Received 17 August 2023, accepted 19 September 2023, date of publication 26 September 2023, date of current version 2 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3319365

RESEARCH ARTICLE

Energy-Efficient HTTP Adaptive Streaming System Over SDN-Enabled Wi-Fi APs

HYUNMIN NOH[1], GI SEOK PARK[2], YUNMIN GO[3], SANG-HEON SHIN[4], YOUNGCHAN JANG[4], AND HWANGJUN SONG[5]

[1]Division of Electronic Engineering, Jeonbuk National University, Jeonju 54896, South Korea
[2]Department of Computer Science and Engineering, Incheon National University, Incheon 22012, South Korea
[3]School of Computer Science and Electrical Engineering, Handong Global University, Pohang 37554, South Korea
[4]Intelligent C4I Team, Hanwha Systems, Seongnam 13524, South Korea
[5]Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang 37673, South Korea

Corresponding author: Hwangjun Song (hwangjun@postech.ac.kr)

This work was supported in part by the Korea Research Institute for Defense Technology Planning and Advancement (KRIT) Grant funded by the Korean Government, Defense Acquisition Program Administration (DAPA), under Grant KRIT-CT-21-040; and in part by the GEOINT Cloud-Based Real World Realization Technology, in 2023.

ABSTRACT This paper presents an energy-efficient Hypertext Transfer Protocol (HTTP) adaptive streaming system to maximize the overall quality of video streaming services for all clients in an energy-efficient way over software-defined networking (SDN)-enabled Wi-Fi access points (APs). To achieve this goal, the proposed system employs multi-path technology to overcome the limitations of a single Wi-Fi AP and adopts the Luby transform (LT) code as forward error correction to support flexible and reliable data transmission via multiple APs. Furthermore, the SDN controller manages the segment bitrate, code rate of the LT code, and video packet transmission via multiple APs based on a global view of the network status, buffer occupancy, and energy consumption. The proposed system is implemented by using network simulator 3 (NS-3) to verify on a large-scale simulation environment and fully implemented in a real testbed to demonstrate its feasibility. The experimental results show that the proposed system can provide superior performance with lower energy consumption than any other existing system.

INDEX TERMS Adaptive streaming, software-defined network, multi-path, bitrate adaptation, segment scheduling, fountain code.

I. INTRODUCTION

Global mobile traffic has been growing rapidly due to the popularity of video streaming services such as YouTube [1], Netflix [2], and Disney+ [3]. According to the International Telecommunication Union (ITU) report [4], overall mobile traffic is expected to grow at a 55% annual rate from 2020 to 2030. According to Ericsson [5], video data traffic will account for 77 % of all mobile data traffic by 2026. Wi-Fi is one of the key solutions to meet the growing demand for wireless network resources. According to the Cisco Visual Network Index [6], there will be 628 million public Wi-Fi hotspots available worldwide by 2023. However, it is still difficult to provide high-quality and smooth video streaming services due to the limited and time-varying wireless network conditions.

Recently, HTTP adaptive streaming (HAS) has been widely used to provide seamless video streaming services over time-varying wireless networks. In 2012, the Moving Picture Experts Group (MPEG) standardized HAS as dynamic adaptive streaming over HTTP (DASH) [7]. In DASH, a video is encoded at multiple bitrates and divided into smaller segments in a time unit. The media presentation description (MPD) includes a list of encoded bitrates and uniform resource locators (URLs) of the segments. The video segments and MPD are stored at the DASH server. The DASH client requests an MPD file to the DASH server in order to initiate video streaming. After receiving the MPD file, the DASH client parses the MPD file to obtain information about the selectable segments, and then continuously determines

The associate editor coordinating the review of this manuscript and approving it for publication was P. Venkata Krishna.

the bitrate of the next segment by considering the current network condition and playback buffer time.

So far, a considerable amount of research efforts has been devoted to seamless video streaming over time-varying networks. Spiteri et al. [8] employed Lyapunov optimization to segment bitrate adaptation to ensure seamless video streaming. Liu et al. [9] defined a smoothed HTTP throughput based on the segment fetch time to detect bandwidth fluctuations using only application layer information. Additionally, they proposed a novel rate adaptation algorithm based on the smoothed HTTP throughput to ensure stable video streaming. Jiang et al. [10] proposed FESTIVE, which performs rate adaptation while considering the tradeoffs among fairness, efficiency, and stability on the bottleneck link. Li et al. [11] proposed PANDA that utilizes the Transmission Control Protocol (TCP) congestion control to maintain a stable playback buffer state. However, in these studies, quality degradation may still occur when multiple DASH clients compete for a shared bottleneck link. Generally, each DASH client independently determines and requests the next segment by considering only the local network condition estimated by itself. This process may cause several issues, such as instability, unfairness, and underutilization of limited wireless network bandwidth [12], [13]. If the segment download intervals of two DASH clients do not overlap, each DASH client overestimates the available bandwidth of the shared bottleneck link. As a result, these clients choose a higher segment bitrate than the current network conditions for the next segment request. Subsequently, if the next segment transmissions of the two clients overlap, congestion may occur because the sum of the segment bitrates exceeds the current available bandwidth of the shared link. This is known as an ON-OFF problem [12]. These problems must be addressed in enterprise/private networks where numerous clients share a common link.

Software-defined networking (SDN) [14] is a network configuration technology, in which a network is easily programmable by separating the control and forwarding planes. In an SDN, the entire network is managed by a centralized controller with open interfaces. SDN provides application programming interfaces (APIs) that enable the implementation of network applications. Network applications can be implemented and deployed to satisfy the specific requirements. Thus, combining SDN and HAS is effective for enterprise/private networks. Recently, considerable efforts have been dedicated to SDN-enabled HAS. In [15], Bhat et al. proposed a network-assisted adaptive bitrate streaming system. It employs SDN for data flow management to optimize cache content utilization and improve the quality of experience for DASH clients. In [16], Bentaleb et al. proposed an SDN-enabled HAS architecture to optimize user QoE by using reinforcement learning. In [17], Noh et al. proposed an SDN-assisted HAS system in which an SDN controller manages HAS requests in a local Wi-Fi network to provide high-quality video streaming services while supporting user fairness over time-varying network conditions. However, it is still difficult to provide high-quality video streaming services without frozen video when tens and hundreds of mobile devices attempt to associate with a single Wi-Fi access point (AP) due to the limited wireless bandwidth.

Thus far, many studies have been devoted to methods for effectively applying multiple wireless interface devices and APs to overcome the limitations of wireless resources. Schepper et al. [18] analyzed that the overall network throughput of a local area network (LAN) consisting of a variety of APs, can be improved by dynamic flow redirection with multiple network interface devices. Chen et al. [19] proposed a smart AP architecture that seamlessly migrates devices to connected APs by using multiple interfaces to balance the network load. So et al. proposed an effective wireless interface binding and a scheduling method to create a virtual link between two nodes. However, these multiple connections are vulnerable to the sudden degradation of wireless channels. When the DASH client receives out-of-order packets owing to the different channel statuses of multiple APs, goodput can be seriously degraded because of the head-of-line blocking (HOL) problem [20]. To handle this problem, a fountain code can be employed. If a sender transfers source data after being encoded with the fountain code to a receiver, the receiver can obtain the source data regardless of packet loss or out-of-order packets by decoding the fountain-encoded data. Thus, the goodput can be improved significantly. Kwon et al. [21] proposed a multi-path multimedia transport protocol that controls fountain encoding parameters such as symbol size, number of source symbols, and code rate by considering the remaining playback buffer time and network conditions. Pokhrel et al. [22] proposed a low-latency scheduling method for vehicular Internet by combining multi-path TCP with congestion control and forward error correction (FEC). Another issue with supporting multiple connections is energy consumption. Activating multiple network interfaces incurs high energy consumption. Thus, these issues must be addressed to enable multiple network interfaces for DASH clients successfully. Until now, considerable studies have been devoted to methods for energy-efficient data transmission. Hoque et al. [23] proposed EStreamer, which defines the relationship between traffic shaping and energy consumption for TCP. Based on the relationship, the traffic burst size was determined to provide smooth video streaming services with low energy consumption. Bui et al. [24] proposed GreenBag for real-time data streaming with energy-efficient bandwidth aggregation. Abou-Zeid et al. [25] proposed an energy-efficient predictive green streaming framework that leverages the data rates of a wireless network to minimize the base station power consumption by reducing the transmission time of video streams. However, these studies did not consider segment bitrate adaptation to provide seamless streaming services over time-varying wireless networks.

In this paper, we propose an energy-efficient HTTP adaptive streaming system for all DASH clients via SDN-enabled

Wi-Fi APs to maximize the overall video streaming quality while minimizing energy consumption. The proposed system utilizes multi-path technology to overcome the limitations of a single Wi-Fi AP wireless bandwidth. Furthermore, fountain code is employed as a forward error correction to support flexible and reliable data transmission via multiple APs. Moreover, the SDN controller manages the segment bitrate, code rate, and packet distribution vector among multiple APs and DASH clients during streaming service by using the overall network status, buffer occupancy, and energy consumption. The key contributions of this paper are summarized as follows:

- Proposing an energy-efficient HAS system over multiple SDN-enabled Wi-Fi APs, which employ the fountain code, novel metafile, and energy-efficient packet scheduling to provide high-quality streaming services without frozen video.
- Employing multiple wireless network interfaces with fountain code not only to overcome the limitation of wireless bandwidth of a single AP but also to support flexible and reliable data transmission via multiple APs.
- Designing the code rate of the LT code and packet distribution determining process via multiple Wi-Fi APs by considering the overall network status, remaining playback buffer, and energy consumption.
- The proposed system is implemented by using network simulator 3 (NS-3) [26] to validate its performance in a large-scale environment. Additionally, a real Wi-Fi testbed is implemented with ONOS and Raspberry PI 4s to verify the feasibility.

The rest of the paper is presented as follows. The basic idea of fountain code and HAS is briefly introduced in Section II. The details of the proposed energy-efficient HAS system are described in Section III. The experimental results are presented in Section IV, and finally, the concluding remarks are provided in Section V.

II. PRELIMINARY BACKGROUND

In this section, we briefly review the basic ideas of the fountain code and HAS in Sections II-A and II-B, respectively.

A. FOUNTAIN CODE

Fountain codes (e.g. Luby transform (LT) [27], Online code [28], Raptor [29], and RaptorQ [30]) are a class of erasure codes with high coding efficiency, flexibility, and low encoding/decoding processing times. The fountain code can generate encoding symbols infinitely from a finite number of source symbols. Due to this feature, it is also called rateless code. In fountain codes, the code rate (c) plays an important role as it determines the amount of redundant data used for error protection. That is defined by

$$c = k/n,$$

where k is the number of source symbols and n is the number of fountain encoding symbols. When a receiver receives a sufficient number of encoded symbols, it can reconstruct all source symbols, even if some packets are lost or out-of-order. The overhead of the fountain code is negligible because its average reception overhead of these fountain codes is approximately 0.2 % when a source block consists of more than 1,000 source symbols [31]. Due to these characteristics, fountain codes have been widely employed to provide smooth video streaming services over error-prone wireless networks.

B. HTTP ADAPTIVE STREAMING AND MPD

Has been widely used for commercial video streaming services such as YouTube, Netflix, Disney, and Microsoft. In HAS, a video is encoded at multiple bitrates, and then divided into smaller segments within a certain time unit. The segment information is contained in the MPD. The content server stores both the video segments and the corresponding MPD file. The MPD file is configured as an extensible markup language (XML) that contains detailed segment information and other metadata that is helpful to the client. The MPD is organized hierarchically with Period, AdaptationSet, Representation, and Segment. At the top layer, the MPD contains an ordered list of one or more consecutive non-overlapping periods. The Period layer represents the start time or duration to provide a starting position of the video stream. In each Period, there are several AdaptationSets. The AdaptationSet layer specifies the media type, such as video, audio, and caption. In the case of video, AdaptationSet contains Representation layers. Each Representation layer contains segment encoding features, such as resolution, encoding bitrate, frames per second (FPS), and video codec). The Representation layer includes Segment layers that provide the playback duration and URL of each segment. To initiate HAS, the client requests and receives an MPD from a content server. After receiving the MPD, the client analyzes it to select an appropriate segment by considering the network and remaining playback buffer status.

III. PROPOSED ENERGY-EFFICIENT COOPERATIVE HTTP ADAPTIVE STREAMING SYSTEM

The goal of the proposed system is to maximize the overall spatial video quality of all DASH clients in an energy-efficient way without noticeable frozen video instants, even if network congestion may occur at several Wi-Fi APs. To achieve our research goal, the proposed system utilizes multiple network interfaces to overcome the limited resources of a single Wi-Fi network. The LT-code is employed as an FEC scheme to support flexible and reliable data transmission via multiple APs. Additionally, an SDN is employed to fulfill efficient segment packet scheduling with a global view of the network.

The overall architecture of the proposed system is shown in Fig. 1. The proposed system consists of a DASH server, gateway server, SDN controller, SDN-enabled Wi-Fi APs, and DASH clients. The DASH server stores video segments and extended MPD, including the segment bitrate, URL, and video distortion information. Fig. 2 represents an

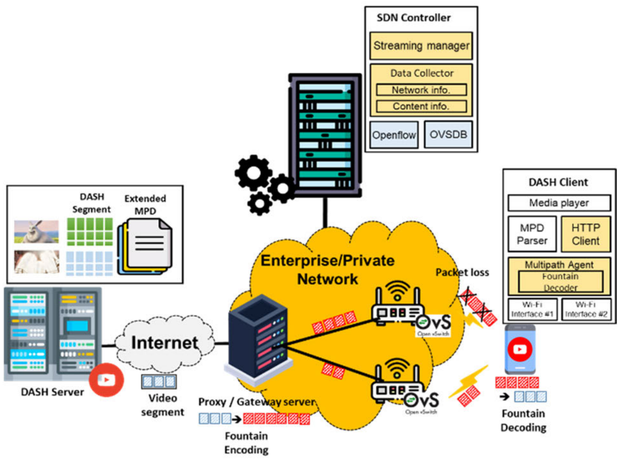
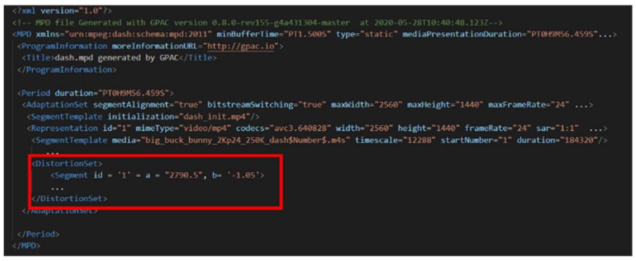


FIGURE 1. Overall architecture of the proposed system.



```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.8.0-rev155-g4a431304-master  at 2020-05-28T10:40:48.123Z-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500S" type="static" mediaPresentationDuration="PT0H9M56.459S"...>
 <ProgramInformation moreInformationURL="http://gpac.io">
  <Title>dash.mpd generated by GPAC</Title>
 </ProgramInformation>

 <Period duration="PT0H9M56.459S">
  <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth="2560" maxHeight="1440" maxFrameRate="24" ...>
   <SegmentTemplate initialization="dash_init.mp4"/>
   <Representation id="1" mimeType="video/mp4" codecs="avc3.640828" width="2560" height="1440" frameRate="24" sar="1:1"  ...>
    <SegmentTemplate media="big_buck_bunny_2Kp24_250K_dash$Number$.m4s" timescale="12288" startNumber="1" duration="184320"/>
      ...
   <DistortionSet>
       <Segment id = '1' = a = "2790.5", b= '-1.05'>
       ...
   </DistortionSet>
  </AdaptationSet>

 </Period>
</MPD>
```

FIGURE 2. Example of an extended MPD file structure.

example of an extended MPD. The SDN controller monitors and manages enterprise/private network components such as a Proxy/Gateway server, SDN-enabled APs, and DASH clients. The Proxy/Gateway converts the segment into fountain-encoded packets and forwards them to the SDN-enabled APs. SDN-enabled APs contain an SDN switch (i.e. Open vSwitch (OVS) [32]). The DASH client contains a multi-path agent, HTTP client, and media player with multiple Wi-Fi interfaces. It can be connected to multiple adjacent Wi-Fi APs by using multiple Wi-Fi interfaces.

The overall working procedure is presented in Fig. 3. The DASH client requests an extended MPD from the DASH server, and then the DASH server transmits the extended MPD to the DASH client. When the MPD is delivered to the SDN controller and DASH client, the data collector on the SDN controller can obtain video content characteristics (e.g. video encoding bitrates and rate-distortion model parameters) by parsing the MPD. Besides, DASH clients and SDN-enabled APs periodically report the network status information (e.g. available bandwidth, packet loss rate, and received signal strength indicator, etc.) and current playback buffer time. By using this information, the streaming manager on the SDN controller determines the operating parameters such as the segment bitrate, code rate, and packet distribution vector. These operating parameters are provided to the Proxy/Gateway server, SDN-enabled APs, and DASH clients. The multi-path agent on the DASH client requests the video segment from the DASH server based on the packet distribution vector. The DASH server then provides the corresponding segment. The segment is fountain encoded and packetized while passing the Proxy/Gateway server, and the fountain encoded stream is split and forwarded to the SDN-enabled APs by the SDN controller. The SDN-enabled AP manages the transmission of segment data in the time-slot unit to deal with the ON-OFF problem, as shown in Fig. 4. After receiving sufficient encoded segment packets from multiple SDN-enabled APs, the multi-path agent on the DASH client decodes the received packets to obtain the original segment data and stores it in a playback buffer. Then, the DASH client requests the next segment. This process is repeated until the entire streaming service is complete.

A. PROBLEM DESCRIPTION

In this section, we describe the problem formulation in detail. Before presenting a detailed description, some symbols are defined. First, the packet distribution vector is represented by

$$\overrightarrow{pkt}_i = \left(pkt_{i,1}, \ldots, pkt_{i,j}, \ldots, pkt_{i,|\mathbf{AP}|}\right), \tag{1}$$

where $pkt_{i,j}$ represents the number of packets to be delivered via the AP #j to the DASH client #i within the time slot unit, and $\mathbf{AP}$ is the set of available APs. In the proposed system, the video segment is fountain encoded with c_i to generate redundant packets. Thus, the segment bitrate that can be transmitted to the DASH client #i via multiple APs is represented by

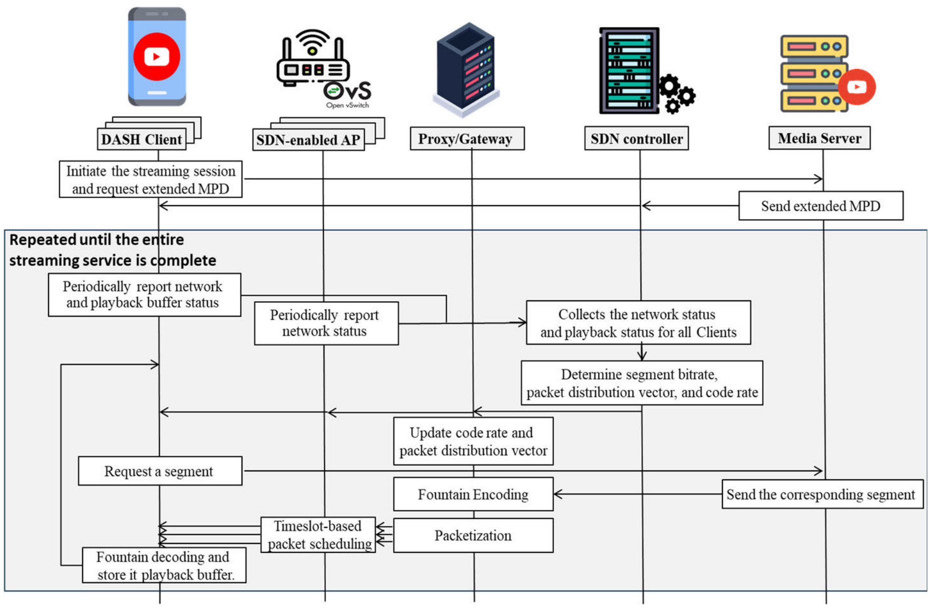
$$r_i^{seg}\left(\overrightarrow{pkt}_i, c_i\right) = \sum_{j \in \mathbf{AP}} pkt_{i,j} \cdot S_{pkt} \cdot c_i / T^{seg}, \tag{2}$$

where S_{pkt} is the size of the packet payload and T^{seg} is segment playback duration. The rate-distortion model [33] is used to estimate the objective video segment quality at the DASH client. The distortion of the video segment transferred to DASH client #i is modeled by

$$dist_i\left(\overrightarrow{pkt}_i, c_i\right) = \alpha_i \cdot r_i^{\beta_i}, \tag{3}$$

where α_i and β_i are the coefficients of the rate-distortion model of the video content provided to the DASH client #i. In the proposed system, α_i and β_i are obtained from a curve-fitting process by using the trust-region algorithm [34], and the DASH client can obtain the coefficients from the expanded MPD file, as shown in Fig. 2.

When multiple wireless network interfaces are enabled on DASH clients, SDN controllers can dynamically manage the wireless resources to enhance the overall video quality. However, activating multiple network interfaces consumes additional energy and computing power at Wi-Fi interfaces. In the proposed system, the Wi-Fi interface energy consumption model [35] is employed as follows (please refer to [35]



DASH Client
SDN-enabled AP
Proxy/Gateway
SDN controller
Media Server
Initiate the streaming session and request extended MPD
Send extended MPD
Repeated until the entire streaming service is complete
Periodically report network and playback buffer status
Periodically report network status
Collects the network status and playback status for all Clients
Determine segment bitrate, packet distribution vector, and code rate
Update code rate and packet distribution vector
Request a segment
Fountain Encoding
Send the corresponding segment
Timeslot-based packet scheduling
Packetization
Fountain decoding and store it playback buffer.

FIGURE 3. Overall working procedure of the proposed system.

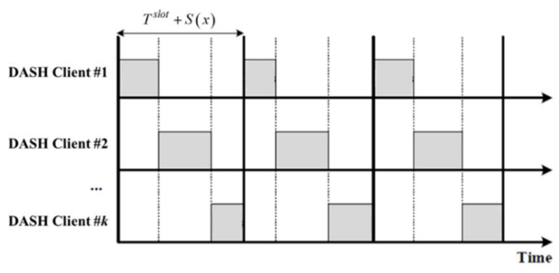


FIGURE 4. Example of synchronous control in the time slot unit.

for more details).

$$e_i\left(\overrightarrow{pkt}_i\right) = \sum_{j=1}^{N_{net}^i}\left\{p_{i,j}^{recv}\cdot t_{i,j}^{recv} + p_{i,j}^{tail}\cdot t_{i,j}^{tail} + p_{i,j}^{idle}\cdot t_{i,j}^{idle}\right\}$$

$$+\, p_i^{base}\cdot\left\{T^{slot} - \min_{1\le i\le N_{net}^i}\left(t_{i,j}^{rtt}\right)\right\}, \qquad (4)$$

$$p_{i,j}^{recv} = \rho_i\cdot\widetilde{bw}_{i,j} + \tau_i, \qquad (5)$$

where $p_{i,j}^{recv}$ is the consumed power of the DASH client #i to receive $pkt_{i,j}$ packets via AP #j, $p_{i,j}^{tail}$ is the consumed power of the Wi-Fi tail duration of the network interface of the DASH client #i connected to the AP #j, $p_{i,j}^{idle}$ is the consumed power of Wi-Fi idle state of the DASH client #i connected to the AP #j, p_i^{base} is the base power of the DASH client #i, $t_{i,j}^{recv}$ is the consumed time of the DASH client #i to receive $pkt_{i,j}$ packets via the AP #j, $t_{i,j}^{tail}$ is the tail duration of the Wi-Fi interface, $t_{i,j}^{idle}$ is the idle time of the Wi-Fi interface, $\widetilde{bw}_{i,j}$ is the available bandwidth between the DASH client #$i$and the AP #j, and ρ_i and τ_i are receiving power consumption model parameters, respectively. Now, we can formulate the optimal problem to achieve our goal as follows.

Problem formulation: Determine $\overrightarrow{pkt}_i$ and c_i with the given n_i^{net} for $\forall i\in\mathbf{C}$ to minimize the following cost functions

$$\sum_{i\in\mathbf{C}}\left(\omega\cdot dist_i\left(\overrightarrow{pkt}_i, c_i\right) + (1-\omega)\cdot e_{i,j}^{net}\left(\overrightarrow{pkt}_i\right)\right) \qquad (6)$$

$$\text{subjectto }\left\|\overrightarrow{pkt}_i\right\|_o \le n_i^{net}\text{ for }\forall i\in\mathbf{C}, \qquad (7)$$

$$r_i^{seg}\left(\overrightarrow{pkt}_i, c_i\right) \le r_i^{req}\text{ for }\forall i\in\mathbf{C}, \qquad (8)$$

$$r_i^{seg}\left(\overrightarrow{pkt}_i, c_i\right) \in \mathbf{R}_i^{MPD}\text{ for }\forall i\in\mathbf{C}, \qquad (9)$$

$$\varphi^{dec}\left(\overrightarrow{pkt}_i, c_i\right) \le \prod_{blk}^{\max}\text{ for }\forall i\in\mathbf{C}, \qquad (10)$$

$$\text{and}\sum_{i\in\mathbf{C}}\frac{pkt_{i,j}\cdot S_{pkt}}{\widetilde{bw}_{i,j}} \le T^{slot} + S\left(t_{avg}^{buf} - T_{th}^{buf}\right)\text{ for }\forall j\in\mathbf{AP}, \qquad (11)$$

where ω is a weighting factor ($0 \le \omega \le 1$), which is a system parameter to pursue an effective tradeoff between video quality and energy consumption, $\left\|\overrightarrow{pkt}_i\right\|_o$ is the number of non-zero entries of the $\overrightarrow{pkt}_i$, r_i^{req} is the segment bitrate requested by the DASH client #i, $\mathbf{R}_i^{MPD}$ is the set of segment bitrates available in the MPD by the DASH client #i, $\mathbf{C}$ is the set of DASH clients, t_{avg}^{buf} is the average remaining playback buffer time of all DASH clients, $\varphi^{dec}\left(\overrightarrow{pkt}_i, c_i\right)$ is the fountain decoding failure rate, $\prod_{blk}^{\max}$ is the tolerable maximum fountain decoding failure rate, T^{slot} is the time slot duration, and $S(\cdot)$ is the penalty function to avoid the playback buffer underflow caused by sudden network fluctuations. Eq. (7) indicates the constraint of the available wireless network interfaces that the number of APs delivering segment data to the DASH client should not exceed the number of enabled Wi-Fi interfaces, Eq. (8) is the constraint that the segment bitrate served to the DASH client must be less than or equal to the bitrate requested by the DASH client, and Eq. (11) means that all DASH clients must download a segment within a time constraint. The time constraint is changed based on the t_{avg}^{buf} to avoid playback buffer underflow and maintain a stable playback buffer status. If t_{avg}^{buf} is larger than the threshold T_{th}^{buf}, then $S(x)$ should be increased to improve the bitrate of new-coming segments while consuming redundant playback buffer times. Otherwise, $S(x)$ should be decreased to fill the playback buffer rapidly by decreasing the segment bitrate. Therefore, the third-degree polynomial function $S(x)$ is defined by

$$S(x)=\begin{cases} \min\left(\gamma_1 x^3+\gamma_2 x^2+\gamma_3 x+\gamma_4, T^{slot}/2\right) & x\geq 0 \\ \max\left(\gamma_1 x^3+\gamma_2 x^2+\gamma_3 x+\gamma_4, T^{slot}/2\right) & \text{otherwise,} \end{cases}$$

where $\gamma_1, \gamma_2, \gamma_3$, and γ_4 are coefficients of $S(x)$. The range of $S(x)$ is limited to $[-T^{slot}/2, T^{slot}/2]$ to avoid sudden segment bitrate fluctuations, which may occur blinking artifacts during video streaming services.

By the way, the above optimal problem is difficult to solve in real time because of the high computational complexity. In fact, c_i and $\overrightarrow{pkt}_i$ (for $i \in \forall\mathbf{C}$) are tightly coupled. The overall packet loss rate of the DASH client #i depends on the number of packets transmitted via each APs with the wireless channel condition. The code rate of the DASH client #i should be determined based on the overall packet loss rate. The number of transmitted packets (source and redundant packets) via each AP should be determined by the code rate. Thus, it is hard to determine $\overrightarrow{pkt}_i$ and c_i at once. In the proposed system, to achieve a practical and efficient near-optimal solution with reasonable computational complexity, $\overrightarrow{pkt}_i$ and c_i are sequentially and iteratively determined at each AP. This process is repeated until the optimal cost function converges, as shown in Fig. 5.

B. PROPOSED PARAMETER DETERMINING PROCESS

In this section, we describe the parameter determining process. First, the code rate determining algorithm is presented, and then the packet distribution vector deter-mining process is described in detail.

1) CODE RATE DETERMINING PROCESS

In the code rate determining process, the value of the code rate is determined based on $\overrightarrow{pkt}_i$ and the overall packet loss rate. When the packet distribution vector $\overrightarrow{pkt}_i$ is given, the overall PLR of the DASH client #i is calculated by

$$plr_t^{overall}\left(\overrightarrow{pkt}_i\right)=\frac{\sum\limits_{j\in\mathbf{AP}} pkt_{i,j}\cdot plr_{i,j}}{n_{pkt}^{tot}\left(\overrightarrow{pkt}_i\right)}, \tag{12}$$

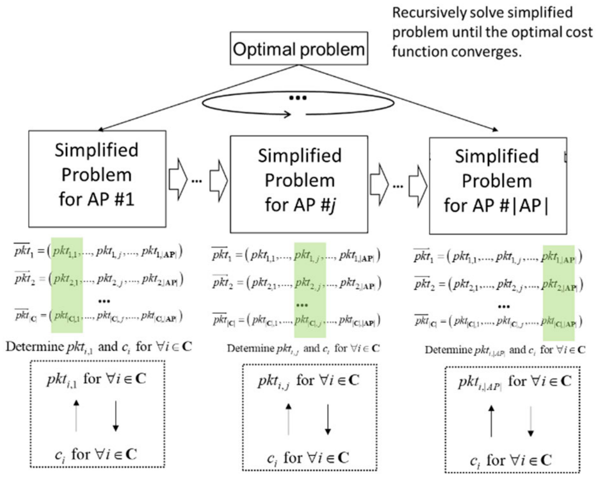

Recursively solve simplified problem until the optimal cost function converges.
Optimal problem
Simplified Problem for AP #1
Simplified Problem for AP #j
Simplified Problem for AP #|AP|
$\overrightarrow{pkt}_1=(pkt_{1,1},\ldots,pkt_{1,j},\ldots,pkt_{1,|\mathbf{AP}|})$
$\overrightarrow{pkt}_2=(pkt_{2,1},\ldots,pkt_{2,j},\ldots,pkt_{2,|\mathbf{AP}|})$
$\overrightarrow{pkt}_{|\mathbf{C}|}=(pkt_{|\mathbf{C}|,1},\ldots,pkt_{|\mathbf{C}|,j},\ldots,pkt_{|\mathbf{C}|,|\mathbf{AP}|})$
Determine $pkt_{i,1}$ and c_i for $\forall i\in\mathbf{C}$
Determine $pkt_{i,j}$ and c_i for $\forall i\in\mathbf{C}$
Determine $pkt_{i,|AP|}$ and c_i for $\forall i\in\mathbf{C}$
$pkt_{i,1}$ for $\forall i\in\mathbf{C}$
$pkt_{i,j}$ for $\forall i\in\mathbf{C}$
$pkt_{i,|AP|}$ for $\forall i\in\mathbf{C}$
c_i for $\forall i\in\mathbf{C}$

FIGURE 5. Illustration of the parameter determining process.

where $n_{pkt}^{tot}\left(\overrightarrow{pkt}_i\right)$ is the total number of packets transferred to DASH client #i via multiple APs (e.g. $\sum_{j\in\mathbf{AP}} pkt_{i,j}$) and $plr_{i,j}$ is the packet loss rate between DASH client #i and AP #j. $plr_{i,j}$ can be achieved by the SDN controller. Then, we can approximately calculate the fountain decoding failure rate with given $\overrightarrow{pkt}_i$ and c_i based on a binomial distribution (13) and (14), as shown at the bottom of the next page, where $n_{pkt}^{\min}\left(\overrightarrow{pkt}_i, c_i\right)$ is the minimum number of received packets for successful fountain decoding, and δ is the minimum symbol overhead. Then, the optimal code rate satisfying Eq. (10) can be calculated by

$$c_i^{opt}\left(\overrightarrow{pkt}_i\right) = \arg\min_{0<c_i\leq 1}\left(c^{diff}\left(\overrightarrow{pkt}_i, c_i\right)\right), \tag{15}$$

$$c^{diff}\left(\overrightarrow{pkt}_i, c_i\right) = \begin{cases} \prod_{blk}^{\max}-\pi_{blk}\left(\overrightarrow{pkt}_i, c_i\right) & \text{if}\,\varphi^{dec}\left(\overrightarrow{pkt}_i, c_i\right)\leq \prod_{blk}^{\max} \\ \infty & \text{otherwise.} \end{cases} \tag{16}$$

2) PACKET DISTRIBUTION VECTOR DETERMINING PROCESS

In the packet distribution vector determining process, the above optimal formulation is simplified for a single AP to achieve a practical and efficient near-optimal solution with low computational complexity.

*Simplified Problem Formulation at the AP #**k**:* Determine $pkt_{i,k}$ with given n_i^{net}, c_i, and $pkt_{i,j}$ for $\forall i \in \mathbf{C}$ and $\forall j \in \mathbf{AP}/\{k\}$ to minimize

$$\sum_{i\in\mathbf{C}} \omega\cdot\Delta dist_i\left(pkt_{i,k}, c_i\right) + (1-\omega)\cdot\Delta e_i\left(pkt_{i,k}\right)\Big|_{\substack{\text{with given } pkt_{i,j} \\ \text{for } \forall j\in AP/\{k\}}} \tag{17}$$

$$\text{subject to } \left\|\overrightarrow{pkt}_i\right\|_o \leq n_i^{net} \text{for } \forall i\in\mathbf{C}, \tag{18}$$

$$r_i^{seg}\left(\overrightarrow{pkt}_i, c_i\right) \leq r_i^{req} \text{for } \forall i \in \mathbf{C}, \quad (19)$$

$$r_i^{seg}\left(\overrightarrow{pkt}_i, c_i\right) \in \mathbf{R}_i^{MPD} \text{for } \forall i \in \mathbf{C}, \quad (20)$$

$$\text{and} \sum_{i\in\mathbf{C}} \frac{pkt_{i,j} \cdot S_{pkt}}{\widetilde{bw}_{i,j}} \leq T^{slot} + S\left(t_{avg}^{buf} - T_{th}^{buf}\right), \quad (21)$$

where $\Delta dist_i\left(pkt_{i,k}, c_i\right)$ and $\Delta e_i\left(pkt_{i,k}\right)$ are the amount of video distortion and energy consumption change of the DASH client #i when the number of packets transferred via the AP #k is $pkt_{i,k}$, respectively.

The streaming manager in the SDN controller selects an AP sequentially and iteratively and determines the number of packets transferred via the selected AP until the cost value (Eq. (6)) converges. For each iteration, the packet distribution values for a single AP #k are determined based on the fountain code rate by using the sliding window and greedy method to provide a near-optimal solution with relatively low computational complexity and the code rate is updated based on the determined packet distribution values. The details of the packet distribution determining process are in the followings.

Step 1) Initialize $\overrightarrow{pkt}_i$ to zeros, c_i to one for $\forall i \in \mathbf{C}$, and sliding window size μ^{wd}.

Step 2) Set AP index k to 1.

Step 3) Set $pkt_{i,k}$ for $\forall i \in \mathbf{C}$ to zeros.

Step 4) Select the DASH client #i (for $\forall i \in \mathbf{C}$) with the lowest value of

$$\omega \cdot \left(\Delta dist_i\left(pkt_{i,k} + \mu^{wd}, c_i\right) - \Delta dist_i\left(pkt_{i,k}, c_i\right)\right) + (1-\omega) \cdot \left(\Delta e_i\left(pkt_{i,k} + \mu^{wd}\right) - \Delta e_i\left(pkt_{i,k}\right)\right),$$

Step 5) Set $pkt_{i,k}$ to $pkt_{i,k} + \mu^{wd}$.

Step 6) If $\sum_{i\in\mathbf{C}} \frac{pkt_{i,j} \cdot S_{pkt}}{bw_{i,j}}$ is less $T^{slot} + S\left(t_{avg}^{buf} - T_{th}^{buf}\right)$, go to **Step 4).**

Step 7) Check the number of available network interfaces. For $\forall i \in C$, if $\left\|\overrightarrow{pkt}_i\right\|_o \geq n_i^{net}$, then find the smallest packet distribution values of the DASH client #i among all APs and set to 0.

Step 8) Change $pkt_{i,j}$ so that $r_i^{seg}\left(\overrightarrow{pkt}_i, c_i\right)$ is the nearest value less than or equal to the segment bitrate in the MPD and r_i^{req}.

Step 9) Update c_i for $\forall i \in \mathbf{C}$ by calling the code rate determining process with $\overrightarrow{pkt}_i \forall i \in \mathbf{C}$.

Step 10) Increase k by 1, and then, if $k > |\mathbf{AP}|$, go to **Step 2),** otherwise, go to **Step 3)** until the cost value (Eq. (6)) converge.

IV. EXPERIMENTAL RESULTS

The proposed system is implemented in a large-scale network environment by using the NS-3 [26], and a relatively small-scale real Wi-Fi network testbed is constructed with Raspberry PI-4s, laptops, and Wi-Fi dongles for practical reasons. In Section IV-A, we verify the performance of the proposed system in a simulation environment with a large number of DASH clients and APs. In Section IV-B, we evaluate the performance and feasibility of the proposed system in a real Wi-Fi testbed.

A. PERFORMANCE VERIFICATION ON NS-3-BASED LARGE-SCALE ENVIRONMENT

In this section, we demonstrate the performance of the proposed system on NS-3-based large-scale environment. During the experiments, the simulation environment is set up as follows.

① Network: We configure a large-scale network topology consisting of 18 DASH clients and three 802.11n APs. The positions and mobilities of the entities are shown in Fig. 6. The DASH clients are associated with APs located within 50 m. We employ yet another network simulator (YANS) [37] and Friis propagation loss model [38] to emulate the Wi-Fi channel. 24 Mbps background traffic is generated to each AP to take into consideration real network conditions.

② Test video: We use three 2K videos, Big Buck Bunny, Elephants Dream, and Sintel. The video stream is encoded at 250, 500, 1000, 1500, 2000, 3000, 3500, and 4000 Kbps using H.265 with 25 FPS and a group of pictures (GOP) composed of 25 frames (IPPP..P). The segment playback duration is set to 2 sec and the total playback time is set to 300 sec, respectively.

③ DASH client and controller: DASH clients #1, 4, 7, 10, 13, and 16 request Big Buck Bunny, DASH clients #2, 5, 8, 11, 14, and 17 request Elephants Dream, and DASH clients #3, 6, 9, 12, 15, and 18 request Sintel. DASH clients start video playback when the initial playback buffer is larger than 2 sec, the maximum playback buffer time is set to 10 sec, T_{th}^{buf} is set to 6 sec, and T^{slot} is set to 2 sec. The penalty function parameters γ_1,

$$\varphi^{dec}(\overrightarrow{pkt}_i, c_i) = P\left(X < n_{pkt}^{\min}\left(\overrightarrow{pkt}_i, c_i\right) | X \sim B\left(n_{pkt}^{tot}\left(\overrightarrow{pkt}_i\right), 1 - plr_t^{overall}\left(\overrightarrow{pkt}_i\right)\right)\right)$$
$$= \sum_{k=0}^{n_{pkt}^{\min}\left(\overrightarrow{pkt}_i, c_i\right)-1} \left(\begin{pmatrix} n_{pkt}^{\min}\left(\overrightarrow{pkt}_i, c_i\right) \\ k \end{pmatrix} \cdot \left(1 - plr_i^{overall}\left(\overrightarrow{pkt}_i\right)\right)^k \left(plr_i^{overall}\left(\overrightarrow{pkt}_i\right)\right)^{n_{pkt}^{tot}\left(\overrightarrow{pkt}_i\right)-k}\right), \quad (13)$$

$$n_{pkt}^{\min}\left(\overrightarrow{pkt}_i, c_i\right) = \left\lceil (1+\delta) \cdot n_{pkt}^{tot}\left(\overrightarrow{pkt}\right) \cdot c_i \right\rceil \quad (14)$$

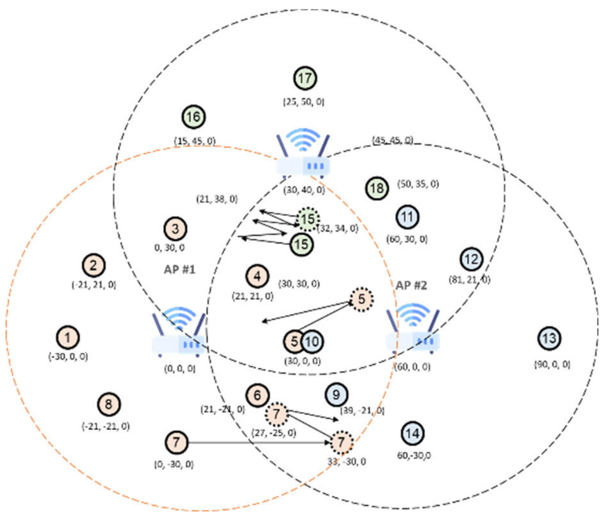


FIGURE 6. Simulation network topology.

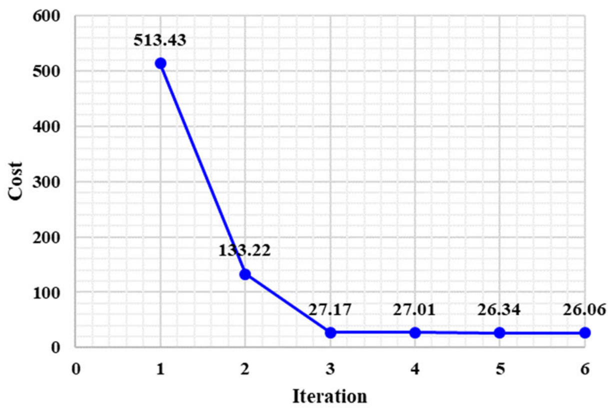


FIGURE 7. Change in the cost value (Eq. (6)) with the iteration.

γ_2, γ_3, and γ_4, are set to 0.0144, 0.0287, 0.0283, and 0.0059, respectively. The energy consumption model parameters p_i^{base}, $p_{i,j}^{tail}$, $p_{i,j}^{idle}$, ρ_i, and τ_i are set to 774.27, 310.44, 24.93, 210.67, and 374.91 mW, respectively, which are measured on the Samsung Galaxy S4 in a real Wi-Fi environment [35].

At first, we examine the proposed packet distribution vector and code rate determining algorithm. Figs. 7, 8, and 9 represent the cost value (including video distortion and energy consumption), packet distribution vectors, and code rates with the iteration, respectively. As shown in Figs. 8 and 9, at each iteration, the packet distribution vector and code rate of each DASH client are updated to minimize the cost value while preventing buffer underflows. The proposed system employs a greedy method with a window unit (a fixed number of packets) instead of byte units, which effectively reduces the computational complexity to find a near-optimal solution. Therefore, the cost value of distortion and energy consumption converges quickly, as shown in Fig. 7.

Secondly, we investigate the tradeoff relationship between video quality and energy according to ω. Table 1 represents the changes in the peak signal-to-noise ratio (PSNR), bitrate, and energy consumption according to ω. When ω is set to 0.000, the proposed algorithm does not consider the energy consumption. Thus, it provides the best video quality with the smallest standard deviation at the cost of the highest energy consumption as shown in Table 1. On the other hand, when ω is increased to 0.020, the energy consumption significantly decreases by 6.1%, but PSNR decreases by only 2.9% because energy consumption plays a more important role than video quality during the optimization process as ω increases.

TABLE 1. Performance summary according to the ω.

ω	PSNR (dB)		Bitrate (Kbps)		Energy (J)
	Avg.	Std.	Avg.	Std.	
0.000	42.230	4.167	3143.02	973.52	200.08
0.001	42.167	4.199	3099.65	990.11	198.46
0.005	41.814	4.202	2888.74	1059.46	195.38
0.010	41.435	4.202	2703.52	1107.87	192.67
0.015	41.260	4.350	2644.99	1126.79	191.36
0.020	40.992	4.383	2574.58	1157.62	187.69
0.100	40.029	4.717	2423.67	1220.66	179.82

Thirdly, we compare our proposed system with the three existing HAS systems.

- DASH.js [39]: Each DASH client independently monitors the available network bandwidth while receiving the previous segment data and then determines the next segment bitrate based on the measured bandwidth.
- BOLA [8]: Each DASH client independently selects the bitrate of the next request segment by using the Lyapunov optimization-based rate control algorithm to avoid buffer underflow.
- MP-HAS-SDN [36]: DASH clients support multiple network interfaces. The SDN controller centrally determines the segment bitrate and the amount of resources among APs and DASH clients. Based on the allocated resources, a segment is split into fragments and delivered to the DASH client via multiple APs.

The experimental results of all DASH clients are summarized in Tables 2, 3, 4, and 5. The detailed results of the bitrate adaptation, PSNR, and remaining playback buffer time of the DASH client #7 are shown in Figs. 10, 11, 12, and 13. As shown in Figs. 10 (a)-(c), DASH.js can provide a seamless streaming service when the network state is stable, but it cannot respond quickly to sudden network fluctuations. Additionally, DASH clients intermittently overestimate the available bandwidth, which causes a sudden playback buffer reduction due to incorrect segment bitrates at 80 sec and 100 sec. In the case of BOLA, it employs Lyapunov optimization, considering only the playback buffer time. Thus, it can rapidly fill the remaining playback buffer. As shown in Figs. 11 (b) and (c), when the playback buffer occupancy of the DASH client #7 becomes smaller, the segment bitrate is rapidly decreased to fill the playback buffer. As a result, buffer underflow occurs less than DASH.js, but more segment

TABLE 2. Performance evaluation Of Dash.Js On a large-scale environment.

DASH client	PSNR (dB)		Bitrate (kbps)		Playback buffer time (sec.)		Num. of buffer underflow
	Avg.	Std.	Avg.	Std.	Avg.	Std.	
1	38.35	3.83	1491.02	464.05	6.87	1.33	0.00
2	38.77	6.71	1667.68	632.85	7.28	0.92	1.00
3	40.12	4.42	1246.99	409.29	6.95	1.15	0.00
4	38.59	3.76	1614.46	556.27	6.61	1.64	0.00
5	38.32	6.20	1421.69	413.09	7.20	0.99	1.00
6	40.36	4.39	1315.87	407.17	7.12	1.15	0.00
7	38.35	4.00	1584.85	651.08	6.37	1.89	3.00
8	34.16	6.22	1416.17	414.33	7.12	1.01	0.00
9	40.32	6.36	1645.22	1075.28	7.21	0.99	0.00
10	41.14	3.55	2904.19	773.02	7.44	0.71	3.00
11	38.19	7.44	1807.43	1078.98	7.37	0.94	2.00
12	42.71	3.83	2571.86	767.63	7.39	0.75	0.00
13	38.87	4.14	1860.39	895.43	7.22	1.14	0.00
14	41.30	6.79	3053.89	827.02	7.41	0.72	0.00
15	42.90	4.30	2849.40	1104.64	6.98	1.14	0.00
16	42.57	3.75	3919.16	543.22	7.58	0.66	0.00
17	40.60	7.04	2774.39	1159.61	6.81	1.43	0.00
18	44.20	3.77	3925.15	597.61	7.48	0.67	0.00
Avg.	39.99	5.03	2170.54	709.48	7.13	1.07	0.56

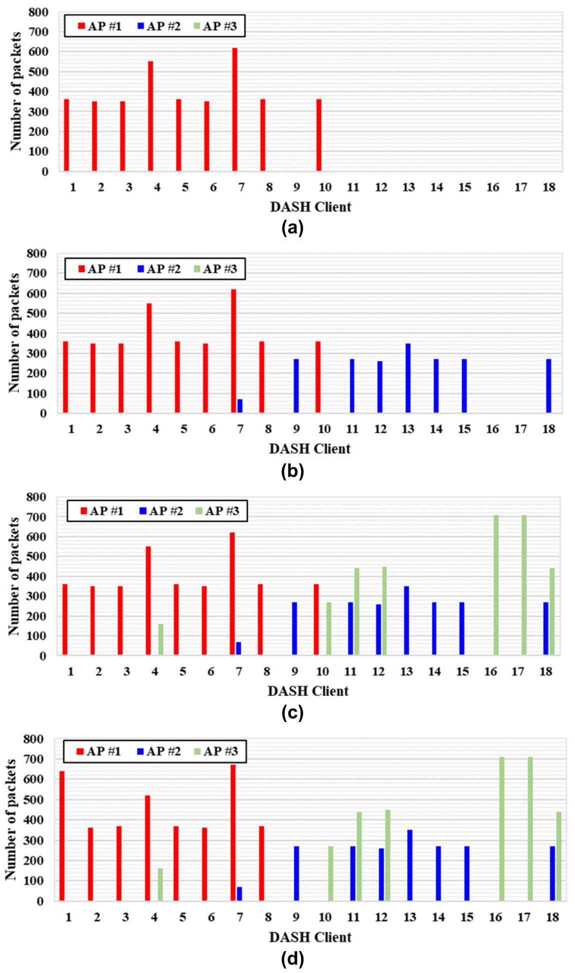


FIGURE 8. Change in number of transmitted packets with the iteration. (a) iteration #1, (b) iteration #2, (c) iteration #3, (d) iteration #4.

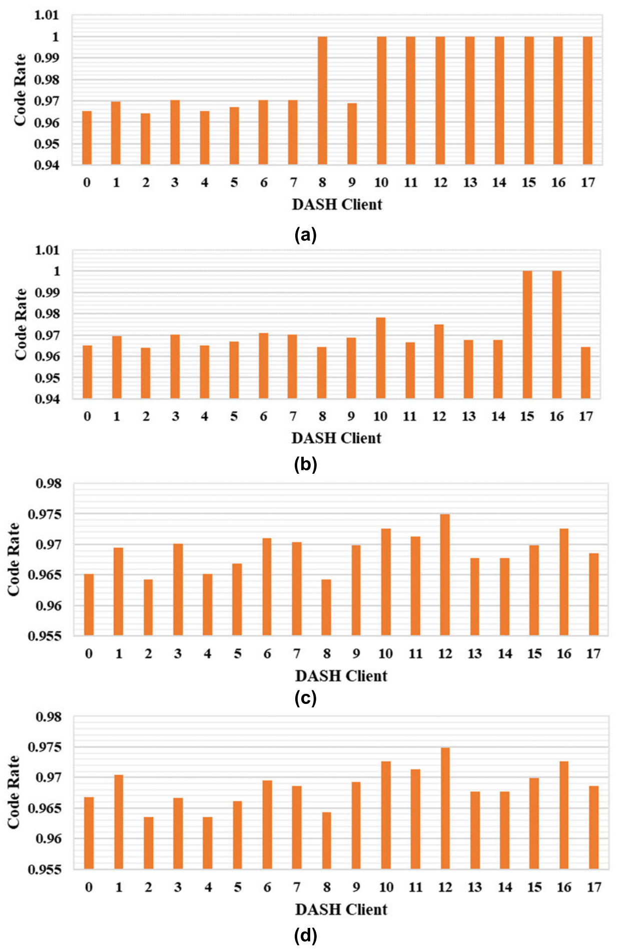


FIGURE 9. Change in partial segment bitrates with the iteration. (a) iteration #1, (b) iteration #2, (c) iteration #3, (d) iteration #4.

bitrate changes are incurred, which may cause blink artifacts. In the cases of the MP-HAS-SDN and the proposed system, they effectively perform bitrate adaptation and segment transmission scheduling via multiple Wi-Fi APs by considering the overall network and buffer statuses of all DASH clients. Thus, they can provide better video quality than DASH.js and BOLA without buffer underflow. However, as shown in Figs 12 and 13, due to instantaneous buffer reduction, the PSNR degradation occurs more frequently in MP-HAS-SDN than the proposed system. MP-HAS-SDN divides a segment into smaller fragments with determined sizes and transmits them to the DASH client via multiple APs. While a fragment is being transferred, if the wireless channel status is degraded due to the movement of the DASH client, it causes an additional transmission delay. It rapidly consumes playback buffer time because the DASH client must wait for all fragments to arrive before reconstructing the segment and storing it in the playback buffer. On the other hand, the proposed system can recover the playback buffer quickly because fountain-encoded packets can be decoded into a segment when a sufficient number of encoded packets are delivered to the DASH client regardless of the APs. Thus, the proposed system decreases the PSNR changes, sustains more playback buffer time of all DASH clients, and consumes less

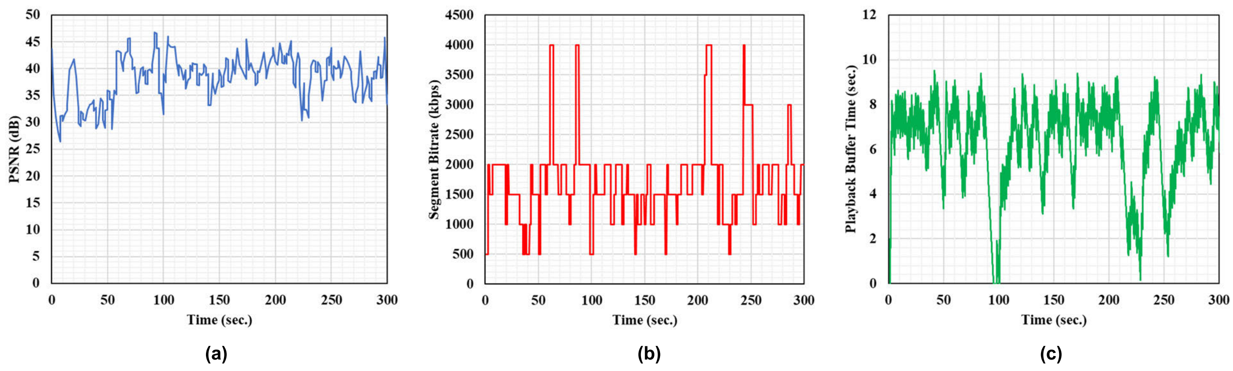


FIGURE 10. Performance evaluation of DASH.js at the DASH Client #7. (a) PSNR, (b) Bitrate, and (c) Playback buffer time.

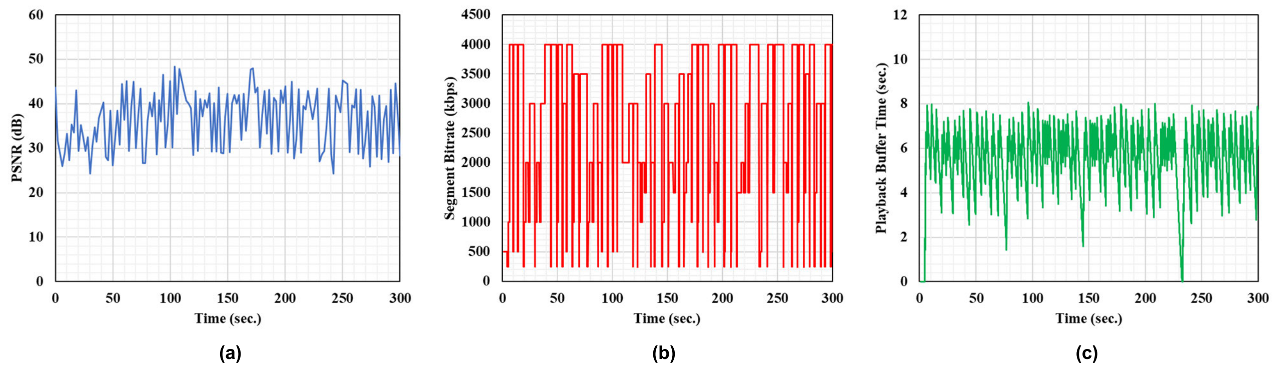


FIGURE 11. Performance evaluation of BOLA at the DASH Client #7. (a) PSNR, (b) Bitrate, and (c) Playback buffer time.

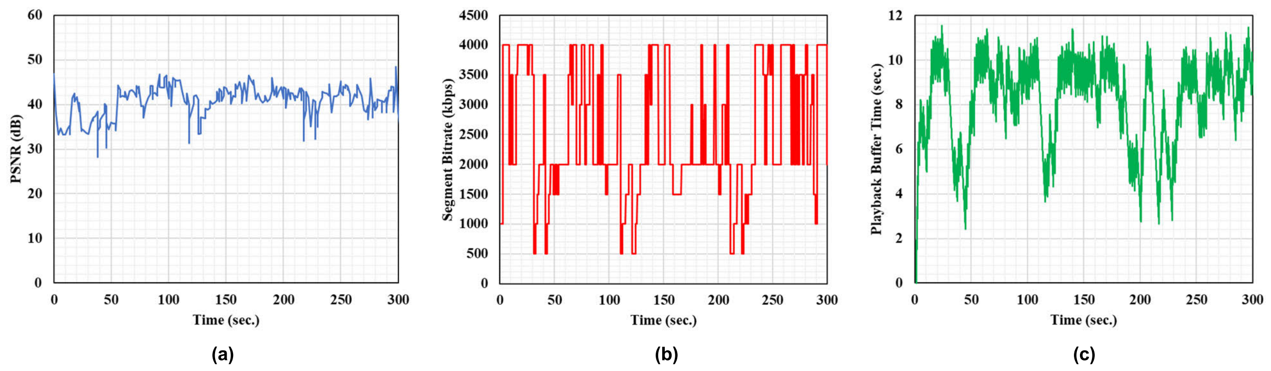


FIGURE 12. Performance evaluation of MP-HAS-SDN at the DASH Client #7. (a) PSNR, (b) Bitrate, and (c) Playback buffer time.

energy than MP-SDN-HAS owing to the energy consumption model-based packet scheduling, as shown in Tables 4 and 5. Consequently, the experimental results represent that the proposed system provides a better PSNR than any other existing system while providing seamless and stable video streaming services.

B. PERFORMANCE VERIFICATION ON REAL WIRELESS NETWORK TESTBED ENVIRONMENT

In this section, we present the performance of the proposed system on a real Wi-Fi environment. The real Wi-Fi testbed consists of a DASH server, SDN controller, SDN-enabled APs, and DASH clients, as shown in Fig. 14. The SDN controller is configured by using ONOS [40] on Ubuntu 18.04. The SDN-enabled Wi-Fi APs are configured by using Raspberry PI-4s installing the OVS and hostapd [41]. Five laptops are used as DASH clients. DASH clients are implemented on laptops and Wi-Fi dongles due to the limited number of Wi-Fi interfaces available on commercial smartphones (additional Wi-Fi interfaces can be installed on the commercial smartphone by using USB adaptors and Wi-Fi dongles. In this case, jailbreaking or rooting may be required to simultaneously

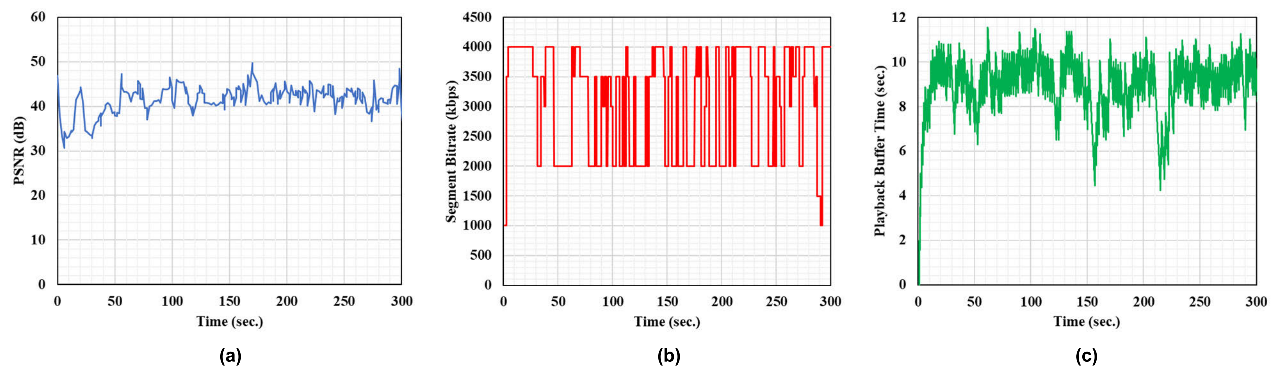


FIGURE 13. Performance evaluation of the proposed system at the DASH Client #7. (a) PSNR, (b) Bitrate, and (c) Playback buffer time.

TABLE 3. Performance evaluation Of Bola on a large-scale environment.

DASH client	PSNR (dB)		Bitrate (kbps)		Playback buffer time (sec.)		Num. of buffer underflow
	Avg.	Std.	Avg.	Std.	Avg.	Std.	
1	36.06	6.39	1700.30	1430.01	5.67	1.21	0.00
2	36.09	7.98	1709.34	1447.28	5.68	1.25	1.00
3	38.41	6.77	1673.19	1465.82	5.64	1.19	0.00
4	35.89	6.51	1699.70	1453.84	5.55	1.41	0.00
5	35.63	7.83	1671.69	1523.36	5.58	1.29	0.00
6	38.51	6.63	1707.83	1500.99	5.64	1.22	0.00
7	35.93	6.64	1694.28	1429.75	5.40	1.39	1.00
8	36.33	7.94	1734.94	1437.25	5.72	1.16	0.00
9	40.28	6.32	2236.45	1676.66	6.22	0.95	0.00
10	38.48	5.63	2336.83	1628.42	6.27	0.92	1.00
11	38.89	7.09	2401.20	1546.32	6.22	1.07	0.00
12	40.79	6.06	2323.35	1651.59	6.28	0.91	0.00
13	38.64	5.32	2353.92	1575.02	6.26	0.93	0.00
14	39.05	7.48	2417.17	1584.69	6.17	0.99	1.00
15	43.90	5.01	3649.70	1242.38	6.68	0.63	0.00
16	42.18	3.67	3676.65	651.90	6.67	0.61	0.00
17	42.11	6.38	3652.69	671.13	6.68	0.63	0.00
18	43.89	3.82	3631.74	684.77	6.66	0.63	0.00
Avg.	38.95	6.30	2348.39	1366.73	6.06	1.02	0.22

TABLE 4. Performance evaluation Of MP-SDN-HAS on a large-scale environment.

DASH client	PSNR (dB)		Bitrate (kbps)		Playback buffer time (sec.)		Num. of buffer underflow	Energy (J)
	Avg.	Std.	Avg.	Std.	Avg.	Std.		
1	40.30	5.24	2874.23	843.60	6.33	1.75	0.00	194.68
2	40.00	7.49	2820.12	871.13	6.88	1.46	0.00	190.83
3	41.34	5.20	2359.57	674.52	6.55	1.94	0.00	180.50
4	41.49	4.94	3003.05	636.35	4.55	1.39	0.00	226.90
5	41.41	7.80	2941.72	660.21	4.65	1.54	0.00	222.96
6	43.05	5.11	2374.23	770.57	6.75	1.45	0.00	222.80
7	40.06	5.24	2904.32	863.79	6.85	1.91	0.00	191.19
8	40.02	7.58	2809.82	854.01	6.87	1.03	0.00	190.80
9	40.75	5.32	1971.04	397.03	6.80	1.15	0.00	172.09
10	41.49	4.94	3152.44	636.35	6.55	1.07	0.00	226.86
11	41.41	7.76	2889.57	651.01	6.80	1.54	0.00	216.57
12	42.66	5.14	2777.78	723.57	6.80	1.74	0.00	203.97
13	39.02	4.92	2420.25	554.91	6.87	1.49	0.00	177.68
14	38.84	7.09	2361.96	355.89	6.78	1.40	0.00	175.07
15	40.61	5.31	2013.89	399.22	6.90	1.74	0.00	171.71
16	39.08	4.85	3254.60	506.40	6.77	1.74	0.00	178.92
17	38.99	7.26	3009.32	512.82	7.04	1.67	0.00	177.87
18	41.69	5.14	2725.46	723.62	6.93	1.24	0.00	204.03
Avg.	40.68	5.91	2703.52	646.39	6.54	1.51	0.00	195.86

activate multiple APs). Each laptop is equipped with two Wi-Fi dongles connected to different SDN-enabled APs to support multiple connections. Additionally, we emulate the energy consumption of the DASH client as a mobile device

TABLE 5. Performance evaluation of the proposed system on a large-scale environment.

DASH client	PSNR (dB)		Bitrate (kbps)		Playback buffer time (sec.)		Num. of buffer underflow	Energy (J)
	Avg.	Std.	Avg.	Std.	Avg.	Std.		
1	41.10	3.65	2936.89	1019.51	8.44	1.78	0.00	183.91
2	40.86	5.58	2760.21	1147.77	8.88	1.12	0.00	191.94
3	42.12	3.52	2157.77	1275.06	8.46	1.60	0.00	178.01
4	41.30	3.67	3707.32	1043.05	8.95	2.85	0.00	192.05
5	41.11	5.74	3655.18	1131.43	8.91	2.69	0.00	195.31
6	42.20	3.56	3588.57	1260.70	8.55	1.22	0.00	184.33
7	40.73	3.49	2783.38	1205.73	7.75	1.20	0.00	199.93
8	40.85	5.57	2768.90	1179.78	8.02	1.25	0.00	189.50
9	41.64	3.76	1726.22	1087.52	8.48	1.30	0.00	189.17
10	41.52	2.96	3707.32	1009.08	8.01	1.60	0.00	192.42
11	41.09	6.54	3655.18	1067.95	8.59	1.30	0.00	196.40
12	43.14	3.48	3351.07	1098.85	8.24	1.17	0.00	195.09
13	40.35	2.57	1969.51	985.12	8.65	1.15	0.00	206.00
14	40.23	6.36	1833.38	1104.75	8.19	1.24	0.00	200.32
15	41.76	3.45	1656.71	1157.56	8.36	1.29	0.00	193.17
16	41.70	2.80	2001.37	955.84	8.65	1.18	0.00	197.80
17	41.18	5.74	1949.24	1138.11	8.56	0.83	0.00	188.93
18	42.98	3.20	3356.86	1073.95	8.82	1.00	0.00	193.85
Avg.	41.44	4.20	2753.62	1107.87	8.47	1.43	0.00	192.67

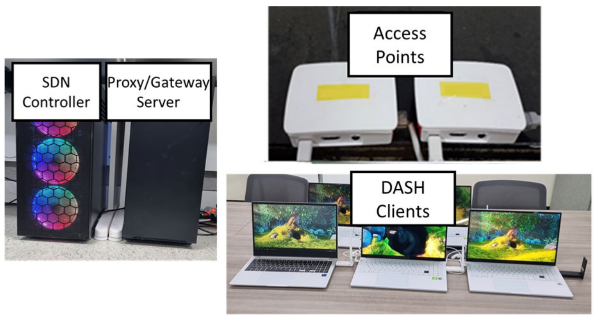


FIGURE 14. Real Wi-Fi testbed for the proposed HAS system.

by using the energy consumption parameters measured on real smartphones in Wi-Fi environments as presented in Section IV-A. Big Buck Bunny is used as the test video. The initial playback buffer time is set to 2 sec and the maximum buffered playback time is set to 10 sec. Background traffics between 35 and 45 Mbps are generated at the APs to emulate time-varying network conditions.

During the experiments, the proposed system is compared with MP-SDN-HAS to compare the PSNR, bitrate,

TABLE 6. Summary of performance comparison on real-testbed environment.

System	DASH client	PSNR (dB)		Bitrate (kbps)		Playback buffer time (sec.)		Num. of buffer underflow	Energy (J)
		Avg.	Std.	Avg.	Std.	Avg.	Std.		
MP-HAS-SDN	1	40.77	4.02	2921.83	694.44	9.25	0.88	0.00	205.68
	2	39.70	7.38	2743.37	674.95	8.66	0.90	0.00	200.60
	3	39.17	5.55	2349.97	717.96	8.99	1.03	0.00	177.61
	4	39.99	4.01	2921.90	537.64	8.77	0.90	0.00	212.24
	5	39.73	5.14	2776.10	732.40	8.63	0.91	0.00	200.96
	Avg.	40.07	5.22	2742.63	671.47	8.85	0.92	0.00	199.42
Proposed	1	41.16	3.59	2945.12	855.79	9.74	0.64	0.00	206.48
	2	41.08	6.79	2837.42	880.54	9.46	0.67	0.00	192.98
	3	40.23	5.09	2478.53	869.85	9.62	0.69	0.00	179.09
	4	41.25	3.63	2993.87	1186.81	9.31	0.68	0.00	196.55
	5	40.96	3.65	2765.03	888.66	9.37	0.67	0.00	191.55
	Avg.	40.94	4.55	2803.99	936.33	9.50	0.67	0.00	193.33

and energy consumption of multiple network interfaces. The experimental results are summarized in Table 6. As shown in the table, the proposed system can provide a higher quality streaming service and stable buffered playback time while consuming less energy than the MP-SDN-HAS. In the proposed system, fountain encoded packets can be flexibly transmitted via multiple APs according to time-varying network conditions, and it is managed in an energy-efficient way. On the other hand, MP-SDN-HAS cannot respond to network fluctuations until the pre-separated partial segments are fully transmitted. Moreover, it only focuses on increasing the PSNR without considering energy consumption.

V. CONCLUSION

In this paper, we proposed an energy-efficient HTTP adaptive streaming system to maximize the overall video streaming service quality of all clients in an energy-efficient way over SDN-enabled Wi-Fi APs. The proposed system employs multi-path technology to overcome the limited wireless bandwidth, and the LT code is adopted as an FEC scheme to support flexible and reliable data transmission via multiple APs. SDN technology is employed to effectively harmonize multiple Wi-Fi APs and DASH clients in a time slot unit as well as centrally control packet transmission by considering the global view of network status, buffer occupancy, and energy consumption. Furthermore, the proposed system can operate in real time by finding a near-optimal solution with low computational complexity. The proposed system is validated on a large-scale simulation environment as well as a real Wi-Fi network environment. The experimental results show that the proposed system can provide high quality video streaming services while maintaining a stable playback buffer time with low energy consumption.

REFERENCES

[1] *YouTube*. Accessed: Sep. 27, 2023. [Online]. Available: http://www.youtube.com
[2] *Netflix*. Accessed: Sep. 27, 2023. [Online]. Available: http://www.netflix.com
[3] *Disney+*. Accessed: Sep. 27, 2023. [Online]. Available: http://www.disneyplus.com
[4] *IMT Traffic Estimates for the Years 2020 to 2030*, ITU, Geneva, Switzerland, 2015.
[5] Ericsson. *Ericsson Mobility Report Data and Forecast*. Accessed: Sep. 27, 2023. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast
[6] Cisco. *Cisco Annual Internet Report (2018–2023)*. Accessed: Sep. 27, 2023. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html
[7] *Information Technology—Dynamic Adaptive Streaming Over HTTP (Dash)—Part 1: Media Presentation Description and Segment Formats*, Standard ISO/IEC 23009-1:2019, 2019.
[8] 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.
[9] 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.
[10] 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.*, Dec. 2012, pp. 97–108.
[11] 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.
[12] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *Proc. 22nd Int. Workshop Netw. Operating Syst. Support Digit. Audio Video*, Jun. 2012, pp. 9–14.
[13] R. Houdaille and S. Gouache, "Shaping HTTP adaptive streams for a better user experience," in *Proc. 3rd Multimedia Syst. Conf.*, Feb. 2012, pp. 1–9.
[14] K. Kirkpatrick, "Software-defined networking," *Commun. ACM*, vol. 56, no. 9, pp. 16–19, Sep. 2013.
[15] D. Bhat, A. Rizk, M. Zink, and R. Steinmetz, "Network assisted content distribution for adaptive bitrate video streaming," in *Proc. 8th ACM Multimedia Syst. Conf.*, Jun. 2017, pp. 62–75.
[16] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous, "SDNHAS: An SDN-enabled architecture to optimize QoE in HTTP adaptive streaming," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2136–2151, Oct. 2017.
[17] H. Noh, H. Lee, Y. Go, H. Park, J. Lee, J. Kim, and H. Song, "Congestion-aware HTTP adaptive streaming system over SDN-enabled Wi-Fi network," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 21, pp. 1–14, Nov. 2020.
[18] T. De Schepper, S. Latré, and J. Famaey, "Flow management and load balancing in dynamic heterogeneous LANs," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 693–706, Jun. 2018.
[19] X. Chen, Y. Zhao, B. Peck, and D. Qiao, "SAP: Smart access point with seamless load balancing multiple interfaces," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1458–1466.
[20] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. 8th USENIX Conf. Netw. Syst. Design Implement.*, Apr. 2011, pp. 99–112.
[21] O. C. Kwon, Y. Go, Y. Park, and H. Song, "MPMTP: Multipath multimedia transport protocol using systematic raptor codes over wireless networks," *IEEE Trans. Mobile Comput.*, vol. 14, no. 9, pp. 1903–1916, Sep. 2015.
[22] S. R. Pokhrel and J. Choi, "Low-delay scheduling for Internet of Vehicles: Load-balanced multipath communication with FEC," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8489–8501, Dec. 2019.
[23] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "TCP receive buffer aware wireless multimedia streaming: An energy efficient approach," in *Proc. 23rd ACM Workshop Netw. Operating Syst. Support Digit. Audio Video*, Feb. 2013, pp. 13–18.
[24] 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.
[25] H. Abou-zeid, H. S. Hassanein, and S. Valentin, "Energy-efficient adaptive video transmission: Exploiting rate predictions in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2013–2026, Jun. 2014.
[26] *The Network Simulator 3 (NS-3)*. Accessed: Sep. 27, 2023. [Online]. Available: http://www.nsnam.org
[27] M. Luby, "LT codes," in *Proc. IEEE Symp. Found. Comput. Sci.*, Nov. 2002, pp. 271–280.
[28] P. Maymounkov, "Online codes," New York Univ., New York, NY, USA, Tech. Rep., TR2002-833, Nov. 2002.

[29] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
[30] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RFC 6330: RaptorQ forward error correction scheme for object delivery," IETF Request Comments, Fremont, CA, USA, Tech. Rep. RFC 6330, 2011.
[31] T. Stockhammer, A. Shokrollahi, M. Watson, M. Luby, and T. Gasiba, "Application layer forward error correction for mobile multimedia broadcasting," in *Handbook of Mobile broadcasting: DVB-H, DMB, ISDB-T and Media FLO*. Boca Raton, FL, USA: CRC Press, 2008, pp. 239–280.
[32] *Open vSwitch*. Accessed: Sep. 27, 2023. [Online]. Available: http://openvswitch.org
[33] D. Jurca and P. Frossard, "Media flow rate allocation in multipath networks," *IEEE Trans. Multimedia*, vol. 9, no. 6, pp. 1227–1240, Oct. 2007.
[34] H.-P. Helfrich and D. Zwick, "A trust region algorithm for parametric curve and surface fitting," *J. Comput. Appl. Math.*, vol. 73, nos. 1–2, pp. 119–134, Oct. 1996.
[35] 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.
[36] H. Noh, G. S. Park, Y. Go, and H. Song, "HTTP adaptive streaming system maximizing overall video quality over SDN-enabled Wi-Fi APs," in *Proc. 18th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2022, pp. 178–183.
[37] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proc. Workshop ns-2: IP Netw. Simulator*, 2006, pp. 1–10.
[38] H. T. Friis, "A note on a simple transmission formula," *Proc. IRE*, vol. 34, no. 5, pp. 254–256, May 1946.
[39] *DASH-IF Dash.js Player*. Accessed: Sep. 27, 2023. [Online]. Available: https://github.com/DASH-Industry-Forum/dash.js/wiki
[40] P. Berdel, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, and B. Lantz, "ONOS: Towards an open, distributed SDN OS," in *Proc. ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 1–6.
[41] *Hostapd*. Accessed: Sep. 27, 2023. [Online]. Available: https://w1.fi/hostapd/

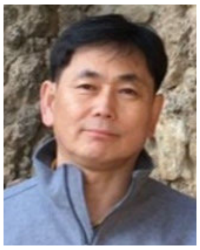
HYUNMIN NOH received the B.S. degree from the School of Computer Science and Engineering, Hanyang University, South Korea, in 2015, and the Ph.D. degree in computer science and engineering from the Pohang University of Science and Technology (POSTECH), South Korea, in 2022. He is currently an Assistant Professor with Jeonbuk National University, South Korea. Prior to joining Jeonbuk National University, he was a Postdoctoral Researcher with the Pohang University of Science and Technology (POSTECH), in 2022. His research interests include AR/VR streaming, 5G/6G networks, and blockchain.

GI SEOK PARK received the degree (summa cum laude) from the Department of Electrical Engineering, Dongguk University, Seoul, South Korea, in 2010, and the M.S. and Ph.D. degrees from the Department of IT Convergence Engineering, Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 2013 and 2018, respectively. From 2018 to 2019, he was a Senior Engineer with Samsung Electronics, Suwon, South Korea. From 2019 to 2023, he was an Assistant Professor with Dongguk University. He is currently an Assistant Professor with Incheon National University (INU), Incheon, South Korea. His research interests include web engineering and distributed systems. He was a recipient of the Dongguk Academic Award, in 2022.

YUNMIN GO received the B.S. degree from the School of Computer Science and Electrical Engineering, Handong Global University, Pohang, South Korea, in August 2010, and the Ph.D. degree from the Division of IT Convergence and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, in August 2018. From 2017 to 2018, he was a Postdoctoral Researcher with the BK 21+ POSTECH Computer Science and Engineering Institute. From 2018 to 2020, he was a Staff Engineer with Samsung Electronics, Suwon, South Korea. He is currently an Assistant Professor with Handong Global University. His research interests include HTTP adaptive streaming, channel coding-based streaming, and energy-efficient wireless networks.

SANG-HEON SHIN received the B.S. degree in electronic engineering and the M.S. and Ph.D. degrees in information communication engineering from Yeungnam University, South Korea, in 1998, 2000, and 2004, respectively. In 2009, he joined the Communication Research and Development La., Hanwha Systems, South Korea. He is currently a Senior Engineer with the Intelligent C4I Team, Hanwha Systems. His research interests include VR/MR streaming, C4I systems, tactical networks, satellite communication, network performance analysis, and network M&S.

YOUNGCHAN JANG received the M.S. degree in computer engineering from Chungnam National University, South Korea, in 2012. From 2012 to 2016, he was a Researcher with the Agency for Defense Development, South Korea. In 2023, he joined the Communication Research and Development La., Hanwha Systems, South Korea. He is currently a Senior Engineer with the Intelligent C4I Team, Hanwha Systems. His research interests include VR/MR streaming, computer graphics, and computer vision.



HWANGJUN SONG received the B.S. and M.S. degrees from the Department of Control and Instrumentation (EE), Seoul National University, Seoul, South Korea, in 1990 and 1992, respectively, and the Ph.D. degree in electrical engineering-systems from the University of Southern California (USC), Los Angeles, CA, USA, in 1999. From 2000 to 2005, he was an Assistant Professor/Vice Dean of Admission Affairs, Hongik University, Seoul. Since February 2005, he has been with the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), South Korea. From 2011 to 2012, he was a Courtesy Associate Professor with the University of Florida on Sabbatical Leave. From 2016 to 2017, he was the Vice President of the Korean Institute of Information Scientists and Engineers. His research interests include media processing and communication, network protocols necessary to implement functional rich media streaming, and blockchain. He was a co-recipient of the Workshops Best Paper Award from IEEE SDS and FMEC 2018 and the Best Paper Award from IEEE WiMob 2022. He was an Editorial Board Member of *Journal of Visual Communication and Image Representation* and an Associate Editor of *Journal of Communications and Networks*. He served as a Guest Editor for a Special Issue on "Network Technologies for Emerging Broadband Multimedia Services" in the *Journal of Visual Communication and Image Representation*. He was an APSIPA Distinguished Lecturer, from 2017 to 2018.