Received March 3, 2022, accepted March 31, 2022, date of publication April 8, 2022, date of current version April 15, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3165657

Scalable Multicast for Live 360-Degree Video Streaming Over Mobile Networks

DUC NGUYEN[1], (Member, IEEE), NGUYEN VIET HUNG[2,3], NGUYEN TIEN PHONG[3], TRUONG THU HUONG[3], (Member, IEEE), AND TRUONG CONG THANG[4], (Senior Member, IEEE)

[1]Tohoku Institute of Technology, Sendai 982-8577, Japan
[2]Faculty of Information Technology, East Asia University of Technology, Hanoi 100000, Vietnam
[3]School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi 100000, Vietnam
[4]Department of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 965-0006, Japan

Corresponding author: Truong Thu Huong (huong.truongthu@hust.edu.vn)

This work was supported in part by the Vingroup Innovation Foundation (VINIF) under Project VINIF.2020.DA03, and in part by the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) under Grant 22K12299.

ABSTRACT Thanks to its ability to provide immersive experience to users, 360-degree video has become one of the key enablers of Virtual Reality. However, the huge data size of 360-degree video poses a challenging problem for live streaming of this special type of video over resource-constrained networks. In this paper, we propose a novel framework for live 360-degree video streaming to multiple users over mobile networks. Our proposed framework jointly utilizes Scalable Video Coding and multicast to deliver 360 video to users in a bandwidth-efficient manner. In particular, 360-degree video is split into small parts called tiles, each is encoded into multiple layers using Scalable Video Encoding. The proposed framework then selects suitable tile layers to maximize the overall Quality of Experience of all users. To support real-time adaptation, we design a Linear Regression-based algorithm to estimate the weights of tiles of individual users. In addition, an efficient algorithm for deciding the suitable tile layers and their transmission modes is proposed. Experimental results show that the proposed method can significantly improve the average viewport quality compared to state-of-the-art methods.

INDEX TERMS Virtual reality, 360-degree video, scalable video coding, quality of experience.

I. INTRODUCTION

Virtual Reality technologies are changing the way people communicate and interact with the digital worlds, bringing transformations to industries such as gaming, entertainments. The global Virtual Reality market is expected to growth by 18% annually over the next seven years [1]. 360-degree video (*360 video* for short) is one of the main content type in Virtual Reality [2]. By capturing a 360-degree view of a scene, 360 videos allow users to freely change their viewing direction and watch the video from different perspectives. Watching 360 video on Head-Mounted Displays (HMD) offers the so-called **immersive experience** where users can feel like they are in a different place rather than their true locations [3]. 360 video-based applications are being applied in many fields from educations, manufacturing, to entertainment, thanks to the popularity of affordable Virtual Reality headsets and high-speed Internet connections.

Different from conventional videos, 360-degree videos are normally consumed on Virtual Reality's Head-Mounted Displays (HMDs) with the displays are very close to viewer's eyes. Combining with the fact that a 360-degree video captures a scene in every direction, 360-degree videos should have a very high resolution (e.g., $\geq$ 4K) in order to provide satisfactory viewing experience to users [4]. This results in high video data rates, and thus streaming 360-degree videos over networks requires a large amount of network bandwidth [5]. Live 360-degree video is one of the most popular type of 360-degree videos. Live streaming of sport/music events usually attracts a large number of audience watching an event at a same time. Moreover, unlike conventional videos where multiple users can share a screen (e.g., a TV screen), users watch a 360-degree video on their own devices (i.e., Head-Mounted Display). Thus, applying the conventional unicast streaming method for the live streaming scenario would consume a huge mount of network resources, which is not desirable in resource-constrained mobile networks.

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

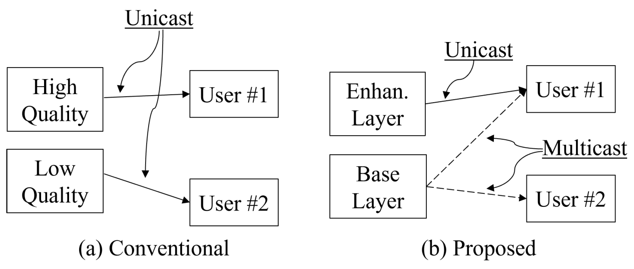


FIGURE 1. (a) Conventional method: Tiles are encoded into separate versions. Different versions of a tile are delivered separately to users. (b) Proposed method: Tiles are encoded into multiple layers. The base layer is multicasted to both users, and the enhancement layer is streamed in unicast to User #1.

To address the above challenge, existing solutions rely on tile-based viewport adaptive streaming and multicast to reduce network bandwidth consumption [6]–[9]. In particular, a full 360-degree video is divided into non-overlapping parts called *tiles*, each of which is encoded into multiple versions with different quality levels. Given users' viewing directions, tiles inside the viewport are delivered at a high quality version while those outside the viewport are delivered at a lower quality version. In addition, if a tile version is requested by multiple users, the tile version will be delivered using multicast to further reduce network resource requirement. In [6], [7], a tile version can be delivered to a user through both unicast and multicast. As a result, if a user receives multiple versions of a tile, only the version with the highest quality is used while the others are discarded, causing waste of available network resources. The methods proposed in [8], [9] deliver a tile version to a user in either the unicast or multicast mode. This way, sending multiple versions of a tile to a same user can be avoided. This method, however, still requires different versions of a same tile to be streamed over the network separately. An example is shown in Fig. 1a. When two users need different versions of a tiles, both versions must be delivered in the unicast mode to individual users. The primary problem with this scheme is that it does not consider the fact that the two versions of a tile share a lot of similarities.

In this paper, we propose a new framework for live streaming of 360-degree videos to multiple users over mobile networks. The proposed framework can improve network resource utilization by jointly optimizing the content preparation and transmission of tiles. Specifically, we propose to encode tiles into multiple layers using Scalable Video Encoding. The layers consist of a base layer and multiple enhancement layers in which the base layer provides the basic quality. Combining the base layer with the enhancement layers provides better quality. By sending the enhancement layers for tiles inside the user viewport, significant amount of the network resource can be saved. In addition, the proposed method exploits cross-user similarity and applies multicast to optimize the transmission of tiles. In particular, the layers requested by multiple users are delivered in the multicast mode to further reduce network bandwidth consumption. Since only the differences between quality versions (i.e., enhancement layers) are needed to be transmitted, the proposed method can better utilize the available network resource than the conventional methods, as can be shown in Fig. 1b. To the best of our knowledge, this is the first work that jointly combines Scalable Video Coding and multicast for live streaming of 360-degree video over mobile networks.

The remaining of the paper is as follows. Section II gives an overview of the related work. The proposed method is described in Section III, followed by an evaluation in IV. Finally, the paper is concluded in Section V.

II. RELATED WORK

A. VIEWPORT ADAPTIVE STREAMING

Due to the limited Field of View of both the Virtual Reality headsets and human eyes, the video part that users can see (i.e., *viewport*) is approximately 15-20% of the full 360-degree video. Thus, it is possible to reduce the transmission data by streaming the viewport at a high quality and the remaining video part at a lower quality. This idea is referred to as Viewport Adaptive Streaming [5].

There are two popular Viewport Adaptive Streaming approaches, namely *Tile-based Approach* and *Viewport-Dependent Approach*. In the Tile-based Approach, the original 360-degree video is spatially divided into non-overlapping parts called *tiles*. Individual tiles are encoded into multiple versions of different quality levels. Given the user's viewport position and available network bandwidth, the system selects the highest possible quality level for the tiles that are inside the viewport, and a lower quality level for the other tiles [10]–[13]. In the Viewport-dependent Approach, the original 360-degree video is not split into tiles. Instead, multiple targeted viewport positions are first selected. After that, for each of the selected viewport positions, the original 360-degree video is encoded so that the quality of the viewport area is higher than the non-viewport ones. For viewport adaptation, the version with the targeted viewport closest to the user's viewport position is selected and delivered [14].

Viewport Adaptive Streaming (VAS) has been reported to be able to reduce the bandwidth requirement by up to 80% compared to conventional streaming approach [15], [16]. To achieve high performance, VAS methods require not only accurate network bandwidth prediction but also accurate viewport prediction [17]. A user's future viewport positions can be predicted from the user's past viewport information [5], cross-user behaviors [18], and video characteristics [19]. Since existing viewport prediction methods are less effective for long prediction windows [16], tiles adjacent to the viewport are also delivered at a high quality [5]. A comparison of tile version selection methods can be found in [17].

Another key issue in designing Viewport Adaptive Streaming solutions is the trade-off between network adaptivity and user adaptivity [20], [21]. In particular, to ensure smooth video playback under network variations, the streaming client needs to buffer some amount of video data prior to playback [22]. A larger buffer size could better cope with network fluctuations. Yet, it could negatively affect the viewport

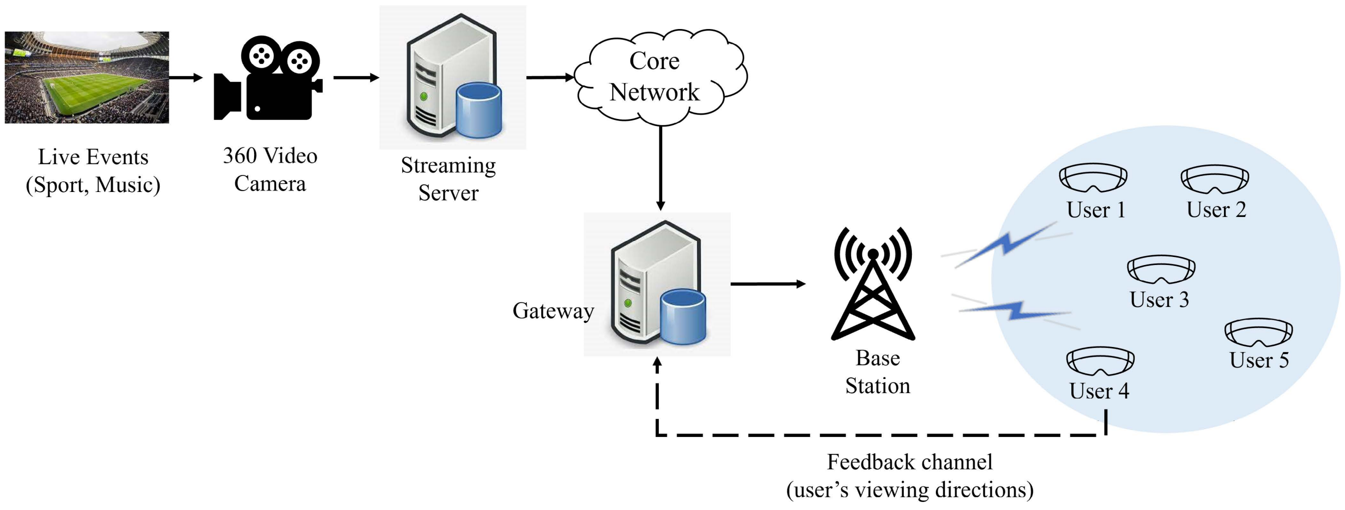


FIGURE 2. General architecture of proposed system.

prediction accuracy, and so reducing the effectiveness of VAS. To solve this problem, some previous methods applied Scalable Video Encoding to encode individual tiles into a base layer and multiple enhancement layers [20], [21]. By buffering only the base layer and downloading the enhancement layers ahead of their playback times, smooth video playback and viewport adaptivity could be achieved at the same time [20].

B. MULTI-USER 360 VIDEO STREAMING

Some previous works have proposed solution for streaming of 360-degree video to multiple users [6]–[9], [23]–[25]. In the early work [23], the authors proposed to transmit partial frames to users based on the user's viewing direction. The partial frames consists of multiple macroblocks. The macroblocks required by a single viewer are unicasted to the viewer. For a mackroblock required by multiple viewers, it will be transmitted to the viewers using multicast to reduce bandwidth consumption. The key problem with this solution is that adaptation at the macroblock level requires modifications of the encoders, and thus increasing system complexity.

Recent works have proposed to combine tile-based viewport adaptive streaming and multicast for streaming 360-degree videos to multiple users [6]–[9], [25], [26]. In [25], users are first grouped into multiple multicast groups based on their network conditions. Users in a same multicast group then receive a same set of tiles' representations (quality). The drawback of this solution is that it does not consider the fact that different users tend to watch different parts of a 360-degree video. In [9], the authors proposed to allow a tile in a group to be delivered in either the unicast or multicast mode to account for the diversity in user viewing behaviors.

In [24], a Multi-Session Multicasting system is leveraged to optimize 360-degree video delivery to multiple users under limited network resources. First, users are grouped into multicast groups based on their channel quality. Then, an optimization framework is proposed to 1) allocate the network resource and 2) determine the tile quality for ach individual group to maximize the overall utility. This method, however, neither considers the difference in viewing interests of individual users. In [6], a framework for live 360-degree video streaming is proposed. Users are classified based on network bandwidths into multiple classes. Each class corresponds to a multicast group and receives the same set of tiles. The tile quality of a class is determined to minimize the total distortion of all users in the class. [26] exploited the multicast opportunities to improve utility for delivering 360-degree videos to multiple users in FDMA networks. Those tiles requested by multiple users are multicasted. In [8], users watching the same 360-degree video are clustered based on the network condition (e.g., throughput), and the viewing pattern (e.g., FoV). Then, the delivery mode (unicast or multicast) and bitrates of tiles in each cluster is determined to maximize the overall QoE of all users. In [7], a cooperative multicast and unicast transmission scheme is proposed, in which a basic version of the video is transmitted to all users in a multicast session, and tiles of enhanced-versions are transmitted to each viewer in a unicast session. The main problem with the existing methods is that because the tiles are encoded into separated versions, similarities across versions can not be exploited, causing sub-optimal resource utilization.

III. PROPOSED METHOD

A. SYSTEM MODEL

The general architecture of the proposed system is shown in Fig. 2. A group of M users watching a live 360-degree video over a mobile network using Head-Mounted Displays. The live 360-degree video is captured from a live event such as a sport match or a music concert. Both the mobile network and the user equipment are assumed to support multicast transmission. In practice, 4G-LTE mobile networks support multicast transmission to multiple users via the Evolved Multimedia Broadcast Multicast Services (eMBMS) [27].

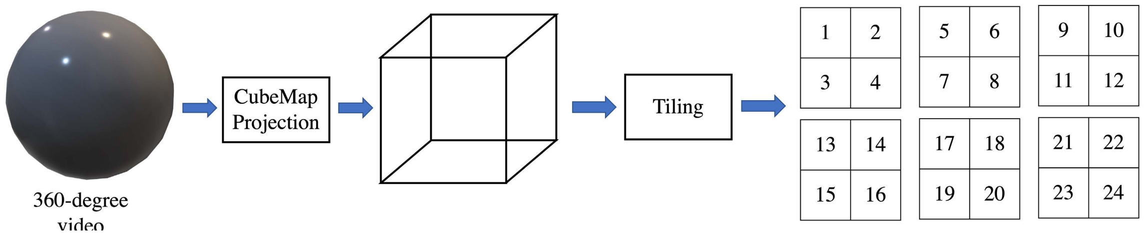


FIGURE 3. Content preparation process at the streaming server.

In addition, multicast services for 5G networks are also being developed [28].

The processing of the live 360-degree video at the streaming server is illustrated in Fig. 3. First, 360-degree video are captured in real-time using a 360 camera. In this paper, we focus on the transmission aspect of 360-degree video. Study on the capturing of 360-degree video is reserved for our future work.The live 360-degree video is then converted into a flat video using CubeMap projection [29]. The CubeMap projection projects a 360-degree video on to a cube consisting of six faces. The faces are then arranged to produce a rectangular video. Compared to the popular Equirectangular projection (ERP), the CubeMap projection can reduce the distortion in the video. Thus, it is considered as one of the best projection for 360-degree videos [14]. The converted video is spatially partitioned into N equal-sized and non-overlapping parts, called *tiles*. Here, the tiles are generated by partitioning six faces of the CubeMap projection into the same number of tiles. Each tile is encoded into multiple layers using Scalable Video Encoding [30]. Each layer represents a different quality representation of the tile. The base layer (BL) provides the lowest quality. The enhancement layers, when combining with the base layer and lower enhancement layers, can provide improved tile quality. It is important to note that, the base layer and all lower enhancement layers are needed to decode an enhancement layer. There are three main types of scalability, namely temporal, spatial, and quality. In our system, we employ the quality scalability where enhancement layers provide improved details or fidelity of the tile.

Our proposed system is mainly deployed at the Gateway, which connects to the Streaming Server via the core network and has two main tasks. The first task is to decide which tile layers to be delivered to individual users and their transmission modes so as to maximize all user's Quality of Experience at a given available network resource. For that purpose, the Gateway receives users' viewing direction information from the user equipment via a feedback channel. It also receives bitrate and quality information of the tile layers from the Streaming Server, as well as the spectral efficiency information of the users from the Base Station. The Gateway's second task is to perform the transmission of the required tile layers to users. Given the decisions on tile layers, the Gateway will first fetch the tile layers from the Streaming Server, then coordinates with the Base Station to perform tile delivery to the user equipment. Each user equipment's main task is to receive and decode the tile layers, then extracts and renders the viewport corresponding to the user's current viewing direction. In addition, it also monitors and transmits the user's viewing direction to the Gateway through the feedback channel.

B. RESOURCE ALLOCATION PROBLEM

In this part, we formulate the resource allocation problem in the live 360-degree video streaming over mobile networks. Assume that a group of users is allocated R network resource blocks. User m $(1 \leq m \leq M)$ is characterized by the spectral efficiency of the wireless link between the user equipment and the base station. The live 360-degree video consists of N tiles, each of which is encoded into $(K+1)$ layers. Layer k $(0 \leq k \leq K)$ of tile n $(1 \leq n \leq N)$ has a bitrate of B_{nk} and quality of Q_{nk}. Each layer is further divided into chunks with a playback duration of τ seconds. The number of network resource blocks required to send layer k of tile n to user m, denoted R_{mnk}, is given by,

$$R_{mnk} = \frac{B_{nk} \times \tau}{\sigma_m} \tag{1}$$

where σ_m denotes the spectral efficiency of the network connection between user m and the base station. A tile layer can be sent in two transmission modes: unicast and multicast. In the unicast mode, the tile layer is sent to a specific user. On the other hand, the tile layer is delivered to a set of users in the multicast mode. Besides, the capacity of multicast is limited by the user with the smallest spectral efficiency among those receiving multicast transmission. The cost to multicast layer k of tile n to a group of users G is calculated as follows.

$$R^G_{nk} = \frac{B_{nk} \times \tau}{\min_{m \in G} \sigma_m} \tag{2}$$

We use a set of binary variables $\{y_{mnk}\}$ to represent the selection of tile layers as follows.

$$y_{mnk} = \begin{cases} 1 & \text{layer k of tile n is transmitted to user m} \\ 0 & \text{otherwise} \end{cases}$$

If a tile layer is requested by more than one user, it will be transmitted in the multicast mode to all requested users.

Otherwise, the tile layer is unicasted to the specific user.

$$\sum_{m=1}^{M} y_{mnk} = \begin{cases} 1 & \text{layer } k \text{ of tile } n \text{ is unicasted to user } m \\ \text{otherwise} & \text{layer } k \text{ of tile } n \text{is multicasted to user } m \end{cases}$$

The resource allocation problem for multi-user 360-degree video streaming can be formulated as an optimization problem as follows.

Find the optimal values of $\{y_{mnk}\}$ *to maximize the overall Quality of Experience of all users.*

$$\max \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{k=0}^{K} w_{mn} Q_{nk} y_{mnk} \tag{3}$$

and to satisfy the following constraints:

$$\sum_{n=1}^{N}\sum_{k=0}^{K} \frac{B_{nk} \times \tau}{\min_{1\le m\le M \& y_{mnk}=1} \sigma_m} \le R \tag{4}$$

$$y_{mn0} = 1, \quad \forall m, \quad \forall n \tag{5}$$

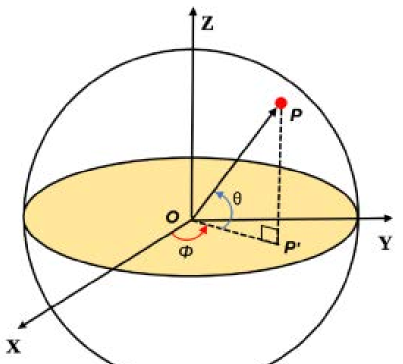
$$y_{mnk} \ge y_{mn(k+1)}, \ \forall k < K, \ \forall m, \ \forall n \tag{6}$$

The objective function in Eq. (3) measures the overall Quality of Experience (QoE) of all users in the group where w_{mn} is the weight of tile n for user m. Quality of Experience (QoE) measures the level of user satisfaction when watching 360-degree videos, and mainly depends on the image quality of the viewport area [4]. In our formulation, the QoE of a user is measured as the weighted sum of tiles overlapping the viewport. The first constraint of the problem is that the total resources blocks allocated to all tile layers (both multicast and unicast) must be less than or equal to the available resource blocks R. This constraint is expressed in Eq. (4) where τ is the playback duration of a video segment. We assume that each tile can be delivered only once over either the multicast or unicast mode, and each user always receives the base layer for every tile (i.e., Eq. (5)). The constraint expressed in Eq. (6) guarantees that if a user receives an enhancement layer k, it must also receive the base layer and all enhancement layers with lower quality. This is to ensure that the tile corresponding to layer k can be correctly decoded by the user equipment.

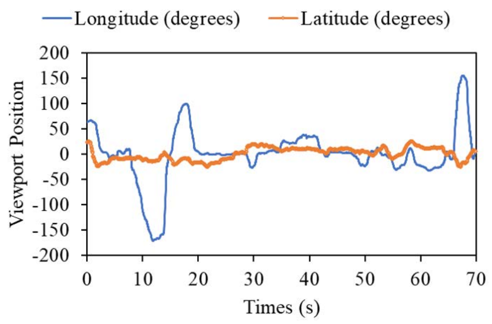
To solve the above resource allocation problem, one first needs to estimate individual users' future viewing direction (i.e., viewport) in order to obtain the weight values w_{mn}. Then, an algorithm for deciding the values of y_{mnk} must be devised. Since users usually change their viewing directions while watching a 360-degree video and the time-varying nature of mobile network condition, the algorithm must be quick to support real-time adaptation.

C. TILE WEIGHT ESTIMATION

The weight of a tile should reflect the importance of the tile for a given user. Due to the limited Field of View (FoV) of the VR headset, users can only see a portion of the full 360-degree video, called *viewport* [20]. Thus, we propose to



(a) Position of a point on 360-degree video.ϕ: Longitude; θ: Latitude



(b) Longitude (ϕ) and latitude (θ) values over time.

FIGURE 4. **User viewport position.**

calculate the weight of a tile based on how that tile overlaps the user viewport. In addition, since there exists a delay from when the decision is made to when the tile is displayed [5], the user viewport position needs to be estimated. Here, the position of a viewport refers to the position of the center point of the viewport, which is defined by two values: longitude (degrees) and latitude (degree) as shown in Fig. 4a. The longitude, denoted ϕ, is in $[-180, 180]$ range. The latitude, denoted θ, is in $[-90, 90]$ range. Viewport positions of a user watching a 360-degree video over time are shown in Fig. 4b.

Our proposed method for calculating the weights of tiles is described in Algorithm 1. In our method, the longitude and latitude are predicted separately, then combine to produce the estimated viewport position. For each user, the proposed algorithm first trains two predictors using past viewport information. The viewport information is sent from user equipment to the gateway via feedback channels. The viewport predictors are based on linear regression. Let t_{now} be the current time and W be the window size in seconds. The predictors h_{ϕ}^{m} and h_{θ}^{m} respectively predict the longitude and latitude of user m at time t as follows

$$h_{\phi}^{m}(t) = \alpha_{\phi} \times t + \beta_{\phi} \tag{7}$$

$$h_{\theta}^{m}(t) = \alpha_{\theta} \times t + \beta_{\theta} \tag{8}$$

Algorithm 1: *Tile Weight Estimation*

```
1  for $m = 1$ to $M$ do
2      Train the predictors $h_\phi^m$ and $h_\theta^m$;
3      Calculate the estimated viewport:
         $(\phi_m^e, \theta_m^e) = (h_\phi^m(t_{now} + D), h_\theta^m(t_{now} + D))$;
4      for $n = 1$ to $N$ do
5          Calculate $p_{mn}$, the number of pixels of tile $n$
             which are inside the estimated viewport;
6          $w_{mn} = 0$;
7          if $p_{mn} > 0$ then
8              $w_{mn} = \frac{p_{mn}}{S}$;
9          end
10     end
11 end
12 return $\{w_{mn}, 1 \le m \le M, 1 \le n \le N\}$;
```

Algorithm 2: *Tile Layer Allocation*

```
1  Assign the base layers of all tiles to all users $y_{mn0} = 0$
     $\forall m, \forall n$;
2  Calculate total number of allocated resource blocks $R^c$;
3  Calculate $UoC(m, n, k)$ using Eq. (12);
4  Sort $UoC(m, n, k)$ in descending order;
5  for $(m, n, k)$ in $UoC$ do
6      Assign layer 1 to layer $k$ of tile $n$ for user $m$:
         $y_{mnj} = 1, 1 \le j \le k$;
7      Update value of $R^c$;
8      if $R^c > R$ then
9          Reverse the decision at line 6;
10         break;
11     end
12 end
13 return $\{y_{mnk}, 1 \le m \le M, 1 \le n \le N, 0 \le k \le K\}$;
```

Here, $\alpha_\phi, \beta_\phi, \alpha_\theta$, and β_θ are trainable model parameters. The predictors are trained using the Ordinary Least Squares method with the training data $\phi^m(t)$ and $\theta^m(t)$ $(t_{now} - W \le t \le t_{now})$. Then, the estimated viewport position is calculated using the trained predictors (i.e., Line 3). Here, D denotes the delay from when the decision is made to when the playback of the tile starts. Next, the number of visible pixels of individual tiles given the predicted viewport is calculated. Finally, the weight of a tile is computed as the ratio of the visible pixels p_{mn} to the viewport size S (Lines 4-10). Since Linear Regression-based models can be trained quickly, the proposed algorithm can estimate the tiles' weights in real-time. It should be noted that our proposed method is general and can work with other viewport estimation methods.

D. RESOURCE ALLOCATION ALGORITHM

To solve the above resource allocation problem, we first re-formulate the problem into a Linear Programming problem, then present a efficient algorithm to find the optimal solution. Without loss of generality, we assume that the values of the spectral efficiency σ_m are sorted in ascending orders.

$$\sigma_1 \le \sigma_2 \le \cdots \le \sigma_M \tag{9}$$

Due to the property of multicast transmission, sending layer k of tile n to user m in the unicast mode consumes the same amount of resources as sending it to users $\{m, m+1, \ldots, M\}$ by the multicast mode. Therefore, if user m receives layer k of tile n, all users from user $(m+1)$ to user M will also receive the tile layer by the multicast mode, i.e.,

$$y_{mnk} \le y_{(m+1)nk}, \quad \forall n, \quad \forall k, \ m < M \tag{10}$$

With the new constraint of Eq. (10), the constraint of Eq. (4) can be re-written in linear form as follows.

$$\sum_{n=1}^{N} \sum_{k=0}^{K} (y_{1nk} \frac{B_{nk}}{\sigma_1} + \sum_{m=2}^{M} ((y_{mnk} - y_{(m-1)nk}) \frac{B_{nk}}{\sigma_m})) \le R \tag{11}$$

With Eq. (4) being replaced by Eq. (11), the resource allocation problem becomes a Binary Integer Programming (BIP) problem. We design an efficient algorithm to find a near-optimal solution for the problem, as shown in Algorithm 2. The basic idea is to assign more layers for a tile with higher utility over the cost ratio. The utility over the cost of layer k $(k \ge 1)$ of tile n of user m, denoted $UoC(m, n, k)$, is calculated as follows.

$$UoC(m, n, k) = \frac{(Q_{nk} - Q_{n(k-1)}) \times w_{mn}}{B_{nk}/\sigma_m} \tag{12}$$

The algorithm first assigns the base layer of all tiles to all users and calculates the number of allocated resource blocks R^c (i.e., Lines 1-2). Then, the utility over cost values of tile layers are calculated and sorted in descending order (Lines 3-4). The algorithm next assigns the tile layers to users, starting from the one with the highest UoC value. At each step, all layers lower than the considered layer k are also assigned to the user to ensure that the layer can be correctly decoded at the user device (Line 6). If the number of allocated resource blocks exceeds the value of R, then the considered layer will not be assigned to the user (i.e., Lines 7-11). The time complexity of the proposed Tile Layer Allocation algorithm is $O(M \times N \times K)$. In practice, the number of tiles N is typically less than 100 [13]. The number of layers (or quality versions) is less than 10 [22]. Also, the proposed algorithm does not require complex computation. Thus, it can be easily implemented and deployed in real-world systems. In our proposed system, Algorithm 1 and Algorithm 2 are both run on the Gateway.

IV. EVALUATION

A. EXPERIMENTAL SETTINGS

In the experiment, we use three 360-degree videos of Rollercoaster, Diving, and Venice taken from the dataset of [31]. Each video is 60-second long, and has a resolution of 3840×2048 in the Equirectangular (ERP) format. The videos are

TABLE 1. Quantization Parameters of scalable layers.

Scalable Layer	Quantization Parameter (QP)
Base Layer	38
Enhance. Layer #1	32
Enhance. Layer #2	28
Enhance. Layer #3	24
Enhance. Layer #4	20

TABLE 2. Average bitrate (kbps) and quality (dB) of the base layer the enhancement layers of three 360-degree videos used in the experiment. (For the proposed and **Multicast-all** methods).

	RollerCoaster		Diving		Venice	
Scalable Layer	Bitrate (kbps)	Quality (dB)	Bitrate (kbps)	Quality (dB)	Bitrate (kbps)	Quality (dB)
Base Layer	55.3	39.5	158.7	34.5	50.3	32.7
Enhance. Layer #1	106.7	42.7	313.8	38.2	114.3	35.9
Enhance. Layer #2	179.1	44.7	504.5	40.8	216.2	38.5
Enhance. Layer #3	389.1	47.1	1037.0	43.9	483.6	41.6
Enhance. Layer #4	632.9	49.2	1418.4	46.2	824.7	44.5

then converted into the CubeMap format using 360Lib software [32]. The converted videos have a resolution of 2890 × 1920, and are divided into 24 tiles, each has a resolution of 480 × 480. Individual tiles are encoded into a base layer and 4 enhancement layers ($K = 4$) using Scalable Video Encoding Extension of HEVC standard [30]. We employ the SNR-scalability mode where the Quantization Parameters (QP) of individual layers are fixed as in Table 1. Table 2 shows the average tile bitrate in kbit/s and tile quality in Peak Signal to Noise Ration (PSNR) of three videos used in the experiment. The number of users M is set to 15. As for the spectral efficiency of users, we use the setting values as in [9]. In particular, the σ_m values of the users are [0.02, 0.031, 0.05, 0.079, 0.116, 0.155, 0.195, 0.253, 0.318, 0.36, 0.439, 0.515, 0.597, 0.675, 0.733] with the minimum spectral efficiency σ of 0.02 (kbits/RB). The viewing direction over time of the users are also taken from the dataset of [31]. The segment duration τ and the prediction delay D are both set to 0.13 seconds. Our proposed method is implemented in Python and tested on a 64-bit Windows 10 PC with 8GB Memory and 3.8Ghz Intel core i7 CPU.

The proposed method is compared to three reference methods as follows.

- **VRCast** [25]: In this method, tiles are encoded into separate versions. Users in a same group receive a same set of tile versions via multicast. The tile versions are chosen to maximize the weighted product of tiles bitrates.
- **JUMPS** [9]: Similar to the **VRCast** method, a tile is also encoded into separate versions in this method. However, unlike the **VRCast** method, the **JUMP** method allows a tile to be delivered in either unicast or multicast.

TABLE 3. Average bitrate (kbps) and quality (dB) of tile versions of three 360-degree videos used in the experiments (for the **VRCast** and **JUMPS** methods).

	RollerCoaster		Diving		Venice	
Version	Bitrate (kbps)	Quality (dB)	Bitrate (kbps)	Quality (dB)	Bitrate (kbps)	Quality (dB)
Version #1	54.9	39.5	158.1	34.5	60.1	32.7
Version #2	131.7	42.9	393.6	38.2	183.0	36.1
Version #3	250.5	45.0	752.3	41.0	384.9	38.7
Version #4	515.9	47.3	1481.5	44.0	826.4	41.8
Version #5	933.8	49.4	2499.0	46.6	1520.5	44.8

The versions and delivery modes of tiles are decided to maximize the weighted sum of FoV bitrates of all users.

- **Multicast-all** [33]: This is a simpler version of our proposed method where all users are constrained to receive a same set of tile layers.

Unlike our proposed method, both the **VRCast** and **JUMPS** methods employ a different content preparation scheme where a tile is encoded into separated versions. To evaluate these methods, we encode tiles into 5 separate versions using HEVC encoder with the same Quantization Parameters (QP) values used to encode the layers in our method (i.e., Table 1). The bitrates and quality values of tile versions are shown in Table 3. We implement the reference methods based on the descriptions in the original papers. The performance of the proposed method is evaluated using two metrics: average viewport PSNR and processing time.

- **Average Viewport PSNR**: The average viewport PSNR is given by,

$$VPSNR = \frac{1}{M}\sum_{m=1}^{M} VPSNR_m. \tag{13}$$

where $VPSNR_m$ is the viewport PSNR of user m, and is calculated as a weighted sum of visible tiles' quality values as follows.

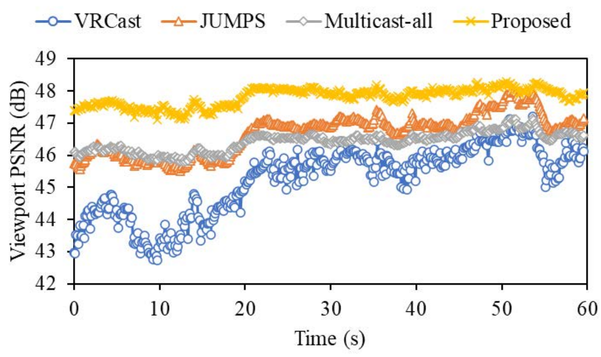
$$VPSNR_m = \sum_{n=1}^{N} w_{mn}^{*} \times Q_{nk_m^*} \tag{14}$$

with k_m^* is the highest layer tile n streamed to user m.

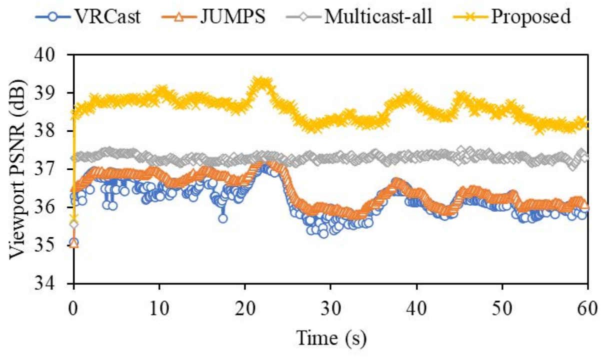
- **Processing time**: The time taken to decide the tile layers for users and their transmission modes. Essentially, this is the total processing time of Algorithm 1 and Algorithm 2.

B. EXPERIMENTAL RESULTS

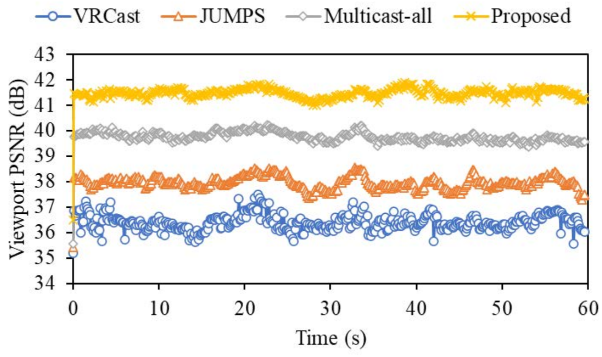
In this section, we investigate the performance of our proposed framework when compared to the reference method. Figure 5 shows the average viewport PSNR over time of the proposed and reference methods for three considered videos when $R = 250000$ (RBs). It can be seen that the proposed method always achieves the highest viewport PSNR for all three videos. In case of the RollerCoaster video, the proposed method achieves an viewport PSNR of 47.1 dB $\sim$ 48.3 dB. Compared to the reference methods, our method can increase the viewport PSNR by 0.2 dB $\sim$ 2.6 dB with the largest



(a) RollerCoaster Video



(b) Diving Video



(c) Venice Video

FIGURE 5. Average viewport PSNR over time of the proposed and reference methods when R=250000 (RBs).

improvement achieved during the first 20 seconds of the video playback. In case of the Diving video, the viewport PSNR provided by the proposed method is 35.7 dB ∼ 39.3 dB. It can be noted that the viewport PSNR in this case is lower than that of the RollerCoaster video. This is due to the fact that the bitrate of the Diving video is much higher than the bitrate of the RollerCoaster video, as can be seen in Table 2. Compared to the three reference methods, our method attains an increase of 0.8 dB ∼ 2.3 dB. Also, the performance of the **VRcast** and **JUMPS** methods are very similar in this case.

For the Venice video, the performance of all methods are quite stable over time, with the proposed method yields

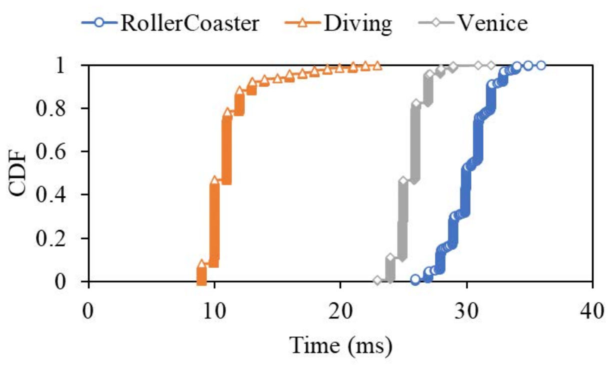
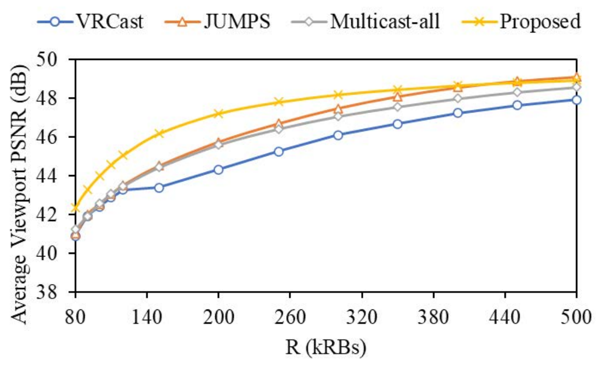


FIGURE 6. Cumulative distribution function (CDFs) of the processing time of the proposed method when R=250000 (RBs).

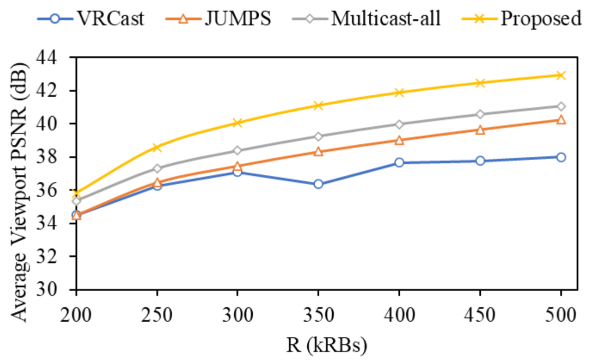
the best viewport PSNR. In addition, our method can significantly improve the viewport quality in comparison with the reference method. Especially, compared to the **VRCast** method, our method can increase the viewport PSNR by up to 5.1 dB. The cumulative distribution function (CDF) of the processing time of the proposed method is shown in Fig. 6. It can be seen that the proposed method takes less than 35 ms to decide the layers of the tiles for all users. This result indicates that our method is effective for real-time adaptation. It can also be noted that, the lower the video bitrate is, the higher the processing time would become. This is because that lower bitrate allows for more tile layers to be sent over the network, so that more feasible solutions are needed to be checked in our proposed Algorithm 2.

Next, we compare the performance of the proposed and the reference methods at different values of R up to 500000 (RBs). The results are shown in Fig. 7. Here, the minimum values of R of different videos are not the same because a video with higher bitrate (e.g., the Diving video) requires more resource for the lowest quality. It can be seen that the higher the value of R is, the better the performance of the proposed method becomes. For the RollerCoaster video, our method consistently outperforms the **VRCast** and **Multicast-all** methods at all considered values of R. In particular, our method can improve the viewport PSNR by 0.9 dB ∼ 2.9 dB compared to the **VRCast** method, and by 0.4 dB ∼ 1.8 dB compared to the **Multicast-all** method. The **JUMPS** method is slightly better than the **Multicast-all** method. For $R \geq$ 400000 (RBs), the **JUMPS** method has a similar performance to that of the proposed method.

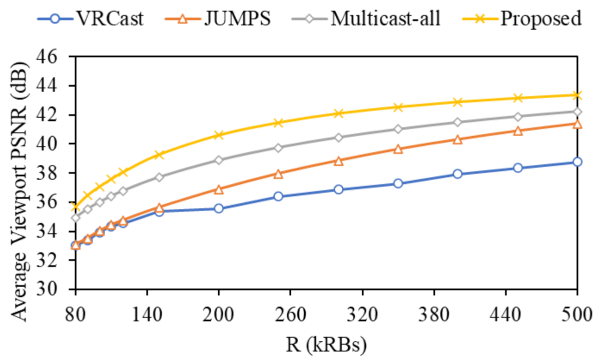
In case of the Diving and Venice videos (i.e., Fig. 7b and Fig. 7c), the proposed method always achieve higher viewport PSNR than that of three reference methods. For the Diving video, the improvement gain provided by our method is 1.3 dB ∼ 4.9 dB compared to the **VRCast** method, 1.3 dB ∼ 2.8 dB compared to the **JUMPS** method, and 0.4 dB ∼ 1.9 dB compared to the **Multicast-all** method. The improvement gain in case of the Venice video are highest among three considered videos. In particular, the proposed method improves the viewport PSNR by 2.6 dB ∼ 5.0 dB



(a) RollerCoaster Video

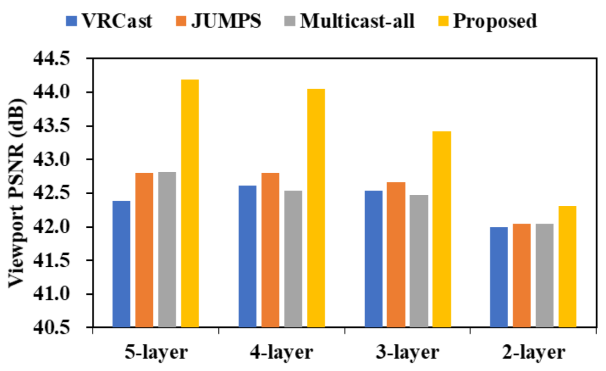


(b) Diving Video

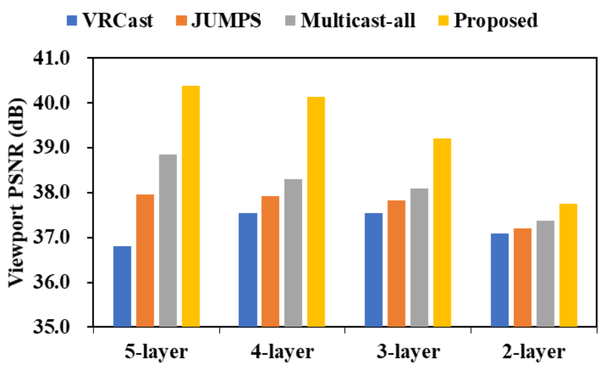


(c) Venice Video

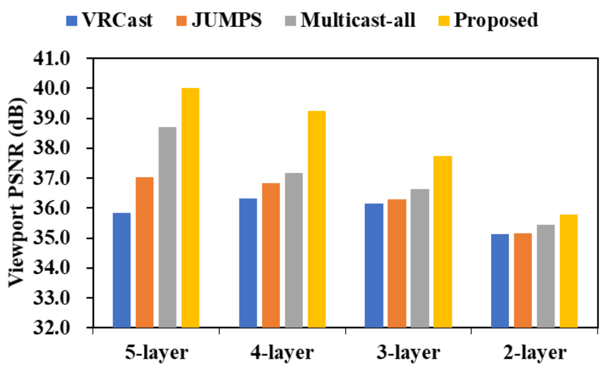
FIGURE 7. **Average viewport PSNR over time of the proposed and reference methods across different values of the number of resource blocks R.**



(a) RollerCoaster Video



(b) Diving Video



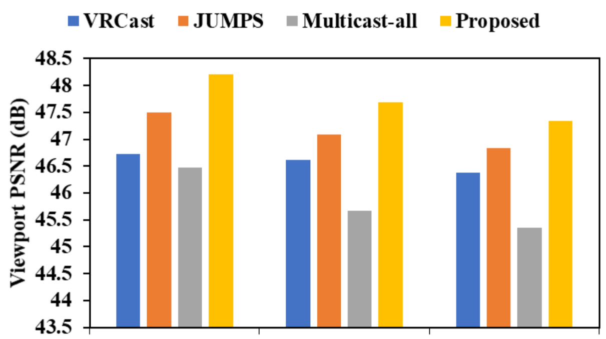
(c) Venice Video

FIGURE 8. **Performance of the proposed and reference methods across different number of scalable layers.**

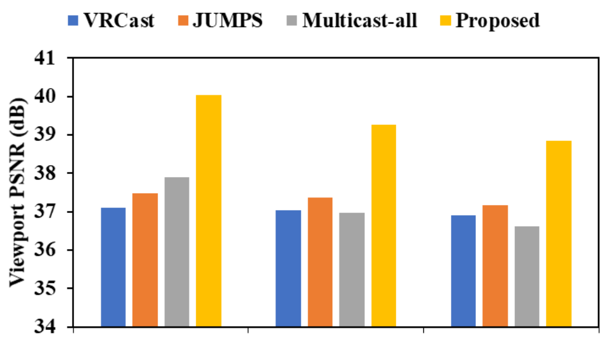
compared to **VRCast**, by 2.6 dB $\sim$ 3.7 dB compared to **JUMPS**, and by 0.7 dB $\sim$ 1.7 dB compared to **Multicast-all** method.

C. IMPACT OF NUMBER OF SCALABLE LAYERS

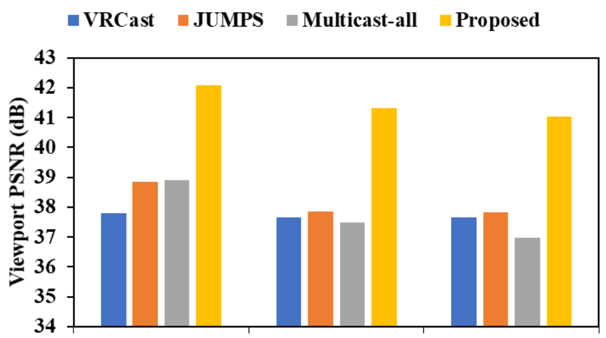
In this part, we evaluate the impact of number of scalable layers on the performance of the proposed method. For that purpose, we carried out experiments with four values of K, namely 2, 3, 4, 5. With VRcast and JUMPS methods, the number of versions for each tile is correspondingly set to 2, 3, 4, 5. The experiments have the same settings as that used in Section IV-B. For the RollerCoaster video, the highest number of resource blocks R is changed to 200000 (RBs). After obtaining the result at each value of R, the average performance across different R values are computed. The comparison results are shown in Fig. 8. For the Roller-Coaster video (i.e., Fig. 8a), it can be seen that the proposed method offers better viewport PSNR than that of the reference methods across different values of the number of scalable layers. The improvement gain provided by our method is 0.3 dB $\sim$ 1.8 dB. It can be noted that the proposed method achieve better viewport quality as more scalable layers are



(a) RollerCoaster Video



(b) Diving Video



(c) Venice Video

FIGURE 9. Performance of the proposed and reference methods across different number of users (R=300000 RBs).

available. The comparison result for the Diving and Venice videos are shown in Fig. 8b and Fig. 8c, respectively. These results indicate that the proposed method is effective in improving the viewport quality at different number of scalable layers.

D. IMPACT OF NUMBER OF USERS

In this part, we investigate the impact of the number of users M on the performance of the proposed methods. In particular, three values of M of 15, 30, and 45 are considered in our experiments. The viewport PSNR of the proposed and reference methods at $R = 300000$ RBs are shown in Fig. 9. We can see that our method consistently outperforms the reference methods across different values of M. As the number of the users increases, the viewport PSNR of the proposed method does decrease, but with a small margin. Compared to the performance at $M = 15$, the viewport PSNR decreases by less than 2 dB at $M = 45$. This can be explained that increase in the number of users would increase the chance that more users watching similar tiles. In such a case, multicast can deliver the tile to new users with a same amount of network resource.

V. CONCLUSION

In this paper, we propose a novel framework for multi-user VR video streaming over mobile networks by combining Scalable Video Coding and multicast to deliver popular tiles to users in a bandwidth-efficient manner. In the proposed framework, tiles are encoded into multiple layers using Scalable Video Coding, which are then delivered to users using multicast. Two online algorithms are proposed to decide the appropriate tile layers for individual users and their transmission modes for real-time adaptation. Experimental results show that the proposed method can significantly improve the average viewport quality compared to state-of-the-art methods. In future work, we plan to extend the proposed method to support the scenario where a massive number of users are watching a live 360-degree video at the same time. In addition, the impact of viewport prediction accuracy on the overall performance will be investigated.

REFERENCES

[1] *Virtual Reality Market Share Trends Report*, GV Research, Hyderabad, India, Mar. 2021.

[2] *360-Degree Video*. Accessed: Jan. 27, 2022. [Online]. Available: https://en.wikipedia.org/wiki/360-degree_video

[3] *Virtual Reality*. Accessed: Jan. 27, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Virtual-reality

[4] H. T. T. Tran, N. P. Ngoc, C. T. Pham, Y. J. Jung, and T. C. Thang, "A subjective study on QoE of 360 video for VR communication," in *Proc. IEEE 19th Int. Workshop Multimedia Signal Process. (MMSP)*, Luton, U.K., Dec. 2017, pp. 1–6.

[5] D. V. Nguyen, H. T. T. Tran, and T. C. Thang, "Impact of delays on 360-degree video communications," in *Proc. TRON Symp. (TRONSHOW)*, Tokyo, Japan, Dec. 2017, pp. 1–6.

[6] R. Aksu, J. Chakareski, and V. Swaminathan, "Viewport-driven rate-distortion optimized scalable live 360° video network multicast," in *Proc. ICME*, 2018, pp. 1–6.

[7] J. Yang, J. Luo, J. Wang, and S. Guo, "CMU-VP: Cooperative multicast and unicast with viewport prediction for VR video streaming in 5G H-CRAN," *IEEE Access*, vol. 7, pp. 134187–134197, 2019.

[8] N. Kan, C. Liu, J. Zou, C. Li, and H. Xiong, "A server-side optimized hybrid multicast-unicast strategy for multi-user adaptive 360-degree video streaming," in *Proc. ICIP*, Taiwan, Taipei, 2019, pp. 141–145.

[9] A. Majidi and A. H. Zahran, "Optimized joint unicast-multicast panoramic video streaming in cellular networks," in *Proc. IEEE 28th Int. Conf. Netw. Protocols (ICNP)*, Madrid, Spain, Oct. 2020, pp. 1–6.

[10] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 video streaming using tile-based HTTP adaptive streaming," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 315–323.

[11] M. Graf, C. Timmerer, and C. Mueller, "Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP," in *Proc. 8th ACM Multimedia Syst. Conf.*, Taipei, Taiwan, Jun. 2017, pp. 261–271.

[12] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Rubiks: Practical 360-degree streaming for smartphones," in *Proc. 16th Annu. Int. Conf. Mobile Syst., Appl., Services*, New York, NY, USA, 2018, pp. 482–494, doi: 10.1145/3210240.3210323.

[13] D. V. Nguyen, H. T. T. Tran, A. T. Pham, and T. C. Thang, "An optimal tile-based approach for viewport-adaptive 360-degree video streaming," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 29–42, Mar. 2019.

[14] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. Int. Conf. Commun. (ICC)*, 2017, pp. 1–7.

[15] M. Hosseini and V. Swaminathan, "Adaptive 360 VR video streaming: Divide and conquer," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, Dec. 2016, pp. 107–110.

[16] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proc. 5th Workshop All Things Cellular, Oper., Appl. Challenges*, New York, Oct. 2016, pp. 1–6.

[17] D. V. Nguyen, H. T. T. Tran, and T. C. Thang, "An evaluation of tile selection methods for viewport-adaptive streaming of 360-degree video," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 16, no. 1, pp. 1–24, Feb. 2020, doi: 10.1145/3373359.

[18] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang, "CUB360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2018, pp. 1–6.

[19] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360° video streaming in head-mounted virtual reality," in *Proc. 27th Workshop Netw. Oper. Syst. Support Digit. Audio Video*, Taipei, Taiwan, 2017, pp. 67–72, doi: 10.1145/3083165.3083180.

[20] D. V. Nguyen, H. Van Trung, H. L. Dieu Huong, T. T. Huong, N. P. Ngoc, and T. C. Thang, "The permeability characteristics of silicone rubber," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process. (MMSP)*, Kuala Lumpur, Malaysia, Mar. 2019, pp. 1–6.

[21] A. T. Nasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, "Adaptive 360-degree video streaming using scalable video coding," in *Proc. 25th ACM Int. Conf. Multimedia*, New York, NY, USA, 2017, pp. 1689–1697, doi: 10.1145/3123266.3123414.

[22] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for HTTP live streaming," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 693–705, Dec. 2014.

[23] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu, "Motion-prediction-based multicast for 360-degree video transmissions," in *Proc. 14th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2017, pp. 1–9.

[24] J. Park, J.-N. Hwang, and H.-Y. Wei, "Cross-layer optimization for VR video multicast systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 206–212.

[25] O. Eltobgy, O. Arafa, and M. Hefeeda, "Mobile streaming of live 360-degree videos," *IEEE Trans. Multimedia*, vol. 22, no. 12, pp. 3139–3152, Dec. 2020.

[26] K. Long, C. Ye, Y. Cui, and Z. Liu, "Optimal multi-quality multicast for 360 virtual reality video," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[27] D. Lecompte and F. Gabin, "Evolved multimedia broadcast/multicast service (eMBMS) in LTE-advanced: Overview and Rel-11 enhancements," *IEEE Commun. Mag.*, vol. 50, no. 11, pp. 68–74, Nov. 2012.

[28] V. K. Shrivastava, S. Baek, and Y. Baek, "5G evolution for multicast and broadcast services in 3GPP release 17," *TechRxiv*, Jul. 2021, doi: 10.36227/techrxiv.14985354.v1.

[29] Y. Ye, E. Alshima, and J. Boyce, *Algorithm Descriptions of Projection Format Conversion and Video Quality Metrics in 360lib*, document Jvet-e1003, Joint Video Exploration Team, 2017.

[30] J. M. Boyce, Y. Yan, J. Chen, and A. K. Ramasubramonian, "Overview of SHVC: Scalable extensions of the high efficiency video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 20–34, Jan. 2016.

[31] X. Corbillon, F. De Simone, and G. Simon, "360-degree video head movement dataset," in *Proc. 8th ACM Multimedia Syst. Conf.*, Taipei, Taiwan, Jun. 2017, pp. 199–204.

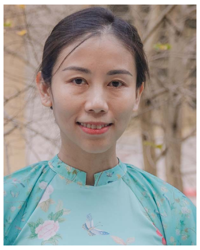
[32] *360lib Software*. Accessed: Jan. 27, 2022. [Online]. Available: https://jvet.hhi.fraunhofer.de/svn/svn-360Lib

[33] D. Nguyen, N. V. Hung, T. T. Huong, and T. C. Thang, "A cross-layer framework for multi-user 360-degree video streaming over cellular networks," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2022, pp. 1–3.

DUC NGUYEN (Member, IEEE) received the M.E. and Ph.D. degrees in computer science and engineering from The University of Aizu, Japan, in 2016, and 2019, respectively. From 2019 to 2021, he worked as a Researcher with KDDI Research Inc., Japan. Since 2021, he has been a Lecturer with the Tohoku Institute of Technology, Sendai, Japan. His research interests include multimedia communications, virtual reality/augmented reality, and machine learning.

NGUYEN VIET HUNG received the B.Sc. (Bachelor of Informatics) degree in pedagogy from the Faculty of Engineering Technology, Ha Tinh University, Vietnam, in 2008, and the M.Sc. (Master of Information Technology) degree from the Faculty of Information Technology, Ho Chi Minh City University of Technology, Vietnam, in 2016. He is currently pursuing the Ph.D. degree in telecommunications engineering with the Hanoi University of Science and Technology, Vietnam. His research interests include multimedia communications, network security, artificial intelligence, traffic engineering in next-generation networks, QoE/QoS guarantee for network services, green networking, and applications.

NGUYEN TIEN PHONG is currently pursuing the bachelor's degree with the Hanoi University of Science and Technology. He is also a Research Assistant with the Future Internet Laboratory, School of Electronics and Telecommunications. His research interests include video streaming and multimedia processing.



TRUONG THU HUONG (Member, IEEE) received the B.Sc. degree in electronics and telecommunications from the Hanoi University of Science and Technology (HUST), Vietnam, in 2001, the M.Sc. degree in information and communication systems from the Hamburg University of Technology, Germany, in 2004, and the Ph.D. degree in telecommunications from the University of Trento, Italy, in 2007. Her research interests include oriented toward network security, artificial intelligence, traffic engineering in next generation networks, QoE/QoS guarantee for network services, green networking, and development of the Internet of Things ecosystems and applications.

TRUONG CONG THANG (Senior Member, IEEE) received the B.E. degree from the Hanoi University of Science and Technology, Vietnam, in 1997, and the Ph.D. degree from the KAIST, South Korea, in 2006. From 1997 to 2000, he worked as a Network Engineer with Vietnam Post and Telecom (VNPT). From 2007 to 2011, he was a member of Research Staff with the Electronics and Telecommunications Research Institute (ETRI), South Korea. Since 2011, he has been an Associate Professor with The University of Aizu, Japan. His research interests include multimedia networking, image/video processing, content adaptation, IPTV, and MPEG/ITU standards. He was also an Active Member of Korean and Japanese delegations to standard meetings of ISO/IEC and ITU-T, from 2002 to 2014.