VRCT: A Viewport Reconstruction-Based 360° Video Caching Solution for Tile-Adaptive Streaming

Ziwen Ye, Qing Li, *Member, IEEE*, Xiaoteng Ma, Dan Zhao, Yong Jiang, *Member, IEEE*, Lianbo Ma, *Senior Member, IEEE*, Bo Yi, *Member, IEEE*, and Gabriel-Miro Muntean, *Fellow, IEEE*

***Abstract*—360° video streaming demands higher bandwidth and lower latency than conventional videos. Some solutions employ tile-adaptive 360° video streaming and edge caching mechanisms to improve the quality of their content delivery. However, it is hard to cache large amounts of popular content with limited cache capacity. Reconstruction technologies, which have been widely adopted for images and conventional videos, can potentially reconstruct complete tile-based viewports from partial observation for 360° videos, thus further relieving the pressure on the caching. However, it is challenging to design a flexible and efficient caching solution that supports viewport reconstruction for 360° videos. In this paper, we propose a Viewport Reconstruction-based 360° video Caching solution for Tile-adaptive streaming (VRCT). To enhance viewers' quality of experience (QoE), a QoE-driven reconstruction trigger scheme is designed to determine whether to perform reconstruction or not based on current cache information and network conditions. To make efficient use of the cache space and facilitate the viewport reconstruction, a heuristic-based solution, named aggregation-based cache replacement scheme, is proposed to improve the probability of viewport reconstruction by carefully selecting which tiles to be stored in the given limited space. Through comprehensive experiments with a real head movement dataset, we show that the proposed VRCT increases the cache hit ratio by up to 29%, reduces the backhaul usage by up to 44% and improves user QoE by up to 32% compared with other existing methods. In addition, experimental results show that our proposed cache replacement scheme facilitates viewport reconstruction and supports different video types.**

***Index Terms*—Adaptive 360° video streaming, caching strategy, viewport reconstruction, tile-based transmission, edge computing.**

Manuscript received 23 February 2023; revised 10 April 2023; accepted 10 April 2023. Date of publication 16 May 2023; date of current version 6 September 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3105000; in part by the National Natural Science Foundation of China under Grant 61972189; in part by the Major Key Project of PCL under Grant PCL2021A03-1; in part by the Shenzhen Science and Technology Innovation Commission: Research Center for Computer Network (Shenzhen) Ministry of Education; and in part by the Shenzhen Key Laboratory of Software Defined Networking under Grant ZDSYS20140509172959989. The work of Gabriel-Miro Muntean was supported in part by the Science Foundation Ireland (SFI) Frontiers Projects under Grant 21/FFP-P/10244 (FRADIS), and in part by the SFI Research Centres under Grant 12/RC/2289_P2 (INSIGHT). *(Corresponding author: Qing Li.)*

Ziwen Ye and Yong Jiang are with the Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China, and also with the Department of Mathematics and Theories, Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: yezw21@mails.tsinghua.edu.cn; jiangy@sz.tsinghua.edu.cn).

Qing Li and Dan Zhao are with the Department of Mathematics and Theories, Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: liq@pcl.ac.cn; zhaod01@pcl.ac.cn).

Xiaoteng Ma is with the Tsinghua–Berkeley Shenzhen Institute, Tsinghua University, Shenzhen 518055, China, and also with the Department of Mathematics and Theories, Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: maxt17@mails.tsinghua.edu.cn).

Lianbo Ma is with the College of Software, Northeastern University, Shenyang 110819, China (e-mail: malb@swc.neu.edu.cn).

Bo Yi is with the College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China (e-mail: yibo@cse.neu.edu.cn).

Gabriel-Miro Muntean is with the School of Electronic Engineering, Dublin City University, Dublin 9, D09 DXA0 Ireland (e-mail: gabriel.muntean@dcu.ie).

Digital Object Identifier 10.1109/TBC.2023.3274350

I. Introduction

THE RECENT evolution of virtual reality (VR), VR devices and network communication technologies have significantly stimulated the production and consumption of 360° videos (a.k.a. panoramic videos) [1]. Many leading content providers, such as Facebook and YouTube, have deployed support for 360° videos and are increasingly promoting immersive user experiences in a wide range of areas, including education, sports and entertainment [2]. According to a recent market report [3], this trend will continue and a whopping number of 112.62 million smartphone-based VR devices and Head-Mounted Displays (HMDs) is expected by 2026. Unfortunately, due to the ultra-high resolution (up to 8K) and bitrate (up to 200 Mbps) of 360° videos, the bandwidth to deliver such multimedia content is 4-5 times larger than that required to deliver conventional videos [4]. Besides, the delivery latency of the immersive videos must be kept small (e.g., less than 20 ms) to guarantee users' quality of experience (QoE) [5]. In spite of the effective increase in the available network bandwidth and reduced network latency enabled by the 5G communication technology, it is still a challenge to stream 360° videos from remote content servers to multiple users.

To alleviate bandwidth consumption and transmission latency, some solutions [6], [7], [8] propose adaptive 360° video streaming by leveraging information about users' field of view (FoV). FoV is a portion of the 360° video around the user's line of sight. It can be spatially partitioned into non-overlapped rectangular regions called tiles. In tile-based techniques, each tile is transcoded into multiple representations with different bitrates. Both bandwidth consumption and latency can be significantly decreased by delivering only the tiles within the user's FoV at high resolution, while the other tiles are served at low resolution or not delivered at all.

To avoid the transmission redundancy associated with repeated requests and reduce the traffic load on the backhaul

path, some works [9], [10], [11] bring popular content closer to users by deploying transcoding-enabled cache servers at the edge of the Content Delivery Network (CDN). The application of cache servers effectively reduces the load on the content servers and improves users' QoE on the client side. However, with the volume of the 360° videos exploding, it has become increasingly more challenging to store massive popular content in the cache server. Exploiting the multi-access edge computing (MEC) paradigm, [12], [13], [14] propose more advanced caching systems to optimize the edge computing capacity. However, these solutions require that all tiles requested to participate in edge computing be cached before the optimization methods are executed, which is unpractical given the limited cache capacity.

Reconstruction technologies are widely used in images [15], [16], [17] and conventional videos [18], [19], [20] because they can infer missing regions from partial observation, which is helpful to reduce additional requests for uncached content. However, unlike traditional videos, where the entire frames can be reconstructed, it is difficult and also unnecessary to reconstruct the entire frames of 360° videos due to two reasons. First, the volume of 360° videos is usually much larger than that of conventional videos. It will waste excessive amount of bandwidth to transmit the whole frames. Second, as will be shown in Section II-C, only the users' viewports, rather than the entire frames, need to be reconstructed in 360° videos. As such, it is necessary to design a flexible cache system that can facilitate viewport reconstruction, to store appropriate tiles given the limited cache space. To the best of our knowledge, no work has yet incorporated reconstruction techniques into tile-based 360° video streaming.

In this paper, we propose a Viewport Reconstruction-based 360° video Caching solution for Tile-adaptive streaming (VRCT). The proposed VRCT addresses the following two key challenges: (i) how to decide whether to perform the viewport reconstruction task, and (ii) how to choose which tiles to evict in the cache server when the cache capacity is exceeded.

To solve the first problem, a QoE-driven Reconstruction Trigger Scheme (QRTS) is proposed to determine whether to perform the viewport reconstruction or not based on current cache information and network conditions. VRCT first estimates the viewport quality and associated delay by introducing a fitting function and employing queuing theory. Then it calculates user QoE with/without viewport reconstruction based on the estimations, and finally makes a decision to enhance user QoE. Related to the second problem, we propose a heuristic-based solution, named Aggregation-based Cache Replacement Scheme (ACRS), to select specific cached tiles to be removed, aiming to increase the viewport hit ratio of requests and improve the probability of viewport reconstruction. It can efficiently improve the spatial aggregation of the cached tiles and the temporal uniformity of the cache proportion. In other words, tiles of the same segment are stored centrally in the spatial dimension, while the number of cached tiles in any segment remains within a desirable range.

To assess the performance of VRCT, we compare it with several existing caching policies through comprehensive experiments, using a head-tracking dataset consisting of real traces. Experimental results show that: (i) VRCT outperforms other schemes, increasing up to 29% cache hit ratio and 32% user QoE, and effectively reducing the load on the backhaul path by up to 44%; (ii) The proposed cache replacement algorithm, ACRS, makes it easier to reconstruct viewports by improving the spatial aggregation of the cached tiles and the temporal uniformity of the cache proportion; (iii) The introduction of ACRS improves the probability of viewport reconstruction in different video types by 2% to 7%, which shows great robustness, and an average of 10% more users can benefit from the reconstruction mechanism.

The contributions of this paper are as follows:

- We propose VRCT, a 360° video caching solution based on viewport reconstruction for tile-adaptive streaming;
- We design a system model to estimate user QoE and develop a QoE-driven reconstruction trigger scheme for the viewport reconstruction task;
- We propose an aggregation-based cache replacement scheme to change the cache features of the cached content and improve the probability of viewport reconstruction;
- We conduct comprehensive experiments in a real-world testbed with a head movement dataset to demonstrate the superior performance of VRCT.

The rest of the paper is organized as follows. In Section II, we briefly introduce related works and discuss our motivation. In Section III, we present the system model and detailed definitions. Section IV details the design of the proposed VRCT solution. Section V describes the experimental testing setup, and Section VI discusses the testing results. Section VII summarizes the advantages and limitations of VRCT. Finally, Section VIII concludes the paper.

II. Related Works and Motivation

A. Adaptive 360° Video Streaming

Over the years, 360° video streaming solutions have progressed from viewport-independent schemes and viewport-dependent schemes to tile-based schemes. The viewport-independent schemes [21] treat 360° videos similarly as traditional ones and stream the whole video frame area at the same quality level, without considering the user's FoV. In viewport-dependent schemes [22], not only the bitrate level, but also the transmission area of the frame can be dynamically selected, so the streaming is adaptive. Compared with the first two approaches, tile-based schemes are more flexible. In tile-based schemes, the 360° videos are spatially partitioned into a number of non-overlapping rectangular regions, called tiles, which are further split temporally into many fixed-duration segments according to the dynamic adaptive streaming over HTTP (DASH) standard [23]. The employment of the DASH-based approach makes it easier to choose the appropriate quality of requested videos for users according to different network situations.

Many efforts are put to reduce bandwidth consumption and enhance viewport quality in tile-based 360° video streaming. Wang et al. [24] design a saliency-driven mobile 360° video streaming system which takes full advantage of gaze information to improve the QoE under insufficient wireless

network bandwidth. Zheng et al. [25] present a novel ShiftTile-Tracking (STC) streaming system, which crops and transmits video by tracking the FoV movement of users. Wu et al. [26] propose a dual-queue streaming framework to enable the Deep Reinforcement Learning (DRL) agent to determine and change the tile download order without incurring overhead. Madarasingha and Thilakarathna [27] present a computational geometric approach-based adaptive tiling mechanism, which can take visual attention information as the input and provide a suitable non-overlapping variable size tile cover on the frame. Yaqoob and Muntean [28] introduce CFOV, a combined FoV tile-based adaptive streaming solution, which introduces a practical-oriented tile selection method and actively allocates the video bitrate budget to different video frame areas to enhance the VR perception quality. The same authors have gone a step further and have proposed DVS, a prioritised bitrate adaptation approach based on dynamic viewport adjustment [29]. Additionally, Madarasingha et al. [30] also propose an Inductive Logic Programming (ILP)-based mechanism to devise optimal cache tile configuration at MEC server which reduces the data fetched from the content server.

Although adaptive streaming technologies can reduce bandwidth consumption, traditional network architectures based on centralized storage and computing no longer meet the requirements of high-bitrate content request and delivery. Since popular content is often requested repeatedly, this approach drives up the load on the content server and wastes a lot of bandwidth on the backbone network.

B. Edge Caching and Edge Computing

To reduce duplicate content delivery and alleviate the load on the backbone network, alternative solutions based on edge caching and edge computing should be introduced in the 360° video streaming process.

He et al. [31] present an edge caching scheme for 360° video streaming at the video and tile levels to reduce traffic load and improve user QoE. He et al. [31] model the process of collaborative transcoding and caching as a Markov decision process, and use a model-free DRL approach to obtain an efficient caching replacement and computing power allocation strategy. Furthermore, Fu et al. [32] propose a sequential reinforced 360-degree video streaming scheme, which boosts the performance of long-term viewpoint prediction with cross-user attentive network. Dai et al. [13] design a view synthesis-based VR caching system that can synthesize the user's current desired FoV by cached nearby FoVs. Dasari et al. [14] combine traditional video encoding with super-resolution techniques by training small micro-models, significantly increasing video quality while reducing bandwidth requirements.

However, the above solutions require all the requested tiles to be cached before their optimization methods can be executed. As such, they always have to fetch extra uncached tiles from the remote content servers, causing escalated backhaul traffic. Fortunately, reconstruction technology can alleviate this problem, among which the masked autoencoders (MAE) approach [17] is the most advanced. MAE is an encoder-decoder architecture. The encoder maps the partial observed signal to a latent representation, from which the decoder can reconstruct the original signal. Finally, the masked image can be reconstructed from partial observations. Intuitively, by reconstructing complete tile-based viewports from the cached tiles already available in the MEC-Cache server, the reconstruction technology can reduce additional requests for uncached tiles.

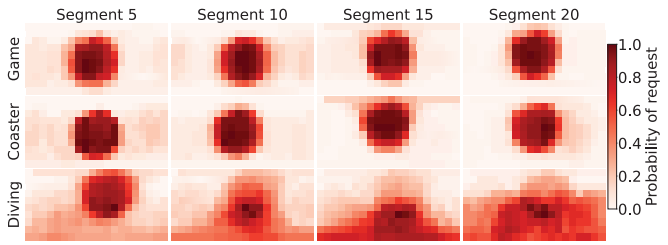


Fig. 1. Some examples of heatmaps for 360° videos.

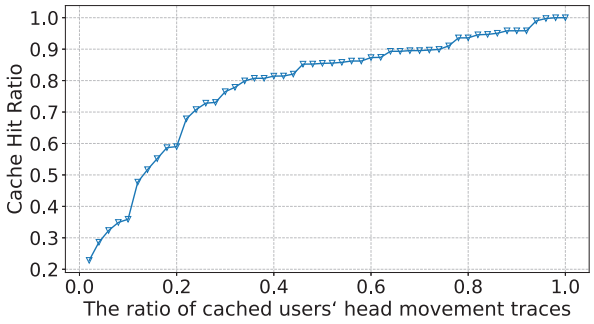


Fig. 2. Hit ratio vs. Cached ratio of users' watching traces.

C. Motivation

To verify the above-mentioned ideas, we analyze a head-tracking dataset with real traces [33], and compute the heatmap with three different types of 360° videos. As depicted in Figure 1, the salient regions that attract users' attention always appear in the same tiles, and these areas are defined as regions of interest (ROI). Additionally, we further investigate the similarity of 50 users' head movement traces, and Figure 2 shows the experimental result in detail. An important conclusion is that a 60% hit ratio can be achieved when only 20% of the watching traces are stored, indicating users share similar behavioral patterns when watching 360° videos. These two phenomena reveal that users' demands for tiles are highly correlated in the spatial dimension, which means that only a small number of viewports need to be reconstructed to meet the viewing needs of most users. Besides, highly overlapping cached tiles can also improve the viewport hit ratio and guarantee the quality of reconstructed viewports.

However, due to the limited computational resources and storage space available at the edge, the introduction of reconstruction technology brings new challenges to the design of tile-based caching and streaming systems. These challenges include 1) determination of the appropriate timing for viewport reconstruction instead of executing it whenever cache misses are encountered; 2) specially designed cache replacement strategy to facilitate the reconstruction technology. In

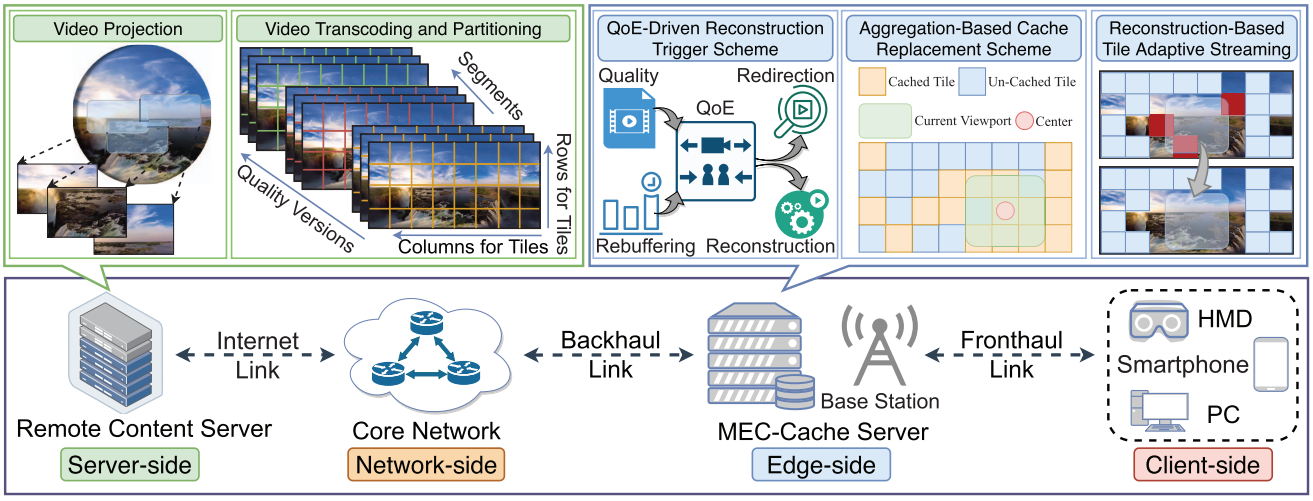


Fig. 3. The system overview of VRCT.

this context, we propose VRCT, a viewport reconstruction-based 360° video caching solution for tile-adaptive streaming, to reduce the number of requests for uncached tiles.

III. System Overview

A. System Model

Figure 3 illustrates the system overview of VRCT. Our proposed VRCT framework considers a wireless network scenario involving multiple mobile terminals, and there is a MEC-Cache server between the remote content server and these clients. On the server side, the created 360° videos undergo preprocessing such as projection, transcoding and partitioning before streaming. On the edge side, the MEC-Cache server is able to store a portion of 360° videos, as well as reconstruct viewports as needed to enhance the 360° video streaming quality.

To address two aforementioned challenges in Section I, we first develop a QoE-driven reconstruction trigger scheme, which allows the flexibility to choose whether to reconstruct the incomplete viewports or fetch the uncached tiles from the remote content server. Then, we design an aggregation-based cache replacement scheme to facilitate viewport reconstruction by storing appropriate tiles in the given limited space. Combining both, the viewport reconstruction-based tile adaptive streaming can be achieved.

B. 360-Degree Video Model

Assume that there are V 360° videos stored on a remote content server, indexed by $\mathcal{V} = \{1, \ldots, v, \ldots, V\}$, which can be transmitted and cached in edge servers. Each video is spatially partitioned into $M \times N$ non-overlapped tiles and a tile is indexed by m and n, where $m \in \mathcal{M}$, $\mathcal{M} = \{1, \ldots, m, \ldots, M\}$, is the column index and $n \in \mathcal{N}$, $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$ is the row index. Then, each tile is further split temporally into S consecutive segments, indexed by $\mathcal{S} = \{1, \ldots, s, \ldots, S\}$. To facilitate adaptive 360° video streaming, each 360° video is transcoded into K different representations indexed by $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$. And there are U users in the network, indexed by $\mathcal{U} = \{1, \ldots, u, \ldots, U\}$. Therefore, $v^k_{s,m,n}$ denotes the tile at the *m-th* column and *n-th* row in the *s-th* segment of the *v-th* video, transcoded at the *k-th* bitrate level.

In addition, we use $\mathcal{D}$ to indicate the set of tiles within the requested viewport, and $\mathcal{D}(v_s)$ means the set of tiles within the specific viewport in the *s-th* segment of the *v-th* video. For tile $v^k_{s,m,n}$, $x^{v^k_{s,m,n}}_{u,t} = 1$ indicates that $v^k_{s,m,n}$ is located in the FoV of user u at time slot t, and $x^{v^k_{s,m,n}}_{u,t} = 0$ otherwise.

C. Adaptive Bitrate Algorithm

On the client side of the 360° video streaming network, a hybrid adaptive bitrate (ABR) algorithm, which takes both throughput and buffer size into account, is adopted to decide for users which representations of 360° videos to request based on historical network conditions.

First, the network throughput $B(t)$ at time slot t is estimated based on the historic downloading capacity as follows:

$$B(t) = \frac{\sum_{v^k_{s,m,n} \in \mathcal{D}} size\left(v^k_{s,m,n}\right)}{\sum_{v^k_{s,m,n} \in \mathcal{D}} DT\left(v^k_{s,m,n}\right)}, \tag{1}$$

where $size(\cdot)$ and $DT(\cdot)$ represent the file size and download time of the input tile, respectively. Then, the estimation of the throughput at time slot $t+1$ is defined as the harmonic mean of the downloading throughput for the last I segments:

$$B(t+1) = \frac{I}{\sum_{i=0}^{I-1} 1/B(t-i)}, \tag{2}$$

where the input length I is set to 5 segments to cope with frequent fluctuations in network bandwidth [34].

When the current buffer size BS is within the preset range $[BS_{min}, BS_{max}]$, the user will request the next viewport based on the estimated throughput $B(t+1)$, that is, we first select a subset of $\mathcal{K}$ in which the bitrates of all files are less than $B(t+1)$, and then choose the representation with the highest bitrate in the subset as the request quality. Otherwise, if BS is less than the preset minimum BS_{min}, the client will fetch the tiles at the lowest quality to fill the buffer as soon as

possible. And if BS exceeds the preset maximum BS_{max}, the requests will be suspended and the buffer will be consumed continuously.

D. Quality of Experience Model

To evaluate the user's viewing experience, inspired by [35], [36], [37], we build an objective QoE evaluation model by jointly considering the following metrics.

1) *Perceived Quality:* In 360° videos, only the tiles in the user's FoV can affect the perceived quality. Therefore, the average perceived quality of the requested tile set $\mathcal{D}(v_s)$ is defined as:

$$Q_p(\mathcal{D}(v_s)) = \frac{1}{N(\mathcal{D}(v_s))} \sum_{v^k_{s,m,n} \in \mathcal{D}(v_s)} q\left(v^k_{s,m,n}\right), \tag{3}$$

where $N(\mathcal{D}(v_s))$ represents the number of tiles in $\mathcal{D}(v_s)$, and $q(\cdot)$ calculates the perceived quality of the input tile based on its bitrate. Note that a user can only request one representation of the tile in a fixed position, as such, the following constraint is satisfied:

$$\sum_{k \in \mathcal{K}} x^{v^k_{s,m,n}}_{u,t} = 1. \tag{4}$$

2) *Rebuffering*: When a user starts to request a video, the initial buffer is empty. During video playback, the buffer may also run out due to network fluctuations. In both cases, the client player needs to stop playing until the requested viewport is prepared. So we define the rebuffering duration as the time spent waiting:

$$Q_r(\mathcal{D}(v_s)) = [PT(\mathcal{D}(v_s)) - BS]^+, \tag{5}$$

where $PT(\cdot)$ calculates the preparation time required to acquire the requested viewport consisting of the input tile set, which will be discussed in Section IV-A2.

3) *Switching Quality:* When the network bandwidth fluctuates greatly, the selected quality version will also fluctuate accordingly, resulting in frequent switching of video quality across segments. Therefore, the smoothness of video quality across the segments is defined as

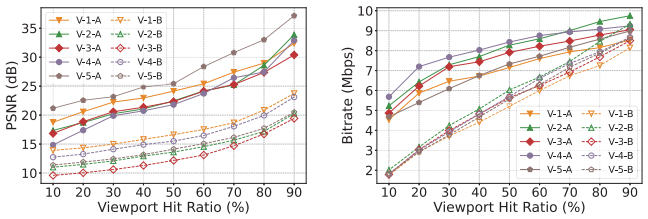
$$Q_s(\mathcal{D}(v_s)) = \left|Q_p(\mathcal{D}(v_s)) - Q_p(\mathcal{D}(v_{s-1}))\right|. \tag{6}$$

Finally, we define the QoE objective by a weighted summation formulation:

$$\begin{aligned} QoE(\mathcal{D}(v_s)) = {} & \alpha_1 \cdot Q_q(\mathcal{D}(v_s)) - \alpha_2 \cdot Q_r(\mathcal{D}(v_s)) \\ & - \alpha_3 \cdot Q_s(\mathcal{D}(v_s)), \end{aligned} \tag{7}$$

$$QoE = \frac{1}{\bar{s} - \underline{s} + 1} \sum_{s=\underline{s}}^{\bar{s}} QoE(\mathcal{D}(v_s)), \tag{8}$$

where α_1, α_2 and α_3 are weighting factors. Due to the introduction of reconstruction technology, we are most concerned with the perceived quality of the viewports. Therefore, the weighting factors $(\alpha_1, \alpha_2, \alpha_3)$ are set to $(1, 0.25, 0.25)$, which is the setting also used in [35] and [36]. Besides, $\underline{s}$ and $\bar{s}$ denote the start and end segments of user requests, respectively. As can be seen, QoE is enhanced by higher video quality, shorter rebuffering duration and fewer bitrate switches.



(a) PSNR (b) Bitrate

Fig. 4. The Impact of viewport hit ratio on reconstruction tasks: (a) The PSNR of the viewport after reconstruction (V-N-A) and before reconstruction (V-N-B) (b) The bitrate of the viewport after reconstruction (V-N-A) and before reconstruction (V-N-B).

IV. VRCT Framework Design

A. QoE-Driven Reconstruction Trigger Scheme

In this subsection, we first introduce two key metrics for evaluating the impacts of reconstruction, i.e., the quality of the reconstructed viewport and the time consumed by the process. Then, the QoE-driven reconstruction trigger scheme is detailed.

1) Quality of the Reconstructed Viewport: As for the improvement in viewport quality, we run a series of tests with multiple 360° videos. In the experiments, we randomly mask a certain proportion of tiles as black areas and then reconstruct the viewport. Figure 4 demonstrates how the viewport hit ratio affects the peak signal-noise ratio (PSNR) and bitrate of the reconstructed viewport, where the viewport hit ratio refers to the percentage of unmasked tiles in all requested tiles. As illustrated in Figure 4(a), the PSNR of the viewport after/before reconstruction increases with the viewport hit ratio, and the reconstruction mechanism can effectively improve the viewport quality. From Figure 4(b) we can observe that a higher viewport hit ratio brings a higher bitrate of the reconstructed viewport and finally benefits the quality.

In the experiments, we find that the bitrate of the reconstructed viewport can be roughly approximated by linear regression. We use $f(\cdot)$ to represent the fitting function. The output of $f(\cdot)$ is the bitrate of the reconstructed viewport, which is decided by two inputs, i.e., the viewport hit ratio and the perceived quality of the cached tiles that are used to reconstruct the viewport.

2) Time to Prepare the Viewport: The time to prepare the viewport consists of the computation delay for reconstructing viewports and the transmission delay for streaming viewports.

$$t_{cd} = \frac{1}{\mu - p \cdot \lambda} + \frac{1}{\mu} = \frac{1}{\mu - \frac{N_{vr}}{N_{req}} \cdot \lambda} + \frac{1}{\mu}, \tag{9}$$

where μ is the number of viewports that the cache server can reconstruct per second, λ is the average rate at which requests are arriving, and p is the proportion of the reconstruction tasks. We use N_{req} and N_{vr} to represent the number of requests and viewport reconstruction tasks, respectively. In order to obtain more accurate delay estimates, we update p in real-time by counting N_{req} and N_{vr} within sliding time windows which are set to 5 minutes. In equation (9), $1/(\mu - p \cdot \lambda)$ is the average queuing delay of each task with $\mu - p \cdot \lambda > 0$, and $1/\mu$ is the average execution delay.

In fact, μ, p and λ are time-varying parameters, so we divide the time series into successive time windows (usually set to 1 minutes empirically), and update these parameters in real-time to obtain more accurate delay estimates.

The transmission delay in reconstruction tasks is given by:

$$t_{rv} = \frac{size\left(\hat{\mathcal{D}}\right)}{B_{eu}}, \tag{10}$$

where $\hat{\mathcal{D}}$ represents the reconstructed viewport, while B_{eu} and B_{oe} are the estimated throughputs of the fronthaul and backhaul links based on equations (1)–(2), respectively. If the viewport can not be reconstructed, the uncached tiles should be fetched from the origin server, and then all requested tiles will be sent to users. In this case, we use t_{oe} and t_{eu} to indicate the transmission delays from the origin server to the edge server and from the edge server to users, respectively. Their formulas are given in equation (11):

$$\begin{aligned} t_{oe} &= \frac{\sum_{v^k_{s,m,n} \in \mathcal{D}_{miss}} size\left(v^k_{s,m,n}\right)}{B_{oe}}, \\ t_{eu} &= \frac{\sum_{v^k_{s,m,n} \in \mathcal{D}} size\left(v^k_{s,m,n}\right)}{B_{eu}}, \end{aligned} \tag{11}$$

where $\mathcal{D}_{miss}$ indicates the subset of the uncached tiles in $\mathcal{D}$. Similarly, we use $\mathcal{D}_{hit}$ to represent the subset of the cached tiles in $\mathcal{D}$, which will be used later.

Finally, the total delay for reconstructing and delivering the viewport in viewport reconstruction tasks is defined as t_{vr}:

$$t_{vr} = t_{cd} + t_{rv}, \tag{12}$$

while the total delay for fetching the uncached tiles and sending them to users along with the cached ones is t_{fetch}:

$$t_{fetch} = t_{oe} + t_{eu}. \tag{13}$$

They can be used to represent the viewport preparation time $PT(\cdot)$ in equation (5) when calculating the corresponding user QoE, namely QoE_{vr} and QoE_{fetch}.

3) QoE-Driven Reconstruction Trigger Scheme (QRTS): Algorithm 1 summarizes the decision-making process for viewport reconstruction. The algorithm describes the actions that the MEC-Cache server takes whenever it receives a user request. We use $\mathcal{C}$ to indicate the set of all cached content in the MEC-Cache server, and $\mathcal{C}(v^k_s)$ indicates the set of cached tiles in the *s-th* segment of the *v-th* video at the *k-th* level.

First, QRTS performs adaptive representation selection based on the ABR algorithm mentioned in Section III-C, and use the obtained representation level k to retrieve the subset of the cached tiles in $\mathcal{D}(v^k_s)$, called $\mathcal{D}_{hit}(v^k_s)$. Then, it counts the number of tiles in $\mathcal{D}(v^k_s)$ and $\mathcal{D}_{hit}(v^k_s)$, respectively defined as $N(\mathcal{D}(v^k_s))$ and $N(\mathcal{D}_{hit}(v^k_s))$, and calculates the viewport hit ratio δ. After that, it estimates the bitrate of the reconstructed viewport based on the fitting function $f(\cdot)$, and computes t_{vr} and t_{fetch} by equations (9)-(13). In order to maximize QoE, we calculate $QoE_{vr}(\mathcal{D}(v_s))$ and $QoE_{fetch}(\mathcal{D}(v_s))$ for two different scenarios, and choose the one that achieves the higher QoE as our action.

Algorithm 1: QoE-Driven Reconstruction Trigger Scheme

Input: Requested tile set $\mathcal{D}$; Cached content $\mathcal{C}$; Historical network conditions $\mathcal{H}$; Current buffer size BS and its preset range $[BS_{min}, BS_{max}]$.
Output: Whether to reconstruct the viewport or not.

Function *QRTS*():
  Adaptive representation selection: $k \leftarrow ABR(\mathcal{H}, BS, BS_{min}, BS_{max})$;
  Retrieve the hit requested file: $\mathcal{D}_{hit}(v^k_s) = \mathcal{D}(v^k_s) \cap \mathcal{C}(v^k_s)$;
  Calculate the viewport hit ratio: $\delta = N(\mathcal{D}_{hit}(v^k_s))/N(\mathcal{D}(v^k_s))$;
  Calculate the quality of the reconstructed viewport: $Q_p(\mathcal{D}(v_s)) \leftarrow f(\delta, Q_p(\mathcal{D}_{hit}(v^k_s)))$;
  Calculate the delays used to prepare the viewport: t_{cd}, t_{rv}, t_{oe}, t_{eu}, t_{vr} and t_{fetch};
  Calculate the user QoE in two different scenarios: $QoE_{vr}(\mathcal{D}(v_s))$ and $QoE_{fetch}(\mathcal{D}(v_s))$;
  if $QoE_{vr}(\mathcal{D}(v_s)) > QoE_{fetch}(\mathcal{D}(v_s))$ **then**
    N_{vr} += 1;
    return True;
  else
    return False;
  N_{req} += 1;

B. Aggregation-Based Cache Replacement Scheme

In this subsection, we first analyze the cache features of panoramic videos. Then, we propose an aggregation-based cache replacement scheme that selects appropriate tiles to store in the limited space and improves the probability of viewport reconstruction.

1) Cache Feature Analysis: In order to improve the viewport hit ratio and enhance the quality of the reconstructed viewport, it is necessary to increase the aggregation degree of cached tiles in the spatial dimension. It means that the tiles of a segment should be cached centrally rather than distributively.

To characterize the aggregation degree of cached content, we first define $\xi(v^k_{s,m,n})$ as the aggregation degree of the cached tile $v^k_{s,m,n}$ by

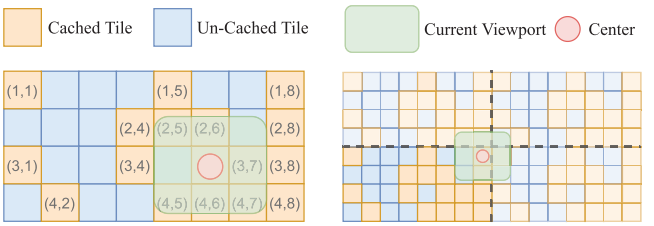
$$\xi\left(v^k_{s,m,n}\right) = \frac{N_c\left(v^k_{s,m,n}\right)}{N_{view}}, \tag{14}$$

where $N_c(v^k_{s,m,n})$ represents the number of cached tiles that current viewport contains when $v^k_{s,m,n}$ is located in the center of user's FoV, and N_{view} represents the total number of tiles in the viewport.

As illustrated in Figure 5, there are two different scenarios when calculating the aggregation degree. In general, the viewport is located within the frame, i.e., $\xi(v^k_{s,m,n}) = 7/9 = 0.78$ in Figure 5(a). Otherwise, we need to stitch together the boundary of the frame before calculation, i.e., $\xi(v^k_{s,m,n}) = 5/9 = 0.56$ in Figure 5(b). The aggregation degree of segment v^k_s is defined as the average aggregation degree of the cached tiles in $\mathcal{C}(v^k_s)$:

$$\xi\left(v^k_s\right) = mean\left\{\xi\left(v^k_{s,m,n}\right) | v^k_{s,m,n} \in \mathcal{C}\left(v^k_s\right)\right\}. \tag{15}$$

However, if all tiles of a segment are cached, the aggregation degree of this segment will reach the maximum value of 1. Unfortunately, this is not what we expect, since we only want



(a) A viewport is within the frame (b) A viewport crosses the boundary

Fig. 5. Two scenarios for calculating the aggregation degree.

to store the frequently requested tiles in the given limited space rather than caching all of them. Therefore, in addition to the aggregation degree in the spatial dimension, the uniformity degree of cached content in the temporal dimension also needs to be considered. We define the cache proportion of segment v_s^k, denoted by $\psi(v_s^k)$, as the ratio of the cached tiles to all tiles in the segment:

$$\psi\left(v_s^k\right) = \frac{N\left(\mathcal{C}\left(v_s^k\right)\right)}{M \cdot N}, \tag{16}$$

where $N(\mathcal{C}(v_s^k))$ represents the number of cached tiles in v_s^k.

2) Problem Formulation: The results of many experiments perform that the ROI size in a segment only accounts for 20%∼30% area [38] (approximately equal to the size of one viewport), which is also intuitive in Figure 1. Therefore, we assume that only viewport-sized tiles are stored for each segment. Then, we define $\vartheta(v_s^k)$ as the composite indicator of segment v_s^k:

$$\begin{aligned}\vartheta\left(v_s^k\right) &= \xi\left(v_s^k\right) - \beta\left[\frac{N\left(\mathcal{C}\left(v_s^k\right)\right) - N_{view}}{M \cdot N}\right]^+ \\ &= \xi\left(v_s^k\right) - \beta\left[\psi\left(v_s^k\right) - \psi_{view}\right]^+,\end{aligned} \tag{17}$$

where β is the penalty for the tiles that exceed the preset storage range, and ψ_{view} represents the cache proportion of a viewport. Further, $\vartheta(v)$ is defined as the average of $\vartheta(v_s^k)$ for the *v-th* video overall segments and quality versions:

$$\vartheta(v) = \frac{\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}}\vartheta\left(v_s^k\right)}{K \cdot S}. \tag{18}$$

The goal of our cache policy is to maximize the composite indicator of the cached content, which can be achieved by selecting appropriate tiles to be stored in the limited cache capacity. The optimization problem can be formulated as

$$\max \quad \vartheta(\mathcal{C}) = \sum_{v\in\mathcal{V}} w_v \cdot \vartheta(v) \tag{19a}$$

$$\text{s.t.} \sum_{v\in\mathcal{V}}\sum_{s\in\mathcal{S}}\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}} size\left(v_{s,m,n}^k\right)\cdot y\left(v_{s,m,n}^k\right) \le C_B, \tag{19b}$$

$$\sum_{v\in\mathcal{V}} w_v = 1, \tag{19c}$$

where w_v is the request probability of the *v-th* video.

Constraint (19b) ensures that the size of cached data does not exceed the cache capacity C_B, where $y(v_{s,m,n}^k) = 1$ if $v_{s,m,n}^k$ is stored in the cache server. Otherwise, $y(v_{s,m,n}^k) = 0$. Constraint (19c) normalizes the popularity of videos.

Problem (19) is an Integer Linear Programming (ILP) problem. As the problem is NP-hard, it is extremely challenging to solve the problem optimally within polynomial time. Therefore, we propose a heuristic-based solution in Algorithm 2 for cache replacement, named the aggregation-based cache replacement scheme.

Algorithm 2: Aggregation-Based Cache Replacement Scheme

Input: Tile to be cached $v_{s,m,n}^k$; Cached content $\mathcal{C}$; Cache capacity C_B.

```
C(v_s^k).append(v_{s,m,n}^k) (store v_{s,m,n}^k on the cache);
while cache.size() > C_B do
    if ψ(v_s^k) > ψ_view then
        for v_{s,x,y}^k ∈ C(v_s^k) do
            Assume that v_{s,x,y}^k is removed from C(v_s^k);
            Compute ξ(v_s^k) of the rest tiles in C(v_s^k);
        Remove the tile with the largest ξ(v_s^k);
    else
        Remove the tile with the smallest γ(v_{s,m,n}^k);
```

3) Aggregation-Based Cache Replacement Scheme (ACRS): Algorithm 2 shows the process of ACRS, in which a cache replacement metric, $\gamma(v_{s,m,n}^k)$, that takes both age and frequency into consideration is utilized:

$$\gamma\left(v_{s,m,n}^k\right) = \sum_{t=0}^{T}\left[R_t\left(v_{s,m,n}^k\right)\cdot\phi(T-t)\right], \tag{20}$$

$$R_t\left(v_{s,m,n}^k\right) = \sum_{u\in\mathcal{U}} x_{u,t}^{v_{s,m,n}^k}, \quad \phi(T-t) = e^{-\sigma(T-t)}, \tag{21}$$

where $R_t(v_{s,m,n}^k)$ refers to the number of requests to tile $v_{s,m,n}^k$ made by all users at time slot t, T represents the current time slot, and $\phi(T-t)$ is a kernel function that decreases the weights of old tiles. Meanwhile, we use σ to represent a non-negative empirical constant in the kernel function.

When the MEC-Cache server decides to remove some cached tiles to make room for a new one, the new tile $v_{s,m,n}^k$ will be stored on the cache first, and then one of the following two cases happens.

If the condition $\psi(v_s^k) > \psi_{view}$ is satisfied, the MEC-Cache server will remove tiles in $\mathcal{C}(v_s^k)$ to ensure that the cache proportion does not increase anymore, which avoids the occurrence of excessive caching. In this situation, we calculate the aggregation degree $\xi(v_s^k)$ of the remaining cached tiles by assuming each tile in $\mathcal{C}(v_s^k)$ is removed. Finally, the tile with the largest remaining $\xi(v_s^k)$ will be actually removed because its removal has minimal effect on the aggregation degree. Consider an example in Figure 5(a), assuming that tile $v_{s,1,1}^k$ is removed, the aggregation degree of the remaining cached tiles in $\mathcal{C}(v_s^k)$ is $\xi(v_s^k) = 0.59$, while this value would be $\xi(v_s^k) = 0.65$ if tile $v_{s,4,2}^k$ were to be removed. Similarly, we can get the results assuming any one of the tiles is removed and find the maximum $\xi(v_s^k)$ among them is 0.65, which happens when tile $v_{s,4,2}^k$ is removed. As a result, tile $v_{s,4,2}^k$ is

Algorithm 3: Reconstruction-Based Tile Adaptive Streaming

Input: Requested tile set $\mathcal{D}$; Subsets $\mathcal{D}_{hit}$ and $\mathcal{D}_{miss}$; Cached content $\mathcal{C}$; Cache capacity C_B.

```
while receive a request for D do
    if D ∈ C (all requested tiles are hit) then
        Send D to the user;
    else
        if QRTS() then
            Reconstruct the viewport and send it back;
        else
            for v^k_{s,m,n} ∈ D do
                if v^k_{s,m,n} ∈ D_hit then
                    Send v^k_{s,m,n} to the user;
                else
                    Fetch v^k_{s,m,n} from the origin server;
                    Send v^k_{s,m,n} to the user;
                    if cache.size() < C_B then
                        Store v^k_{s,m,n} on the cache;
                    else
                        Cache replacement using ACRS;
```

actually removed. Otherwise, if the condition is not satisfied, the same process will be performed by removing the tile with the smallest $\gamma(v^k_{s,m,n})$, which only considers the popularity of tiles. These operations are repeated until the cache capacity is sufficient to store the new tile.

C. Reconstruction-Based Tile Adaptive Streaming

Algorithm 3 shows the overview of the reconstruction-based tile adaptive streaming in the MEC-Cache server. If all the tiles within the requested viewport are hit, the requested content will be sent to the user. Otherwise, the MEC-Cache server will determine whether to reconstruct the viewport or not by using the QRTS in Algorithm 1. When the outcome of the algorithm is true, the complete viewport will be reconstructed and sent back. Otherwise, the cache server will be ordered to fetch the uncached tiles one by one from the remote content server, and the ACRS in Algorithm 2 will be executed when the cache capacity is insufficient to accommodate more content.

D. Time Complexity Analysis

Let $\mathcal{D}(v^k_s)$ and r represent the requested tile set and the number of user requests, respectively. The computational complexity of QRTS is $O(N(\mathcal{D}(v^k_s)))$, while that of ACRS is $O(N(\mathcal{C}(v^k_s))^2)$. Consequently, the overall time complexity of the proposed reconstruction-based tile adaptive streaming algorithm is $O(r \cdot N(\mathcal{D}(v^k_s)) \cdot N(\mathcal{C}(v^k_s))^2)$.

V. Experimental Setup

A. Experimental Settings

The performance evaluation is based on head movement data traces of 50 users watching ten 360° videos [33], including fast-paced and slow-paced videos. The details of the 360° videos used in our experiments are presented in Table I. In addition, the head-tracking dataset contains the information of users' FoVs and requested tiles for every frame.

TABLE I
Specifications of Video Dataset

ID	Video Name	Category	Used Segment
1	Hog Rider	NI, slow-paced	0:00 - 1:00
2	Kangaroo Island	NI, slow-paced	0:01 - 1:01
3	SFR Sport	NI, slow-paced	0:16 - 1:16
4	Shark Shipwreck	NI, slow-paced	0:30 - 1:30
5	Roller Coaster	NI, fast-paced	0:20 - 1:20
6	Mega Coaster	NI, fast-paced	1:30 - 2:30
7	Driving with	NI, fast-paced	0:48 - 1:48
8	Perils Panel	NI, slow-paced	0:10 - 1:10
9	Pac-Man	CG, fast-paced	0:10 - 1:10
10	Chariot Race	CG, fast-paced	0:02 - 1:02

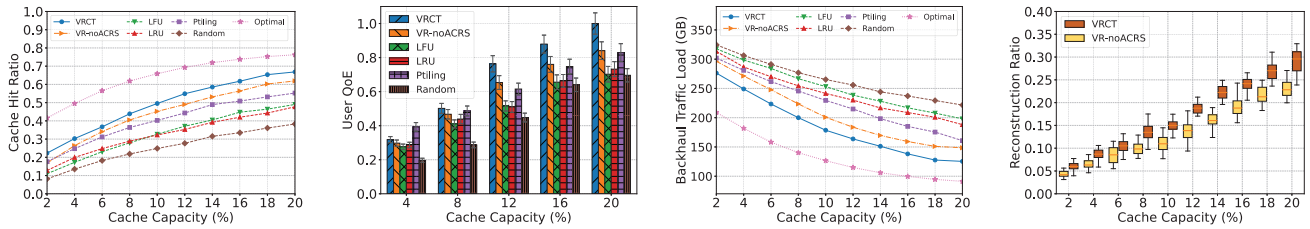
[1] CG: Computer-Generated; NI: Natural Image; Video Dataset [33].

On the server side, each 360° video is cut to a fixed duration of 1 minute, encoded and projected by equirectangular projection with 4K resolution (3840×1920) and 30 frames per second. In order to acquire multiple quality versions, we use the HEVC encoder kvazaar [39] to transcode 360° videos into 5 different representations at 2 Mbps, 3 Mbps, 5 Mbps, 7 Mbps and 10 Mbps. Then, we spatially partition the 360° videos into 4×8 tiles (expressed by rows × columns). Finally, GPAC MP4Box [40] is used to package the encoded videos in MP4 containers and generate Media Presentation Description (MPD) files which split each video temporally into 2-second long segments.

On the client side, We conduct a total of ten rounds of experiments and in each round, a total of 5000 requests for 360° videos from 50 simulated users were sent to the edge server, following a Poisson distribution [41] with rate $\lambda_r = 20$ [requests/min]. We assume that the video popularity is decided by using a Zipf distribution [42] with shape parameter $\eta_v = 1.2$. Hence, the probability of the *v-th* popular 360° video to be selected is given by $p_v = \frac{1/v^{\eta_v}}{\sum_{v \in \mathcal{V}} 1/v^{\eta_v}}$.

The experiments are conducted in a real-world testbed with a prototype especially built to perform the delivery of 360° videos with the support of three x86 servers. We utilize one server as the source server, one server as the edge server, and one client-server acting as multiple clients. We implement Nginx [43] to all servers to enable the DASH-based video delivery and use Django [44] with uWSGI to implement VRCT at the edge server. In addition, we use the Linux Traffic Control tool (i.e., tc) [45] to simulate the dynamic bandwidth, which is generated based on a 4G/LTE dataset [46] with varying throughput patterns. To support the playback of 360° videos, we linearly scale the network throughput, and the throughput values fall between 2.7 Mbps and 13.1 Mbps, with an average of 5.4 Mbps. The average number of viewports that the cache server can reconstruct per second is $\mu = 1.8$ [requests/s].

For viewport reconstruction, we choose the MAE architecture [17], which is a well-known encoder-decoder architecture implemented by Vision Transformers (ViT) [47], to reconstruct the viewport frames. To achieve a better reconstruction effect, ViT-Huge [47] is employed as our pre-training model.



(a) Cache Hit Ratio (b) QoE Performance (c) Backhaul Traffic Load (d) Reconstruction Ratio

Fig. 6. Evaluation of 360° video streaming performance with respect to change in cache capacity.

B. Baseline Algorithms

The performance of the proposed VRCT for tile-adaptive 360° video streaming is evaluated by choosing among the following existing caching algorithms.

1) *Random:* Random caching policy randomly selects tiles to remove whenever the cache capacity is exceeded.

2) *LRU (Least Recently Used):* In LRU scheme, whenever the cache capacity is exceeded, the tile with the longest idle time has the highest rank for being removed.

3) *LFU (Least Frequently Used):* In LFU scheme, whenever the cache capacity is exceeded, the tile that is least frequently accessed has the highest rank for being removed.

(4) *VR-noACRS:* VR-noACRS is a caching policy that utilizes viewport reconstruction and always removes the tile with the smallest metric in equation (20), instead of using ACRS as the cache replacement algorithm. It is designed for ablation experiments.

5) *Ptiling:* Ptiling [35] is a popularity-aware 360° video streaming algorithm, which encodes the popularly viewed areas as macrotiles to save bandwidth and enhance the QoE.

6) *Optimal:* For the optimal solution, we assume that the prior knowledge about the user requests and the video popularity is known in advance, and the tiles with the highest requested probability are cached in the MEC-Cache server.

C. Performance Metrics

In terms of metrics, we use the following metrics to verify the performance of different schemes.

1) *Cache Hit Ratio:* The fraction of the requested tiles that are stored in the cache server and obtained by the viewport reconstruction technology. A higher cache hit ratio indicates less need to retrieve new data from the remote content server.

2) *User QoE:* User QoE is first calculated by equation (3)-(8), and then the QoE metrics of different schemes are limited to the range [0, 1] through numerical normalization, where the largest QoE value under all parameters is set to 1.

3) *Backhaul Traffic Load:* The bandwidth consumed for streaming 360° videos from the content server to the MEC-Cache server, which reflects the load of the backhaul path.

4) *Reconstruction Ratio:* The proportion of reconstructed viewports in all viewports delivered to users.

VI. Performance Evaluation

A. Effect of Cache Capacity

We first study the impact of cache capacity on 360° video streaming. To this end, we set $\eta_v = 1.2$ and vary the cache capacity in the range [2, 20]% of the video library size to evaluate the aforementioned performance metrics, and the results are shown in Figure 6.

As we can see from Figure 6(a), the cache hit ratio increases with the cache capacity for all schemes. Due to the introduction of the reconstruction mechanism, VRCT and VR-noACRS can further improve the hit ratio. Meanwhile, VRCT outperforms other schemes (except the optimal one) in all ranges of cache size. Specifically, for large cache capacities, i.e., 20%, VRCT improves the cache hit ratio by 5%, 11%, 18%, 19% and 29% compared with VR-noACRS, Ptiling, LFU, LRU and Random, respectively.

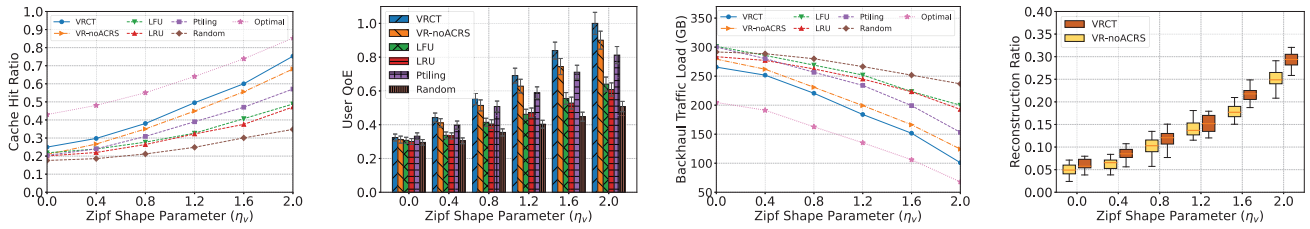
As illustrated in Figure 6(b), the QoE performance of VRCT is lower than Ptiling but still higher than the others when the cache size is small. This is because there are not enough cached tiles that can be used to reconstruct the complete viewport, and the missing tiles need to be fetched from the remote content server. However, as the cache capacity increases, the QoE of VRCT increases significantly and outperforms all baselines when the cache capacity is 8% and above. Specifically, the proposed VRCT improves QoE by at least 17% when the cache capacity is 20% of the video library.

The results in Figure 6(c) and Figure 6(d) depict that VRCT can significantly reduce backhaul traffic load by reconstructing part of the requested viewports regardless of the size of the cache capacity. For example, it can reduce the backhaul usage by 12%∼44% compared with other schemes when the cache capacity is 20% of the video library. Furthermore, we can note that VRCT outperforms VR-noACRS, because the introduction of ACRS optimizes the cache features of the cached content, which will be discussed in Section VI-D, and makes it easier to reconstruct viewports, thus reducing bandwidth consumption for uncached content.

B. Effect of Video Popularity Distribution

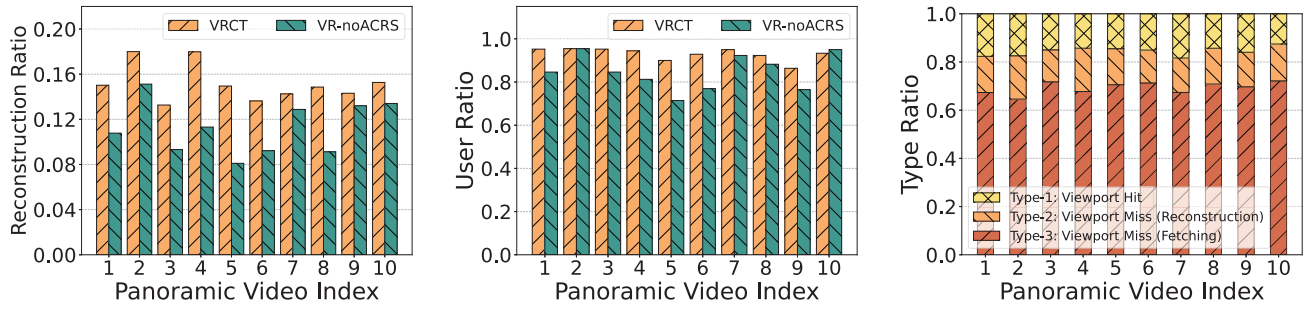
To observe the effect of the video popularity distribution on different schemes, we set the cache capacity to 10% of the video library and vary the Zipf shape parameter η_v in the range [0, 2] for different popularity distributions.

The results in Figure 7(a) show that VRCT outperforms all baselines (except the optimal one) under all settings. Moreover, an increase in η_v leads to a smaller performance gap between the optimal scheme and VRCT, and the gap reduces to less than 10% when $\eta_v = 2$. This is because a larger η_v makes the video popularity distribution get steeper, which is beneficial to the viewport hit ratio of the reconstruction tasks.



(a) Cache Hit Ratio (b) QoE Performance (c) Backhaul Traffic Load (d) Reconstruction Ratio

Fig. 7. Evaluation of 360° video streaming performance with respect to change in Zipf shape parameter.



(a) Ratio of reconstruction tasks (b) Ratio of benefited users (c) Ratio of three feedback types

Fig. 8. Ratio of reconstruction tasks, users who benefit from reconstruction, and three feedback types with respect to change in video type.

As shown in Figure 7(b), the QoE of VRCT increases sharply as η_v increases. VRCT provides over 10% improvement compared with other schemes when η_v is 2. Moreover, Figure 7(c) and Figure 7(d) show that skewed user request behavior will increase the reconstruction ratio, thus alleviating backhaul traffic load.

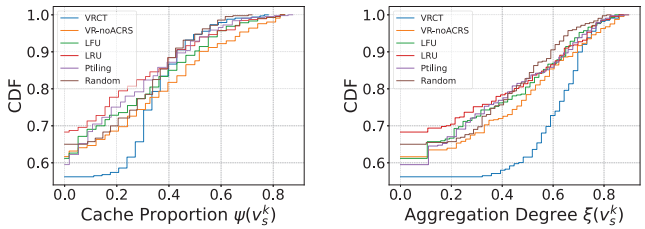
C. Effect of Video Types

In this experiment, we analyze the effect of the proposed ACRS with different categories of 360° videos. Figure 8(a) shows the reconstruction ratios in ten different 360° videos. The results demonstrate that no matter what kind of video is requested, the reconstruction ratio of VRCT is higher than that of VR-noACRS. This is because the introduction of ACRS optimizes the cache features, improving the probability of viewport reconstruction. Figure 8(b) shows the percentage of users who benefit from reconstruction. Compared with VR-noACRS, VRCT brings improvements in terms of the benefited user ratio for most videos due to the use of ACRS, which is very important, especially in the context of a network with multiple users who have different video preferences.

In order to assess Algorithm 3, in terms of viewport hits, reconstruction of the viewport and fetching the missing content, we explicitly plot the ratio of the three types in VRCT in Figure 8(c). It can be noted how, VRCT achieves a reconstruction ratio of around 15% for all types of videos, which indicates the robustness of the proposed algorithm across multiple video categories.

D. Effect of Cache Features

In this experiment, we count the distributions of two cache features obtained from VRCT and baselines, namely cache proportion and aggregation degree. Their Cumulative Distribution Functions (CDFs) are illustrated in Figure 9.



(a) CDF of cache proportion (b) CDF of aggregation degree

Fig. 9. CDFs of two different cache features.

As shown in Figure 9(a), all baseline algorithms result in near uniform distribution. On the other hand, the cache proportion of VRCT is mainly distributed in the interval of 20%∼40%, which is a more desirable distribution than the uniform distribution. This is because an overly large cache proportion can lead to a waste of cache capacity, and an overly small cache proportion can make it difficult to trigger the reconstruction mechanism, resulting in an increase in the backhaul traffic load. Thanks to the cache replacement scheme ACRS, VRCT can maintain an appropriate cache proportion and increase the number of different segments stored in the cache by an average of 10%, so that the limited cache capacity can be effectively utilized. Figure 9(b) shows that VRCT significantly increases the aggregation degree of the cached content compared with other schemes, which is important to improve the viewport hit ratio of the reconstruction tasks.

VII. Discussions

The framework design and experimental evaluations presented in this paper demonstrate that VRCT can

intelligently realize viewport reconstruction-based 360° video adaptive streaming by integrating network conditions, client status and cache information. The proposed VRCT flexibly selects between three different approaches to provide viewing tiles to 360° video consumers. It effectively alleviates the transmission pressure of the backhaul path, significantly enhances the user QoE, and exhibits great robustness in complex multi-user caching scenarios. However, there are still some limitations of the proposed solution.

First, VRCT may degrade the viewing experience of users in non-mainstream viewing modes. To improve the probability of viewport reconstruction, the proposed ACRS aims to increase the aggregation degree of cached tiles during the cache replacement process. In other words, ACRS deliberately removes some cached tiles outside of the ROI to make room for new tiles located inside it. This means that VRCT sacrifices the viewing experience of a few users with non-mainstream watching traces to improve the QoE of the majority of users.

Secondly, the adaptive bitrate streaming of VRCT can be further enhanced in the tile-based mechanism. Since each tile in the requested viewport is transmitted independently, delivering different quality versions of tiles in the same viewport is feasible. However, it is difficult to reconstruct the viewport by using tiles with different quality versions, and the decrease in perceived quality caused by the quality variation among different tiles in the same viewport also needs to be considered in QoE modeling.

Thirdly, the proposed system model is scalable to multi-edge assisted content delivery scenarios. In the experiments, we only use one edge server to cache tiles and reconstruct viewports, but in reality, a large number of edges are expected to be available in 5G network environments. In the multi-edge assisted network, introducing collaborative caching strategies helps to adaptively aggregate video content. For instance, by redirecting user requests and communicating between edge nodes, contents can be congregated automatically in different edges by video type. In this way, each edge can provide better caching and reconstruction services for specific 360° videos.

Addressing these challenges involves complex client behavior analysis, adaptive mechanism optimization and cooperative scheduling models, which is not trivial. Future work will try to improve the proposed method by focusing on some of these limitations.

VIII. Conclusion

In this paper, we propose VRCT, a viewport reconstruction-based 360° video caching solution for tile-adaptive streaming, which consists of a QoE-driven reconstruction trigger scheme and an aggregation-based cache replacement scheme that can support viewport reconstruction. The employment of the reconstruction mechanism reduces the need for retrieving new data, decreasing backhaul traffic load. In order to enhance the user QoE, we design a decision-making process for viewport reconstruction, and choose the action that leads to a higher QoE by building models to predict viewport bitrate and consumed delay. To increase the probability of viewport reconstruction, we propose an innovative cache replacement algorithm, which can improve the spatial aggregation of the cached tiles and store more different segments. Extensive testing results demonstrate that VRCT outperforms other schemes, as it improves the cache hit ratio by up to 29%, reduces the backhaul usage by up to 44% and significantly enhances user QoE. Moreover, VRCT optimizes the cache features to facilitate viewport reconstruction and supports different video types in 360° video streaming.

References

[1] S. F. Langa, M. M. Climent, G. Cernigliaro, and D. R. Rivera, "Toward hyper-realistic and interactive social VR experiences in live TV scenarios," *IEEE Trans. Broadcast.*, vol. 68, no. 1, pp. 13–32, Mar. 2022.

[2] C. Westphal, "Challenges in networking to support augmented reality and virtual reality," in *Proc. IEEE ICNC*, 2017, pp. 1–9.

[3] *Virtual Reality (VR) Market—Growth, Trends, COVID-19 Impact, and Forecasts (2021–2026)*, Morder Intell., Hyderabad, India, Jan. 2021.

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

[5] A. Yaqoob, T. Bi, and G.-M. Muntean, "A survey on adaptive 360 video streaming: Solutions, challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2801–2838, 4th Quart., 2020.

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

[7] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski, "Optimal set of 360-degree videos for viewport-adaptive streaming," in *Proc. 25th ACM Int. Conf. Multimedia*, Mountain View, CA, USA, Oct. 2017, pp. 943–951.

[8] H. Ahmadi, O. Eltobgy, and M. Hefeeda, "Adaptive multicast streaming of virtual reality content to mobile users," in *Proc. Thematic Workshops ACM Multimedia*, Mountain View, CA, USA, Oct. 2017, pp. 170–178.

[9] Q. Lu, C. Li, J. Zou, K. Tang, Q. Wang, and H. Xiong, "Transcoding-enabled edge caching and delivery for tile-based adaptive 360-degree video streaming," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Sydney, NSW, Australia, Dec. 2019, pp. 1–4.

[10] T. Yang, Z. Tan, Y. Xu, and S. Cai, "Collaborative edge caching and transcoding for 360° video streaming based on deep reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25551–25564, Dec. 2022.

[11] J. Shi, L. Pu, and J. Xu, "Allies: Tile-based joint transcoding, delivery and caching of 360° videos in edge cloud networks," in *Proc. IEEE 13th Int. Conf. Cloud Comput. (CLOUD)*, Beijing, China, Dec. 2020, pp. 337–344.

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

[13] J. Dai, Z. Zhang, S. Mao, and D. Liu, "A view synthesis-based 360° VR caching system over MEC-enabled C-RAN," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 10, pp. 3843–3855, Oct. 2020.

[14] M. Dasari, A. Bhattacharya, S. Vargas, P. Sahu, A. Balasubramanian, and S. R. Das, "Streaming 360-degree videos using super-resolution," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Toronto, ON, Canada, Jul. 2020, pp. 1977–1986.

[15] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 2536–2544.

[16] H. Bao, L. Dong, and F. Wei, "BeIT: BERT pre-training of image transformers," in *Proc. Comput. Vis. Pattern Recognit.*, 2021, pp. 1–9.

[17] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. B. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, New Orleans, LA, USA, Jun. 2022, pp. 15979–15988.

[18] R. Wang et al., "BEVT: BERT pretraining of video transformers," in *Proc. CVPR*, New Orleans, LA, USA, Jun. 2022, pp. 14713–14723.

[19] A. Gupta, S. Tian, Y. Zhang, J. Wu, R. Martín-Martín, and L. Fei-Fei., "MaskViT: Masked visual pre-training for video prediction." 2022. [Online]. Available: https://arxiv.org/abs/2206.11894

[20] Z. Wang, H. Zhang, Z. Cheng, B. Chen, and X. Yuan, "MetaSCI: Scalable and adaptive reconstruction for video compressive sensing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 2083–2092.
[21] S. Afzal, J. Chen, and K. Ramakrishnan, "Characterization of 360-degree videos," in *Proc. Workshop Virtual Real. Augmented Real. Netw.*, Los Angeles, CA, USA, Aug. 2017, pp. 1–6.
[22] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, San Jose, CA, USA, Dec. 2016, pp. 583–586.
[23] T. Stockhammer, "Dynamic adaptive streaming over HTTP– standards and design principles," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, San Jose CA, USA, Feb. 2011, pp. 133–144.
[24] S. Wang et al., "SalientVR: Saliency-driven mobile 360-degree video streaming with gaze information," in *Proc. 28th Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, Oct. 2022, pp. 542–555.
[25] C. Zheng, J. Yin, Y. Guan, X. Zhang, and Z. Guo, "STC: Enabling 16k VR streaming on mobile platforms with FoV tracking," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Taipei, Taiwan, Dec. 2020, pp. 1–6.
[26] C. Wu, Z. Wang, and L. Sun, "PaaS: A preference-aware deep reinforcement learning approach for 360° video streaming," in *Proc. 31st ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video*, New York, NY, USA, Jul. 2021, pp. 34–43.
[27] C. Madarasingha and K. Thilakarathna, "Vastile: Viewport adaptive scalable 360-degree video frame tiling," in *Proc. 29th ACM Int. Conf. Multimedia*, Chengdu, China, Oct. 2021, pp. 34–43.
[28] A. Yaqoob and G.-M. Muntean, "A combined field-of-view prediction-assisted viewport adaptive delivery scheme for 360° videos," *IEEE Trans. Broadcast.*, vol. 67, no. 3, pp. 746–760, Sep. 2021.
[29] A. Yaqoob, M. A. Togou, and G.-M. Muntean, "Dynamic viewport selection-based prioritized bitrate adaptation for tile-based 360° video streaming," *IEEE Access*, vol. 10, pp. 29377–29392, 2022.
[30] C. Madarasingha, K. Thilakarathna, and A. Zomaya, "OpCASH: Optimized utilization of MEC cache for 360-degree video streaming with dynamic tiling," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Pisa, Italy, Mar. 2022, pp. 34–43.
[31] D. He, J. Jiang, C. Westphal, and G. Yang, "Efficient edge caching for high-quality 360-degree video delivery," in *Proc. Int. Conf. Multimedia Model.*, Daejeon, South Korea, Jan. 2020, pp. 39–51.
[32] J. Fu, Z. Chen, X. Chen, and W. Li, "Sequential reinforced 360-degree video adaptive streaming with cross-user attentive network," *IEEE Trans. Broadcast.*, vol. 67, no. 2, pp. 383–394, Jun. 2021.
[33] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "360 video viewing dataset in head-mounted virtual reality," in *Proc. 8th ACM Multimedia Syst. Conf.*, Taipei, Taiwan, Jun. 2017, pp. 211–216.
[34] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Rubiks: Practical 360-degree streaming for smartphones," in *Proc. 16th Annu. Int. Conf. Mobile Syst. Appl. Services*, New York, NY, USA, Jun. 2018, pp. 482–494.
[35] X. Chen, T. Tan, and G. Cao, "Popularity-aware 360-degree video streaming," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Vancouver, BC, Canada, Jul. 2021, pp. 1–10.
[36] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "DRL360: 360-degree video streaming with deep reinforcement learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Paris, France, Jun. 2019, pp. 1252–1260.
[37] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Favor: Fine-grained video rate adaptation," in *Proc. 9th ACM Multimedia Syst. Conf.*, New York, NY, USA, Jun. 2018, pp. 64–75.
[38] S. Park, A. Bhattacharya, Z. Yang, M. Dasari, S. R. Das, and D. Samaras, "Advancing user quality of experience in 360-degree video streaming," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, Pisa, Italy, Mar. 2019, pp. 1–9.
[39] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen, "Kvazaar: Open-source HEVC/H.265 encoder," in *Proc. 24th ACM Int. Conf. Multimedia*, Amsterdam, The Netherlands, Oct. 2016, pp. 1179–1182.
[40] J. Le Feuvre, "GPAC filters," in *Proc. 11th ACM Multimedia Syst. Conf.*, Istanbul, Turkey, Jun. 2020, pp. 249–254.
[41] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 4, pp. 333–344, 2006.
[42] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. Conf. Comput. Commun. 18th Annu. Joint Conf. IEEE Comput. Commun. Soc. (IEEE INFOCOM)*, New York, NY, USA, Mar. 1999, pp. 1–9.
[43] W. Reese, "NGINX: The high-performance Web server and reverse proxy," *Linux J.*, vol. 2008, no. 173, p. 2, 2008.
[44] J. Forcier, P. Bissex, and W. J. Chun, *Python Web Development With Django*. London, U.K.: Addison-Wesley Prof., 2008.
[45] W. Almesberger et al., "Linux network traffic control—Implementation overview," 1999. [Online]. Available: http://marco.uminho.pt/disciplinas/ST/traffic_control_on_linux.pdf
[46] J. van der Hooft et al., "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.
[47] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. ICLR*, 2021, pp. 1–6.

Ziwen Ye received the B.Eng. degree in information engineering from Southeast University, China, in 2017. He is currently pursuing the master's degree in computer technology with Tsinghua University. He is also a Visiting Student with the Peng Cheng Laboratory, Shenzhen, China. His research interests include edge-assisted adaptive multimedia streaming, in-network caching/computing, and load-balanced network.

Qing Li (Member, IEEE) received the B.S. degree in computer science and technology from the Dalian University of Technology, Dalian, China, in 2008, and the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2013. He is currently an Associate Researcher with Peng Cheng Laboratory, Shenzhen, China. His research interests include reliable and scalable routing of the Internet, in-network caching/computing, intelligent self-running network, and edge computing.

Xiaoteng Ma received the B.Eng. degree in electrical engineering from the Huazhong University of Science and Technology, China, in 2017. He is currently pursuing the Ph.D. degree in data science and information technology with Tsinghua–Berkeley Shenzhen Institute. He is also a Visiting Student with the Peng Cheng Laboratory, Shenzhen, China. His research interests include edge-assisted multimedia delivery, adaptive multimedia and multimedia streaming, and resource allocation on hybrid cloud–edge-client network.

Dan Zhao received the bachelor's degree in telecommunications from the Beijing University of Posts and Telecommunications in 2011, and the Ph.D. degree in electronic engineering from the Queen Mary University of London in 2015. She is currently an Assistant Researcher with Peng Cheng Laboratory, Shenzhen, China. She was a Postdoctoral Researcher with the School of Electronic Engineering, Dublin City University.

Yong Jiang (Member, IEEE) received the B.S. and Ph.D. degrees from Tsinghua University in 1998 and 2002, respectively. He is currently a Full Professor with Tsinghua Shenzhen International Graduate School, China. He is also a Professor with the Peng Cheng Laboratory, Shenzhen, China. He mainly focuses on the future Internet, edge computing, multimedia transmission, and AI for networks.

Lianbo Ma (Senior Member, IEEE) received the B.Sc. degree in communication engineering and the M.Sc. degree in communication and information system from Northeastern University, Shenyang, China, in 2004 and 2007, respectively, and the Ph.D. degree from the University of Chinese Academy of Sciences, China, in 2015. He is currently a Professor with Northeastern University, China. He has published over 90 journal articles, books, and refereed conference papers. His current research interests include computational intelligence and machine learning.

Bo Yi (Member, IEEE) is currently a Lecturer of Computer Science and Engineering with Northeastern University, China. He has authored and coauthored more than 20 journal and conference articles on IEEE TPDS, TCC, IEEE COMMUNICATIONS LETTER, IWQoS, AAAI, and *Computer Networks*. His research interests include service computing, routing, virtualization, cloud computing in SDN, NFV, and DetNet. He is currently the Reviewer of IEEE COMMUNICATIONS SURVEY AND TUTORIAL, *Communications Letter*, *Computer Networks*, and *Journal of Network and Computer Applications*.

Gabriel-Miro Muntean (Fellow, IEEE) received the Ph.D. degree from the School of Electronic Engineering, Dublin City University (DCU), Ireland, for his research on quality-oriented adaptive multimedia streaming in 2003, where he is currently a Professor. He is the Co-Director of the DCU Performance Engineering Laboratory. He has published over 500 papers in prestigious international journals and conferences, authored four books and 26 book chapters, and edited seven other books. His research interests include quality-oriented and performance related issues of adaptive multimedia delivery, performance of wired and wireless communications, energy-aware networking, and personalized technology-enhanced learning. He is an Associate Editor of the IEEE TRANSACTIONS ON BROADCASTING, the Multimedia Communications Area Editor of the IEEE COMMUNICATION SURVEYS AND TUTORIALS, and a reviewer for other important international journals, conferences, and funding agencies. He is a Fellow of the IEEE Broadcast Technology Society.