A View Synthesis-Based 360° VR Caching System Over MEC-Enabled C-RAN

Jianmei Dai, *Student Member, IEEE*, Zhilong Zhang, Shiwen Mao, *Fellow, IEEE*, and Danpu Liu, *Senior Member, IEEE*

***Abstract*— With the development of virtual reality (VR) technology, the future of VR systems is evolving from single-user wired connections to multi-user wireless connections. However, wireless online rendering and transmission incur extra processing and transmission latency, as well as higher bandwidth requirements. To meet the requirements of wireless VR applications and enhance the quality of the VR user experience, this paper designs a view synthesis-based 360° VR caching system over Cloud Radio Access Network (C-RAN), where both mobile edge computing (MEC) and hierarchical caching are supported. In the system, an MEC-Cache Server is deployed in the pooled Base band Units (BBU pool) and used for view synthesis and caching. In addition, the remote radio heads (RRHs) can also cache some video contents. If the requested content of a specific view is cached in the BBU pool or RRHs, or can be synthesized with the aid of the cached adjacent views, it is unnecessary to request the content from the remote VR video source server. Therefore, the transmission latency and backhaul traffic load for VR services can be decreased. We formulate a hierarchical collaborative caching problem aiming to minimize the transmission latency, which is proved NP-hard. To address the impractical expenses of the offline optimal method, an online MaxMinDistance caching algorithm with low complexity is proposed. Numerical simulation results demonstrate that the proposed caching strategy provides significantly improved cache hit rate, backhaul traffic load, transmission latency, and Quality of Experience (QoE) performances relative to conventional caching strategies.**

***Index Terms*— Virtual reality (VR), hierarchical caching, view synthesis, MEC, C-RAN, quality of experience (QoE).**

I. Introduction

VIRTUAL reality (VR) is a human computer interface technology that enables users to interact with each other in the virtual environment with three-dimensional spatial information [1]. With their rapid development in recent years, VR technologies have attracted much attention in many fields, ranging from education and military training to entertainment. A recent market report forecasts that the data consumption from mobile VR devices (smartphone-based or standalone) will grow by over 650% between 2017 to 2021 [2], [3]. 360° video is an integral part of VR. As a user can freely change his/her viewing direction while watching, it can provide panoramic and immerse experience. Nevertheless, wireless 360° video delivering incurs 4-5 times higher bandwidth requirements than that of traditional videos. Research by Huawei ilab shows that a general 360° VR video data rate with 4K resolution is 50 Mbps, and the data rate with 8K resolution increases to 200 Mbps [4]. Therefore, with the rapid increase in the number of VR headsets (wireless VR headsets are going to increase to 50 million by 2021) [2], the communication network can potentially become a bottleneck.

Some VR solutions use user's field of view (FoV) streaming to reduce bandwidth consumption. The FoV of a user is defined as the portion of the 360° video that is in the user's line of sight, and a User FoV can be spatially divided into small parts called tiles, each is encoded into multiple versions of different quality levels [5]–[8]. Bandwidth consumption can be reduced by sending tiles in User Fov only in high resolution, while other tiles are sent in low resolution or not at all [9].

While FoV adaptive 360° video streaming is useful for reducing bandwidth requirements, 360 video streaming from remote content servers is still challenging due to network latency. The Latency restriction is critical for VR services. Many studies indicate that the motion-to-photon (MTP) latency for VR should be less than 20 ms; otherwise, the user will feel dizzy. To alleviate the transmission latency, an efficient approach is caching popular VR contents at the edge of the network, such as RRHs and base stations. The existing literature has studied a number of problems related to caching in VR systems [10]–[13]. However, these caching schemes do not take the view synthesis character into consideration. View synthesis is a feature of multi-view video. A multi-view video is generated by capturing a scene of interest with multiple cameras from different angles simultaneously. A view is provided by one camera capturing both texture maps (i.e., images) and depth maps (i.e., distances from objects in the scene). Many methods [14]–[17] can be used for view synthesis, for example, Depth-Image-Based Rendering (DIBR) [14] technique, which is the most widely used method, can synthetically generate free-viewpoint video by using a reference 2D video and its associated depth map.

View synthesis is not only a common way to generate free-viewpoint video from a limited number of views, but also an effective method for predictive coding in multi-view video compression [18]–[20] and can achieve good performance.

Manuscript received May 12, 2019; revised September 19, 2019; accepted October 6, 2019. Date of publication October 11, 2019; date of current version October 2, 2020. This work was supported in part by the Beijing Natural Science Foundation under Grant L172032, in part by the National Natural Science Foundation of China under Grant 61971069 and Grant 61801051, and in part by the NSF under Grant IIP-1822055. This article was recommended by Associate Editor Z. Chen. *(Corresponding author: Danpu Liu.)*

J. Dai, Z. Zhang, and D. Liu are with the Beijing Laboratory of Advanced Information Network, Beijing University of Posts and Telecommunications, China, and also with the Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, China (e-mail: jammy_d-ane@bupt.edu.cn; zhilong.zhang@outlook.com; dpliu@bupt.edu.cn).

S. Mao is with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849-5201 USA (e-mail: smao@ieee.org).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2019.2946755

Moreover, view synthesis can be utilized in VR systems to generate corresponding views according to the viewpoint of users [21]. Indeed, the user's current desired FoV can be synthesized by the previous requested nearby "left and right" or "up and down" FoVs in the 360° VR video streaming, because the adjacent FoVs usually share many similar parts. Based on this, if a required FoV which can be used for synthesizing more incoming FoVs is cached, the user requests can be largely satisfied by transmitting only a part of FoVs (correspondingly, a part of tiles) from the source. Therefore, we propose a new 360° VR system over C-RAN, where both mobile edge computing (MEC) and hierarchical caching are supported. In the system, an MEC-Cache Server is deployed in the BBU pool and used for view synthesis and caching. In addition, the RRHs can also cache some video content. If the requested view is cached in the BBU pool or the RRHs, or can be synthesized with the aid of the adjacent cached views, it is unnecessary to request contents from the remote VR video source server. Therefore, the transmission latency and backhaul traffic load for 360° VR services can be decreased, and the energy consumption on mobile phones is significantly relaxed. Different from [22], *i) in the proposed VR system, the video data do not need to be pre-fetched, so there are no additional remote access cost and local transmission cost*; *ii) the caching is hierarchical and cooperative, which is more suitable for the C-RAN architecture* and *iii) The view synthesis is done by MEC-Cache server in the BBU pool, due to the ample computing resource, the processing latency is less than the smartphones, which can significantly increase the QoE of VR users*.

Furthermore, to fully exploit the benefits of the proposed view synthesis-based 360° VR caching system, several challenges need be addressed. First, view synthesis is a computationally intensive task. The concurrent video synthesis could quickly exhaust the available processing resources of the MEC-Cache Server. Therefore, an efficient cache scheme needs to be designed for the given processing resources. Second, caching multiple views of video incurs high overhead in storage. Although hard disks are now very cheap, storing all of these files is neither economical nor feasible. Finally, the impact of caching data at the BBU pool and at different RRHs should be quantified, and the questions of what contents and where to be placed should be addressed.

In summary, the novelty and technical contributions of this work are as follows.

- We propose a view synthesis-based 360° VR caching system over C-RAN. An MEC-Cache server is deployed in the BBU pool for video synthesis and caching, and the view synthesis feature of 360° VR videos is considered in the caching algorithm.
- We formulate a hierarchical collaborative caching problem as an integer linear program (ILP), which aims to minimize the transmission latency subject to the cache storage and computing capacity constraints.
- We prove the NP-hardness of the formulated problem, and propose a MaxMinDistance online caching algorithm to address the NP completeness of the problem and the impractical expenses of the offline optimal method.
- Numerical simulation results demonstrate that the proposed MaxMinDistance strategy provides significantly improved cache hit rate, backhaul traffic load, transmission latency, and QoE performances relative to conventional caching strategies.

The remainder of this paper is structured as follows. Section II discusses the related work. Section III presents the system model. The hierarchical collaborative caching problem to minimize the average transmission latency is formulated in Section IV. In Section V, the flow of the whole caching process and MaxMinDistance algorithm are discussed. In Section VI, the performance evaluations are illustrated, and finally, in Section VII, we conclude the paper and discuss future research directions for caching about VR video over C-RAN.

II. Related Work

A. VR Transmission Solutions

To improve transmission efficiency of a 360 VR video, many solutions have been proposed by adopting tiling and multicast technologies [23]–[28]. In [23] and [24], the authors studied two optimal multicast transmission schemes for tiled 360° VR video. One is to maximize the received video quality in orthogonal frequency division multiple access (OFDMA) systems by optimizing subcarrier, transmission power and transmission rate allocation, and the other is to minimize average transmission energy by optimizing transmission time and power allocation. The view synthesis multicast is further analyzed in [25]. In [26], the authors optimized the VR video quality level selection, transmission time allocation and transmission power allocation to maximize the total utility under the transmission time and power allocation constraints as well as the quality smoothness constraints for mixed-quality tiles. In [27], the authors proposed a multicast DASH-based tiled streaming solution, including a user's viewports based tile weighting approach and a rate adaptation algorithm, to provide an immersive experience for VR users. In [28], the authors leveraged a probabilistic approach to pre-fetch tiles countering viewport prediction error, and designed a QoE-driven viewport adaptation system, which can achieve a high viewport PSNR. In [29], the problem of resource management was studied for VR application in drone-UEs network. By taking the image quality and format in resource management, the QoE performance of VR is improved.

In this paper, to clarify the significance of caching in wireless VR transmission, while a multi-view 360° VR video to multiple users is considered and the tiling and multicast technologies naturally can be used, we do not focus on the design of multicast scheme and mainly consider the transmission optimization at view level.

B. VR Caching Algorithms

A number of studies have examined the problems related to caching in VR systems, such as [11]–[13], [30]–[35]. In [11], the authors propose a new approach for cached content replacement that allows for transmission delay optimization and design an optimization framework that allows the

base stations to select cooperative caching/rendering/streaming strategies that maximize the aggregate reward they earn when serving the users for the given caching/computational resources at each base station. The authors in [12] study content caching and transmission in a unmanned aerial vehicle (UAV) wireless virtual reality (VR) network and propose a distributed deep learning algorithm that brings together new neural network ideas from a liquid state machine (LSM) and echo state networks (ESNs) to solve the joint content caching and transmission problem. The authors in [13] propose a proactive computing and mmWave communication system for ultra-reliable and low latency wireless virtual reality. By leveraging information about users, proactive computing and caching are used to pre-compute and store HD video frames to minimize the computing latency. The authors in [30] propose a content caching method for three dimensional VR images, which is used to speed up any kind of rasterized rendering on a graphics workstation that supports hardware texture mapping. In [31], the authors introduce the challenges and benefits of caching in wireless VR networks and provide a relaxed analytical treatment of caching, relying on simple toy examples. In [32], the authors provide the specific challenges and opportunities related to caching and VR techniques. In [33], the authors present a novel MEC-based mobile VR delivery framework that is able to cache parts of the field of views (FOVs) in advance and run certain post-processing procedures at the mobile VR device. In [34], the authors proposed an FoV-aware caching policy based on learned probabilistic user request model of common-FoV, which improved cache hit ratio compared to classic caching policies by at least 40%. The authors in [35] designed a network function virtualization (NFV)-based virtual cache (vCache) to dynamically manage video chunks, which strikes a tradeoff between storage and computing costs, and it can reduce the operational costs of ABR streaming.

However, above caching schemes do not take the view synthesis character of 360° VR videos into consideration, and could not get a high performance gain compare to traditional caching schemes. In addition, the caching schemes have not reflected the specific features of C-RAN and can not be used in C-RAN. In this paper, to obtain an enhanced latency performance of 360° VR transmission, we proposed an hierarchical and cooperative caching scheme, which introduces the view synthesis feature.

III. System Model

A. Cache Model

The view synthesis-based 360° VR caching system over C-RAN is shown in Fig.1, which consists of one BBU pool and a set $\mathcal{R} = \{1, 2, \ldots, r, \ldots, R\}$ of R RRHs connected to the BBU pool via low-latency, high-bandwidth fronthaul links.

An MEC-Cache server is deployed at the BBU pool, providing computing, synthesizing, caching and networking capabilities to support context-aware and delay-sensitive applications in close proximity to the users. The cache storage of the MEC-Cache server is denoted by C_b (with a capacity of C_B bytes). An edge-cache is deployed in each RRH, which is

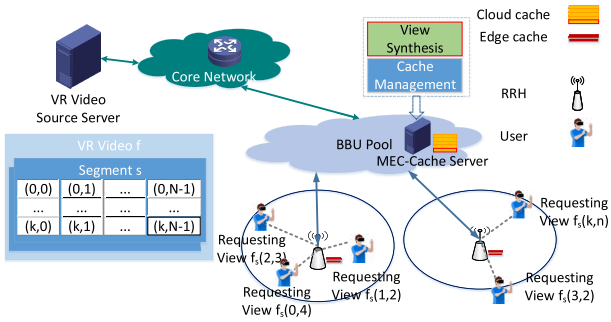


Fig. 1. Illustration of the view synthesis-based 360° VR caching system over Cloud Radio Access Network (C-RAN), including pooled baseband units (the BBU pool) with an MEC-Cache Server to synthesize the views and manage the cloud cache, remote radio heads (RRHs) with their individual edge cache, and users requesting different views. The requested data can be fetched directly from the edge cache/the cloud cache/the VR video source server, or can be synthesized by the MEC-Cache Server.

denoted by C_r. The capacity of C_r is C_R bytes, where C_R is usually much less than C_B.

A set $\mathcal{F} = \{0, 1, 2, \ldots, f, \ldots, F-1\}$ of F 360° VR video files are stored in the VR video source server, which can be transmitted and cached in the C-RAN network. For each VR video f, it is composed a set $\mathcal{S} = \{f_0, f_1, f_2, \ldots, f_s, \ldots, f_{S-1}\}$ of S continuous segments, and each segment f_s can be split to $K \times N$ overlapped views in set $\{\mathcal{K}, \mathcal{N}\}$, and one view can be denoted by $f_s(k, n)$, where $k \in \mathcal{K}, n \in \mathcal{N}$. $f_s(k, n)$ can be synthesized by using its left view $f_s(k-1, n)$ and its right view $f_s(k+1, n)$, or its up view $f_s(k, n-1)$ and its down view $f_s(k, n+1)$, as the reference views by the MEC-Cache server, since the adjacent views are overlapped and share many similar parts. The quality of each synthesized view depends on its distance to its two reference views and the qualities of its reference views. The set of all views that a user can request is $\Psi = \{f_s(k, n) | f \in \mathcal{F}, s \in \mathcal{S}, k \in \mathcal{K}, n \in \mathcal{N}\}$ and the angle of user's FoV is the same as the angle of one view. Further, considering different resolutions and versions of VR videos, the size of $f_s(k, n)$ can be different, and is denoted by $SIZE_{f_s(k,n)}$.

To enable the flexible transmission of views and improve the transmission efficiency of the 360° VR video, the tiling and multicast technologies can be used [23], [24], [26]. However, due to the length of the article, the transmission optimization and the content caching are all at the view level.

We consider that video requests arriving at each RRH following a Poisson process with rate $\delta_r, r \in \mathcal{R}$. The caching design is evaluated in a long time period to accumulate a large number of request arrivals. The set of new request arriving at RRH r in the considered time period is denoted as $\Omega_r \subseteq \Psi$.

A set $\mathcal{U} = \{1, 2, \ldots, u, \ldots, U\}$ of U users are served by the RRHs. Basically, one user only connects to one nearest RRH (in terms of signal strength) at the same time, which is later referred to as the user's associated RRH. The data can be fetched either from the associated RRH cache or from the BBU pool, to offload the traffic and reduce the transmission delay of both the fronthaul and backhaul. If the required view is not cached, it can be synthesized by the MEC-Cache Server

TABLE I
NOTATION

Symbol	Description
$\mathcal{R}$, R, r	RRH set, total number of RRHs, the r-th RRH
$\mathcal{F}$, F, f	Video library, total number of videos, the f-th VR video file
$\mathcal{K}$, K	the view set and view numbers in each row of one segment
$\mathcal{S}$, S	Segment Set, total number of segments in a file
$\mathcal{U}$, U, u	User set, total number of users, the u-th user
$\mathcal{N}$, N	the view set and view numbers in each column of one segment
f_s	the s-th segment in video f
$f_s(k,n)$	the (k,n)-th view of s-th segment in video f
$SIZE_{f_s(k,n)}$	the data size of one view
C_b, C_B	the cache of BBU pool and its capacity
C_r, C_R	the cache of the r-th RRH and its capacity
$c_r^{f_s(k,n)}$, $c_b^{f_s(k,n)}$	0-1 variables, the view cache status in the RRH and BBU pool
$B_{r,u}$, $\gamma_{r,u}$	the bandwidth and the average SINR ratio of user u in its associated RRH r
Ω_r	the set of new request arriving at RRH r in the considered time period
req_r	the proportion of the view synthesis task requested in RRH r
δ_r	the user requests arriving rate at RRH r
Λ_f	the video file popularity
V_B, V_O	the total data rate of fronthaul and backhaul
Ψ	the set of all views that a user can request
ξ	service rates of MEC-Cache server

since the left and right views are cached in the BBU pool. Furthermore, considering that many users mainly download and watch videos with little data uploading to the VR video source server, and the uplink of the fronthaul is idle most of the time, the requested view data can be obtained and synthesized from other unassociated RRH caches. This method can reduce the consumption of backhaul resource much further. Otherwise, the users should obtain the requested view data from the video VR source server.

B. Basic Transmission Latency Model

$B_{r,u}$ and $\gamma_{r,u}$ denote the bandwidth and the average signal-to-interference-plus-noise ratio of user u in its associated RRH r. V_B and V_O represent the total data rate of fronthaul (between RRHs and the BBU pool) and backhaul (between the BBU pool and source server), respectively. V_B and V_O are known beforehand, and $B_{r,u}$ and $\gamma_{r,u}$ can be estimated by the BBU pool. Without loss of generality, the fronthaul and backhaul transmission resources are shared equally by the U users.

Inspired by [36], [37] and [38], we denote by t_O, t_B and t_R the average latency incurred when transferring 1 bit from the origin server to the BBU pool cache, from the BBU pool cache to the RRH via fronthaul link, and from the RRH cache to the user, respectively. In practice, t_O and t_B are usually much greater than t_R [36], [37]. The definitions of them are as follows.

$$t_R = \frac{1}{B_{r,u} \cdot \log_2(1+\gamma_{r,u})} \tag{1}$$

$$t_B = \frac{1}{V_B/U} \tag{2}$$

$$t_O = \frac{1}{V_O/U} \tag{3}$$

C. Computational Latency Model of View Synthesis

We define req_r as the proportion of the view synthesis task requested in RRH r, and the service rates of MEC-Cache server is defined as ξ. According to the queueing theory, we can calculate the computation delay generated by the MEC-Cache server as following [39]. where $\sum_{i=1}^{R}\delta_r$ is the amount of view synthesis task on MEC-Cache server, $\frac{1}{\xi-\sum_{i=1}^{R}\delta_r}$ is the average execution delay of each task at MEC-Cache server with $\xi - \sum_{i=1}^{R}\delta_r > 0$.

$$t_{cmp} = \frac{\sum_{i=1}^{R} req_r\delta_r}{\xi - \sum_{i=1}^{R}\delta_r} \tag{4}$$

If the MEC-cache server can synthesize a requested view, the transmission delay t_{cs} between the user and the MEC-cache server is expressed as

$$t_{cs} = t_B + t_R + t_{cmp} \tag{5}$$

To ensure that users receive the requested view in a timely manner, t_{cs} should be lower than the transmission latency from the source server, thus:

$$t_{cs} \le t_O \tag{6}$$

IV. PROBLEM FORMULATION

Fig.2 illustrates seven possible (exclusive) events that happen when a user is requesting video view data. We introduce $x_{r,u}^{f_s(k,n)}$, $x_{b,u}^{f_s(k,n)}$, $y_{b,u}^{f_s(k,n)}$, $x_{0,u}^{f_s(k,n)}$, $y_{r,u}^{f_s(k,n)}$, $z_{b,u}^{f_s(k,n)}$, and $z_{b',u}^{f_s(k,n)}$ to describe these seven possible events, where

$$x_{0,u}^{f_s(k,n)}, x_{r,u}^{f_s(k,n)}, y_{r,u}^{f_s(k,n)}, x_{b,u}^{f_s(k,n)}, y_{b,u}^{f_s(k,n)}, z_{b,u}^{f_s(k,n)}, z_{b',u}^{f_s(k,n)} \in \{0,1\} \tag{7}$$

- $x_{r,u}^{f_s(k,n)} = 1$ indicates that the desired view data can be accessed in the associate RRH cache directly (Fig.2(a)), and $x_{r,u}^{f_s(k,n)} = 0$ otherwise. We denote by D_{rcache} the latency of downloading the required video data $f_s(k,n)$ from RRH r by user u, which equals $x_{r,u}^{f_s(k,n)} \cdot SIZE_{f_s(k,n)} \cdot t_R$.
- $x_{b,u}^{f_s(k,n)} = 1$ indicates that the user can access the required view data from the BBU pool cache (Fig.2(b)), and $x_{b,u}^{f_s(k,n)} = 0$ otherwise. D_{bcache} denotes the latency of downloading the required video data from the BBU pool, which equals $x_{b,u}^{f_s(k,n)} \cdot SIZE_{f_s(k,n)} \cdot (t_B + t_R)$.
- $y_{b,u}^{f_s(k,n)} = 1$ indicates that the user can access the synthesized view data from the BBU pool (Fig.2(c)), as only the left and right view data are cached in the BBU pool, and $y_{b,u}^{f_s(k,n)} = 0$ otherwise. The latency of downloading the synthesized video data from the BBU pool is defined as D_{synB}, which equals $y_{b,u}^{f_s(k,n)} \cdot SIZE_{f_s(k,n)} \cdot t_{cs}$.
- $x_{0,u}^{f_s(k,n)} = 1$ indicates that the user can access the required view data only from the source (Fig.2(d)), and

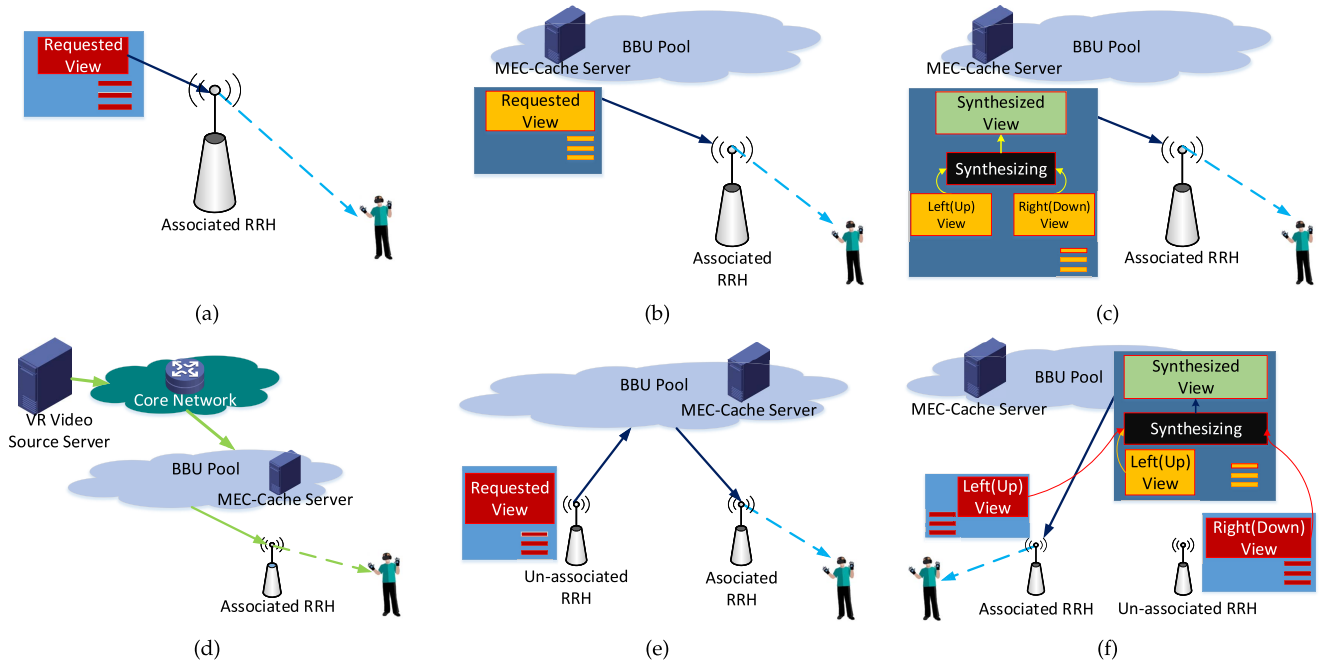


Fig. 2. Illustration of possible (exclusive) events that happen when a user is requesting video view data. (a) The requested view data are obtained directly from the cache of the associated RRH. (b) The requested view data are obtained from the cache of the BBU pool. (c) The requested view data are not cached either in the BBU pool or the RRH; however, the left and right view data are cached in the BBU pool, and the requested view data can be synthesized by the MEC-Cache server in the BBU pool. (d) The requested view data are obtained from the origin server. (e) The requested view data are obtained from the cache of other unassociated RRH caches. (f) The requested view data are not cached either in the BBU pool or RRHs, and there are not enough left and right view data cached in the BBU pool; however, the left and/or right view video data are cached in the RRHs.

$x_{0,u}^{f_s(k,n)} = 0$ otherwise. We define the download latency as D_{remote}, which equals $x_{0,u}^{f_s(k,n)} \cdot SIZE_{f_s(k,n)} \cdot t_O$.

- $y_{r,u}^{f_s(k,n)} = 1$ indicates that the desired view data can be accessed only in the unassociated RRH cache (Fig.2(e)), and $y_{r,u}^{f_s(k,n)} = 0$ otherwise. We denote by $D_{r'cache}$ the latency of downloading the required view data $f_s(k,n)$ from an unassociated RRH r by user u, which equals $y_{r,u}^{f_s(k,n)} \cdot SIZE_{f_s(k,n)} \cdot (2t_B + t_R)$.
- $z_{b,u}^{f_s(k,n)} = 1$ indicates that there are no desired view data cached in the RRHs and the BBU pool, but the required view data can be synthesized by the BBU pool since the needed left and right view data are cached in the RRHs(Fig.2(f)), and $z_{b,u}^{f_s(k,n)} = 0$ otherwise. We define the download latency as D_{synR}, which equals $z_{b,u}^{f_s(k,n)} \cdot SIZE_{f_s(k,n)} \cdot (t_B + t_{cs})$.
- $z_{b',u}^{f_s(k,n)} = 1$ means that there are no desired view data cached in RRHs and the BBU pool, the needed left (right) view data are cached in an RRH, and the right (left) data are cached in the BBU pool (Fig.2(f)). In this situation, the required view data can also be synthesized, and $z_{b',u}^{f_s(k,n)} = 0$ Otherwise. The download latency is defined as D_{synBR}, which equals $z_{b',u}^{f_s(k,n)} \cdot SIZE_{f_s(k,n)} \cdot (t_B + t_{cs})$.

When requesting a view data, only one of these seven events occurs. To ensure this, we impose the following constraints:

$$x_{r,u}^{f_s(k,n)} + x_{b,u}^{f_s(k,n)} + y_{b,u}^{f_s(k,n)} + x_{0,u}^{f_s(k,n)} + y_{r,u}^{f_s(k,n)} + z_{b,u}^{f_s(k,n)} + z_{b',u}^{f_s(k,n)} = 1. \quad (8)$$

To describe the view cache status in the RRH and BBU pool, two 0-1 variables, $c_r^{f_s(k,n)}$ and $c_b^{f_s(k,n)}$ are defined. If the required view has been cached in the r-th RRH, $c_r^{f_s(k,n)} = 1$ and $c_r^{f_s(k,n)} = 0$ otherwise. If the BBU pool has cached the required view, $c_b^{f_s(k,n)} = 1$ and $c_b^{f_s(k,n)} = 0$ otherwise. The following constraints are imposed to ensure that the amount of cached view data can not be larger than the total storage of the RRH and the BBU pool.

$$\sum_{f_s(k,n)\in\Omega_r} c_r^{f_s(k,n)} \cdot SIZE_{f_s(k,n)} \le C_R \quad (9)$$

$$\sum_{f_s(k,n)\in\Omega_r} c_b^{f_s(k,n)} \cdot SIZE_{f_s(k,n)} \le C_B \quad (10)$$

$$c_r^{f_s(k,n)}, c_b^{f_s(k,n)} \in \{0,1\} \quad (11)$$

We know that only if the data is cached in the associated/unassociated RRHs or BBU pool can $x_{r,u}^{f_s(k,n)}$ $x_{b,u}^{f_s(k,n)}$ or $y_{r,u}^{f_s(k,n)}$ be true, the following constraints are imposed:

$$x_{r,u}^{f_s(k,n)} \le c_r^{f_s(k,n)} \quad (12)$$

$$x_{b,u}^{f_s(k,n)} \le c_b^{f_s(k,n)} \quad (13)$$

$$y_{r,u}^{f_s(k,n)} \le \min(\sum_{l\in R, l\neq r} c_l^{f_s(k,n)}, 1) \quad (14)$$

Further, to ensure the availability of the view synthesis, the following constraints are imposed:

$$y_{b,u}^{f_s(k,n)} \le \min((\sum_{j=k-1, j\neq k}^{k+1} c_b^{f_s(j,n)})/2, 1) \quad (15)$$

$$y_{b,u}^{f_s(k,n)} \leq \min((\sum_{i=n-1,i\neq n}^{n+1} c_b^{f_s(k,i)})/2, 1) \quad (16)$$

$$z_{b,u}^{f_s(k,n)} \leq \min((\sum_{l\in R}\sum_{j=k-1,j\neq k}^{k+1} c_l^{f_s(j,n)})/2, 1) \quad (17)$$

$$z_{b,u}^{f_s(k,n)} \leq \min((\sum_{l\in R}\sum_{i=n-1,i\neq n}^{n+1} c_l^{f_s(k,i)})/2, 1) \quad (18)$$

$$z_{b',u}^{f_s(k,n)} \leq \min((c_b^{f_s(k+1,n)} + \sum_{l\in R} c_l^{f_s(k-1,n)})/2, 1) \quad (19)$$

$$z_{b',u}^{f_s(k,n)} \leq \min((c_b^{f_s(k,n+1)} + \sum_{l\in R} c_l^{f_s(k,n-1)})/2, 1) \quad (20)$$

$$z_{b',u}^{f_s(k,n)} \leq \min((c_b^{f_s(k-1,n)} + \sum_{l\in R} c_l^{f_s(k+1,n)})/2, 1) \quad (21)$$

$$z_{b',u}^{f_s(k,n)} \leq \min((c_b^{f_s(k,n-1)} + \sum_{l\in R} c_l^{f_s(k,n+1)})/2, 1) \quad (22)$$

The total download latency in the network for a user request is the sum of the above cases, which is denoted by $D_{u,r}^{f_s(k,n)}$.

$$D_{u,r}^{f_s(k,n)} = D_{rcache} + D_{bcache} + D_{synB} + D_{remote} + D_{r'cache} + D_{synR} + D_{synBR} \quad (23)$$

To minimize the overall downloading latency in the network, we formulate the problem as follows (24).

$$\min_{c_r^{f_s(k,n)}, c_b^{f_s(k,n)}} \sum_{u\in U}\sum_{r\in R}\sum_{f_s(k,n)\in\Omega_r} D_{u,r}^{f_s(k,n)}$$
$$\text{s.t. } (6), (7), (8), (9), (10), (11), (12), (13), (14), (15), (16), (17), (18), (19), (20), (21), (22) \quad (24)$$

V. Hierarchical Video Caching Algorithm

The problem in (24) is NP-Hardness (proved in the Theorem 2) and solving it to optimal in polynomial time is extremely challenging. Therefore, We begin with a brief analysis of an optimal solution to serve as a performance baseline, and then an online view synthesis-based caching algorithm is proposed.

Theorem 1: Problem in (24) *is NP-hard.*

Proof: We prove the NP-Hardness of problem (24) by reduction from a typical knapsack problem (KP) which is NP-Hard. In KP, there is a group of items with different weights and values, and a knapsack with limited capacity. The objective is to select a subset of items that can fit into the knapsack while having the largest total value. Note that KP is a special case of problem (24) if $C_r = 0, \forall r \in R$. In this case, the RRHs are not equipped with caches. Thus, $c_r^{f_s(k,n)} = 0, \forall r, \forall f, \forall s, \forall k, \forall n$. Each view is mapped into an item in KP. The item's weight corresponds to the size of a view $SIZE_{f_s(k,n)}$ and the item's value corresponds to $D_{bcache} + D_{synB} + D_{remote}$. Since the reduction can be done in polynomial time, problem (24) is NP-hard. ∎

A. Optimal Solution

Let us now assume that the network had a priori knowledge about all of the user requests Ω_r^*. In this case, problem (24) corresponds to an integer linear programming problem and can be decoupled into Ω_r^* independent sub-problems, one for each user request. Hence, its optimal solution can be easily computed in a running time of $\mathcal{O}(|RU|log(|RU|))$. The new problem is expressed in (25). Since this solution possessing the optimal performance is achieved under a priori knowledge about user requests, we call it the KP-optimal solution.

$$\min_{c_r^{f_s(k,n)}, c_b^{f_s(k,n)}} \sum_{u\in U}\sum_{r\in R}\sum_{f_s(k,n)\in\Omega_r^*} D_{u,r}^{f_s(k,n)}$$
$$\text{s.t. } (6), (7), (8), (9), (10), (11), (12), (13), (14), (15), (16), (17), (18), (19), (20), (21), (22) \quad (25)$$

However, we can not obtain all of the priori knowledge about user requests in real scenario, the KP-optimal solution is impractical. Therefore, a view synthesis based online caching algorithm is proposed to make view caching immediately and irrevocably upon each video request arrival at one of the RRHs. The whole caching process is showed in subsection V-B, and the proposed cache replacement scheme MaxMinDistance (hereinafter referred to as MMD) is described in subsection V-C.

B. Whole Caching Process

shown in Algorithm 1, the requested view $f_s(k,n)$ is checked at the beginning of every process loop. If $f_s(k,n)$ can be fetched from the BBU pool or the associated/unassociated RRH caches or can be synthesized by the MEC-Cache server, the corresponding data will be sent to the user immediately. Otherwise the system will bring it to the user from the VR video source server. In the meantime, the caching stage is launched. There are two phases in the caching stage. One is the cache placement phase, in which the data is cached immediately since the cache storage is not full; the other is the cache replacement phase, in which the cache storage is full. In the cache replacement phase, if the video segment is new, which means that no views in segment k of video f are cached

Algorithm 1 Whole Caching Process

Require: $f_s(k,n)$
1: **while** TRUE **do**
2: **if** $f_s(k,n)$ is cached $\|$ ($f_s(k,n)$ can be synthesized && $t_B + t_{cmp} \leq t_O$) **then**
3: Send $f_s(k,n)$ to user
4: **else**
5: Fetch $f_s(k,n)$ from source, and send it to user
6: **if** Cache storage is not full **then**
7: Cache the $f_s(k,n)$
8: **else**
9: **if** s is a new segment **then**
10: Cache replacement using LFU
11: **else**
12: Cache replacement using the MaxMinDistance scheme
13: **end if**
14: **end if**
15: **end if**
16: **end while**

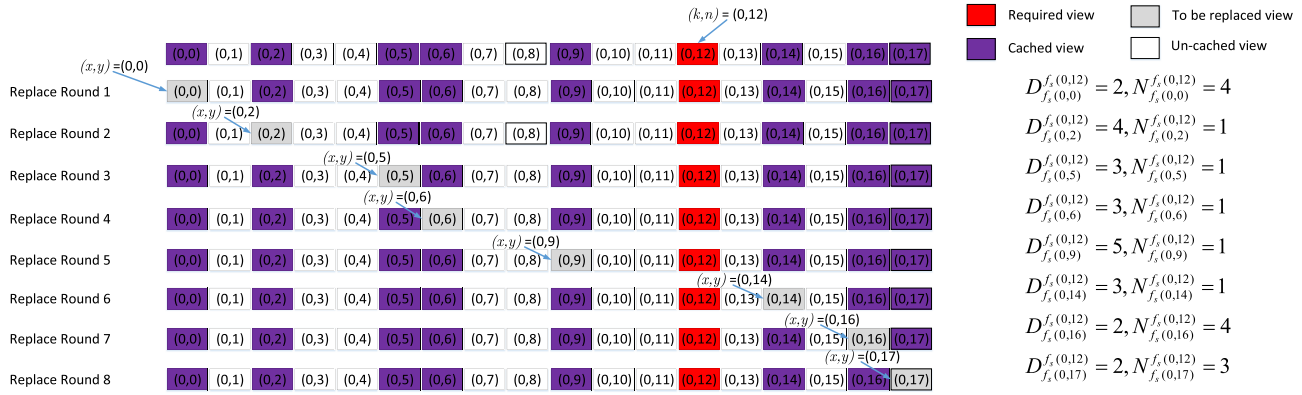


Fig. 3. An example of MaxMinDistance(MMD) caching.

at the RRHs or the BBU pool, the least frequently used data will be replaced. If not, considering the view synthesis feature of 360° VR data, we use the proposed MaxMinDistance (hereinafter referred to as MMD) scheme to replace the cached data.

C. MaxMinDistance (MMD) Scheme

Before introducing the MMD scheme, the definition of view distance is given in definition 1. It is shown that the larger the view distance, the farther the two data views are, and vice versa.

Definition 1: ***View Distance*** *reflects the interval between two views. Considering that $f_s(a,b)$ and $f_s(a',b')$ are different adjacent cached views in segment s of video f, where $\forall a, a' \in K$ and $\forall b, b' \in N$, the view distance of $f_s(a,b)$ and $f_s(a',b')$ is defined as follows.*

$$d^{f_s(a,b)}_{f_s(a',b')} = \begin{cases} |a-a'|-1, & |a-a'| < K/2,\ b=b' \\ (K-|a-a'|)-1, & |a-a'| \geq K/2,\ b=b' \\ |b-b'|-1, & |b-b'| < N/2,\ a=a' \\ (N-|b-b'|)-1, & |b-b'| \geq N/2,\ a=a' \\ \infty, & a \neq a',\ b \neq b' \end{cases} \tag{26}$$

It is obviously that the smaller the maximum distance of arbitrarily two cached adjacent views in a segment, the more opportunities there are for view synthesis and the less latency there are for video transmission. Therefore, we propose the MMD scheme to get the smallest maximum distance for any segment.

Assume that a 2-dimension view set $(\mathcal{X}, \mathcal{Y})$ which includes $X \times Y$ $(X < K, Y < N)$ different views of segment s in video f are cached in the RRH(s) and the BBU pool. $f_s(k,n)$ is the required view number and is not cached. First, the cached view data $f_s(x,y)$ are temporarily replaced with $f_s(k,n)$ one by one, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. After each replacement, the maximum view distance $D^{f_s(k,n)}_{f_s(x,y)}$ and the total number of maximum view distance $N^{(k,n)}_{(x,y)}$ can be obtained, where

$$D^{f_s(k,n)}_{f_s(x,y)} = \max_{\substack{\forall a,a' \in \{\mathcal{X} \setminus x, k\} \\ \forall b,b' \in \{\mathcal{Y} \setminus y, n\}}} d^{f_s(a,b)}_{f_s(a',b')} \tag{27}$$

All $D^{f_s(k,n)}_{f_s(x,y)}$ and $N^{f_s(k,n)}_{f_s(x,y)}$ are in set $\mathcal{D}$ and $\mathcal{N}$. Thereafter, we put the elements with minimum value in $\mathcal{D}$ and their corresponding elements in $\mathcal{N}$ into a two dimensional set $\mathbb{D}$. Finally, if there is only one element pair in $\mathbb{D}$, such as $f_s(x,y)$, then this element will be replaced by $f_s(k,n)$. If there are two or more element pairs in $\mathbb{D}$, then the $f_s(x,y)$ with minimum number of maximum view distances will be replaced.

Considering an simple example in Fig.3, the s-th segment of video file f is split into 6 rows and 18 columns, in the row 0, a set $(\mathcal{X}, \mathcal{Y}) = \{f_s(0,0),\ f_s(0,2),\ f_s(0,5),\ f_s(0,6),\ f_s(0,9),\ f_s(0,14),\ f_s(0,16),\ f_s(0,17)\}$ which includes 8 views are cached, and the requested view $f_s(k,n) = f_s(0,12)$ are not cached. In the MMD scheme, it will start eight rounds of virtual replacement first. In the 1st round of replacement, $f_s(0,0)$ is virtual replaced by $f_s(0,12)$, and we can obtain $D^{f_s(0,12)}_{f_s(0,0)} = 2$ and $N^{f_s(0,12)}_{f_s(0,0)} = 4$, which means that the maximum view distance between the adjacent cached views is *2* and the number of maximum distances is *4*. After eight rounds of virtual replacement, a set $\mathcal{D} = \{D^{f_s(0,12)}_{f_s(0,0)} = 2,\ D^{f_s(0,12)}_{f_s(0,2)} = 4,\ D^{f_s(0,12)}_{f_s(0,5)} = 3,\ D^{f_s(0,12)}_{f_s(0,6)} = 3,\ D^{f_s(0,12)}_{f_s(0,9)} = 5,\ D^{f_s(0,12)}_{f_s(0,14)} = 3,\ D^{f_s(0,12)}_{f_s(0,16)} = 2,\ D^{f_s(0,12)}_{f_s(0,17)} = 2\}$ and a set $\mathcal{N} = \{N^{f_s(0,12)}_{f_s(0,0)} = 4,\ N^{f_s(0,12)}_{f_s(0,2)} = 4,\ N^{f_s(0,12)}_{f_s(0,5)} = 1,\ N_{6,12} = 1,\ N^{f_s(0,12)}_{f_s(0,9)} = 1,\ N^{f_s(0,12)}_{f_s(0,14)} = 1,\ N^{f_s(0,12)}_{f_s(0,16)} = 4,\ N^{f_s(0,12)}_{f_s(0,17)} = 3\}$ are obtained. Based on these sets, we can obtain a set $\mathbb{D} = \{D^{f_s(0,12)}_{f_s(0,0)} = 2,\ N^{f_s(0,12)}_{f_s(0,0)} = 4;\ D^{f_s(0,12)}_{f_s(0,16)} = 2,\ N^{f_s(0,12)}_{f_s(0,16)} = 4;\ D^{f_s(0,12)}_{f_s(0,17)} = 2,\ N^{f_s(0,12)}_{f_s(0,17)} = 3\}$. Seeking $\mathbb{D}$, the minimum pair is $\{D^{f_s(0,12)}_{f_s(0,17)} = 2,\ N^{f_s(0,12)}_{f_s(0,17)} = 3\}$. Therefore, $f_s(0,17)$ is finally replaced by $f_s(0,12)$. The MMD scheme is shown in Algorithm 2.

Theorem 2: The proposed online view synthesis-based caching Algorithm 1 (simplified as online algorithm) has a competitive ratio of $\nabla = 2$, compared with the optimal KP offline algorithm for solving the minimization optimization problem in (24).

Proof: We assume a scenario that the following three conditions are satisfied simultaneously:

1) the users in different RRHs are concerning different segments of different VR video files;
2) the requested views are of new segments;

Algorithm 2 MaxMinDistance (MMD) Scheme

Require: $f_s(k,n)$, $\mathcal{X}$
Ensure: $\mathbb{D}$
for $\forall x, y \in \mathcal{X}, \mathcal{Y}$ **do**
  Virtually replace $f_s(x,y)$ with $f_s(k,n)$
  Compute $D^{f_s(k,n)}_{f_s(x,y)}$, and put it into $\mathcal{D}$
  Compute $N^{f_s(k,n)}_{f_s(x,y)}$, and put it into $\mathcal{N}$
end for
for $\forall D^{f_s(k,n)}_{f_s(x,y)} \in \mathcal{D}$ **do**
  if $D^{f_s(k,n)}_{f_s(x,y)} \leq D^{f_s(k,n)}_{f_s(x',y')}, x', y' \in \mathcal{X}, \mathcal{Y}$ **then**
    Put $D^{f_s(k,n)}_{f_s(x,y)}$ into $\mathbb{D}$
    Put $N^{f_s(k,n)}_{f_s(x,y)}$ into $\mathbb{D}$
  end if
end for
if Element pair in $\mathbb{D}$ is not unique **then**
  Find the least $N^{f_s(k,n)}_{f_s(x,y)}$ in $\mathbb{D}$
  if The least $N^{f_s(k,n)}_{f_s(x,y)}$ is not unique **then**
    Randomly select $D^{f_s(k,n)}_{f_s(x,y)}$
  end if
end if
Determine (x, y)
Replace $f_s(x,y)$ with $f_s(k,n)$

3) the requested views are less popular than the cached views.

It is obvious that this scenario is the worst case for the online algorithm and the largest competitive ratio value will be achieved (i.e., an upper bound), as the cooperative feature of C-RAN cannot be utilized and view synthesis cannot be applied among the RRHs.

Under this circumstance, the original problem in (24) can be divided into $R+1$ independent knapsack problems. We denote by $S_i = \{a_{i,1}, a_{i,2}, a_{i,t}, \ldots a_{i,T_i}\}$ the requested view set *(selected items)* in RRH *(knapsack)* i $(1 \leq i \leq R+1)$, which includes T_i $(2 < T_i \ll C_R)$ requests. For any requested view $a_{i,t}$, its time cost by delivering it from the VR video source can be denoted as function $f(a_{i,t})$ *(item value)*. It should be noted that other time costs, e.g., the view synthesis latency, are ignored for simplification. C_i $(0 \leq C_i \leq C_R)$ denotes the remaining caching space of RRH i. Obviously, if $C_i = 0$ or $C_i = 1$, the online algorithm has the same performance as the offline algorithm, since there will be no chance to utilize view synthesis for both algorithms and all the requested views should be delivered by the VR video source.

For other C_i, the proof, which is based on that the view synthesis range is equal to 2, is given as follows.

(i) $2 \leq C_i \leq T_i/2$.

For the offline algorithm, $\lceil T_i/2 \rceil$ views are selected to be transmitted based on their View Distances *(item weight)* in the best case. For example, there are 5 successive requested views in the i-th knapsack, which is denoted as $S_i = \{(0,0), (0,1), (0,2), (0,3), (0,4)\}$, and the optimal view set S_i^O with 3 views to be transmitted and cached can be $\{(0,0), (0,2), (0,4)\}$, since all the priori knowledge about user requests can be obtained according to the assumption.

However, all T_i requested views should be transmitted in the online algorithm in the worst case. The view set is denoted as $S_i^\dagger$, which equals to S_i.

Let $\mathcal{P}_i^O\left(S_i^O\right)$ and $\mathcal{P}_i^\dagger(S_i^\dagger)$ denote the offline result and online result, respectively. We have

$$\mathcal{P}_i^O\left(S_i^O\right) = \sum_{t=1}^{\lceil T/2 \rceil} f\left(a_{i,2t}\right) \tag{28}$$

and

$$\mathcal{P}_i^\dagger(S_i^\dagger) = \sum_{t=1}^{T_i} f\left(a_{i,t}\right), \tag{29}$$

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. Subsequently, the competitive ratio $\nabla^\dagger$ can be computed by

$$\begin{aligned}\nabla^\dagger &= \frac{\sum_{i=1}^{R+1} \mathcal{P}_i^\dagger\left(S_i^\dagger\right)}{\sum_{i=1}^{R+1} \mathcal{P}_i^O\left(S_i^O\right)} \\ &= \frac{\sum_{i=1}^{R+1}\sum_{t=1}^{T_i} f(a_{i,t})}{\sum_{i=1}^{R+1}\sum_{t=1}^{\lceil T_i/2 \rceil} f(a_{i,2t})} \\ &\leq \frac{\sum_{i=1}^{R+1} T_i \cdot f_{i,\max}}{\sum_{i=1}^{R+1} \frac{T_i}{2} \cdot f_{i,\max}} = 2,\end{aligned} \tag{30}$$

where $f_{i,\max}$ is the maximum value for all $f(a_{i,t})$.

(ii) $C_i \geq T_i/2 + 1$.

In this case, the offline result is equal to $\mathcal{P}_i^O(S_i^O)$. For the online algorithm, its result $\mathcal{P}_i^\ddagger(S_i^\ddagger)$ is always less than $P_i^\dagger(S_i^\dagger)$, since at least one view synthesis can be applied and not all the T_i requested views need to be transmitted as $C_i \geq T_i/2 + 1$. The competitive ratio $\nabla^\ddagger$ is as follows.

$$\begin{aligned}\nabla^\ddagger &= \frac{\sum_{i=1}^{R+1} \mathcal{P}_i^\ddagger(S_i^\ddagger)}{\sum_{i=1}^{R+1} \mathcal{P}_i^O\left(S_i^O\right)} \\ &< \frac{\sum_{i=1}^{R+1} \mathcal{P}_i^\dagger(S_i^\dagger)}{\sum_{i=1}^{R+1} \mathcal{P}_i^O\left(S_i^O\right)} \\ &= \nabla^\dagger.\end{aligned} \tag{31}$$

Finally, we have

$$\nabla = \max(\nabla^\dagger, \nabla^\ddagger) = 2. \tag{32}$$

■

D. Complexity Analysis

For one request, the algorithm searches $\mathcal{O}(KN)$ and calculates $\mathcal{O}(KN)$ cached view data to compute $D_{x,k}$ and $N_{f_s(x,y)}^{f_s(k,n)}$, and the minimum $D_{f_s(x,y)}^{f_s(k,n)}$ also needs $\mathcal{O}(KN)$ iterations. Further, the algorithm needs additional $\mathcal{O}(2R)$ to find the cached left and right view data in all the RRHs. Therefore, the complexity of Algorithm 2 becomes $\mathcal{O}((KN)^2 + KN + 2R)$. Polynomial time is needed by the algorithm, and it is an efficient/easy algorithm due to the small value of K and N ($K \leq 32$, $N \leq 32$ for common three-dimension VR videos) and R.

In terms of space it will consume only linear space complexity which is nothing but size of given elements.

VI. Simulation Results

In this section, numerical simulations are presented to evaluate the performance of the proposed MMD algorithm. We set t_B to 5 ms and $t_O = 10t_B$ to be consistent with actual network conditions. The default cache storage of an RRH and the BBU pool are 40 GB and 160 GB, respectively. V_B and V_O are set to 320 Mbps and 640 Mbps, respectively. There are 10,000 360° VR video files in the library. Each 360° VR video file has 100 segments, and each segment has 32×32 views. The default view synthesis range (d_{syn}) is set to 2, namely, only the adjacent left and right view can be used for synthesizing. The highest video resolution is $4K$, and the maximum compressed video data rate is 50 Mbps.

The video file popularity at each RRH follows a Zipf distribution [40] with the skew parameter $\alpha = 0.8$, and the frequency of the f-th popular video is inversely proportional to f:

$$\Lambda_f = \frac{\frac{1}{f^\gamma}}{\sum_{j=1}^{F} \frac{1}{j^\gamma}}, \quad 1 \leq f \leq F. \tag{33}$$

Considering that the user would watch the successive video segment of the current VR video or change to another new video file at different time, the probability of a user's request for segment f_s follows Markov process [41]. For Further, the the view popularity follows uniform distribution [22]. Video requests arrive one-by-one at each RRH r following a Poisson distribution with rate $\delta_r = 10$ [requests/min]. For each simulation, we randomly generate 50,000 requests at each RRH.

furthermore, the users may watch different versions of a single video file or different video files, the established data sizes to be cached are different for different users. No cache space is occupied at the beginning. The simulation configurations are listed in TABLE I.

A. Baseline Algorithms

Five existing algorithms, including the KP-optimal algorithm, the traditional LFU algorithm, VS-RANDOM algorithm, VS-LFU algorithm and Efficient View Exploration Algorithm (EVEA) [22], are compared with the proposed scheme.

TABLE II

Standard Simulated C-RAN 360° VR Caching System Related Parameters

Parameters	Description	Values
F	Number of video files	10000
S	Number of segments of a video	100
$K \times N$	Number of views of a segment	32×32
$\mathcal{L}$	Available bitrate level set	$[10, 50]Mbps$
$SIZE_{f_s(k,n)}$	Size of view	$[10, 40]MB$
R	Number of RRHs	$[5, 50]$
U	Number of users	$[50, 150]$
C_R	Cache storage of an RRH	$40GB$
C_B	Cache storage of the BBU pool	$160GB$
$t_{r,u}$	Average latency from an RRH to a user	–
t_B	Average latency from the BBU pool to a user	$5ms$
t_O	Average latency from the source to a user	$50ms$
$B_{r,u}$	Bandwidth of user u in its associated RRH r	–
V_B	Total data rate between RRHs and the BBU pool	$320Mbps$
V_O	Total data rate between the BBU pool and source server	$640Mbps$

(1) KP-optimal algorithm. An impractical algorithm possessing the optimal performance, which assumes a priori knowledge about the complete user requests, as described in Subection V-A.

(2) LFU algorithm. The LFU algorithm only keeps a sorted list recording the frequency of the cached views. Whenever a cache miss happens, the last element (with the lowest frequency) in the list is replaced.

(3) VS-LFU algorithm. Similar to the traditional LFU, the VS-LFU algorithm further considers the view synthesis feature.

(4) VS-RANDOM algorithm. Different from the VS-LFU, the cache replacements are randomly.

(5) EVEA algorithm. The EVEA algorithm is a heuristic approach based on the Markov decision process that leverages DIBR in multi-view 3D videos.

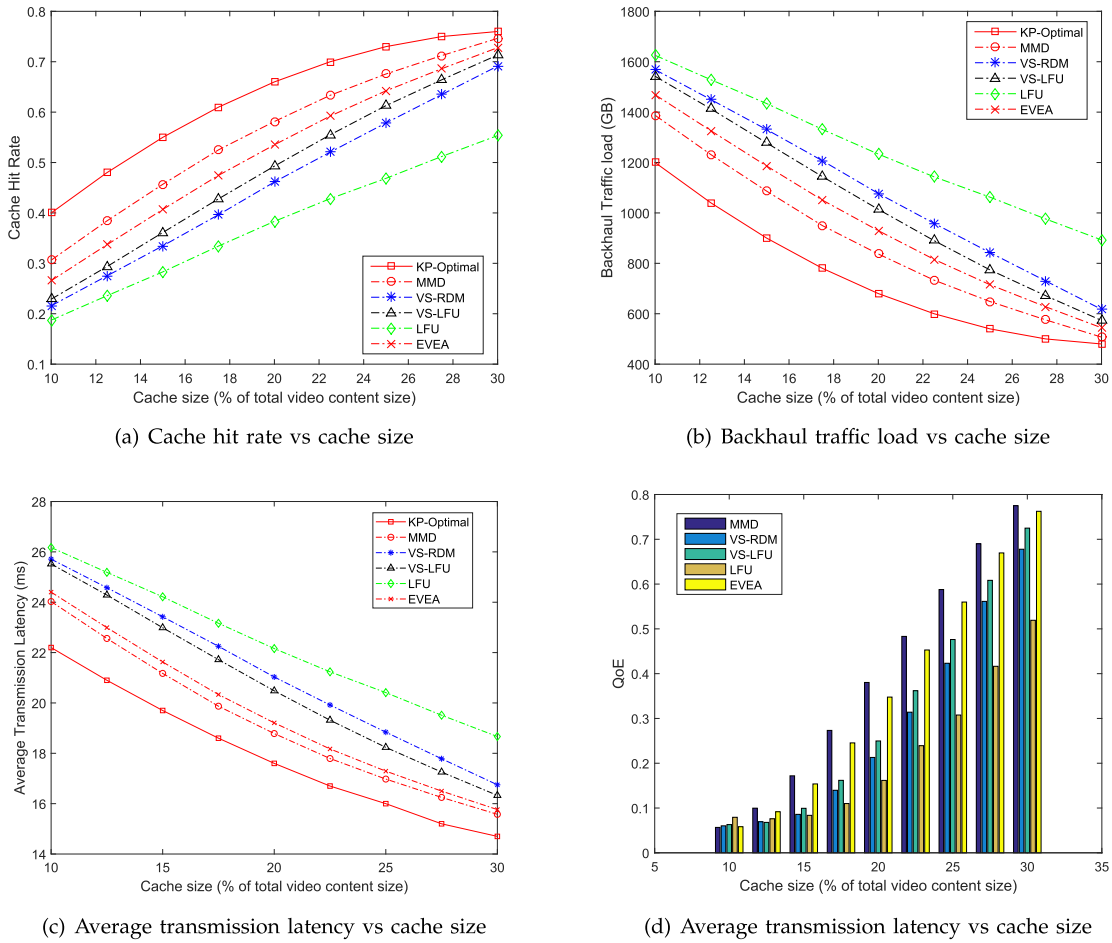
B. Performance Metrics

The performance metrics used for the evaluation are as follows.

(1) Average cache hit rate: we define $\mathcal{C}_{hit}$ as the number of cache hits during window w, $\mathcal{C}_{syn}$ as the number of syntheses during window w and $\mathcal{C}_{req}$ as the request number during window w. The cache hit rate performance can be evaluated using the average cache hit rate (AHR), which is defined as follows.

$$AHR = \frac{1}{W} \sum_{w \in W} \frac{\mathcal{C}_{hit} + \mathcal{C}_{syn}}{\mathcal{C}_{req}} \tag{34}$$

A higher AHR indicates that a higher number of video segment requests are directly downloaded from caches over a succession of W time windows of length w, which consequentially reduces the average response time.



(a) Cache hit rate vs cache size

(b) Backhaul traffic load vs cache size

(c) Average transmission latency vs cache size

(d) Average transmission latency vs cache size

Fig. 4. Simulated cache hit rate, backhaul traffic load, average transmission latency and QoE performance among the MMD algorithm, KP-Optimal algorithm, LFU algorithm, VS-LFU algorithm, VS-RDM algorithm, and EVEA algorithm with respect to different cache capacity.

(2) Backhaul traffic load [GB]: the volume of video downloaded by the user from the source server going through the backhaul network.

(3) Average latency [ms]: the average transmission delay of 360° VR video content to a requesting user, including the delay of traveling from the RRH and the BBU pool caches, the synthesis delay, and the delay of fetching directly from the source server.

(4) Quality-of-experience (QoE): inspired by [42], the QoE model reflects the user-perceived performance of the 360° VR video during a period of time (i.e., T) and can be enumerated by the function that has the following key elements.

i) Average video quality: Assuming that $\upsilon(\tau, u)$, which is a collection of symbols about the view $f_s(k, n)$, represents the data required by user u at time τ, the bitrate and the quality of $\upsilon(\tau, u)$ can be denoted by $L_{\upsilon(\tau,u)}$ and $q(L_{\upsilon(\tau,u)})$, respectively, where $q(L_{\upsilon(\tau,u)})$ is a non-decreasing function that maps the selected bitrate $L_{\upsilon(\tau,u)}$. The higher the bitrate selected, the higher the video quality perceived by the user will be.

Different from traditional video system, the user will receive a synthesized view, whose quality is always lower than that of the original view. Based on the evaluation in [43], for example, if the PSNR of the original data is 42.3 dB, when the view distances are 2, 3, 4, and 5, the PSNR of the synthesized data becomes 41.9 dB, 41 dB, 40.7 dB, and 40.2 dB, respectively, with slight, linear degradation.

Therefore, the average per-view quality $\mathcal{Q}$ over all requested view data is denoted by (35).

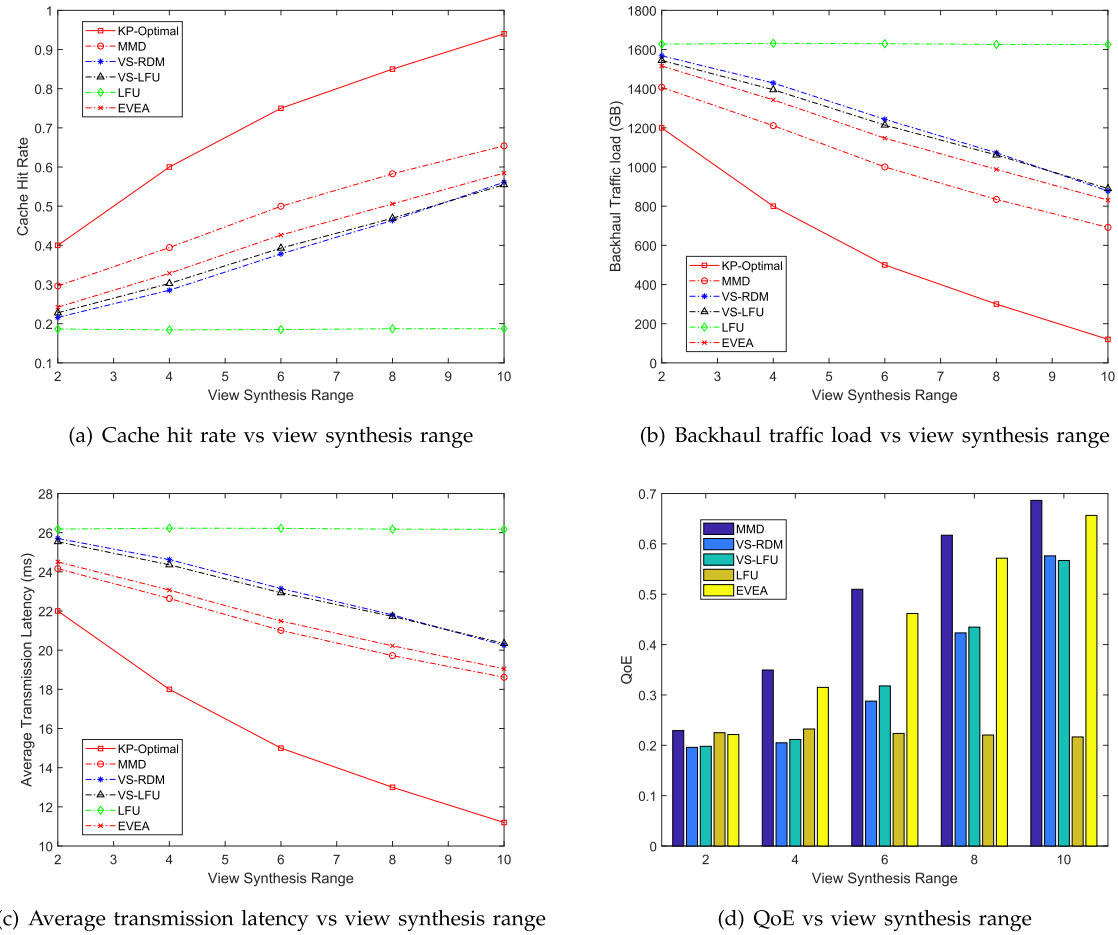
$$Q = \frac{1}{T}\sum_{\tau=0}^{T}\left(q(L_{\upsilon(\tau,u)}) - \beta \cdot d_{\text{syn}}\mathbb{1}_{\{\text{synthesized }\upsilon(\tau,u)\}}\right) \quad (35)$$

where β is the decline slope of view quality caused by synthesizing, and $\mathbb{1}_{\{x\}} = 1$ when x is true and $\mathbb{1}_{\{x\}} = 0$ when x is false.

ii) Average quality variations ($\mathcal{V}$): This tracks the magnitude of the changes in the quality from one set of view data to another. Once the quality of the data required at time $\tau + 1$ is lower than that of the data required at time τ, the QoE will decrease.

$$\mathcal{V} = \frac{1}{T}\sum_{\tau=0}^{T}(q(L_{\upsilon(\tau,u)}) - q(L_{\upsilon(\tau+1,u)})) \quad (36)$$

iii) Rebuffer time (T): For each required dataset, rebuffering occurs only if the download time is longer than the play-out time of buffered video when the view download begins (i.e., P_τ). Thus, the total rebuffer time is expressed by (37), where V_τ is the download data rate of $\upsilon(\tau, u)$. It should be



(a) Cache hit rate vs view synthesis range

(b) Backhaul traffic load vs view synthesis range

(c) Average transmission latency vs view synthesis range

(d) QoE vs view synthesis range

Fig. 5. Simulated cache hit rate, backhaul traffic load, average transmission latency and QoE performance among the MMD algorithm, KP-Optimal algorithm, LFU algorithm, VS-LFU algorithm, VS-RDM algorithm, and EVEA algorithm with respect to different view synthesize range.

replaced by $B_{r,u}\log_2(1+\gamma_{r,u})$, V_B/U, or V_O/U according to the location of $v(\tau,u)$.

$$\mathcal{T}=\sum\nolimits_{\tau=0}^{T}\left(\frac{SIZE_{v(\tau,u)}}{V_\tau}-P_\tau\right)_+ \tag{37}$$

iv) Startup delay (T_{start}): Assume $T_{start}\ll P_{\max}$, where $P_{\max}$ means the playout time of full buffered video.

As users may have different preferences on which of the four components is more important, we define the QoE of user u by a weighted sum of the aforementioned components:

$$QoE_u=\mathcal{Q}-\lambda\mathcal{V}-\mu\mathcal{T}-\eta T_{start} \tag{38}$$

Here λ, μ, and η are non-negative weighting parameters corresponding to video quality variations, rebuffering time and startup delay, respectively. In the 360° VR system, the weights λ, μ, and η are set to 0.1, 8 and 10, respectively, which means that the user is deeply concerned about rebuffering time and prefers a low startup delay.

C. Impact of Cache Storage

In this subsection, we change the total cache size from 160 GB (10% of the total file size) to 480 GB (30% of the total file size) to observe the performance in terms of the cache hit rate, backhaul traffic load, average latency and QoE.

Fig.4 presents the cache hit rate, average transmission latency and backhaul traffic load of the six algorithms with $R=50$, respectively. the performance of the six algorithms improves as the cache size grows, and the MMD algorithms always achieve an obviously superior performance, which is benefit from considering view synthesis feature in caching and processing.

As shown in Fig.4(a), When the cache capacity is small (160 GB), the cache hit rate of the proposed algorithm is almost 1.67 times that of the traditional LFU algorithm. When the cache capacity increases to 480 GB, the cache hit rate of the proposed algorithm is still 1.4 times that of the traditional LFU algorithm and 5% higher than that of the EVEA algorithm. While there is a 10% gap between the proposed MMD algorithm and the KP-Optimal algorithm (160 GB), the gap becomes smaller as the cache storage increases.

As shown in Fig.4(b), the backhaul traffic load of the proposed algorithm is 25% lower than that of the other algorithms in the worst case.

As shown in Fig.4(c), the MMD algorithm always achieves the lowest average latency. In the best case, the average latency of the MMD algorithm is less than 16 ms, which is only 1 ms more than the KP-Optimal algorithm, and almost 30% less than that of the LFU algorithm.

The QoE of the six algorithms with respect to cache capacity is presented in Fig.4(d). When the cache capacity is low, the proposed algorithm is not the best because not enough views can be synthesized, and the requested data should be fetched from the source server. As the cache capacity increases, the QoE of the MMD algorithm increases sharply and is higher than that of other algorithms. In the best case, the QoE of the proposed algorithm is 60% higher than the results of the LFU algorithm.

D. Impact of View Synthesis Range

In subsection VI-C, only the adjacent left and right view is used for synthesizing. Knowing that he quality of each synthesized view depends on its distance to its two reference views, we fix the cache capacity to 160 GB and change the view synthesize range from default 2 to 10 to observe the influence on the performance in this subsection.

From Fig.5, we can see that the larger the view synthesis range, the more obvious the advantage of this algorithm is, in addition to the KP-optimal algorithm. For KP-optimal algorithm, since almost all the necessary views can be cached with the increasing of view synthesize range, the gap between KP-optimal algorithm and the proposed MMD algorithm becomes larger and larger.

Further, while the average video quality will decrease when the view synthesis range increases, the rebuffer time and start up delay will decrease sharply. Therefore, we can see from Fig.5(d) that the QoE remains increasing as the view synthesis range increases and the performance of MMD algorithm is the best among all of imported schemes.

VII. Conclusion

In this paper, we design a view synthesis-based 360° VR caching system over C-RAN, where MEC is enabled for view synthesizing and hierarchical caches are deployed at both BBU pool and RRHs. To decrease the transmission latency and backhaul traffic load for VR services, an integer linear program (ILP) problem aimed at minimizing the total transmission latency for 360° VR video contents is formulated and is proved to be NP-Hard. Due to the NP-completeness of the problem and the absence of the request arrival information in practice, we propose an efficient online MMD caching replacement algorithm, which performs cache replacement upon arrival of each new request. Rigorous numerical simulations show that the proposed algorithm always yields better performance in terms of cache hit rate, backhaul traffic load, average transmission latency and QoE than the other employed caching algorithms.

References

[1] N. S. S. Hamid, F. A. Aziz, and A. Azizi, "Virtual reality applications in manufacturing system," in *Proc. Sci. Inf. Conf.*, Aug. 2014, pp. 1034–1037.

[2] *Forecast 'Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021'*, Cisco, San Jose, CA, USA, 2017.

[3] Juniper. (2017). *Virtual Reality Markets: Hardware, Content & Accessories 2017–2022*. [Online]. Available: https://www.juniperresearch.com/researchstore/innovationdisruption/virtual-reality/hardware-content-accessories

[4] Huawei. (2016). *White Paper: Hosted Network Requirements Oriented on VR Service*. [Online]. Available: http://www.imxdata.com/archives/17346

[5] 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.

[6] M. Graf, C. Timmerer, and C. Mueller, "Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP: Design, implementation, and evaluation," in *Proc. 8th ACM Multimedia Syst. Conf.*, 2017, pp. 261–271

[7] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Hevc-compliant tile-based streaming of panoramic video for virtual reality applications," in *Proc. 24th ACM Int. Conf. Multimedia*, 2016, pp. 601–605.

[8] R. Skupin, Y. Sanchez, D. Podborski, C. Hellge, and T. Schierl, "HEVC tile based streaming to head mounted displays," in *Proc. 14th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2017, pp. 613–615.

[9] 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.

[10] S. Sukhmani, M. Sadeghi, M. Erol-Kantarci, and A. El Saddik, "Edge caching and computing in 5G for mobile AR/VR and tactile Internet," *IEEE Multimedia Mag.*, vol. 26, no. 1, pp. 21–30, Jan./Mar. 2019.

[11] J. Chakareski, "VR/AR immersive communication: Caching, edge computing, and transmission trade-offs," in *Proc. Workshop Virtual Reality Augmented Reality Netw.*, Aug. 2017, pp. 36–41.

[12] M. Chen, W. Saad, and C. Yin, "Echo-liquid state deep learning for 360° content transmission and caching in wireless VR networks with cellular-connected UAVs," 2018, *arXiv:1804.03284*. [Online]. Available: https://arxiv.org/abs/1804.03284

[13] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Edge computing meets millimeter-wave enabled VR: Paving the way to cutting the cord," 2018, *arXiv:1801.07614*. [Online]. Available: https://arxiv.org/abs/1801.07614

[14] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," in *Proc. International Society for Optics and Photonics, Stereoscopic Displays and Virtual Reality Systems XI*, vol. 5291, pp. 93–104, 2004.

[15] G. Chaurasia, O. Sorkine, and G. Drettakis, "Silhouette-aware warping for image-based rendering," *Comput. Graph. Forum*, vol. 30, no. 4, pp. 1223–1232, 2011.

[16] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *ACM Trans. Graph.*, vol. 32, no. 3, p. 30, Jun. 2013.

[17] S. Wanner and B. Goldluecke, "Variational light field analysis for disparity estimation and super-resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 606–619, Mar. 2014.

[18] M. Domański *et al.*, "High efficiency 3D video coding using new tools based on view synthesis," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3517–3527, Sep. 2013.

[19] F. Zou, D. Tian, A. Vetro, H. Sun, O. C. Au, and S. Shimizu, "View synthesis prediction in the 3-D video coding extensions of AVC and HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 10, pp. 1696–1708, Oct. 2014.

[20] Y. Gao, G. Cheung, T. Maugey, P. Frossard, and J. Liang, "Encoder-driven inpainting strategy in multiview video compression," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 134–149, Jan. 2016.

[21] G. Luo, Y. Zhu, Z. Weng, and Z. Li, "A disocclusion inpainting framework for depth-based view synthesis," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.

[22] J.-T. Lee, D.-N. Yang, and W. Liao, "Efficient caching for multi-view 3D videos," in *Proc. IEEE Global Commun. Conf.*, Dec. 2016, pp. 1–7.

[23] C. Guo, Y. Cui, and Z. Liu, "Optimal multicast of tiled 360 VR video in OFDMA systems," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2563–2566, Dec. 2018.

[24] C. Guo, Y. Cui, and Z. Liu, "Optimal multicast of tiled 360 VR video," *IEEE Wireless Commun. Lett.*, vol. 8, no. 1, pp. 145–148, Feb. 2019.

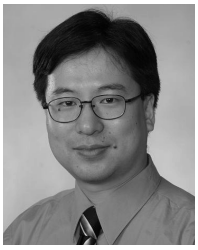
[25] W. Xu, Y. Wei, Y. Cui, and Z. Liu, "Energy-efficient multi-view video transmission with view synthesis-enabled multicast," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–7.

[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] H. Ahmadi, O. Eltobgy, and M. Hefeeda, "Adaptive multicast streaming of virtual reality content to mobile users," in *Proc. Thematic Workshops ACM Multimedia (Thematic Workshops)*, New York, NY, USA, 2017, pp. 170–178. doi: 10.1145/3126686.3126743.
[28] 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. ACM Multimedia Conf. (MM)*, Mountain View, CA, USA, Oct. 2017, pp. 315–323. doi: 10.1145/3123266.3123291.
[29] M. Chen, W. Saad, and C. Yin, "Echo state learning for wireless virtual reality resource allocation in UAV-enabled LTE-U networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[30] G. Schaufler and W. Stürzlinger, "A three dimensional image cache for virtual reality," *Comput. Graph. Forum*, vol. 15, no. 3, pp. 227–235, Aug. 1996.
[31] E. Bastug, M. Bennis, M. Médard, and M. Debbah, "Toward interconnected virtual reality: Opportunities, challenges, and enablers," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 110–117, Jun. 2017.
[32] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," 2017, *arXiv:1710.02913*. [Online]. Available: https://arxiv.org/abs/1710.02913
[33] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Communications, caching and computing for mobile virtual reality: Modeling and tradeoff," 2018, *arXiv:1806.08928*. [Online]. Available: https://arxiv.org/abs/1806.08928
[34] A. Mahzari, A. T. Nasrabadi, A. Samiei, and R. Prakash, "FoV-aware edge caching for adaptive 360° video streaming," in *Proc. ACM Multimedia Conf. Multimedia Conf.*, 2018, pp. 173–181.
[35] G. Gao, Y. Wen, and J. Cai, "vCache: Supporting cost-efficient adaptive bitrate streaming," *IEEE MultimediaMag.*, vol. 24, no. 3, pp. 19–27, Aug. 2017.
[36] L. Gkatzikis, V. Sourlas, C. Fischione, I. Koutsopoulos, and G. Dán, "Clustered content replication for hierarchical content delivery networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2015, pp. 5872–5877.
[37] T. X. Tran, D. V. Le, G. Yue, and D. Pompili, "Cooperative hierarchical caching and request scheduling in a cloud radio access network," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2729–2743, Dec. 2018.
[38] M. Chen, W. Saad, C. Yin, and M. Debbah, "Echo state networks for proactive caching in cloud-based radio access networks with mobile users," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3520–3535, Jun. 2016.
[39] H. Wang, R. Li, L. Fan, and H. Zhang, "Joint computation offloading and data caching with delay optimization in mobile-edge computing systems," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2017, pp. 1–6.
[40] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, 2007, pp. 1–14.
[41] J. Dai, Z. Zhang, and D. Liu, "Proactive caching over cloud radio access network with user mobility and video segment popularity awared," *IEEE Access*, vol. 6, pp. 44396–44405, 2018.
[42] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 325–338, 2015.
[43] J.-T. Lee, D.-N. Yang, Y.-C. Chen, and W. Liao, "Efficient multi-view 3D video multicast with depth-image-based rendering in LTE-advanced networks with carrier aggregation," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 85–98, Jan. 2018.

Jianmei Dai (S'07) received the B.S. degree in communication engineering and the M.S. degree in communication and information systems, in 2004 and 2007, respectively. He is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications, Beijing. He is also a Lecturer with the Beijing University of Posts and Telecommunications. He is also a Visiting Scholar with Auburn University, AL, USA. His research interests include optimization theory and its applications in wireless video transmission and wireless networks.

Zhilong Zhang received the B.E. degree in communication engineering from the University of Science and Technology, Beijing, China, in 2007, and the M.S. and Ph.D. degrees in communication and information systems from the Beijing University of Posts and Telecommunications (BUPT), Beijing, in 2010 and 2016, respectively. From 2010 to 2012, he was a Software Engineer with TD Tech Ltd., Beijing. From 2014 to 2015, he was a Visiting Scholar with Stony Brook University, NY, USA. He is currently a Lecturer with BUPT. His research interests include optimization theory and its applications in wireless video transmission, cross-layer design, and wireless networks.



Shiwen Mao (S'99–M'04–SM'09–F'19) received the Ph.D. degree in electrical and computer engineering from Polytechnic University, Brooklyn, NY. He is currently a Samuel Ginn Professor with the Department of Electrical and Computer Engineering, and the Director of the Wireless Engineering Research and Education Center (WEREC), Auburn University, Auburn, AL. His research interests include wireless networks, multimedia communications, and smart grid. He was a recipient of the Auburn University Creative Research and Scholarship Award in 2018 and the NSF CAREER Award in 2010. He was a co-recipient of the IEEE ComSoc MMTC Best Conference Paper Award in 2018, the Best Demo Award from IEEE SECON 2017, the Best Paper Awards from IEEE GLOBECOM 2016 & 2015, IEEE WCNC 2015, and IEEE ICC 2013, and the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems. He is on the Editorial Board of the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE INTERNET OF THINGS JOURNAL, the IEEE MULTIMEDIA, the IEEE NETWORKING LETTERS, and ACM *GetMobile*.

Danpu Liu (S'98–M'02–SM'15) received the Ph.D. degree in communication and electrical systems from the Beijing University of Posts and Telecommunications, Beijing, China, in 1998. She was a Visiting Scholar with the City University of Hong Kong in 2002, University of Manchester in 2005, and Georgia Institute of Technology in 2014. She is currently with the Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications. Her research involved MIMO, OFDM, and broadband wireless access systems. She has published more than 100 articles and three teaching books, and submitted 26 patent applications. Her recent research interests include 60 GHz mmWave communication, wireless high definition video transmission, and wireless sensor network.