[1]STEALIEN Inc., Seoul 04315, South Korea
[2]Department of Computer Engineering, Inha University, Incheon 22212, South Korea

Corresponding author: Minseok Song (mssong@inha.ac.kr)

This work was supported in part by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT, under Grant 2017M3C4A7080248, in part by the Basic Science Research Program through the NRF funded by the Ministry of Education under Grant NRF-2018R1D1A1B07050614, and in part by the Inha University Research Grant.

ABSTRACT In dynamically adaptive streaming over HTTP (DASH), which is the de facto standard for streaming, each video is divided into segments, and each segment is further transcoded into multiple bitrate versions. This allows a client device to select the most appropriate bitrate version that matches the network bandwidth to avoid jitters or stalls. However, Wi-Fi download of a high-bitrate version may consume significant energy, especially when network conditions are good. To address this, we propose a new streaming method that limits the energy consumed by mobile devices but maintains an acceptable video quality. First, we derive a power model to analyze how bitrate selection affects power consumption in smartphones. Based on this, we propose two algorithms that determine the bitrate of each segment with the aim of maximizing overall video quality while limiting energy consumption. We use dynamic programming and heuristics to address the tradeoff between algorithm complexity and video quality. The proposed scheme was implemented on an Android-based DASH streaming platform, and various issues were resolved to cope with varying network conditions. Experimental results demonstrated that our scheme effectively optimized the video quality while limiting the energy consumption. For example: 1) our scheme uses 4% and 10% less power than DASH while maintaining an excellent video quality, and 2) the average difference between estimated and actual power consumption is 0.8%, thus keeping a precise energy bound.

INDEX TERMS Video streaming, power capping, WiFi energy, bitrate selection.

I. INTRODUCTION

With recent advances in wireless networks, people can access video streaming services anytime anywhere using their smartphones. To adapt video streaming to varied wireless network conditions, dynamically adaptive streaming over HTTP (DASH) techniques have been widely adopted by major streaming companies, including Netflix, Hulu, and YouTube [1]. Each video is divided into segments, and each segment is transcoded into various bitrate versions. This allows each client device to choose the most appropriate bitrate version to match the network conditions, providing reliable streaming even with varied network conditions [1].

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

In a DASH environment, bitrate selection is crucial to provide reliable streaming over various network conditions [2]. Three methods of bitrate selection exist in the literature: throughput-based, buffer-based, and hybrid algorithms [2]. To determine the bitrate of the next segment, the throughput-based schemes estimate network throughput by monitoring network bandwidth [3], while the buffer-based schemes check the current buffer level [4]. The hybrid schemes use data on the current buffer level and network throughput to take advantage of both schemes [5].

These schemes aim to select the highest possible bitrate that matches the current network bandwidth. Therefore, if the network conditions are good, then the highest bitrate is usually selected, resulting in a high Wi-Fi power consumption to process a large number of packets [6]. However, people tend to miss slight differences in video quality when using

different bitrates, especially on mobile devices with small screens, and this difference also varies depending on the scene characteristics [7]. However, bitrate selection for DASH did not consider this issue.

For mobile devices, it is essential to optimize the energy consumption [6], [8]. For example, if the battery level is low, the remaining energy must be used efficiently. In addition, some users prefer low-power streaming, even if their quality of experience (QoE) degrades. To achieve this, we introduce the concept of energy budget to ensure that the energy consumption during the streaming session is as close as possible to this budget value. This approach requires an exact prediction of how the power consumption is related to the bitrate.

In video streaming, network interfaces consume a significant amount of power, and many methods exist to minimize the power consumption [9]–[14]. Devices typically have low-power modes, so most of the power-saving schemes attempt to extend the amount of time when the network device is turned off. For this purpose, these methods may involve prefetching operations and minimize unnecessary prefetching while minimizing rebuffering operations. However, despite many studies on matching bitrate selection to network conditions, to the best of our knowledge, there has been no previous study on limiting the Wi-Fi energy consumed by a mobile device while maintaining QoE.

We propose a new bitrate adaptation scheme for DASH-based streaming systems. First, we derive an energy consumption model according to the bitrate by considering the hardware characteristics. We formulate an optimization problem that aims at maximizing overall video quality while limiting the energy consumption. Next, we propose two bitrate selection algorithms that use dynamic programming and heuristic techniques to achieve a tradeoff between video quality and computational complexity. Finally, we report the implementation details of our scheme on real DASH systems.

The remainder of the paper is organized as follows. Section II reviews related work. Section III describes our approach and the power model. Section IV proposes new bitrate algorithms. Section V presents implementation details. Section VI shows the experimental results. Section VII concludes the paper.

II. RELATED WORK

Several video streaming techniques aim to reduce the power consumption of radio interfaces, such as long-term evolution (LTE), with three power states: idle, active, and tail. Because the idle state consumes almost zero power, most of power-saving techniques prefetch future video data to a buffer to extend the length of the idle time. For example, Li *et al.* [10] found that accurate prediction of video playback length is important to maximize the length of idle time (while avoiding unnecessary data download overhead) and presented a scheme that adjusts the download buffer size dynamically based on users' viewing history and network conditions. To accurately determine when to switch to idle mode to maximize the energy efficiency, Hu and Cao [9] presented a technique that determined whether users typically watched video for a long time or tended to skip, based on their usage patterns of the progress bar of the media player. Yang and Cao [15] generalized this prefetch-based energy optimization problem to find a prefetching schedule that minimizes the energy consumption of the data transmission for specific network conditions.

Because a Wi-Fi device cannot receive packets in the sleep mode, Wei *et al.* [16] presented a statistical prediction scheme to determine when the interface should go to the sleep mode. The IEEE 802.11e standard defines the power saving mode (PSM) for storing packets at the access point while the device is in sleep mode, for which it is important to calculate when the Wi-Fi device enters the sleep mode [14]. To address this, Csernai and Gulyas [17] presented a scheme that controlled sleep cycles of the Wi-Fi interface based on video quality by using the PSM. Zou *et al.* [12] compared Wi-Fi with LTE devices in terms of energy efficiency against video quality, number of clients, and so on, and found that Wi-Fi consumed less power than LTE. Bui *et al.* [13] combined Wi-Fi with LTE streaming to provide energy-efficient bandwidth aggregation while adapting to network fluctuations. However, none of these works have provided power management schemes related to bitrate selection, which is essential in a DASH environment.

Many studies on bitrate selection primarily focused on estimating the network bandwidth. The throughput-based schemes estimate the network throughput between clients and servers to determine the bitrate of the next segment [2]. For example, Jiang *et al.* [18] identified problems of the video bitrate adaptation over the HTTP, and proposed several schemes that address the tradeoffs between stability, fairness, and efficiency. Li *et al.* [3] proposed a bitrate selection method called ''probe and adapt'' (PANDA) to resolve the bitrate oscillation problem that occurs when multiple streaming clients share the network. Next, the buffer-based algorithms mainly use the buffer fullness level to determine the bitrate of the next segment. For example, Huang *et al.* [4] found that buffer capacity estimation was unnecessary in the steady streaming phase, even though simple capacity estimation based on past network throughput was necessary during the startup phase. Unlike conventional heuristics, Spiteri *et al.* [19] formulated bitrate adaptation as a utility maximization problem and presented an online algorithm called BOLA to minimize rebuffering and maximize video quality.

The hybrid schemes considered the throughput and buffer information to take advantage of the two schemes above [2]. For example, Beben *et al.* [5] proposed a scheme called ABMA+ that determined the bitrate of the next segment based on the predicted probability of video freezing, which can be derived from both download time and playout buffer map. Qin *et al.* [20] presented a bitrate adaptation technique for variable-bitrate videos using a control-theoretic information based on scene complexity, bandwidth variability, and

quality level. However, these studies did not consider energy efficiency.

Several works have dealt with energy issues in DASH systems. For example, Go *et al.* [21] presented a DASH system that considered both the state of the wireless network and the energy consumption of mobile devices. They have implemented schemes on an Android platform to allow video streaming over multiple wireless networks and provide seamless video streaming with low energy consumption and networking costs. Wei *et al.* [22] presented an energy-aware video streaming mechanism on an HTTP2 DASH system. They used a server-push mechanism in the HTTP/2 protocol to allow wireless interfaces to frequently enter low-power mode. Bezerra *et al.* [23] evaluated the tradeoff between QoE and energy consumption of the DASH system in terms of network technology, bandwidth, and video type. Petrangeli *et al.* [24] considered battery lifetime characteristics and presented an algorithm that optimized QoE and the battery consumption of a video streaming session. Yan and Chen [25] presented a new bitrate and display brightness adaptation platform to reduce LCD display power in a DASH system. For power-saving display adaptation, they produced additional server-side video versions, which were adaptively delivered to the clients to minimize the display power with a low distortion rate. However, to the best of our knowledge, none of the existing bitrate selection methods limited energy consumption of DASH-based mobile systems to maximize video quality.

III. BASIC IDEA AND POWER MODEL FOR WI-FI STREAMING

A. BASIC IDEA

We aim to maximize overall video quality while limiting the amount of the smartphone energy consumption. The basic idea of our scheme is as follows:

1) Power modeling: To limit energy consumption, it is essential to accurately model the power consumption of the smartphones. Many studies derived power consumption models for each hardware component of a smartphone, often with multiple variable regression [6], [8], [11], [13]. These regression models reflect the power consumption of the smartphone when device-specific parameters are derived from real power measurements. For example, power monitoring apps such as Appscope [6] and Powertutor [8] used real measurements to derive parameters of the power model. We also made real power measurements to extract device-specific regression parameters. Unlike other methods, our scheme establishes the relationship between bitrate and power to help the algorithms select the bitrate of each segment based on the power per the amount of bits streamed.
2) Problem formulation: Using power parameters, we formulate an optimization problem that maximizes overall video quality subject to energy limit. Structural similarity (SSIM) index, one of the most popular video quality metrics, is used to represent video quality, because it is known to provide an estimate of the exact perceptual quality of video by effectively reproducing the human visual system [26], [27]. Therefore, the SSIM index is used to express video quality.
3) Algorithm development: We propose two bitrate selection algorithms to solve the optimization problem using dynamic programming and heuristics to address the tradeoff between complexity and video quality.
4) Design and implementation: Implementing the proposed scheme on a real DASH platform requires addressing two problems: 1) coping with changing network conditions and 2) minimizing algorithmic complexity. We resolved these issues which will be discussed in Section V.

B. POWER MODEL FOR WI-FI STREAMING

The power consumption of each component can be expressed as a regression function of component utilization, where the parameters of the regression depend on the smartphone characteristics obtained from power measurements [6], [8], [11], [13], [28]. In smartphones, dedicated hardware decoders perform video decoding operations, which require a small amount of CPU power [29]. Therefore, Wi-Fi and display devices typically dominate the power consumption during Wi-Fi video streaming [11], [12]. We consider smartphones with LCD displays to model power consumption.

Because the backlight brightness primarily determines the power consumption of the LCD displays, the amount of their power consumption is directly proportional to their brightness [6], [8]. Additionally, the energy consumed by the Wi-Fi device is proportional to the size of the data transmitted [6], [8], [12], [13]. Because the bitrate of each video segment determines the amount of data transferred, we can now model the relationship between bitrate and power as a linear function of the bitrate [11], [13]. For example, let $P(k)$ be the estimated power consumption at bitrate k, which can be calculated by a linear relationship with the bitrate as follows:

$$P(k) = \alpha k + \beta + \gamma, \tag{1}$$

where α and β represent the parameters of Wi-Fi power consumption that depend on the bitrate of the video segments, and γ represents the power parameter for the LCD display that is proportional to brightness [6], [8]. These parameters are smartphone-specific, so they are typically derived from real power measurements [6], [8], [13].

To determine how the bitrate affects power consumption, we measured the amount of power with various bitrates for 32 different video clips. For the sample videos, the bit-per-pixel (bpp) was set to 0.15, and the bitrates were derived when five typical resolutions (240p, 360p, 480p, 720p, and 1080p) were used in a DASH. The average SSIM values for six of the 32 clips are shown in Figure 1. The SSIM values vary significantly with the video clip and the segment. For example, for the same resolution of 240p, the SSIM

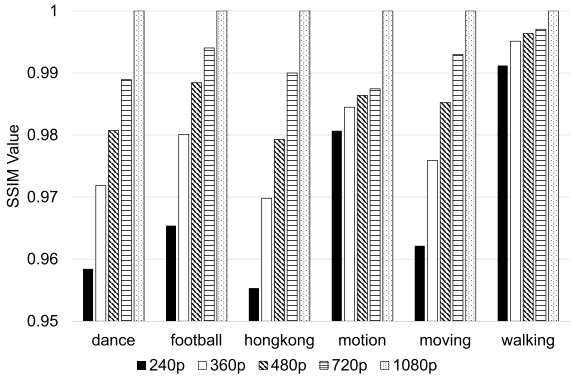


FIGURE 1. Average SSIM values for six of the 32 clips.

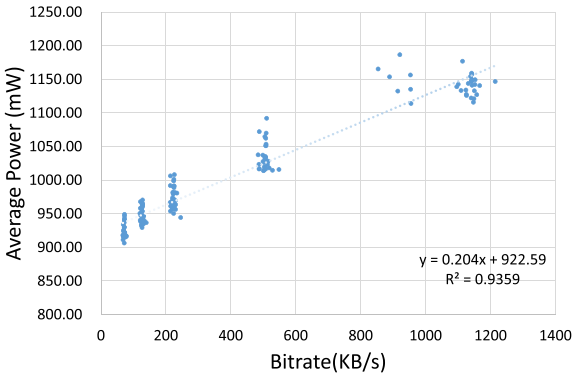


FIGURE 2. Power consumption against bitrate on an LCD smartphone.

value is 0.9553 in the "dance" video clip but 0.9912 in the "hongkong" video, as can be seen in Figure 1.

We measured the power of the LG V20 phone with a 25% display brightness, using a Monsoon power monitor [30] when 32 different video clips were streamed. We observed a linear regression relationship between the bitrate and the amount of power consumed, as shown in Figure 2, where a high correlation factor (say, R^2) of 0.9359 was found. For the V20 phone with a 25% display brightness, the smartphone-specific parameters for $P(k)$ is calculated as: $P(k) = 0.0204k + 922.59$, where the units for k and $P(k)$ are Bytes/s and mW, respectively.

IV. BITRATE SELECTION ALGORITHMS

A. BITRATE SELECTION PROBLEM

The objective of the bitrate selection algorithm is to maximize overall video quality while limiting energy consumption. Let N_{br} be the number of bitrate levels supported by the DASH players. Let N_{seg} be the number of segments. Let $Q_{i,j}$ be the quality index at a bitrate level j of segment i, $(i = 1, \ldots, N_{seg})$. We use SSIM values for $Q_{i,j}$. The SSIM values can be obtained by the comparison with the highest bitrate; thus, $Q_{i,N_{br}} = 1$. As can be seen in Figure 1, these SSIM values can significantly differ from one segment to another, and bitrate selection needs to reflect these quality variations.

Let T_i be the time length of segment i in seconds. Suppose that the jth bitrate level of segment i has bitrate $b_{i,j}$. Let $E_{i,j}$ be the energy consumed while the jth bitrate version of segment i is being streamed. It takes a power parameter $P(k)$ in Equation (1) as follows:

$$E_{i,j} = P(b_{i,j})T_i. \tag{2}$$

Let E_{limit} be the amount of energy that should be limited while the video clip is being streamed. The value of E_{limit} can vary over time. For example, if the smartphone is running low on battery, then E_{limit} value needs to be reduced to save energy, whereas some users may prefer low bit-rate video to reduce energy [9].

We then define an optimization problem, called the bitrate selection problem ($\mathcal{BSP}$), to find B_i, the bitrate of segment i, $(i = 1, \ldots, N_{seg})$, with the aim of maximizing the video quality while limiting the energy consumption by E_{limit} as follows:

$$\text{Maximize} \sum_{i=1}^{N_{seg}} Q_{i,B_i}$$

$$\text{Subject to} \sum_{i=1}^{N_{seg}} E_{i,B_i} \leq E_{limit}$$

$$B_i \in 1, \ldots, N_{br}$$

To find solutions to $\mathcal{BSP}$, we used two techniques: dynamic programming and greedy heuristic, each of which has its advantages and disadvantages. Although the dynamic programming scheme produces the optimal solution, it may not be suitable for a large number of segments due to high complexity. On the other hand, the greedy heuristic has polynomial time complexity so is scalable but may not be optimal. To examine this tradeoff, both methods are presented in the following subsections.

B. BITRATE SELECTION ALGORITHM BASED ON DYNAMIC PROGRAMMING (BSA-DP)

We use dynamic programming to find an optimal value of B_i. Let $Q_{i,m}^{max}$ be the maximum SSIM value when i is the index of the segment and m is the amount of energy consumed, $(i = 1, \ldots, N_{seg}$ and $m = 1, \ldots, E_{limit})$. Let $Q_{i,m}^{level}$ be the bitrate level of segment i to achieve the maximum SSIM value, $Q_{i,m}^{max}$.

The idea of the algorithm is to construct a table listing a pair of values, $(Q_{i,m}^{max}, Q_{i,m}^{level})$ for each segment i when the amount of energy used is m, $(i = 1, \ldots, N_{seg}$ and $m = 1, \ldots, E_{limit})$ using a recurrence relationship between consecutive segments, as shown in Figure 3. The last column in the table, $Q_{N_{seg},E_{limit}}$, represents the maximum SSIM value when the energy consumption is limited by E_{limit} and the number of segments is N_{seg}.

The details of the bitrate selection algorithm using dynamic programming (BSA-DP) are shown in Figure 4. The algorithm has three steps: 1) an initialization step that fills the first row of the table with the values of $(Q_{1,m}^{max}, Q_{1,m}^{level})$ (lines 1–10), 2) a recurrence establishment step that constructs the entire

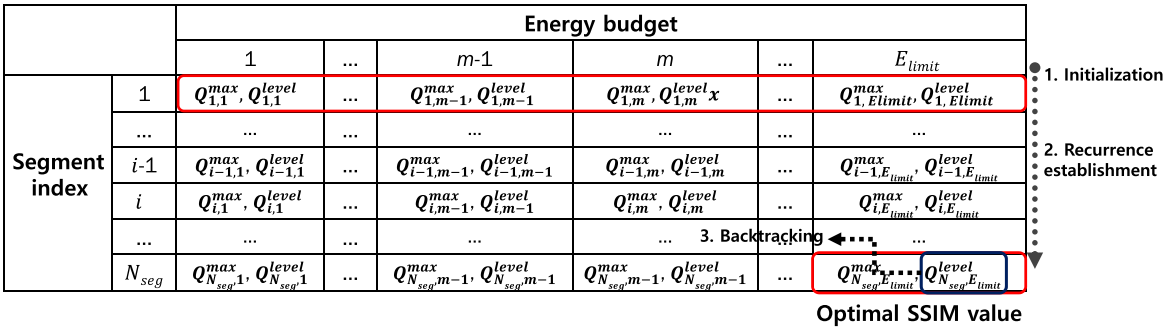


FIGURE 3. Table of $(Q^{max}_{i,m}, Q^{level}_{i,m})$ values for BSA-DP.

table with $(Q^{max}_{i,m}, Q^{level}_{i,m})$ values, $(i = 2, \ldots, N_{seg}$ and $m = 1, \ldots, E_{limit})$ (lines 11–23), and 3) a backtracking step that finds the value of B_i for each segment i (lines 24–30). These three steps can be summarized as follows:

1) Initialization: First, $Q^{max}_{i,m}$ are all initialized to 0 $(i = 1, \ldots, N_{seg}$ and $m = 1, \ldots, E_{limit})$. Next, $Q^{max}_{1,m}$ is initialized as follows:

$$Q^{max}_{1,m} = \max_{j \in \{j | E_{i,j} \le m\}} Q_{1,j}.$$

2) Recurrence establishment: $Q^{max}_{i,m}$ is updated using the following recurrence relationship:

$$Q^{max}_{i,m} = \max(Q^{max}_{i,m-1}, \max_{j=1,\ldots,N_{br}} (Q^{max}_{i-1,m-E_{i,j}} + Q_{i,j})).$$

Based on this recurrence, all the columns of the table in Figure 3 can be filled. The maximum value, $Q^{max}_{i,m}$ can be derived from one of the values of $Q^{max}_{i-1,m}$'s for the previous segment $i-1$, by adding $Q_{i,j}$ to $Q^{max}_{i-1,m-E_{i,j}}$ so that $Q^{max}_{i-1,m-E_{i,j}}, (j = 1, \ldots, N_{br})$. Obviously, the maximum value among the values of $Q^{max}_{i-1,m-E_{i,j}} + Q_{i,j}$ needs to be chosen for $Q^{max}_{i,m}$.

3) Backtracking: $Q_{N_{seg},E_{limit}}$ represents the optimal SSIM value. Starting from the last column in Figure 3 for segment N_{seg} and E_{limit}, the backtracking phase finds the bitrate level of segment i, B_i, by looking up the values of $Q^{level}_{i,m}$. For this purpose, the values of $Q^{level}_{i,m}$ are recorded whenever the values of $Q^{max}_{i,m}$ are updated in the recurrence establishment step (line 18). Suppose that B_i is chosen for segment i when the energy budget is m. Then, the energy budget for the previous segment $i-1$ can be set to $m - E_{i,B_i}$, because the amount of energy E_{i,B_i} is used for segment i to achieve $Q^{max}_{B_i,m}$. These operations are repeated by decreasing the value of i until the value of B_1 for the first segment is found (lines 26–30).

Figure 4 has three nested "for" loops of N_{seg}, E_{limit}, and N_{br}. Thus, time complexity of the BSA-DP can be expressed as $O(N_{seg}E_{limit}N_{br})$.

```
1: Temporary variables: i, m, and k;
2: for i = 1 to N_seg do
3:     for m = 1 to E_limit do
4:         Q^max_{i,m} ← 0;
5:     end for
6: end for
7: for m = E_{1,1} to E_limit do
8:     Q^max_{1,m} = max_{j∈{j|E_{1,j}≤m}} Q_{1,j};
9:     Q^level_{1,m} = arg max_{j∈{j|E_{1,j}≤m}} Q_{1,j};
10: end for
11: for i = 2 to N_seg do
12:     for m = 2 to E_limit do
13:         Q^max_{i,m} ← Q^max_{i,m-1};
14:         for j = 1 to N_br do
15:             if m − E_{i,j} ≥ 0 then
16:                 if Q^max_{i,m} < Q^max_{i-1,m-E_{i,j}} + Q_{i,j} then
17:                     Q^max_{i,m} ← Q^max_{i-1,m-E_{i,j}} + Q_{i,j};
18:                     Q^level_{i,m} ← j;
19:                 end if
20:             end if
21:         end for
22:     end for
23: end for
24: i ← N_seg;
25: m ← E_limit;
26: while i > 0 do
27:     B_i ← Q^level_{i,m};
28:     i ← i − 1;
29:     m ← m − E_{i,B_i};
30: end while
```

FIGURE 4. Bitrate Selection Algorithm using Dynamic Programming (BSA-DP).

C. BITRATE SELECTION ALGORITHM BASED ON HEURISTICS (BSA-H)

Although BSA-DP provides an optimal solution, it is not scalable because of its time and space complexity, so it may not be executed on mobile devices with limited computational capability. Therefore, we present a bitrate selection algorithm

1: Temporary variable: $E_{temp} \leftarrow \sum_{i=1}^{N_{seg}} E_{i,1}$;
2: Temporary variable: $Q_{temp} \leftarrow \sum_{i=1}^{N_{seg}} Q_{i,1}$;
3: Temporary variables: $V_i, \forall\, i\ V_i \leftarrow 1, (i = 1, ..., N_{seg})$;
4: **while** $E_{temp} < E_{limit}$ **do**
5: Choose the highest value of $I_{i,H}$ from the set $I_{i,j}$'s and removes $I_{i,H}$ from the set;
6: **if** $Q_{temp} - Q_{i,V_i} + Q_{i,H} > Q_{temp}$ **then**
7: $Q_{temp} \leftarrow Q_{temp} - Q_{i,V_i} + Q_{i,H}$;
8: $E_{temp} \leftarrow E_{temp} - E_{i,V_i} + E_{i,H}$;
9: $V_i \leftarrow H$;
10: **end if**
11: **if** $A = \phi$ **then**
12: Break the loop;
13: **end if**
14: **end while**
15: **for** $i = 1$ to N_{seg} **do**
16: $B_i \leftarrow V_i$;
17: **end for**

FIGURE 5. Bitrate Selection Algorithm using Heuristic (BSA-H).

based on heuristic (BSA-H), as shown in Figure 5. The algorithm has two steps: initialization and bitrate search.

1) Initialization step: We introduce temporary variables, V_i for each segment i, $(i = 1, \ldots, N_{seg})$ to store bitrate index during algorithm execution. All values of V_i are initialized to 1, corresponding to the lowest video quality and power. Next, these values are increased to maximize quality while minimizing an increase in energy consumption. For this purpose, we define parameters, $I_{i,j}$ for each bitrate j $(j = 2, \ldots, N_{br})$ of each segment i, as follows:

$$I_{i,j} = \frac{Q_{i,j} - Q_{i,1}}{E_{i,j} - E_{i,1}}. \tag{3}$$

2) Bitrate search step: At this step, the values of V_i are increased to improve video quality until energy usage is limited by E_{limit}. The numerator and denominator of $I_{i,j}$ represent the increase in video quality and energy, respectively, in comparison with $V_i = 1$. Therefore, to maximize video quality with little energy increase, the highest value of $I_{i,H}$ in the set of $I_{i,j}$'s should be chosen to increase the value of V_i. This operation is then repeated until $\sum_{i=1}^{N_{seg}} \sum_{j=1}^{N_{br}} E_{i,j} \leq E_{limit}$.

D. EXTENSION OF THE BSA-H TO LIMIT THE NUMBER OF BITRATE CHANGES

Frequent bitrate changes can affect QoE. To address this, the BSA-H in Figure 5 can be extended in each step to limit the number of bitrate changes by L as follows:

1) Initialization step: We introduce the L_{temp} variable to store the number of bit-rate changes during algorithm execution. In the initialization step, the values of V_i are all initialized to the same value 1 (the lowest bit-rate), so $L_{temp} = 0$.

2) Bitrate search step: In the bitrate search step (lines 4–14 in Figure 5), the BSA-H increases the bit-rate of each segment while the energy constraint is satisfied. However, to limit the number of bitrate changes by L, additional condition, $L_{temp} \leq L$ needs to be added to the original condition, which is $Q_{temp} - Q_{i,V_i} + Q_{i,H}$ in the "if statement" in line 5 in Figure 5. If both conditions can be met, then the value of L_{temp} is increased by 1 to update the number of bit-rate changes. This additional condition guarantees that the number of bitrate changes is limited to L.

V. IMPLEMENTATION

A. IMPLEMENTATION DETAILS

We implemented our approach both on server-side and client-side. We have developed a DASH server on the Node.js platform [31] using an express module to access storage devices that store various bitrate versions and media presentation description (MPD) files. To develop the client part, we have modified an Android DASH media player called ExoPlayer [32].

To use our bitrate selection algorithms for DASH, we resolved the following issues:

1) Coping with network bandwidth variation: The objective of BSA-DP and BSA-H is to select a bitrate value between 1 and N_{br} for every segment. The client device also monitors the network bandwidth to find the highest bitrate that matches the current network condition, which is defined by B_i^{bw} for segment i. If B_i^{bw} is higher than B_i, which is the selected bitrate by the algorithms, then B_i is chosen for segment i to limit energy. However, if the network conditions are poor, so that $B_i^{bw} < B_i$, then B_i^{bw} is chosen for segment i to match bandwidth, which under-utilizes the energy budget. In this case, bitrate selection algorithms can be executed again to find the bitrate for future segments to optimally use the remaining energy budget. Suppose that the algorithm is executed after segment cur is streamed. The remaining energy budget E_{remain} is then calculated as follows: $E_{remain} = E_{limit} - \sum_{i=1}^{i=cur} E_{i,B_i}$. Next, E_{remain} is used as a limit value in the algorithm to find the values of B_i for the remaining segments.
2) Energy limit setting: We added a new user interface to the DASH media player to allow users to control the energy budget by a percentage value compared with the highest bitrate streaming, as shown in Figure 6.

We have implemented two options to run a bitrate selection algorithm: server-side and client-side. Each option has its advantages and disadvantages. Server-side application reduces computational overhead on the mobile device, but it requires the device to receive the bitrate selection results from the server. However, client-side application may conform to the typical DASH guideline that selects the bitrate solely on the client side, but this can cause algorithm processing

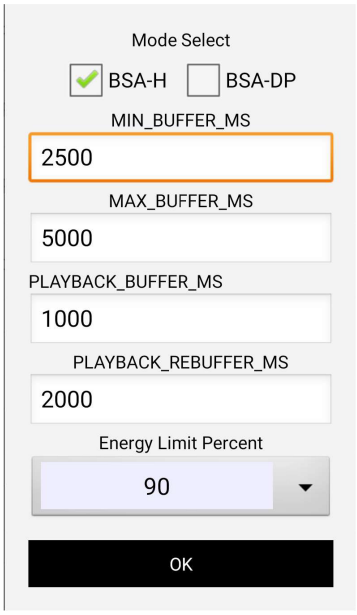


FIGURE 6. User interface for setting energy limit on an Android smartphone.

overhead on the mobile device. The details of the implementation can be described as follows:

1) Server-side execution: A DASH server has a file that contains information about each video clip, such as SSIM values, and the regression relationship between the bitrate and power for each smartphone. It first receives the value of E_{limit} from the client device, runs the algorithm to find the bitrate level of future segments, and delivers these values to the client device. This step needs to be repeated if a lower bitrate is selected than that selected by the algorithm. This approach maintains the SSIM value on the server, so there is no need to send the SSIM file to the client.
2) Client-side execution: When a DASH client requests a video clip, it receives the SSIM and MPD files from the server. It then runs a bitrate selection algorithm to construct a bitrate table for all the segments. Whenever each segment is streamed, it monitors the current network conditions to run the algorithm for the remaining segments, if necessary. Although a list of SSIM values needs to be sent to the client device in this case, only one transfer of the SSIM file is required before starting video streaming, which requires very little overhead as with the previous approach of constructing a simple SSIM XML file [26].

B. REDUCING ALGORITHM OVERHEAD

To choose the optimal bitrate for the current network conditions, the bitrate selection algorithms need to be executed again when $B_i^{bw} < B_i$, which can be time-consuming. Here, we describe how to handle this issue in our implementation.

Because BSA-DP always requires table construction, as shown in Figure 3, based on a recurrence relationship between the consecutive segments, the algorithm in Figure 4 needs to be re-executed for the remaining segments whenever $B_i^{bw} < B_i$. However, the complexity of BSA-H can be significantly reduced if it can eliminate the sorting operation of $I_{i,j}$ values, which causes the most overhead in the algorithm. Based on this, the BSA-H in Figure 5 can be modified to reduce time complexity as follows:

1) Before the first segment is streamed, BSA-H sorts the values of $I_{i,j}$ and stores them as a sorted list. This requires a sorting overhead, $O(nlogn)$, where n is the number of $I_{i,j}$ values and a quick sorting algorithm is used. It then runs the algorithm in Figure 5 and stores the bitrate of each segment as an array.
2) If $B_i^{bw} \geq B_i$, then no change is required for the array of the bitrate per segment, which requires no additional calculation.
3) If $B_i^{bw} < B_i$, so that the bitrate of future segments may be increased, the algorithm finds the bitrate values of future segments from an ordered list of $I_{i,j}$ values constructed when the first segment was streamed.

VI. EXPERIMENTAL RESULTS

We validated our scheme when six video clips of 90 seconds encoded in MPEG-4 were streamed; "motion", "dance", "moving", "hongkong", "walking" and "football" in Figure 1. The focus of the experiment was to examine 1) algorithm efficiency, 2) video quality and power under various network conditions, and 3) whether the proposed scheme optimally used the energy budget. The 802.11a/b/g/n/ac protocol was used for Wi-Fi configuration. To model the energy limit, we introduce variable E_{ratio}, which represents the percentage of energy usage compared with when the highest bitrate is always chosen.

A. ALGORITHM COMPARISON BETWEEN BSA-DP AND BSA-H

First, we calculated how the SSIM values varied with the values of E_{ratio} for two algorithms, as shown in Table 1. No SSIM difference was observed between them when $E_{ratio} = 85\%$, and we skipped these values from the table. The SSIM difference between two algorithms can be bounded by 0.00033, which is almost negligible. This clearly shows that BSA-H provides a near-optimal solution to $\mathcal{BSP}$, as BSA-DP produces an optimal solution.

To compare BSA-DP with BSA-H in terms of time complexity in a streaming scenario, we measured the time taken to run the algorithms on the LG V20 smartphone by inserting the time calculation function into our DASH client app on the Android system. Because the number of algorithm executions is affected by the network conditions, we consider four network scenarios when the bandwidth is selected randomly in the range in Table 2.

TABLE 1. Comparison of SSIM values between BSA-H and BSA-DP.

	Motion		Dance		Moving		Hongkong		Walking		Football	
E_{ratio}	90%	95%	90%	95%	90%	95%	90%	95%	90%	95%	90%	95%
BSA-DP	0.99672	0.99706	0.99001	0.99219	0.99279	0.99394	0.99014	0.99183	0.99739	0.99809	0.99478	0.99598
BSA-H	0.99670	0.99706	0.99000	0.99164	0.99276	0.99373	0.99013	0.99183	0.99716	0.99804	0.99477	0.99565
Difference	0.00002	0	0.00001	0.00005	0.00003	0.00021	0.00001	0	0.00023	0.00005	0.00001	0.00033

TABLE 2. Bandwidth setting for various network conditions.

Network conditions	Bandwidth range
Poor	200–500 kbps
Medium	500–3000 kbps
Good	3000–5000 kbps
Random	200–5000 kbps

Table 3 shows the total algorithm execution time during a 90-second streaming session. BSA-DP requires significantly more time than BSA-H, because BSA-DP always requires a table construction to run the algorithm, which increases with the value of energy limit. When the network conditions are poor or medium, more algorithm executions are required, which increases the total execution time. From these results on time complexity and video quality, we use BSA-H for the rest of the experiment, because it produced excellent video quality with a low computational overhead.

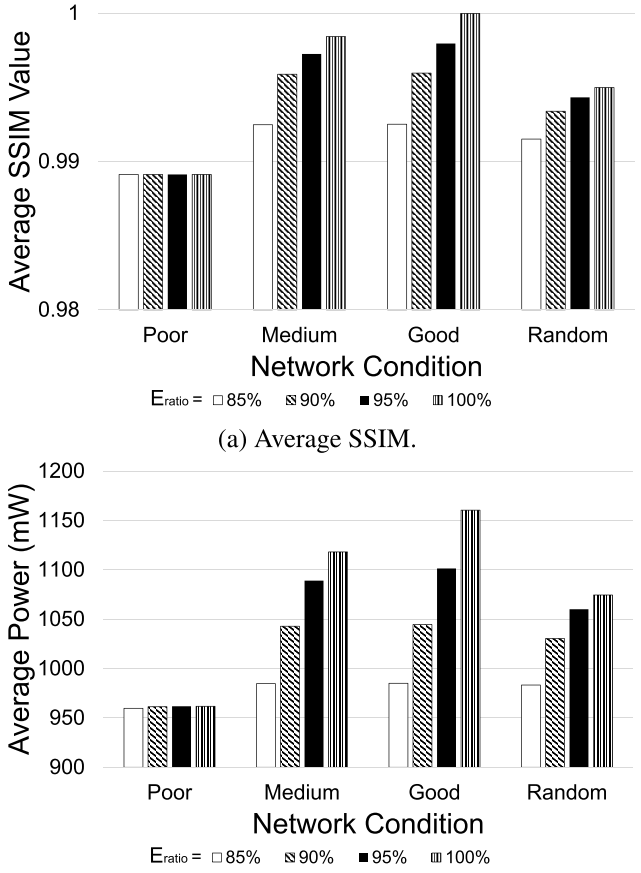
B. EFFECT OF VARIOUS NETWORK CONDITIONS

Next, we examined the effect of various network conditions on power and video quality. WiFi network conditions are affected by various factors, and according to actual measurement studies [35], network conditions are classified into: 1) when the highest bitrate is always supported, 2) when the highest bitrate can be sometimes supported, and 3) the highest bitrate is never be selected. Based on these measurement results, we generated three WiFi network conditions (poor, medium and good) synthetically to examine the effect of various network conditions as tabulated in Table 2.

Figure 7 shows the average SSIM values and the amount of average power consumption for six video clips tested for different values of E_{ratio}. The power values in Figure 7 (b) are derived based on smartphone-specific power parameters from actual power measurement results in Figure 2.

All algorithms used in DASH choose the highest bitrate that matches the current network conditions estimated by various network bandwidth prediction methods [2]–[5]. We compared the BSA-H with this DASH scheme that selected the highest bitrate for segment to match the network condition; for example, in Figure 7, DASH corresponds the the case when $E_{ratio} = 100\%$, because this allows the highest bitrate to be selected to match the network condition. From Figure 7, we make the following observations:

1) Obviously, the average SSIM increases with E_{ratio}. For example, the percentage difference between $E_{ratio} = 85\%$ and DASH is between 0.8% and 2%. The SSIM difference decreases as the network bandwidth decreases, because low bitrate can be selected to support low network bandwidth even when $E_{ratio} = 100\%$ for the poor network conditions.
2) The power difference between $E_{ratio} = 85\%$ and 100% is between 0.2% and 15%. This difference is the highest when the network conditions are good, because the highest bitrate is usually chosen for DASH, and it is fairly high (12%) even when the network conditions are medium.
3) It was reported that an SSIM value of 0.99 corresponds to the highest mean opinion score of 5, which indicates an excellent video quality [33]. The SSIM results are almost identical under poor network conditions regardless of the E_{ratio} = values. However, for other network conditions, when $E_{ratio} \geq 90\%$, all SSIM values exceed 0.99, showing an excellent video quality, but 4% and 10% less power is used than with DASH.



(a) Average SSIM.

(b) Average power consumption.

FIGURE 7. SSIM and power comparison against E_{ratio}.

TABLE 3. Total algorithm execution time during an entire streaming session.

	Poor			Medium			Good			Random		
E_{ratio}	85%	90%	95%	85%	90%	95%	85%	90%	95%	85%	90%	95%
BSA-DP	7312ms	7747ms	8508ms	5737ms	6013ms	8644ms	805ms	863ms	884ms	3532ms	4148ms	63145ms
BSA-H	17ms	12ms	14ms	16ms	14ms	14ms	1.2ms	1ms	1.2ms	11ms	7ms	8ms

TABLE 4. Average, highest, and lowest values of R.

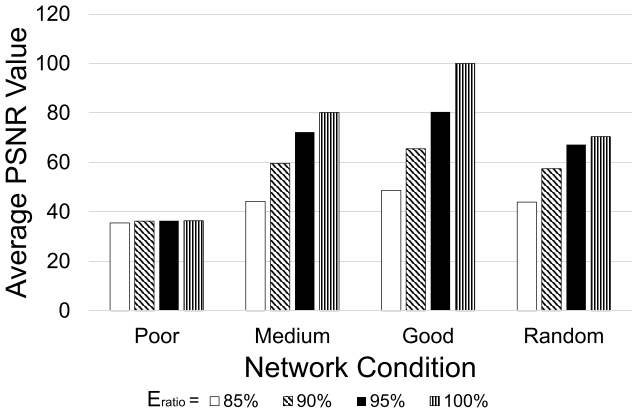
	Motion			Dance			Moving			Hongkong			Walking			Football		
E_{ratio}	85%	90%	95%	85%	90%	95%	85%	90%	95%	85%	90%	95%	85%	90%	95%	85%	90%	95%
R_{avg}	1.004	1.000	0.990	0.999	1.008	1.017	1.009	1.007	1.022	0.996	1.001	1.001	0.996	1.009	1.022	1.003	1.005	1.020
R_{high}	1.009	1.005	0.995	1.006	1.011	1.022	1.014	1.010	1.023	1.000	1.013	1.006	1.000	1.015	1.028	1.010	1.007	1.027
R_{low}	0.996	0.994	0.986	0.993	1.007	1.010	1.003	1.004	1.019	0.991	0.991	0.996	0.993	1.003	1.018	0.995	1.002	1.013

This demonstrates the effectiveness of our scheme: it uses a small amount of energy but maintains an excellent video quality.

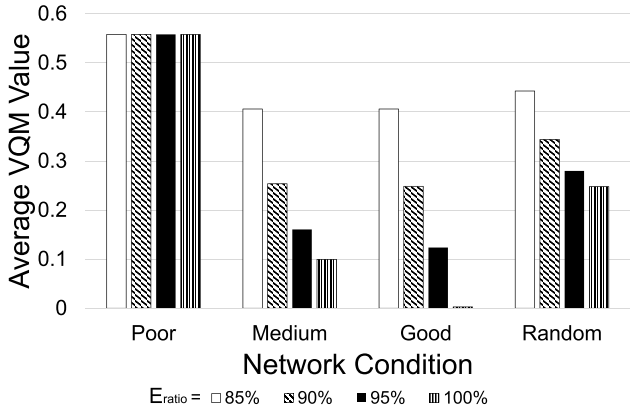
C. EFFECT OF VARIOUS VIDEO QUALITY METRICS

In this subsection, we used different video quality metrics such as PSNR and VQM to see how energy limits affect video quality. The VQM values range from 0 and 5, where 0 is the best quality and 5 is the worst quality [34]. Figure 8 shows average PSNR and VQM values for the different values of E_{ratio}. From the figure, the following observations can be drawn:

1) As expected, video quality decreases as energy budgets decrease regardless of video quality metrics.
2) It was reported that the PSNR value above 37 corresponds to the highest mean opinion score of 5 [34]. Therefore, except in poor network conditions, the proposed method shows the best video quality in terms of PSNR index regardless of the energy limit value.
3) Regardless of the video quality metrics, the results are similar to the SSIM case. For example, in poor network conditions, the video quality is almost the same regardless of the energy limit value. In good network conditions, the video quality depends on the energy limit value, but still exhibits excellent video quality even at low energy limits.



(a) Average PSNR.



(b) Average VQM.

FIGURE 8. PSNR and VQM comparison against E_{ratio}.

D. REAL POWER MEASUREMENT RESULTS

To examine whether our scheme effectively used the energy budget, we measured power three times using a Monsoon power monitor [30] on an LG V20 smartphone, and we calculated the real energy consumption relative to the energy budget, R, as follows: $R = \frac{E_{real}}{E_{limit}}$, where E_{real} is the real energy usage. In this power measurement scenario, the network conditions are very good, so the highest bitrate is always selected for DASH, indicating that $E_{ratio} = 100\%$.

If $R > 1$, then the real energy usage exceeds the energy budget; otherwise, it keeps the energy usage below the budget. Table 4 shows the average, highest, and lowest values of R ($R_{avg}, R_{high}, R_{low}$), which shows that the average value of $|R_{avg}|$ for all videos was 0.8%, suggesting that our scheme optimally uses the allocated energy budget. The maximum difference ($\max|1 - R|$) was also bounded by 2.7%, which represents a negligible variation in our scheme. Therefore, our scheme, in contrast to DASH, can use less energy and limit energy usage, as specified by the E_{limit} value.

VII. CONCLUSION

We proposed a new DASH-based streaming platform that limits the amount of energy consumed by mobile devices while maintaining a high quality of Wi-Fi video streaming. First, we built a power model that set a regression relationship

between bitrate and power based on the extracted device-specific parameters. We have used this power model to formulate an optimization problem that aimed to maximize overall video quality while limiting energy consumption. To address the tradeoff between algorithm complexity and video quality, we developed two bitrate selection algorithms: 1) BSA-DP that used dynamic programming to produce an optimal solution and 2) BSA-H that used a greedy heuristic to reduce algorithm complexity. For the practical use of our scheme, we have solved several issues, such as addressing various network conditions and reducing the overhead of algorithms. Finally, we provided our implementation details for DASH servers and Android smartphones.

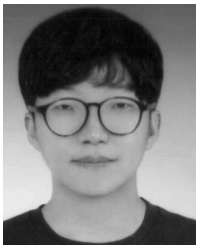
The experimental results show that the proposed scheme can reduce overall smartphone energy consumption from 4% to 10% (compared with standard DASH) while maintaining an excellent video quality. We also demonstrated that (1) BSA-H provides a near-optimal solution to the optimization problem at a small overhead, thus demonstrating usability in real-world streaming compared with BSA-DP, and (2) our scheme is accurate in terms of power limitations, and it shows that power difference between actual power usage and energy limits is 0.8% on average.

We are currently extending our scheme to optimally limit the energy usage of the DASH client devices with OLED displays that require additional parameters such as color information for power modeling. As future work, we consider the use of machine learning for power modeling in DASH streaming environments.

REFERENCES

[1] L. Toni, R. Aparicio-Pardo, G. Simon, A. Blanc, and P. Frossard, "Optimal set of video representations in adaptive streaming," in *Proc. 5th ACM Multimedia Syst. Conf. (MMSys)*, 2014, pp. 271–282.
[2] K. Spiteri, R. Sitaraman, and D. Sparacio, "From theory to practice: Improving bitrate adaptation in the DASH reference player," in *Proc. 9th ACM Multimedia Syst. Conf.*, Jun. 2018. pp. 123–137.
[3] 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.
[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," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 187–198, 2015.
[5] A. Beben, P. Wisniewski, J. M. Batalla, and P. Krawiec, "ABMA+: Lightweight and efficient algorithm for HTTP adaptive streaming," in *Proc. 7th Int. Conf. Multimedia Syst.*, vol. 2, May 2016, pp. –11.
[6] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "Appscope: Application energy metering framework for Android smartphones using kernel activity monitoring," in *Proc. USENIX Conf. Annu. Tech. Conf.*, Jun. 2012, p. 36.
[7] C. Wu, W. Zhu, Q. Li, and Y. Zhang, "Rethinking HTTP Adaptive Streaming with the Mobile User Perception," in *Proc. ACM Multimedia Conf. (MM)*, Oct. 2017, pp. 925–933.
[8] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate Online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardware/Softw. Codesign Syst. Synth. (CODES/ISSS)*, 2010, pp. 105–114.
[9] W. Hu and G. Cao, "Energy-aware video streaming on smartphones," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 1185–1193.
[10] X. Li, M. Dong, Z. Ma, and F. C. Fernandes, "GreenTube: Power optimization for mobile videostreaming via dynamic cache management," in *Proc. 20th ACM Int. Conf. Multimedia*, Oct. 2012, pp. 279–288.
[11] J. Zhang, G. Fang, C. Peng, M. Guo, S. Wei, and V. Swaminathan, "Profiling energy consumption of DASH video streaming over 4G LTE networks," in *Proc. 8th Int. Workshop Mobile Video*, May 2016, pp. 1–11.
[12] L. Zou, A. Javed, and G.-M. Muntean, "Smart mobile device power consumption measurement for video streaming in wireless environments: WiFi vs. LTE," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2017, pp. 1–6.
[13] D. H. Bui, K. Lee, S. Oh, I. Shin, H. Shin, H. Woo, and D. Ban, "GreenBag: Energy-efficient bandwidth aggregation for real-time streaming in heterogeneous mobile wireless networks," in *Proc. IEEE 34th Real-Time Syst. Symp.*, Dec. 2013, pp. 57–67.
[14] M. Kennedy, A. Ksentini, Y. Hadjadj-Aoul, and G.-M. Muntean, "Adaptive energy optimization in multimedia-centric wireless devices: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 768–786, Aug. 2013.
[15] Y. Yang and G. Cao, "Prefetch-based energy optimization on smartphones," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 693–706, Jan. 2018.
[16] Y. Wei, S. Bhandarkar, and S. Chandra, "A client-side statistical prediction scheme for energy aware multimedia data streaming," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 866–874, Aug. 2006.
[17] M. Csernai and A. Gulyas "Wireless adapter sleep scheduling based on video QoE: How to improve battery life when watching streaming sideo?" in *Proc. IEEE Conf. Comput. Commun. Netw.*, Jul. 2011, pp. 57–67.
[18] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol.*, 2012, pp. 97–108.
[19] 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.*, Apr. 2016, pp. 1–9.
[20] Y. Qin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue, "ABR streaming of VBR-encoded videos: Characterization, challenges, and solutions," in *Proc. 14th Int. Conf. Emerg. Netw. Exp. Technol.*, 2018, pp. 97–108.
[21] Y. Go, O. C. Kwon, and H. Song, "An energy-efficient HTTP adaptive video streaming with networking cost constraint over heterogeneous wireless networks," *IEEE Trans. Multimedia*, vol. 17, no. 9, pp. 1646–1657, Sep. 2015.
[22] S. Wei, V. Swaminathan, and M. Xiao, "Power efficient mobile video streaming using HTTP/2 server push," in *Proc. IEEE 17th Int. Workshop Multimedia Signal Process. (MMSP)*, Oct. 2015, pp. 1–6.
[23] C. Bezerra, A. D. Carvalho, D. Borges, N. Barbosa, J. Pontes, and E. Tavares, "QoE and energy consumption evaluation of adaptive video streaming on mobile device," in *Proc. 14th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2017, pp. 1–6.
[24] S. Petrangeli, P. Van Staey, M. Claeys, T. Wauters, and F. De Turck, "Energy-aware quality adaptation for mobile video streaming," in *Proc. 12th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2016, pp. 1–6.
[25] Z. Yan and C. W. Chen, "RnB: Rate and brightness adaptation for rate-distortion-energy tradeoff in HTTP adaptive streaming over mobile devices," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw.*, 2016, pp. 308–319.
[26] A. Bentaleb, A. C. Begen, and R. Zimmermann, "SDNDASH: Improving QoE of HTTP Adaptive Streaming Using Software Defined Networking," in *Proc. ACM Multimedia Conf.*, Oct. 2016, pp. 1296–1305.
[27] T.-S. Ou, Y.-H. Huang, and H. H. Chen, "SSIM-based perceptual rate control for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 682–691, May 2011.
[28] W. Jung, C. Kang, C. Yoon, D. Kim, and H. Cha, "DevScope: A nonintrusive and Online power analysis tool for smartphone hardware components," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, 2012, pp. 353–362.
[29] Y. G. Kim, M. Kim, and S. W. Chung, "Enhancing energy efficiency of multimedia applications in heterogeneous mobile multi-core processors," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1878–1889, Nov. 2017.
[30] *Power Monitor*. Accessed: Jan. 28, 2020. [Online]. Available: http://msoon.github.io/powermonitor/
[31] *Node.js*. Accessed: Jan. 28, 2020. [Online]. Available: https://nodejs.org/en/
[32] *Google ExoPlayer*. Accessed: Jan. 28, 2020. [Online]. Available: https://google.github.io/ExoPlayer/
[33] T. Zinner, O. Hohlfeld, O. Abboud, and T. Hossfeld, "Impact of frame rate and resolution on objective QoE metrics," in *Proc. 2nd Int. Workshop Quality Multimedia Exp. (QoMEX)*, Jun. 2010, pp. 29–34.

[34] R. Gomes, W. Junior, E. Cerqueira, and A. Abelem, "A QoE fuzzy routing protocol for wireless mesh networks," in *Proc. Int. Workshop Future Multimedia Netw.*, Jun. 2010, pp. 1–12.
[35] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "MP-DASH: Adaptive video streaming over preference-aware multipath," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2016, pp. 129–143.



GYUWHAN KIM received the B.S. and M.S. degrees in computer science and engineering from Inha University, South Korea, in 2017, and 2019, respectively. He is currently working with STEALIEN Inc. His research interests include embedded software and ubiquitous computing.

DAYOUNG LEE received the B.S. and M.S. degrees in computer science and engineering from Inha University, South Korea, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree. His research interests include embedded software and ubiquitous computing.

MINSEOK SONG (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer engineering from Seoul National University, South Korea, in 1996, 1998, and 2004, respectively. Since 2005, he has been with the Department of Computer Engineering, Inha University, Incheon, South Korea, where he is currently a Professor. His research interests include embedded software and multimedia systems.