Rendering-Aware VR Video Caching over Multi-cell MEC Networks

Yanwei Liu, *Member, IEEE,* Jinxia Liu, Antonios Argyriou, Liming Wang and Zhen Xu

***Abstract*—Delivering high fidelity virtual reality (VR) video over mobile networks is very challenging since VR applications usually require very high bandwidth and ultra low latency. With the evolution of 5G mobile networks, multi-cell multi-access edge computing (MEC) networks enable low latency data communication. However, even in this setting, the requirements of VR applications are tough to meet. To optimize the end-to-end latency for VR video delivery over multi-cell MEC networks, we propose a rendering-aware tile caching scheme. As a first step we propose collaborative tile caching in 5G MEC networks by enabling multiple cell sites to share caches so that the network caching performance is improved. Hence, the amount of redundant data delivered and eventually the latency are both significantly reduced. Second, our scheme offloads viewport rendering of VR video to the MEC server and closely couples this with cache placement, allowing thus the rendering-induced latency to be reduced. Finally, a low-delay request routing algorithm is integrated with the proposed cache placement scheme to further optimize the end-to-end latency of VR video delivery. Extensive simulation results show that the proposed rendering-aware caching scheme can achieve better latency performance than the state-of-the-art decoupled caching/rendering schemes.**

***Index Terms*—VR video, Multi-cell MEC networks, Rendering-aware tile caching.**

I. Introduction

In recent years, virtual reality (VR) video has gained popularity due to the exciting immersive experience that it offers. As a result of that, VR video has been widely used to events like exhibitions, sports, concerts and films [1]. Currently, most VR video applications are deployed locally off-line and very few applications obtain data remotely via network communication. However, the increasingly powerful mobile platforms are now already able to support VR applications. In this context, the need for efficient wireless networking of VR video is very urgent.

VR videos are typically characterized by larger file sizes when compared to traditional planar video since they provide a super high-definition (HD) 360-degree viewing experience. The high-resolution of VR video brings a major challenge for delivering it over the current bandwidth-limited wireless networks. Though high efficiency video coding (HEVC) can be used to encode VR video, compressed VR video still has $5\sim10\times$ the data size of HD video. One of the key differences from the traditional planar video is that VR video is interactively viewed on a user's head-mounted display (HMD) by freely selecting the field of view (FoV), namely the viewport, from a 360-degree space. During VR video viewing, the user moves the HMD to capture the interesting objects in any viewing direction, and thus the frequent viewport requests require more bandwidth to deliver these time-varying viewports.

Besides the huge amount of data, the end-to-end latency is another practical challenge for achieving high-fidelity wireless VR video streaming. It is well known that the Motion-To-Photon (MTP) latency required by VR video viewing is less than 20ms [2]. Under such a strict constraint, every element in the end-to-end VR video processing chain that starts from the source server and ends with the video playback should be optimized to reduce latency. Among these elements, data transmission and VR video processing (decoding and rendering at the receiver) are the two major latency sources.

To reduce the network-induced latency, content caching must be used to push data near the end user during off-peak hours. According to the multi-access edge computing (MEC) paradigm [3], the content cache can be placed at the base station. In 5G mobile networks, the MEC servers accompany the deployment of base stations (gNBs in 5G) in multiple radio cells. Due to the powerful computational capabilities of MEC servers, computations related to viewport rendering can be offloaded [4] from the user equipment (UE) to the MEC server. Consequently the rendering operation will be accelerated and the overall latency for viewport requests will be reduced.

Even though video caching and offloading of viewport rendering are two ideal solutions for improving VR video delivery performance, they are typically viewed as two independent operations. However, when rendering is enabled at the MEC server, the rendering operation depends heavily on the cached data. To ensure high rendering speed, the data for the rendering operation needs to be provided quickly. Consequently, rendering needs to be accompanied with the cached data. Thus, cache optimization and the edge rendering computation allocation should be closely coupled in order to optimize the VR video delivery performance.

To facilitate fast access of viewport data in one entire frame, VR video is usually encoded with tile partitions [5]. The tiles

This work was supported in part by National Natural Science Foundation of China under Grant 61771469 and Ningbo Natural Science Foundation under Grant 2019A610109.

Y. Liu is with Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, 100049, China (e-mail: liuyanwei@iie.ac.cn).

J. Liu is with the Zhejiang Wanli University, Ningbo 315100, China (e-mail: liujinxia@zwu.edu.cn).

A. Argyriou is with the Department of Electrical and Computer Engineering, University of Thessaly, Greece (e-mail: anargyr@uth.gr).

Liming Wang and Zhen Xu are with Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, 100049, China (e-mail: wangliming@iie.ac.cn, xuzhen@iie.ac.cn)

have of course different popularity that depends on the way the HMD moves. As for the mobile networks, multiple gNBs in 5G networks are deployed very close and form a hierarchical multi-cell MEC network. Due to the high volume of mobile data, the overall cache space is a limited and hence precious resource. At the same time, the computational capabilities of the MEC servers are also not infinite. On top of all these, the latency requirement for viewport request is very stringent and is quite more critical than traditional video.

Given the above issues, the key challenges for VR video caching over multi-cell MEC networks are:

- *How to predict efficiently tile popularity in VR video caching?*
- *How to jointly optimize tile caching and the associated rendering computation to enable low-latency tiled VR video delivery over multi-cell MEC networks?*
- *How to route the viewport request to satisfy the low-latency requirement as much as possible?*

Motivated by the above open questions, in this paper we propose a rendering-aware caching scheme with a low-delay request scheduling algorithm for tiled VR video delivery over multi-cell MEC networks. Specifically, we consider the joint optimization of tile cache placement and viewport request routing to minimize the delivery latency of all possible requests in the system. Although the proposed scheme has similarities to coded caching [6], in our case we consider cooperative cache placement among the different cache nodes that is significant difference from coded caching. More importantly, coded caching is not concerned with the issue of rendering computation provisioning for reducing the VR video delivery latency, which is the main goal of this paper. The contributions of the paper are listed next.

First, we propose a tiled VR video cache placement scheme in multi-cell 5G MEC networks. On one hand, we define the tile as the basic data unit in the caching system so as to enable rapid spatial data access capability in the whole frame. Consequently, tile caching can quickly respond to the interactive viewport rendering and further reduce the viewport request latency. On the other hand, we propose a tile popularity prediction approach based on the estimation of the tile saliency using the Choquet integral. Next, the tiles are collaboratively cached in the multi-cell 5G MEC network to enable the cache node to share content and thus the network caching performance is improved.

Second, we propose to couple tile caching with rendering. With mobile edge computing, the rendering operation is offloaded to the cache server (MEC server). Then, rendering and caching are jointly coupled to optimize the complete multi-cell mobile edge caching system. Besides the rendering-related cost saving, the rendering computation constraint for each cache node is also considered in the cache placement optimization.

Third, based on the tile-based cache placement, a low-delay viewport request routing scheme is proposed to further reduce the viewport delivery latency. The requests are scheduled in a time granularity of a single request. Under the strict constraints of latency and backhaul uplink bandwidth, the proposed routing scheme can find the optimal cache node to provide the viewport rendering service rapidly.

The rest of the paper is organized as follows. Section II introduces the related work and Section III describes the background of the asynchronous MEC-UE rendering. The system model is given in Section IV. The proposed rendering-aware tile caching problem over multi-cell MEC networks is formulated and solved in Section V. Experimental results are provided in Section VI. Finally, Section VII concludes the paper.

II. Related Work

A. VR Video Streaming

To improve VR video delivery efficiency over networks, researchers over the last few years have proposed several VR video streaming schemes. Existing VR streaming approaches are typically classified into two categories. One is full-frame VR video streaming where the complete frame data including the standby data outside the viewport is fetched to the user. This approach can ensure smooth viewport switching during viewing. But this approach requires more bandwidth for transmitting the huge data volume of the complete frame. Another one is FoV-dependent streaming [7] [8] where only the necessary viewport data are delivered to the user. This approach divides the whole frame into tiles that can be chosen flexibly and thus reduces the transmitted volume of data by discarding the redundant tiles outside the viewport.

With network services gradually transitioning to the cloud, the above two types of approaches have been extended to cloud-based systems [4] for primarily exploiting the computational capabilities of the cloud. In cloud-based VR streaming, VR video rendering is offloaded to the cloud to reduce the rendering latency.

In recent years, 5G mobile networks embrace the paradigm shift from centralized cloud computing towards MEC. As the extension of cloud computing to the edge networks, MEC technologies have achieved significant progresses [9]. MEC has been investigated thoroughly in the context of Internet of Things [10]. Several novel approaches were proposed to optimize edge applications, such as dynamic resource allocation that considers end-user mobility [11], an approach that could potentially give VR applications a way to adapt to the dynamic context. To exploit the benefit of MEC, edge-assisted rendering [12][13][14][15] was used for VR video streaming. With the assistance of cloud or edge computing, the above-mentioned approaches can efficiently deliver VR videos when bandwidth is not an issue. However, when we want to deliver high fidelity VR videos over networks that have limited bandwidth (e.g. due to competition by multiple users), the low latency requirement is still an unaddressed issue.

B. VR Video Caching

To reduce both the transmission latency and duplicate traffic, in-network caching is used to copy content in servers close to the users. As the last mile in the video transmission chain, the mobile network is the closest link to the user end. As a result, video caching was further extended to mobile networks.

Especially with the development of 5G networks, mobile network caching has been used to reduce content delivery latency [16][17]. Since the storage space of mobile networks is usually limited and precious, several cache optimization approaches in mobile networks have been proposed. In [18], Ahlehagh et al. proposed a video-aware caching scheme in radio access network (RAN). In [19], the caching framework in both the evolved packet core (EPC) and the RAN of LTE networks was designed to achieve better performance than single-tier only caching.

In 5G systems, the heterogeneous cellular network (HetNet) paradigm has been identified as a promising network architecture. In multi-tier HetNets, traffic offloading [20] via Device-to-device (D2D) communication has been used so as to optimize cache placement [21], while the joint design of the transmission and caching policies was also studied in [22]. Furthermore, as a supplement to network caching, device caching [23][24][25] was also investigated for offloading the base stations by reducing cellular transmissions. All the above approaches enhance the content delivery performance through cache optimization in wireless networks and provide us a hint why mobile caching can be used also for VR video to reduce the delivery latency.

Caching has been considered for VR video delivery. Following the pioneering idea that integrated networking, caching, and computing in wireless systems [26], edge caching and computing for 5G VR applications were studied in [27] and [28]. To exploit the benefits of edge computing in VR caching, the fundamental trade-offs between caching, edge computing, and computation-offloading-related communication cost for VR/AR applications were discussed in [29], [30], [31] and [32]. Given the relatively lax latency constraint, this class of techniques optimized the viewport fidelity efficiently for mobile VR video delivery. However, the considered scenarios cannot meet the rigid MTP requirement yet in practical applications due to the complex global system optimization process.

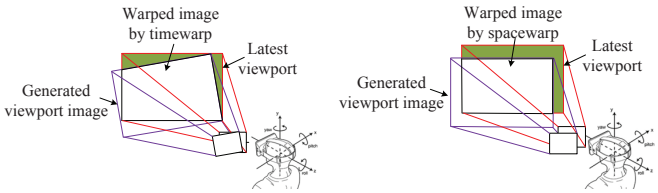
Considering the particular features of VR video, several VR video caching optimization approaches have been proposed. In [33], an optimization scheme for tiled VR video delivery in content delivery networks (CDN) was proposed. In this work, network stack optimization in the CDN was proposed as a method to reduce VR video transmission latency. In our previous work [34], we proposed a joint EPC and RAN caching scheme for tiled VR video. In [35], a FoV-aware caching policy was proposed to improve caching performance. In [36] and [37], the tile-based panoramic video caching optimization was studied. Even though the above studies can optimize VR video caching efficiency, they lack a cooperative optimization that is necessary for offloading the rendering computation in a multi-cell edge computing environment.

In content caching systems, request routing is the process of directing client requests to the best cache servers. Obviously, it is also a crucial procedure that affects end-to-end VR video delivery latency. Typically, routing decisions are updated every few minutes in the CDN [38]. To make traditional routing algorithms suitable for VR video caching, Poirion et al. [39] proposed a scalable request routing approach to optimize VR video delivery. So far, this paper was the one and only study aiming for VR video request routing. However, this approach made routing decisions for the whole-frame VR video request (not tile-based viewport request) in a fine-granular way and was not concerned with viewport requests on top of tile-based caching.

III. Background of Asynchronous MEC-UE Rendering

Rendering offloading can reduce rendering latency by making use of the computational resources of the MEC server. Unlike conventional remote rendering applications that render the displayed frame in a fixed viewpoint, VR video rendering requires a continuously time-varying viewport rendering. Since the viewport changes very rapidly, viewport rendering needs to be performed frame by frame at the MEC server so as to keep pace with the viewport changing speed.

Generally, besides the rendering operation, the necessary processing at the MEC server related to a VR video service includes decoding VR video and further encoding the rendered viewport. These processing operations can be performed sequentially. This sequential processing sometimes cannot meet the requirement of the MTP latency and results in missing delivery deadlines to the HMD. To further reduce rendering latency, parallel processing can be designed by asynchronously performing several procedures [4][40] between the MEC server and the UE, namely asynchronous MEC-UE rendering. This asynchronous rendering is based on the asynchronous timewarp (ATW) and asynchronous spacewarp (ASW) [41] techniques. Timewarp [42] works in tandem with spacewarp to modify a rendered VR image in terms of the very latest head tracking data. The basic principles of them are shown in Fig. 1. Generally, timewarp generates a VR viewport frame via a homogeneous transform based on the latest head orientation, and spacewarp uses motion estimation and camera translation of previous frames to predict the next viewport frame based on the very latest head position. To increase the perceived frame rate, the asynchronous operation is often used for timewarp and spacewarp to make them occur on another thread in parallel (asynchronously) with viewport rendering.



(a) Timewarp (b) Spacewarp

Fig. 1. Basic concept of timewarp and spacewarp

Fig. 2 illustrates the parallel processing pipeline of asynchronous MEC-UE rendering. The MEC server renders the $(n+1)$th frame of the viewport according to the latest viewport position and posture information (The orientation angles along the x, y and z axes that are called yaw, pitch and roll) before displaying the current nth frame at UE. When the UE receives the previously rendered viewport frame, it will be re-projected in terms of the latest viewport posture and position.

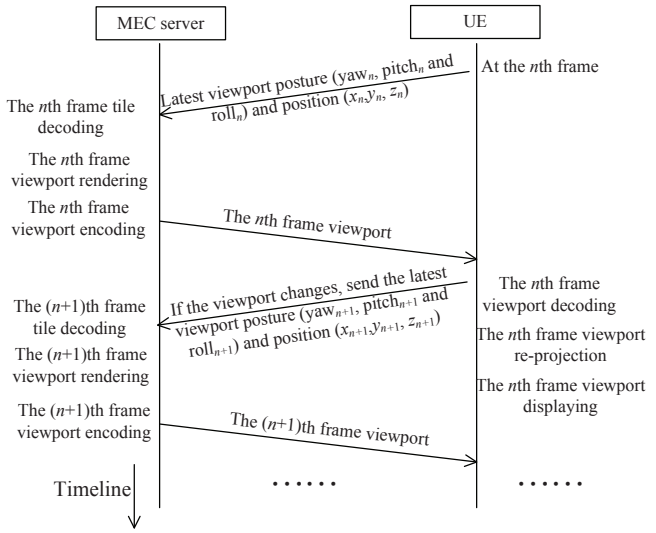


Fig. 2. The parallel processing pipeline of asynchronous MEC-UE rendering

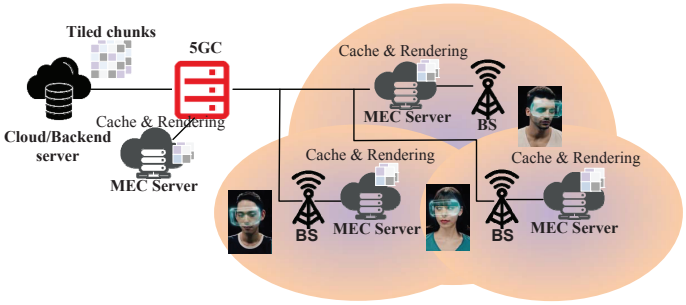


Fig. 3. VR video caching architecture over multi-cell 5G MEC networks

The re-projection is based on the combined ATW and ASW. Specifically, the received frame is first re-projected to the latest viewport posture with ATW and is then re-projected to the latest viewport position with ASW.

Another aspect is that tile decoding, viewport rendering at the cache node, and also decoding of the rendered viewport at the UE, can all be performed in parallel [41]. Moreover, the rendered frame will be encoded with temporal prediction from the previously rendered frames. Temporal prediction in the tile encoding will significantly reduce the volume of transmitted data and consequently shorten the transmission time of rendered data.

In one MEC server, the computational capability is also limited. When multiple users compete for one MEC server for viewport rendering, the rendering speed may be deteriorated due to the computation overload. To avoid this situation, we consider rendering-awareness in advance, that is during the stage of cache placement so as to reserve the appropriate computational resource for each cache node.

IV. System Model

A. Network model

In 5G mobile networks, gNBs are densely deployed. This creates a complex multi-cell environment [13] where users are concurrently in range of multiple gNBs with overlapping coverage. Usually, multiple MEC servers are associated with multi-cell gNBs and the 5G Core (5GC) network to enhance their computational capabilities, and they thus naturally form a two-tier serving topology in terms of their proximities to the users. In the stand-alone (SA) 5G architecture, direct communication between gNBs is already supported through the Xn interface. Thus, the operator can use multiple link paths (forwarding the content via direct communication among gNBs) to route content to the end user. Fig. 3 shows the system architecture of multi-cell 5G MEC networks for VR video streaming. The multiple MEC servers in 5G networks can cache VR video from the backend server/cloud and also perform real-time viewport rendering tasks for the end user.

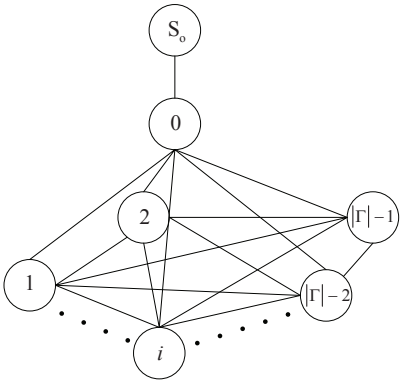


Fig. 4. Logical topology of cache system

In such multi-cell environment, caches are deployed in different RAN cells. The logical caching topology is abstracted in Fig. 4. MEC servers that are usually deployed in two tiers in mobile networks are used as a set of cache nodes $\Gamma = \{0, 1, ..., i, ..., |\Gamma| - 1\}$, where $|\Gamma|$ denotes the number of total cache nodes. One tier is the User Plane Function (UPF)/Access and Mobility management Function (AMF) position in 5GC network and it is numbered as 0. Another tier is located at gNBs in the multiple RAN cells and it is denoted as $i (i = 1, ..., |\Gamma|-1)$. Together with the backend source server S_o, three tiers are created in the proposed caching system. For ease of reference, the key notations introduced in the paper are listed in Table I.

B. VR Content Model

We assume that the VR content uses a separate cache space from the other types of content in the system. Though the available cache space for VR video content is affected by the volume of other types of cache data, the cache placement performance is independent of the cache refresh ratio for all other types of content. Given the overall cache space S for the VR videos in the system, the set of VR videos to be cached is denoted as $\mathcal{K}$, and the number of VR videos in $\mathcal{K}$ is computed as $|\mathcal{K}| = \frac{S}{\overline{S}_{VR}}$, where $\overline{S}_{VR}$ is the average data size of VR video streams. Thus, in terms of the content popularity ranking in a descending order, the top $|\mathcal{K}|$ VR videos in the rank list will be placed in the multi-cell 5G MEC network.

For VR video streaming, video frames are generally divided into several tiles that can be decoded independently when delivered. Tile partitioning for VR video is illustrated in Fig. 5. In the proposed caching system, the tiled streams are used

TABLE I
KEY NOTATIONS

Notation	Definition/explanation
Π	The set of chunks in one VR video
Γ	The set of cache nodes in the system
l_i	The unit latency for transferring VR video tiles from UPF/AMF to gNB in the ith cache node
l_0	The unit latency when transferring VR video tiles from the backend server to UPF/AMF
$l_{i,j}$	The unit latency when transferring VR video tiles between cells i and j
$v_{\tau,i}^{k,m,n}$	The tile at the nth row and the mth column in the τth chunk of the kth VR video at the ith cache node
$s_{\tau,i}^{k,m,n}$	The data size of the tile $v_{\tau,i}^{k,m,n}$ before rendering at the caching node i
$R(s_{\tau,i}^{k,m,n})$	The data size of the tile $v_{\tau,i}^{k,m,n}$ after rendering at the cache node i
$p_{\tau,i}^{k,m,n}$	The tile popularity of $v_{\tau,i}^{k,m,n}$
$C(s_{\tau,i}^{k,m,n})$	The rendering computation costs for the tile $v_{\tau,i}^{k,m,n}$ at the caching node i
B_i	The maximal storage space for the ith cache node
C_i	The computational constraint for rendering task at the ith cache node
$req_{\tau,i}^{k,m,n}$	The request for $v_{\tau,i}^{k,m,n}$

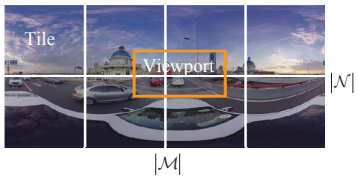


Fig. 5. Illustration of the tile partition

as the cache data. For the kth VR video in the caching system, $|\mathcal{M}| \times |\mathcal{N}|$ tiles are obtained after the tiling process, where $\mathcal{M} = \{1, 2, ..., m, ... |\mathcal{M}|\}$ and $\mathcal{N} = \{1, 2, ..., n, ... |\mathcal{N}|\}$, and $|\mathcal{M}|$ and $|\mathcal{N}|$ are the total tile columns in vertical dimension and tile rows in horizontal dimension in one frame, respectively. Similarly, in the temporal dimension, the tiled VR videos were also segmented into many chunks with equal playback length. Assume each tiled VR video is segmented into $|\Pi|$ chunks, where $\Pi = \{1, 2, ..., \tau, ... |\Pi|\}$ and τ is the chunk index.

C. System Flow

Due to the limitation of the FoV of human eye, only a small part in the whole frame is watched in one moment. This means that only the VR video tiles covered by the viewport are displayed on the user's HMD in one moment. Thus, the UE needs to interactively send the viewport request to the server for watching the 360-degree content. In the 5GC, there is a centrally-deployed content controller, which is connected to UPF/AMF. The content controller is responsible for recognizing the viewport requests from UEs in each MEC cell and then performs the routing algorithm for the user requests. On the UE side, the UE tracks the HMD position and posture in real-time and determines the necessary tiles for rendering the viewport. Once the requested viewport moves out the region of current tiles, the UE will send the new HMD position and posture data to the content controller for next viewport request.

When the viewport request is launched, the request will be served by the cache node within its local cell. Once the local cell cache node cannot satisfy the request, the request will be forwarded to the content controller. When the request arrives to the content controller, the controller will decide whether the requested viewport tiles can be found in the cache system by looking up its global view of cache content. Once the cache nodes for serving the viewport request are determined, the asynchronous MEC-UE rendering service will be initiated to serve the current request. The viewport generally covers several tiles that are possibly distributed at different cache nodes. Thus, the UE requests simultaneously multiple tiles for viewing a viewport. Usually, how the contents are placed at different caching nodes is based on their request probability. Applying this idea to tiled VR video caching, the prediction of the popularity of each tile at different cache nodes needs to be done first.

D. Tile Popularity Model

Given the dynamic nature of user viewport request, accurate tile popularity prediction is of paramount importance for caching performance. Tile popularity is closely related to the VR video content saliency. In this subsection, we introduce the tile popularity prediction methodology that is based on fusing the tile saliency and the VR video popularity.

The interactive viewing nature of VR video makes its saliency detection different from that of traditional video. Considering both the features of the human visual system and the viewport viewing feature in saliency estimation, in the previous work [43] we proposed a VR saliency estimation approach by fusing the visual frequency feature and the viewing behavior feature. In that work, we used a normalized and maximum fusion approach. Though the used fusion approach showed that it can achieve good performance, it cannot fully exploit the inter-dependency among the two features towards building the final saliency map. By taking into account the advantage of the Choquet integral [44] in non-additive measurements, in this work we propose to use this Choquet integral to fuse the visual frequency feature and the viewing behavior feature when performing saliency estimation, and then predict the popularity of each tile. The pipeline of tile popularity prediction is illustrated in Fig. 6.

1) Choquet Integral: The Choquet integral [44] has been shown to be an effective nonlinear aggregation tool for information fusion. Its non-additivity of the signed fuzzy measure is able to effectively capture the interaction among the contributions of feature attributes to the objective attribute. Let $Z = \{z_1, z_2, \cdots, z_h\}$ be a set of feature attributes (i.e. the factors contributing on saliency), where h is the number of feature attributes. Assume the attribute data consists of ℓ observations $f_1, f_2, \cdots, f_\ell$ of feature attributes $z_1, z_2, \cdots, z_h$ and the objective attribute y_o (such as the final saliency). Each observation of $z_1, z_2, \cdots, z_h$ can be regarded as a transform function $f : Z \rightarrow (-\infty, +\infty)$. Correspondingly, the jth observation of $z_1, z_2, \cdots, z_h$ is denoted by f_j, and we write $f_{ji} = f_j(z_i)$ where $1 \leq i \leq h$ and $1 \leq j \leq \ell$.

In the Choquet integral model, the inter-dependency among the feature attributes Z contributing to objective attribute y_o

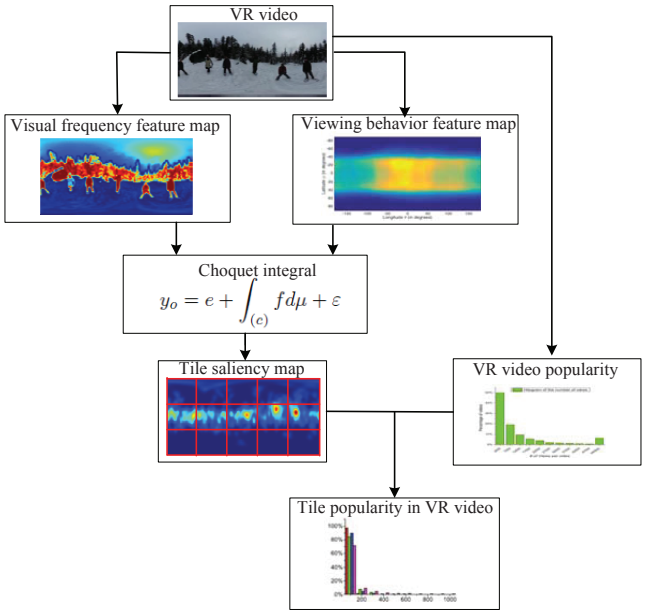


Fig. 6. Tile popularity prediction pipeline

is characterized by a set of functions $\mu : P(Z) \rightarrow \Re$ with $\mu(\emptyset) = 0$, where $P(Z)$ is the power set of Z, and $\Re$ is the real number. Furthermore, the interaction among Z towards y_o can be expressed as a multi-regression form [45],

$$y_o = e + \int_{(c)} f d\mu + \varepsilon \tag{1}$$

where e is a constant and ε is a normally distributed random perturbation following $N(0, \delta^2)$ with expectation 0 and variance δ^2. For the given set function μ and function f, the integral in (1) is calculated as

$$\int_{(c)} f d\mu = \sum_{j=1}^{2^h-1} (\omega_j \cdot \mu_j) \tag{2}$$

where μ_j denotes the fuzzy measurement and ω_j denotes the corresponding contribution coefficient of μ_j in the whole integral. Let $H_j = \min_{i:\mathrm{frc}\left(\frac{j}{2^i}\right) \in \left[\frac{1}{2},1\right)} (f(z_i)) - \max_{i:\mathrm{frc}\left(\frac{j}{2^i}\right) \in \left[0,\frac{1}{2}\right)} (f(z_i))$, where $frc(\frac{j}{2^i})$ is the fractional part of $\frac{j}{2^i}$ and the maximum operation on the empty set is zero, and then

$$\omega_j = \begin{cases} H_j, & \text{if } H_j > 0 \text{ or } j = 2^h - 1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

2) Tile Saliency Prediction: We differentiate from the traditional saliency detection approach, by extracting the spatial frequency information in the viewport plane. Based on the Difference-of-Gaussian (DoG) model [43] in the viewport plane, the visual responses of the human eye on viewports are captured and then the center-surround contrast of them over the whole panoramic image is computed. Finally, the visual frequency feature map for saliency detection is obtained with high accuracy.

In the 360-degree space, the popularity of different FoVs is affected by the human viewing habits [46]. The viewing probability for each latitude over the 360-degree VR image generally follows a Gaussian distribution [43]. Hence, the viewing behavior feature map is attained by fitting a Gaussian model using the users' head motion data that are known in advance.

Together with the visual frequency feature map, the two feature maps are considered as feature attributes z_i $(i = 1, ..., h)$ towards the final saliency map. After that, the Choquet integral transform in (1) is used to fuse the two feature maps to form the saliency map of each VR image.

By having each pixel saliency in one saliency map, the average saliency value $\hat{v}^{k,u,m,n}$ over all pixels in the tile $v^{k,u,m,n}$ of the uth frame in one chunk is computed. Assume that $v_\tau^{k,m,n}$ denotes the tile at the nth row and the mth column in the τth chunk of the kth VR video and the chunk includes $\mathcal{L}$ frames. The average saliency over the $\mathcal{L}$ frames in $v_\tau^{k,m,n}$ is $\bar{v}_\tau^{k,m,n} = \frac{1}{\mathcal{L}} \sum_{u=1}^{\mathcal{L}} \hat{v}^{k,u,m,n}$, and it is used to compute the saliency of the tile at the nth row and mth column in the τth chunk of the kth VR video as,

$$p_\tau^{k,m,n} = \frac{\bar{v}_\tau^{k,m,n}}{\sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} \bar{v}_\tau^{k,m,n}} \tag{4}$$

3) Tile Popularity Prediction: Same as the conventional video, the VR video popularity also follows the Zipf law [47]. The kth VR video's popularity at the ith cache node is described as

$$p_i^k = \beta / \gamma_{i,k}^{\alpha} \tag{5}$$

$$\beta = (\sum_{k=1}^{|\mathcal{K}|} \gamma_{i,k}^{-\alpha})^{-1} \tag{6}$$

where $\alpha \geq 0$ is the parameter of Zipf model and $\gamma_{i,k}$ is the rank of the kth VR video in $\mathcal{K}$ at the ith cache node. Let $p_{\tau,i}^{k,m,n}$ be the popularity requested from the ith cache node for the tile at nth row and mth column in the τth chunk of the kth VR video and p_i^k the popularity of kth VR video requested from the ith cache node, then we have

$$p_{\tau,i}^{k,m,n} = p_i^k \times p_\tau^{k,m,n} \tag{7}$$

Usually, the knowledge of video popularity is obtained from the complete frame content. However, for VR video, tiles are requested depending on how the viewport is viewed and so the popularity difference among tiles in the spatial domain is not captured by the traditional video popularity model. The proposed tile popularity prediction model in (7) uses the tile saliency distribution to predict the viewport content popularity based on the complete frame video popularity.

By using practical viewport request data available in [48], we measured the accuracy of the proposed tile popularity model. Fig. 7 shows the average correlation between the actual tile rank and the predicted one for each category of VR videos. The three categories of VR videos including (i) Computer-Generated, fast-paced (CG(Fast)), (ii) Natural Image, fast-paced (NI(Fast)) and (iii) Natural Image, slow-paced (NI(Slow)) are used. The rank correlation in Fig. 7 verifies that the proposed tile popularity prediction model can obtain a accuracy larger than 0.75, and thus can be used to guide the tile cache placement optimization.

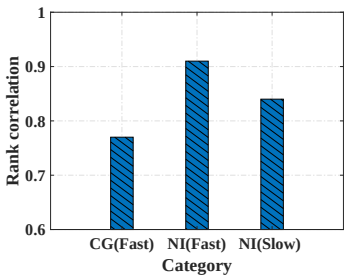


Fig. 7. Tile popularity prediction accuracy

V. Problem Formulation and Solution

For VR video streaming, caching at the edge of mobile networks is a crucial procedure for reducing the end-to-end latency. Hence, VR video caching should be carefully optimized under the storage and computation constraints of multiple MEC servers. Traditional VR video caching approaches deal with data caching and viewport rendering as two independent procedures. Namely, the tiles are cached at different MEC servers during the cache placement stage and the rendering operation is performed on the UE during the viewport request stage. When the rendering computation is offloaded to the MEC server, the viewport rendering and tile caching can be also done separately at the MEC server, and the rendering service placement can also be optimized by allocating computational resources among different MEC servers from a global perspective. However, such global optimization will increase the latency for transferring the rendering-dependent data among gNBs. To avoid this additional latency, we propose to couple tile caching with rendering, and consider the rendering optimization during the tile cache placement stage. To this aim, we design a rendering-aware tile caching scheme. With this idea, computational resources are optimized locally for each MEC server. As rendering is coupled with caching at the same MEC server, it does not introduce additional processing latency.

In the proposed caching system, the cache nodes automatically place or update the VR video tiles according to the cache placement optimization algorithm. With our optimization, the VR video tiles within each user viewport are more likely to be cached near the UE. Once a viewer requests a VR video viewport, the cell cache node will check whether the requested viewport tiles already exist in the cell cache. If the requested data is available, the cache node in the local cell will serve the request by transmitting the rendered viewport data to the UE. If the requested data is not available locally in this cell, the request will be transferred to the UPF/AMF cache node and other cell cache nodes for checking whether they have cached the VR video tiles that are being requested. If they are cached, the cache node with the minimal bandwidth cost will serve the request. Specifically, the VR video tiles will be rendered into the viewport data and then transmitted to the corresponding cell, and finally transmitted to UE. If none of the cache nodes had already cached the requested tiles, the request can only be served by the source server on the Internet.

Note that the proposed rendering-aware caching systems are essentially networks of interconnected caches. Caching optimization needs to decide which cache node to cache what VR video tiles and how to route the user's viewport request to the cache node where the requested item is available. Therefore, two sets of optimization variables need to be decided. One is a 0-1 cache placement decision variable $x_{\tau,i}^{k,m,n}$ that indicates whether the VR video tile $v_{\tau,i}^{k,m,n}$ in the kth VR video is cached in the cache node i. If node i had already cached the tile $v_{\tau,i}^{k,m,n}$, $x_{\tau,i}^{k,m,n}=1$; otherwise $x_{\tau,i}^{k,m,n}=0$. Another is the 0-1 variable $y_{\tau,i,j}^{k,m,n}, (i \neq j)$, and it indicates whether the request $req_{\tau,i}^{k,m,n}$ at the ith node is routed to the jth node. If the $req_{\tau,i}^{k,m,n}$ is satisfied by the local cache node i, $y_{\tau,i,i}^{k,m,n}$=1. Otherwise $y_{\tau,i,i}^{k,m,n}$=0. For all the video tiles in the system, the cache placement and request routing decision vectors are given as

$$\boldsymbol{x} = (x_{\tau,i}^{k,m,n} \in \{0,1\} : \tau \in \Pi, i \in \Gamma, k \in \mathcal{K}, m \in \mathcal{M}, n \in \mathcal{N}) \tag{8}$$

$$\begin{aligned} \boldsymbol{y} =& (y_{\tau,i,j}^{k,m,n} \in \{0,1\} : \tau \in \Pi, i \in \Gamma, j \in \Gamma \backslash \{0,i\}, \\ & k \in \mathcal{K}, m \in \mathcal{M}, n \in \mathcal{N}) \\ & \cup (y_{\tau,i,i}^{k,m,n} \in \{0,1\} : \tau \in \Pi, i \in \Gamma \backslash \{0\}, k \in \mathcal{K}, \\ & m \in \mathcal{M}, n \in \mathcal{N}) \\ & \cup (y_{\tau,i,0}^{k,m,n} \in \{0,1\} : \tau \in \Pi, i \in \Gamma, k \in \mathcal{K}, \\ & m \in \mathcal{M}, n \in \mathcal{N}) \end{aligned} \tag{9}$$

where $y_{\tau,i,0}^{k,m,n}$ indicates whether the request $req_{\tau,i}^{k,m,n}$ at the ith node is routed to the UPF/AMF cache node.

The rendering-aware cache placement and request routing optimization need to satisfy several constraints. First, the user's viewport request needs to be routed to one of the cache nodes or the backend server, and thus

$$\begin{aligned} & \{ \sum_{j \in \Gamma \backslash \{0,i\}} y_{\tau,i,j}^{k,m,n} \} + y_{\tau,i,i}^{k,m,n} + y_{\tau,i,0}^{k,m,n} \leq 1, \forall i \in \Gamma, \\ & \forall \tau \in \Pi, \forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \end{aligned} \tag{10}$$

Second, the request can only be routed to the cache node where the requested item is available,

$$\begin{aligned} & y_{\tau,i,j}^{k,m,n} \leq x_{\tau,j}^{k,m,n}, \ \forall i \in \Gamma \backslash \{0\}, \forall j \in \Gamma \backslash \{0,i\}, \forall \tau \in \Pi, \\ & \forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \\ & y_{\tau,i,i}^{k,m,n} \leq x_{\tau,i}^{k,m,n}, \ \forall i \in \Gamma \backslash \{0\}, \forall \tau \in \Pi, \forall k \in \mathcal{K}, \\ & \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \\ & y_{\tau,i,0}^{k,m,n} \leq x_{\tau,0}^{k,m,n}, \ \forall i \in \Gamma, \forall \tau \in \Pi, \forall k \in \mathcal{K}, \\ & \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \end{aligned} \tag{11}$$

Third, the total amount of VR videos that are placed in each MEC server should be no more than its storage limitation,

$$\sum_{\tau \in \Pi} \sum_{k \in \mathcal{K}} \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} s_{\tau,i}^{k,m,n} \cdot x_{\tau,i}^{k,m,n} \leq B_i, \forall i \in \Gamma \tag{12}$$

where B_i is the maximal storage space for the ith cache node.

Fourth, the total rendering computations that users concurrently request can not exceed the computational capability of each cache node and this constraint explicitly embodies the additional rendering-aware function compared to the traditional

cache placement approach,

$$\sum_{\tau\in\Pi}\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}}\eta_i\cdot C(s_{\tau,i}^{k,m,n})\cdot x_{\tau,i}^{k,m,n}\leq C_i,\forall i\in\Gamma \tag{13}$$

where η_i denotes the ratio of concurrent rendering requests for all cached tiles in the ith cache node, C_i denotes the computational constraint for rendering task at the ith cache node, and $C(s_{\tau,i}^{k,m,n})$ denotes the rendering computation costs for the tile $v_{\tau,i}^{k,m,n}$ at the ith cache node.

To formally capture this proposed VR video caching problem, let us denote as l_i the unit latency for transferring VR video tiles from UPF/AMF to gNB in the ith cache node, l_0 as the unit latency when transferring tiles from the backend server to UPF/AMF and $l_{i,j}$ as the unit latency when transferring tiles between cells i and j. According to the content placement strategy, there are four ways to fetch a viewport for viewers: i) If the cache node i can satisfy the request from UE locally for the tile $v_{\tau,i}^{k,m,n}$, the unit latency saving is l_0+l_i, and $l_0\neq 0$, $l_i\neq 0$; ii) If the request cannot be satisfied by cell node i but can be satisfied by the other cache nodes in adjacent cells, for instance node $j\in\Gamma\backslash\{i,\ 0\}$, the unit latency savings can be written as $l_0+l_i-l_{i,j}$ and $l_{i,j}\neq 0$; iii) If the request can be satisfied by the UPF/AMF cache node in 5GC, the unit latency saving is l_0; iv) If the request can only be satisfied by the backend server on the Internet, the unit latency saving is 0.

More specifically, when the request $req_{\tau,i}^{k,m,n}$ at the ith node is satisfied by the UPF/AMF cache node in the 5GC, the unit latency saving is $l_0\cdot y_{\tau,0}^{k,m,n}$. Similarly, when the request $req_{\tau,i}^{k,m,n}$ is satisfied by another cell gNB j, the maximal unit saving latency is $\max\limits_{j\in\Gamma\backslash\{i,\ 0\}}\{(l_0+l_i-l_{i,j})\cdot y_{\tau,i,j}^{k,m,n}\}$.

For all the requests in the system, the objective of the rendering-aware VR video caching problem is to find the joint tile cache placement and routing policy by maximizing the overall latency savings when compared to obtaining VR video from the source server:

$$\begin{aligned}\text{P1}:\quad &\max_{\boldsymbol{x},\boldsymbol{y}}\sum_{\tau\in\Pi}\sum_{k\in\mathcal{K}}\sum_{i\in\Gamma}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}}\Theta_{\tau,i}^{k,m,n}\\ \text{s.t.}\quad &\text{constraints}\ \ (8)-(13)\end{aligned} \tag{14}$$

where

$$\begin{aligned}\Theta_{\tau,i}^{k,m,n}=&R(s_{\tau,i}^{k,m,n})\cdot[y_{\tau,i,i}^{k,m,n}(l_0+l_i)+(1-y_{\tau,i,i}^{k,m,n})\\ &\cdot\max\{\max_{j\in\Gamma/\{i,\ 0\}}\{(l_0+l_i-l_{i,j})y_{\tau,i,j}^{k,m,n}\},\ l_0\cdot y_{\tau,i,0}^{k,m,n}\}]\end{aligned} \tag{15}$$

where $R(s_{\tau,i}^{k,m,n})$ denotes the data size of tile $v_{\tau,i}^{k,m,n}$ after rendering.

Eq.(14) formulates the joint rendering-aware tile cache placement and viewport request routing problem. It is an integer problem that is typically hard to solve. Moreover, the above formulation assumes the updating of viewport request routing and the tile cache placement are with the same timescale. This assumption is not consistent with the practical case and it also can not satisfy the extremely stringent latency requirement of VR applications. Usually, tile cache placement and viewport request routing occur at different stages and also with different time granularity. The tile cache placement is updated in longer timescales, while the viewport request routing is updated in a very short time span. Hence, we propose a two-stage optimization so as to first optimize the rendering-aware tile cache placement sub-problem, and then optimize the request routing sub-problem given the tile cache placement solution that is previously obtained.

A. Rendering-aware Cache Placement

The tile cache placement policy directly affects the tile delivery latency. Following the formulation of P1, the objective of tile cache placement is still to maximize the latency savings for all the tile requests. However, during the tile cache placement stage both the requests and their routing policies are not known. Consequently, in the formulation of tile cache placement, we have to assume that the tiles are probably requested based on their popularity. Moreover, the request is assumed to be routed to the cache node where the hit tile can be delivered with maximal saving latency (when compared to obtaining VR video from the source server).

Assume that the latency saving $L_{\tau,i}^{k,m,n}$ when the request for the tile $v_{\tau,i}^{k,m,n}$ at node i is satisfied by the UPF/AMF cache node is given by

$$L_{\tau,i}^{k,m,n}=l_0\cdot x_{\tau,0}^{k,m,n} \tag{16}$$

Also, the maximal saving latency $D_{\tau,i,j}^{k,m,n}$ when the request for the tile $v_{\tau,i}^{k,m,n}$ at node i is hit in another cell gNB j is defined as

$$D_{\tau,i,j}^{k,m,n}=\max_{j\in\Gamma\backslash\{i,\ 0\}}\{(l_0+l_i-l_{i,j})x_{\tau,j}^{k,m,n}\} \tag{17}$$

and it indicates the request for the tile $v_{\tau,i}^{k,m,n}$ from UE connecting to cache node i is transferred to cache node j.

Differently from P1, we consider the overall latency savings for all the possible requests (when compared to obtaining VR video from the source server) as the objective function for the tile cache placement problem. It is calculated as

$$\Psi=\sum_{\tau\in\Pi}\sum_{k\in\mathcal{K}}\sum_{i\in\Gamma}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}}\phi_{\tau,i}^{k,m,n} \tag{18}$$

with

$$\begin{aligned}\phi_{\tau,i}^{k,m,n}=&p_{\tau,i}^{k,m,n}\cdot R(s_{\tau,i}^{k,m,n})[x_{\tau,i}^{k,m,n}\cdot(l_0+l_i)\\ &+(1-x_{\tau,i}^{k,m,n})\cdot\max\{L_{\tau,i}^{k,m,n},D_{\tau,i,j}^{k,m,n}\}]\end{aligned}$$

where $p_{\tau,i}^{k,m,n}$ denotes the popularity (also indicates the request probability) of the tile $v_{\tau,i}^{k,m,n}$ at the ith cache node. Thus, when considering rendering-aware caching, the rendering computational resource is considered to be a constraint for cache placement. By maximizing the latency saving under the storage and computational constraints for each cache node, the tile cache placement problem can be formulated as

$$\begin{aligned}\text{P2}:\quad &\max_{\boldsymbol{x}}\ \Psi\\ \text{s.t.}\quad &\text{constraints}\quad (8),(12)\ \text{and}\ (13)\end{aligned} \tag{18}$$

where $\boldsymbol{x}$ is a vector indicating the tile cache placement solution for P2.

Problem P2 is a typical Multi-Knapsack Problem (MKP) that is NP-hard and remarkably complex. To solve it we have

to apply relaxation and decompose the complex P2 into a series of standard single knapsack problems. The upper bound of P2 can be attained by solving the subproblems in a parallel fashion.

The Lagrangian relaxation of P2, relative to a nonnegative vector ($\lambda_i \geq 0$), is defined as

$$\begin{aligned} \text{P3}: \quad & \max_{\boldsymbol{x}} \ \Phi(\boldsymbol{x}, \boldsymbol{\lambda}) \\ \text{s.t.} \quad & \text{constraints} \quad (8) \text{ and } (12) \end{aligned} \tag{19}$$

where

$$\begin{aligned} &\Phi(\boldsymbol{x}, \boldsymbol{\lambda}) \\ &= \sum_{\tau\in\Pi}\sum_{k\in\mathcal{K}}\sum_{i\in\Gamma}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} \phi_{\tau,i}^{k,m,n} \\ &\quad - \sum_{i\in\Gamma}\lambda_i \cdot [\sum_{\tau\in\Pi}\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} (\eta_i \cdot C(s_{\tau,i}^{k,m,n}) \cdot x_{\tau,i}^{k,m,n}) - C_i] \\ &= \sum_{i\in\Gamma}\{\sum_{\tau\in\Pi}\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} [\phi_{\tau,i}^{k,m,n} - \lambda_i \cdot (\eta_i \cdot C(s_{\tau,i}^{k,m,n}) \cdot x_{\tau,i}^{k,m,n})]\} \\ &\quad + \sum_{i\in\Gamma}\lambda_i C_i \end{aligned} \tag{20}$$

and $\boldsymbol{\lambda} = (\lambda_0, ...\lambda_i, ..., \lambda_{|\Gamma|-1})$ with λ_i indicating the ith dual variable. The optimal value of P3 gives an upper bound of the optimum value of P2 for arbitrary nonnegative λ_i. To solve P3, the approximately optimum value of λ_i needs to be obtained first. Then, the Lagrangian dual problem is given by

$$\text{P4}: \quad \min_{\boldsymbol{\lambda}} \ \Phi(\boldsymbol{x}, \boldsymbol{\lambda}) \tag{21}$$

and the approximately optimal λ_i can be achieved by a sub-gradient method [49]. Assume that $h_{\tau,i}^{k,m,n} = \phi_{\tau,i}^{k,m,n} - \lambda_i \cdot (\eta_i \cdot C(s_{\tau,i}^{k,m,n}) \cdot x_{\tau,i}^{k,m,n})$, the ith sub-problem (the decomposed ith standard single knapsack problem) is given as

$$\begin{aligned} &\max_{\boldsymbol{x}} \sum_{\tau\in\Pi}\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} h_{\tau,i}^{k,m,n} \\ &\text{s.t.} \quad \text{constraints} \quad (8) \text{ and } (12) \end{aligned} \tag{22}$$

Let us denote the maximal value of $\sum_{\tau\in\Pi}\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} h_{\tau,i}^{k,m,n}$ for the ith single knapsack problem as π_i, the upper bound of the original multiple knapsack problem P2 can be attained as $\sum_{i\in\Gamma}(\pi_i + \lambda_i \cdot C_i)$. Next, the branch and bound [50] algorithm that includes branching, fathoming, and bounding is used to seek the optimal solution for P2. During the bounding stage, Lagrangian relaxation is used at each node of the search tree to provide an upper bound so as to discard the subset of solutions that cannot possibly contain an optimal one. Moreover, a heuristic procedure for determining the lower bound is also used to evaluate and prune the solution branches [51]. The details of the algorithm follow the one described in chapter of 6.4.3 in [51] and are thus omitted.

For the proposed solution, its computational complexity comes largely from the branch and bound algorithm. The branch and bound algorithm uses an enumeration of the search space and so it has an exponential time complexity in the worst case. However, it is efficient in the average case because many branches can be terminated very early with the upper and lower bounds. It is difficult to formulate the average-case complexity due to its dynamic termination of the search. Thus, we evaluate the average-case complexity of the proposed tile cache placement schemes in terms of the actual running time in the following Section VI.

Algorithm 1: Low-delay request routing

Input: Request $\sum_{m\in\mathcal{M}_v}\sum_{n\in\mathcal{N}_v} req_{\tau,i}^{k,m,n}$
Output: Request routing decision

```
for m ∈ M_v do
    for n ∈ N_v do
        if The request req^{k,m,n}_{τ,i} is found locally in the ith caching node then
            The ith caching node renders the viewport data r^{k,m,n}_{τ,i}
            y^{k,m,n}_{τ,i,i} =1;
        else
            while The cache controller discovers the request req^{k,m,n}_{τ,i} from the ith cache node do
                if Request req^{k,m,n}_{τ,i} is found in the caching system by looking up the global cache view then
                    Select the best proximal cache node p_b (p_b ≠ i) in terms of Eq. (24) to render viewport data r^{k,m,n}_{τ,i}
                    B^u_i = B^u_i − R̂(s^{k,m,n}_{τ,i})
                    y^{k,m,n}_{τ,i,p_b} =1, (p_b ≠ i);
                else
                    Select the backend server to render the request for viewport data r^{k,m,n}_{τ,i}
        Update the available bandwidth for all cache nodes
```

B. Low-delay Viewport Request Routing

In the formulation of P1, the objective is to maximize the overall latency savings for delivering all the cached tiles. For the viewport request routing problem, this objective with such global optimization is not suitable for the practical VR delivery applications. VR video requests are significantly different from those of traditional video. Usually, the user interactively requests the viewport and the request changes frequently. The high dynamic requests demand a rapid response at the cache server, that is a very tight latency bound on the viewport request routing. However, the viewport request routing optimization in P1 is a global optimization for a relatively long time-span. More specifically, video request routing periodically performs a globally optimized scheduling of the aggregated requests by considering the cache node workload and the network conditions. This type of request routing requires more time to aggregate requests, which means significant waiting latency. Thus, globally optimized routing cannot meet the VR latency requirement.

To satisfy the viewport request latency, a straightforward way is to make routing decisions on the time granularity of

each request. This is a locally optimized approach and request routing decisions are sequentially made for each request. After the request $req_{\tau,i}^{k,m,n}$ is launched, the request will be served locally by the ith cache node. Once the ith cache node cannot satisfy the request, it will be forwarded to content controller and the the content controller will discover the best proximal cache node to serve the request from its global view of the content placement.

Let $P_{\tau,i,j}^{k,m,n} = \max_{j \in \Gamma/\{i,\,0\}} \{(l_0 + l_i - l_{i,j}) y_{\tau,i,j}^{k,m,n} \cdot R(s_{\tau,i}^{k,m,n})\}$ and $Q_{\tau,i,0}^{k,m,n} = l_0 \cdot y_{\tau,i,0}^{k,m,n} \cdot R(s_{\tau,i}^{k,m,n})$. By maximizing the saved latency cost relative to obtaining VR video from the backend server, the problem of seeking the best proximal cache node for each request is formulated as

$$\begin{aligned} & \max\{P_{\tau,i,j}^{k,m,n},\ Q_{\tau,i,0}^{k,m,n}\} \qquad (23) \\ \text{s.t.} \quad & y_{\tau,i,j}^{k,m,n} \le x_{\tau,j}^{k,m,n},\ \forall i \in \Gamma\backslash\{0\}, \forall j \in \Gamma\backslash\{i\}, \forall \tau \in \Pi, \\ & \forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \\ & \sum_{j \in \Gamma\backslash\{i\}} y_{\tau,i,j}^{k,m,n} \le 1, \forall i \in \Gamma\backslash\{0\}, \forall j \in \Gamma\backslash\{i\}, \forall \tau \in \Pi, \\ & \forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \end{aligned}$$

where the first constraint indicates that a tile request can be routed to the jth cache node only when it has already cached the requested tile and the second constraint imposes the condition that one request must be directed to at most one cache node.

Eq. (24) is easily solved by an exhaustive search algorithm. Considering the request load changes, the uplink bandwidth of the cache node is updated on-time after initiating the viewport request. The specific low-delay per request routing algorithm is shown in Algorithm 1. In Algorithm 1, B_i^u denotes the available uplink bandwidth of the cache node i and $\hat{R}(s_{\tau,i}^{k,m,n})$ denotes the bit-rate of the rendered viewport data $r_{\tau,i}^{k,m,n}$ that corresponds to the tile $v_{\tau,i}^{k,m,n}$. For each request, the exhaustive search algorithm has to check every single tile in the requested viewport. Thus Algorithm 1 requires a time complexity of $O(|\mathcal{M}_v| \cdot |\mathcal{N}_v|)$, where $\mathcal{M}_v$ and $\mathcal{N}_v$ are the sets of tile column indexes in vertical dimension and tile row indexes in horizontal dimension respectively in one viewport request.

For mobile applications, the UE may possibly move. When the UE moves out the coverage of its local gNB, the proposed routing algorithm can easily adapt to the UE handover process among gNBs. Once the UE switches between the gNBs during the serving period of one request, the request routing will be activated by default to re-select the suitable cache node to continue to serve the current request.

VI. Experimental results

To evaluate the performance of the proposed rendering-aware caching scheme, a detailed custom MATLAB simulation tool was developed. The VR video clips in Youtube [52] [53] and SJTU Media Lab [54] were downloaded for simulating tile caching. The videos with 4K (3840×1920) or 8K (7680×3840) spatial resolution were encoded with tiles in HEVC [55]. The VR video popularity follows a Zipf model. The tile popularity for each VR video was estimated by the saliency-assisted approach. To make the tile popularity estimation more accurate, the chunk playback length was set to 1 second in the experiment.

During the simulation, besides the 5GC cache node, we considered a multi-cell MEC network with 36 gNBs that are regularly deployed on a grid network inside a 1000m×1000m area, where each gNB covers a circular region of 150m radius. One gNB is accompanied with one MEC server. In this setting, one MEC server has at least three adjacent MEC servers for transferring the tile request. For each gNB, 802.11ad link model [56] was used to simulate the new radio features in 5G system. Based on practical measurements, the uplink and downlink bandwidths were set to 500Mbps and 1500Mbps, respectively.

The VR video tiles are usually concurrently requested, while both the amount and the accumulation speed of requests directly reflect the payload of the cache node. To accurately characterize the payload on each cache node, we modeled the request queuing process. We also assumed 50000 potential requests of viewports (One VR video clip may incur a lot of viewport requests) that follow Poisson arrival and departure model with mean inter-arrival time of 3ms. The number of concurrent active requests was estimated by an $\mathrm{M/M/\infty}$ queuing model [57] that follows Little's theory [58] with $N_v = \lambda_v \cdot T_a$, where T_a is the request active time (denotes the duration time that the request keeps unchanged) and $1/\lambda_v$ is the mean request inter-arrival time.

Viewport requests for each video were simulated from traces. The traces were recorded from the 32 viewers for watching the VR test video [53]. For each request in the trace, the rendering time, rendering-related encoding and decoding time were also recorded. We assume that the computed data volume for rendering (given in bits) in one CPU cycle on the MEC server is Z_b and the computational capability (cycles/second) of MEC server CPU is Z_c. Accordingly, the rendering computational capability of MEC server is $Z_b \cdot Z_c$ (bits/second), and thus it can be indirectly characterized by the rendering time (given in seconds). The actual trace data of the rendering time were scaled to those of the MEC server to accurately simulate the impact of MEC on rendering speed. The detailed parameters in the simulations are shown in Table II. Since the network was assumed to not be congested during the stage of cache placement, the fixed delay parameters l_0, l_i and $l_{i,j}$ in Table II were used. However, during the request routing stage, these delay parameters were dynamically estimated in terms of the available bandwidth.

In the proposed Rendering-Aware Caching with Offloaded Rendering plus Low-delay request routing (RAC-OR-L) scheme, the caching result is obtained by solving (19) and then the low-delay routing algorithm is used for each viewport request. We also considered a decoupled rendering and caching scheme as a reference since it is usually adopted by state-of-the-art VR video caching systems. In addition, the widely adopted Most Popular Content (MPC) [59] caching policy was also used for comparison. Regarding routing, the scalable request routing algorithm [39] with batch processing was used for comparison. In this way, a batch of viewport requests are directed towards the best proximal group of servers based on

TABLE II
SIMULATION PARAMETERS

Parameters	Values
Viewport size	1200×1080
Tile size	384×240
Chunk length	1s
RAN cache number	36
Cache size per base station B_i	10G
UE number per base station	varying from 10 to 100
Zipf model parameter α	0.8
l_0	100ms
l_i	10ms
$l_{i,j}$	varying from 2 to 8 ms
C_i	1440s
η_i	varying from 0.2 to 0.4
Z_c	50GHz
Z_b	1Mb
Downlink bandwidth for each node	1500Mbps
Uplink bandwidth for each node	500Mbps
Request arrivals	Poisson, mean inter-arrival time per request = 3ms
Request active time	200ms

a coarse granularity of time slot. In this paper, we assume that the routing time slot length in scalable routing algorithm is 10 ms for aggregating each batch of viewport requests.

Thus, except the proposed scheme, a total of four baseline schemes were simulated:

- *Decoupled Rendering and Caching with Offloaded Rendering plus Low-delay request routing (DRC-OR-L)*: In this scheme, the caching result is obtained by solving an optimization problem similar to (19) without the constraint (13), and in the optimization objective, the data size $s_{\tau,i}^{k,m,n}$ of the tile before rendering (not the size of rendered data) is used. During the delivery stage, the delivered data are the already rendered viewport data.
- *Independent Caching for each node with Local Rendering plus Low-delay request routing (IC-LR-L)*: Contrary to the DRC-OR-L scheme, the IC-LR-L scheme caches the tiles without cooperation among multiple MEC servers. Rendering is performed on the UE and the delivered viewport data are the tiled chunk data.
- *MPC caching with Local Rendering plus Low-delay request routing (MPC-LR-L)*: This scheme caches VR video tiles by applying the MPC strategy for each MEC server and rendering is performed on the UE.
- *Rendering-Aware Caching with Offloaded Rendering plus Scalable request routing (RAC-OR-S)*: Contrary to the proposed RAC-OR-L scheme, scalable request routing is used in this scheme.

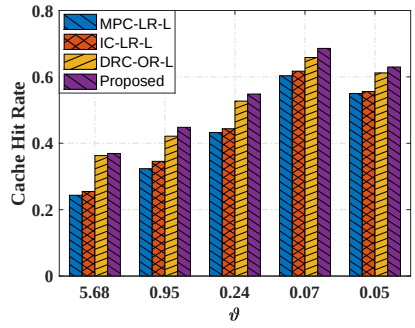
To evaluate the proposed caching scheme in more detail, we first focus on the evaluation of cache placement by comparing our proposed scheme with the first three baseline schemes using the same request routing approach. Afterwards, we compare our proposed low-delay viewport request routing scheme with the fourth baseline scheme to assess the performance of viewport request routing.

A. Caching Performance

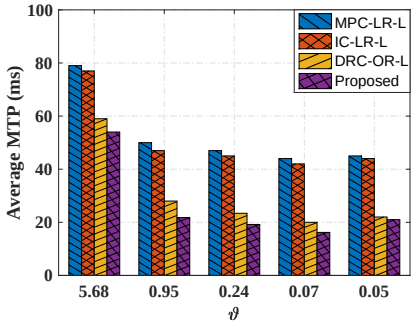
Impact of tile size. Usually, the tile size affects not only the amount of data, but also the random access flexibility of the viewport data in the whole frame. To capture the variation of tile sizes we define a parameter $\vartheta = S_T/S_V$, where S_T denotes the tile size and S_V denotes the viewport size. In our simulation, 4K VR video was segmented into 1×1 (full frame), 6×4 (that means the full frame will be segmented into 6 tiles horizontally and 4 tiles vertically), 8×6, 10×8 and 12×10 tiles. We measured the changes of the cache hit rate and viewport request latency under different cache capacity ratios that indicate different cache spaces. Fig. 8 shows the obtained results with the cache capacity ratio of 0.6. In Fig. 8, the value of 5.68 for a tile size ratio means the tile is the whole frame. Fig. 8(a) shows the cache hit rate changes for different tile size ratios. It can be seen from Fig. 8(a) that the best tile size ratio is approximately 0.07 (corresponding to the 10×8 tile partition for 4K VR video) because it reached an approximate trade-off between the tile's data size and the viewport access flexibility in the entire frame and consequently achieved the highest cache hit rate and the smallest MTP among all the candidate tile size ratios. When the tile size ratio is up to 0.05, the cache hit rate decreases since the corresponding number of tiles covering viewport increases and it results in a significantly negative effect on the cache hit rate. For the four scenarios, the cache hit rate varies slightly when the same tile size is used. The proposed scheme responds successfully to a higher number of requests at the higher system load so that it achieves a slightly higher cache hit rate than the other three schemes.

In Fig. 8(b), latency varies significantly for different tile sizes. Large cache hit rate reduces the requests to the backend server and accordingly leads to the short latency value. At this point, the results in Fig. 8(b) are in line with those of Fig. 8(a). The latency performance at the tile size ratio of 0.07 is the best among all the candidate tile sizes. Moreover, in these simulations, only the proposed rendering-aware caching scheme for several tile sizes can satisfy the requirement that MTP is less than 20ms. It indicates that coupling rendering and caching at the MEC server can significantly reduce latency.

Impact of cache capacity and rendering computational capacity. Fig. 9(a) shows the latency performance for different cache capacity ratio. It can be seen from Fig. 9(a) that, the



(a)



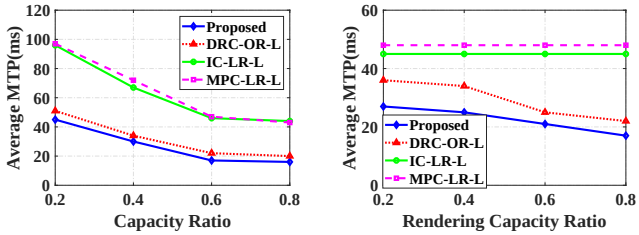
(b)

Fig. 8. Caching performance for different tile sizes

average MTP for viewport requests is gradually decreasing with a higher cache capacity for all four schemes. IC-LR-L achieves similar poor latency performance with the MPC-LR-L scheme. Besides the fact that MPC-LR-L and IC-LR-L achieve slightly lower cache hit rate than the other two schemes, there are two additional reasons for the poor MTP. On one hand, the local rendering on the UE for either MPC-LR-L or IC-LR-L takes more time than rendering on the MEC server. On the other hand, both the MPC-LR-L and IC-LR-L approaches deliver the entire chunk that is used for rendering, while the other two schemes deliver only the already rendered and requested viewport data. Since viewport switching usually occurs at a frame in the chunk, the remaining frames from the switching point that has been already fetched will be no longer useful for the current viewport. Consequently, with frequent viewport switching, the delivery time for either MPC-LR-L or IC-LR-L is typically longer than that of the other two schemes.

In Fig. 9(a), we notice that the proposed RAC-OR-L scheme reduces the latency approximately 6ms on average over the DRC-OR-L scheme for several cache capacity ratios. In fact, caching and rendering are only two elements among many latency sources in the entire VR delivery chain. Since the MTP usually requires the delivery latency to be less than 20ms, even one millisecond latency reduction is very helpful. In such case, the reduction of 6ms latency indicates that the benefit of the proposed scheme is significant.

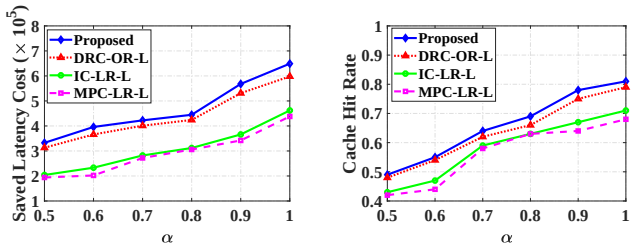
Fig. 9(b) shows the latency performance for different rendering computational capacity ratio. The cache capacity ratio is set to 0.6. The average MTPs of viewport requests for the proposed RAC-OR-L and DRC-OR-L schemes in Fig. 9(b) are slowly decreasing with a higher rendering computational capacity. Moreover, no matter how much the rendering computational capacity is, the proposed RAC-OR-L scheme obtains better latency performance than the other two schemes. At the low rendering capacity regime (around 0.2 of rendering capacity ratio), the benefit from rendering-awareness when compared to DRC-OR-L is significant (at least reducing 10ms latency). Since the MPC-LR-L and IC-LR-L schemes render the video frame on the UE, they are not affected by the rendering computational capacity of MEC server and maintain a constant MTP over the different rendering capacity ratios.



(a) (b)

Fig. 9. Caching performance for different cache and rendering capacities

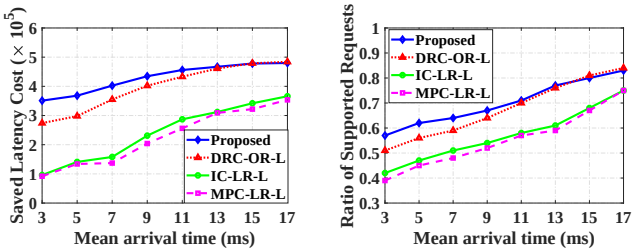
Impact of Zipf parameter. The Zipf parameter directly affects the popularity of VR videos and as a result the popularity of tiles for each VR video. Fig. 10 shows the impact of the Zipf parameter on caching performance. It can be seen from Fig. 10 that both the latency cost savings and the cache hit rates for the three schemes achieve higher values for a larger Zipf parameter. It illustrates that the Zipf parameter changes the distribution of popularity of each tile and further results in the changes of cache hit rate. Among the four schemes, the proposed scheme achieves the best performance.Since the major difference between the proposed RAC-OR-L and DRC-OR-L is the rendering computational resource constraint that allows the proposed RAC-OR-L to serve more requests successfully within the MTP constraint, the proposed RAC-OR-L achieves a modest improvement over DRC-OR-L. Similarly, both IC-LR-L and MPC-LR-L do not cache content cooperatively among multiple MEC servers and thus they achieve lower cache hit rate than the other two schemes.



(a) (b)

Fig. 10. Caching performance for different values of the Zipf parameter

Impact of request load. System performance for different request loads was also investigated. Specifically, the ratio of supported requests and the saved latency cost were used as



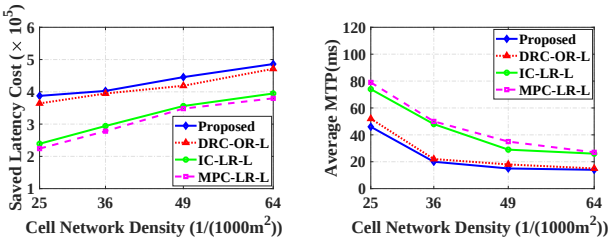
(a) (b)

Fig. 11. Caching performance for different values of the request load

the evaluation metrics. The ratio of the supported requests is defined by the ratio of the number of supported viewport requests to the total number of viewport requests. Fig. 11 presents the latency cost savings and the ratio of supported requests for different mean arrival time between requests. The mean arrival time implicitly controls the load situation in the caching system. The smaller the mean arrival time is, the heavier the load in the system is. It can be seen from Fig. 11(a) that the saved latency cost rises with an increased mean arrival time of requests, and the proposed scheme always results in the highest latency cost savings among all the four schemes. When the mean arrival time of requests is less than 13ms, the performance of the proposed scheme is far superior than the other three schemes. It indicates that the proposed scheme has a significant advantage in saving rendering computational time at heavier system loads. Specifically, when the heavier request load is initiated, the computations for rendering at the cache node will take more time for DRC-OR-L. Similarly, the proposed RAC-OR-L scheme always maintains a higher rendering computation efficiency since it considers in advance the effects of rendering computational load on cache placement.

In Fig. 11(b), it can be seen that the ratios of supported requests for all schemes are increasing while reducing the arrival request rate (increasing the mean arrival time). This is because for lower values of the mean arrival time, the heavier request load results in lower utilization of the uplink bandwidth of gNB which results in lower ratio of supported requests. When the mean arrival time between requests takes values below 11 ms, the proposed scheme achieves a higher ratio of supported requests than the DRC-OR-L scheme. This indicates that the previous rendering awareness for cache placement plays a key role in improving the ratio of supported requests when the mean arrival time of the requests is less than 11ms.

Impact of multi-cell MEC network density. In another set of simulations 25, 36, 49 and 64 gNBs were deployed in a grid network inside a 1000m$\times$1000m area to investigate different multi-cell MEC network densities. Fig. 12 shows the caching performances for different MEC network densities. It can be seen from Fig. 12(a) that the curve of latency cost savings has an increasing trend with an increasing number of gNBs in a 1000m$\times$1000m area for all four schemes. This is also in line with the results in Fig. 12(b). The proposed scheme and the DRC-OR-L scheme both achieve significantly higher latency cost saving and lower MTP latency than the other two local rendering schemes. It illustrates that direct communications among the gNBs contributes to higher cooperative caching efficiency that eventually leads to low latency. In the meantime, the increased number of gNBs provides more cache space and stronger rendering computational capabilities. These aspects make VR video delivery achieve low latency. And also in Fig. 12(b), the schemes with rendering offloading obviously reduce latency more than the case of local rendering. This indicates that rendering offloading is another contributing factor towards reducing the end-to-end latency.



(a) (b)

Fig. 12. Caching performance for different MEC network densities

Cache placement complexity. We measured the complexity for the proposed caching scheme in terms of the running time. The simulations were performed in MATLAB on a computer with an Intel Core i9-10900 2.8-GHz CPU and 32 GB of memory. The number of cached tiles is 48000. Fig. 13 shows the running time averaged over 20 instances for different schemes with varying number of cache nodes. It can be seen from Fig. 13 that the running time of the proposed scheme increases by more than 10 times compared with the independent caching scheme IC-LR-L. Since cache placement is usually updated in longer time scales, the current computational time complexity is acceptable for practical applications.

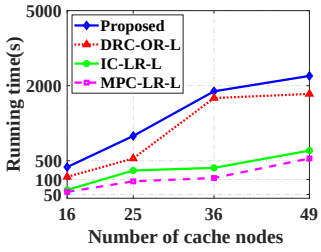


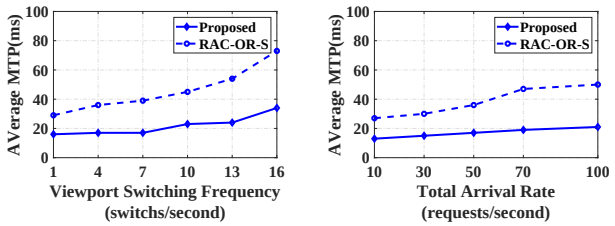
Fig. 13. Comparison of running time over varying numbers of cache nodes.

B. Request Routing Performance

In this subsection, we evaluate the performance of the proposed request routing scheme. The viewport switching frequency changes the viewport request speed. To obtain more realistic results, the viewport browsing trajectory for request routing was simulated with actual VR video browsing trace data. The average simulation result over the ten traces was computed. In the system the content controller manages the cache placement information with a global view and it increases a small delay for forwarding the request to the suitable cache node. The time for viewport request forwarding

from the controller to the cache node that has been hit was computed based on the bandwidth values in Table II with considering the time-varying networking loads. Currently, the request forwarding latency increases about 2ms on average in our experiments. Fig. 14(a) shows the average MTP for the tile-based caching scheme with different request routing algorithm. It can be seen from Fig. 14(a) that the proposed low-delay request routing approach can satisfy the MTP constraint of 20ms when the viewport switching frequency is less than 7, while the scalable routing approach incurs a waiting time for aggregating similar requests leading to an overall MTP more than 30ms. With frequent viewport switching, both routing approaches will suffer from higher latency. This is because frequent viewport switching increases viewport requests and so it requires more time for their routing decisions.

Fig. 14(b) shows the latency performance of the routing algorithm for different total request arrival rates. The total request arrival rates reflect the situation of the system load. Generally, faster total request arrival rates indicate heavier request load. It can be seen from Fig. 14(b) that though the MTP rises with the increasing total request arrival rate, the proposed routing algorithm can achieve satisfactory MTP when the total request arrival rate is below 100 requests/second. As for the scalable routing scheme, it obtains a MTP more than 28ms even at the lightest load which is unacceptable for VR applications.

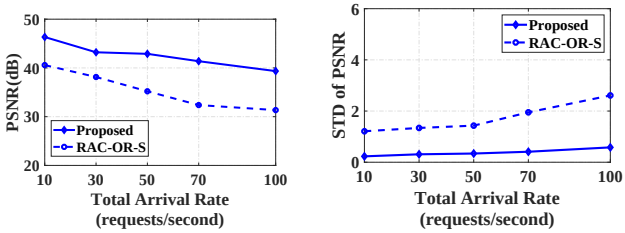


(a) (b)

Fig. 14. Request routing performance

For VR video delivery, the viewport quality received by a user is the objective metric for evaluating the proposed routing and caching approach. In the simulation, we reconstructed the requested viewport video in terms of the user's viewport requesting traces for different request arrival rates in a cell RAN and then computed the Peak Signal to Noise Ratio (PSNR) values for these reconstructed viewport videos. Since for VR video clips in Youtube we cannot find the raw video, we computed the PSNR values only for those VR video clips from SJTU media lab. During viewport reconstruction, the delayed tile data in the viewport are repeated from the previously requested data. The average PSNR value over users for different request arrival rates is shown in Fig. 15(a). In the figure, because the RAC-OR-S scheme incurs a longer latency for aggregating requests than the proposed RAC-OR-L scheme, it achieves a higher number of tile duplication artifacts that lead to a lower PSNR value than the proposed RAC-OR-L scheme.

Considering that the viewport quality variation in the temporal dimension also affects the user quality of experience (QoE), we measure the standard deviation (STD) of PSNR values over frames for the two schemes, as shown in Fig. 15(b). It can be seen in Fig. 15(b) that the proposed routing scheme achieved lower STD values than the scalable routing scheme when the request arrival rate is below 100 requests/second. It illustrates that the proposed routing scheme can obtain smoother QoE in the temporal dimension than the scalable routing scheme. This is because it delivered the necessary tiles timely to avoid the serious tile repeating distortion that the scalable routing often incurred.



(a) (b)

Fig. 15. The viewport quality received by the users at different request arrival rates

VII. Conclusion

Aiming for end-to-end latency optimization for VR video delivery, we proposed a rendering-aware tile caching scheme suitable for multi-cell MEC networks. By coupling rendering and caching, the computational benefits for rendering in multi-cell MEC networks are exploited. With cooperative hierarchical caching and rendering service placement of tiles in multi-cell MEC networks, the rendering latency and viewport data delivery latency are both reduced significantly. Moreover, a low-delay request routing scheme that makes routing decisions on a time granularity of each request is integrated with the cache placement scheme to further optimize VR video caching efficiency. Extensive simulations were carried out and the results show that the proposed scheme achieves superior latency performance over the decoupled tile caching/rendering systems.

References

[1] MPEG Experts, "Summary of survey on virtual reality", ISO/IEC JTC 1/SC 29/WG 11, m16542, Oct. 2016, Chengdu, China.
[2] S. Ohl, M. Willert, and O. Staadt, "Latency in distributed acquisition and rendering for telepresence systems", IEEE Trans. Visual. Comput. Graphics, vol. 21, no. 12, pp. 1442-1448, Dec. 2015.
[3] ETSI, "Mobile Edge Computing - Introductory Technical White Paper", Sep. 2014.
[4] Huawei-ilab, "Whitepaper on Cloud VR solution", 2018, http://www.huawei.com/.
[5] V. R. Gaddam, M. Riegler, R. Eg, P. Halvorsen, "Tiling in interactive panoramic video: Approaches and evaluation", IEEE Trans. Multimedia, vol. 18, no. 9, pp. 1819-1831, Sep. 2016.
[6] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching", IEEE Trans. Inf. Theory, vol. 60, no. 5, pp. 2856-2867, May 2014.

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

[8] J. Le Feuvre and C. Concolato, "Tiled-based adaptive streaming using MPEG-DASH", in Proc. 7th Int. Conf. Multimedia Syst., Wörthersee, Austria, May 2016.

[9] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog Computing: Principles, Architectures, and Applications", Internet of Things: Principles and Paradigms, 2016, ch. 4.

[10] M. D. Donno, K. Tange and N. Dragoni, "Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog", IEEE Access, vol. 7, pp. 150936-150948, Oct. 2019.

[11] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, "Joint container placement and task provisioning in dynamic fog computing", IEEE Internet Things J., vol.6, no.6, pp. 10028-10040, Aug. 2019.

[12] Y. Liu, J. Liu, A. Argyriou, S. Ci, "MEC-assisted Panoramic VR Video Streaming over Millimeter Wave Mobile Networks", IEEE Trans. Multimedia, vol.21, no. 5, pp. 1302-1316, May 2019.

[13] R. Schmoll, S. Pandi, P. J. Braun, F. H.P. Fitzek, "Demonstration of VR/AR offloading to Mobile Edge Cloud for low latency 5G gaming application", in Proc. 15th IEEE Annu. Consum. Commun. & Netw. Conf. (CCNC), 2018.

[14] W. Lo, C. Huang, C. Hsu, "Edge-Assisted Rendering of 360 Videos Streamed to Head-Mounted Virtual Reality", in Proc. IEEE Int. Symp. Multimedia (ISM), 2018.

[15] S. Shi, V. Gupta, R. Jana, "Freedom: Fast Recovery Enhanced VR Delivery Over Mobile Networks", in Proc. 17th ACM Int. Conf. Mobile Sys., Appl., and Services (MobiSys 2019), June 17-21, 2019, Seoul, Korea.

[16] J. Qiao, Y. He, and X. Shen, "Proactive caching for mobile video streaming in millimeter wave 5G networks", IEEE Trans. Wireless Commun., vol. 15, no .10, pp. 7187-7198, August 2016.

[17] X. Wang, M. Chen, T. Taleb, A. Ksentini, V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems", IEEE Commun. Mag., vol. 52, no. 2, pp. 131-139, Feb. 2014.

[18] H. Ahlehagh, S. Dey, "Video-aware Scheduling and Caching in the Radio Access Network", IEEE Trans. Netw., vol. 22, no. 5, pp.1444-1462, Oct. 2014.

[19] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers", in Proc. of IEEE INFOCOM, pp.1107-1115, 2012.

[20] W. Cao, G. Feng, S. Qin, and M. Yan, "Cellular offloading in heterogeneous mobile networks with D2D communication assistance", IEEE Trans. Veh. Technol., vol. 66, no. 5, pp. 4245-4255, May 2017.

[21] W. Jaafar, W. Ajib and H. Elbiaze, "Caching Optimization for D2D-Assisted Heterogeneous Wireless Networks", in Proc. IEEE 30th Annu. Int. Symp. Pers., Indoor and Mobile Radio Commun. (PIMRC), 2019.

[22] M. Gregori, J. Gmez-Vilardeb, J. Matamoros, and D. Gndz, "Wireless content caching for small cell and D2D networks", IEEE J. Select. Areas Commun., vol. 34, no. 5, pp. 1222-1234, May 2016.

[23] N. Giatsoglou, K. Ntontin, E. Kartsakli, A. Antonopoulos and C. Verikoukis, "D2D-Aware Device Caching in mmWave-Cellular Networks", IEEE J. Select. Areas Commun., vol. 35, no. 9, pp. 2025-2037, Sept. 2017.

[24] M. Rim and C. G. Kang, "Content Prefetching of Mobile Caching Devices in Cooperative D2D Communication Systems", IEEE Access, vol. 8, pp. 141331-141341, 2020.

[25] G. Kollias and A. Antonopoulos, "Joint Consideration of Content Popularity and Size in Device-to-Device Caching Scenarios", in Proc. IEEE Int. Conf. Commun. (ICC), 2020.

[26] C.Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching and computing in wireless systems: A survey, some research issues and challenges", IEEE Commun. Surveys Tuts., vol. 20, no. 1, pp. 7-38, First Quarter 2018.

[27] H. Pang, J. Liu, X. Fan and L. Sun, "Toward Smart and Cooperative Edge Caching for 5G Networks: A Deep Learning Based Approach", in Proc. IEEE/ACM Int. Symp. Qual. of Service, 4-6 June 2018.

[28] S. Sukhmani, M. Sadeghi, M. Erol-Kantarci, A. El Saddik, "Edge Caching and Computing in 5G for Mobile AR/VR and Tactile Internet", IEEE Multimedia, vol.26, no.1, pp.21-30, 2019.

[29] Y. Sun, Z. Chen, M. Tao, H. Liu, "Communication, computing and caching for mobile VR delivery: Modeling and trade-off", in Proc. IEEE Int. Conf. Commun. (ICC), 2018.

[30] J. Chakareski, "VR/AR immersive communication: Caching, edge computing, and transmission trade-offs", in Proc. ACM SIGCOMM Workshop Virtual Reality and Augmented Reality Netw., Aug. 2017.

[31] Y. Sun, Z. Chen, M. Tao and H. Liu, "Communications, Caching, and Computing for Mobile Virtual Reality: Modeling and Tradeoff", IEEE Trans. Commun., Vol. 67, No. 11, pp.7573-7586, November 2019.

[32] J. Chakareski, "Viewport-Adaptive Scalable Multi-User Virtual Reality Mobile-Edge Streaming", IEEE Trans. Image Processing, vol. 29, pp. 6330-6342, May 2020.

[33] V. Changrani, E. Zhang, "CDN optimization for VR streaming", Akamai White Paper.

[34] K. Liu, Y. Liu, J. Liu, A. Argyriou, and Y. Ding, "Joint EPC and RAN caching of tiled VR videos for mobile networks", in Proc. Int. Conf. Multimedia Model., Springer, January 2019, pp.92-105.

[35] A. Mahzari, A. Nasrabadi, A. Samiei and R. Prakash, "FoV-Aware Edge Caching for Adaptive 360°Video Streaming", in Proc. 26th ACM int. conf. Multimedia, 2018.

[36] G. Papaioannou and I. Koutsopoulos, "Tile-based caching optimization for 360°videos", in Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. and Comput., July 2019, pp. 171-180.

[37] P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of 360 videos in heterogeneous networks", IEEE Trans. Multimedia, vol. 22, no. 9, pp. 2382 - 2395, Sept. 2020.

[38] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery", in Proc. ACM SIGCOMM CCR.,vol. 45, no. 3, pp. 52-66, 2015.

[39] P. Poirion, L. Jérémie, R. Liu, "Scalable Request Routing for VR-ready CDNs", In Proc. 21st Int. Conf. Innov. in Clouds, Internet and Netw. (ICIN 2018), February 2018.

[40] L. Liu, R. Zhong, W. Zhang, Y. Liu, et al., "Cutting the Cord: Designing a High-quality Untethered VR System with Low Latency Remote Rendering", in Proc. 16th Int. Conf. Mobile Syst., Appl., and Services, June 2018.

[41] J. M. P. Van Waveren, "The asynchronous time warp for virtual reality on consumer hardware", in Proc. 22nd ACM Conf. Virtual Reality Softw. and Technol., 2016, pp.37-46.

[42] T. C. Nguyen, S. Kim, J. Son, and J.-H. Yun, "Selective timewarp based on embedded motion vectors for interactive cloud virtual reality", IEEE Access, vol. 7, pp. 3031-3045, 2019.

[43] Y. Ding, Y. Liu, J. Liu, K. Liu, et al., "Panoramic Image Saliency Detection by Fusing Visual Frequency Feature and Viewing Behavior Pattern", in Proc. Pacific-Rim Conf. Multimedia, Sept. 2018.

[44] G. Choquet, Theory of capacities. Ann. Inst. Fourier, vol. 5, pp. 131-295, 1954.

[45] Z. Wang, K.-S. Leung, and G. J. Klir, "Applying fuzzy measures and nonlinear integrals in data mining", Fuzzy Sets Syst., vol. 156, no. 3, pp. 371-380, 2005.

[46] V. Sitzmann, et al., "Saliency in VR: how do people explore virtual environments?", IEEE Trans. Visual. Comput. Graphics, vol. 24, no.4, pp.1633-1642, 2018.

[47] M. E. J. Newman, "Power laws, Pareto distributions and Zipf"s law", Contemporary Physics, vol. 46, p. 323, 2005.

[48] W. Lo, C. Fan, J. Lee, C. Huang, K. Chen, and C. Hsu, "360 Video Viewing Dataset in Head-Mounted Virtual Reality", in Proc. ACM Multimedia Syst. Conf., Jun. 2017.
[49] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2003.
[50] F. S. Hillier, Introduction to Operations Research. McGraw-Hill, 2004.
[51] S. Martello and P. Toth, Knapsack Problems: Algorithms and Computer Implementations. Wiley, 1990.
[52] https://www.youtube.com/channel/UCzuqhhs6NWbgTzMuM09WKDQ
[53] A. T. Nasrabadi, A. Samiei, A. Mahzari, R. P. McMahan, R. Prakash, M. C. Farias, and M. M. Carvalho, "A Taxonomy and Dataset for 360° Videos", in Proc. 10th ACM Multimedia Syst. Conf., pp. 273-278. ACM, 2019.
[54] X. Liu, Y. Huang, L. Song, R. Xie, X. Yang, "The SJTU UHD 360-Degree Immersive Video Sequence Dataset", in Proc. Int. Conf. on Virtual Reality and Visualization, 2017.
[55] HEVC Test Model, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/
[56] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz", 3GPP, Tech. Rep. 38.901 V14.3.0, Dec. 2017.
[57] D.P.Bertsekas, R.G.Gallager, Data Networks, Second Edition, Prentice-Hall International New Jersey, 1992.
[58] J. D. C. Little, S. C. Graves, "Little's law", in Proc. Int. Series in Operations Res. & Manage. Sci., vol. 115, pp. 81-100, 2008.
[59] D. Raychaudhuri, K. Nagaraja and A. Venkataramani, "MPC: Popularity-based caching strategy for content centric networks", in Proc. IEEE Int. Conf. Commun. (ICC), 2012.

Yanwei Liu received the B.S. degree in applied geophysics from Jianghan Petroleum University, China, in 1998, the M.S. degree in computer science from China Petroleum University (Beijing) in 2004 and the Ph.D. degree in computer science from Institute of Computing Technology, Chinese Academy of Sciences in 2010. Currently, he is an Associate Professor with the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include digital image/video processing, multiview/3D video/VR video processing, wireless multimedia networking. He serves in the TPC of several international conferences in the area of multimedia, communications, and networking.

Jinxia Liu received the B.S. degree and M.S. degree in physics from Harbin Normal University, China, in 1994, and 2005, respectively. In 2005, she joined the Zhejiang Wanli University. Currently, she is a Professor with Zhejiang Wanli University. Her research interests include laser imaging, digital image/video processing, multiview and 3D video coding, and wireless communication.

Antonios Argyriou received the Diploma in electrical and computer engineering from Democri-tus University of Thrace, Greece,and the M.S. and Ph.D. degrees in electrical and computer engineering as a Fulbright scholar from the Georgia Institute of Technology, Atlanta, USA. Currently, he is an Associate Professor at the department of electrical and computer engineering, University of Thessaly, Greece. From 2007 until 2010 he was a Senior Research Scientist at Philips Research, Eindhoven, The Netherlands. Dr. Argyriou serves in the TPC of several international conferences and workshops. His research interests are in the areas of communication systems, and statistical signal processing.

Liming Wang received his Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, Beijing, China in 2007. He is now a professor and working at the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His current research interests include network security and cloud computing.

Zhen Xu received his Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, Beijing, China in 2005. He is now a professor and a director of research center in the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His research interests include network and system security, and mobile computing.