Edge Cache Replacement Strategy for SVC-Encoding Tile-Based 360-degree Panoramic Streaming

Ju Zhang
School of Computer Science and Technology, Nanjing Normal University
Nanjing, China
2446465156@qq.com

Qian Gao
School of Computer Science and Technology, Nanjing Normal University
Nanjing, China
gaoqian@njnu.edu.cn

Guoqiang Zhang
School of Computer Science and Technology, Nanjing Normal University
Nanjing, China
guoqiang@ict.ac.cn

2020 3rd International Conference on Hot Information-Centric Networking (HotICN) | 978-1-7281-9216-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/HotICN50779.2020.9350769

***Abstract*—In recent years, with the rapid development of virtual reality (VR), 360-degree panoramic video has grown up to become a popular Internet service. Streaming 360-degree video on today's Internet is challenging due to its ultra-high bandwidth and ultra-low motion-to-photon latency requirements. Although some VR solutions use tiling technology and Scalable Video Coding(SVC) encoding for 360-degree video transmission to reduce bandwidth consumption, it is still being challenging to transmit 360-degree video streams from remote content servers due to network latency. Caching popular video content at the edge of the network can decrease network latency and bandwidth consumption by reducing the number of future requests that have to be sent all the way to remote content servers. Because of the limited cache size, it is essential to decide which content to cache and how to replace the cache when the cache size is insufficient. In this paper, we propose an edge cache replacement strategy named Size-Popularity-Layer-FoV strategy(SPLF) for tile-based 360-degree video streaming using SVC. To improve caching performance, every video chunks' cache value is estimated according to its popularity, size, SVC layer and whether it is in FoV. Simulation results manifest that the proposed strategy can not only improve the hit rate and video quality compared to various baseline algorithms, but also decrease the playback stalls and the duration of rebuffering.**

***Index Terms*—Edge caching, Cache replacement strategy, 360-degree panoramic streaming, Scalable Video Coding, Field of View**

I. Introduction

As providing a fully immersed user experience, virtual reality (VR) and augmented reality(AR) contents and applications have received increasing attention from both research and industry communities. According to Cisco's Visual Networking index [1], the traffic generated by VR/AR applications will increase 12 folds from 2017 to 2022, with a compound annual growth rate of 65%. However, providing 360-degree video streaming services in today's Internet still faces severe challenges: ultra-high bandwidth requirements and ultra-low Motion-to-Photon (MTP)delay (delay between head movement and video rendering) requirements. The bandwidth required to stream a 360-degree video is an order of magnitude larger than that required for a traditional (2-D) video. The data rate of a 360-degree video that delivers a 4k stream to each eye and allows a full 360-degree viewing range requires about 400 Mb/s, compared to about 25 Mb/s for a traditional 4k video [2]. Also, in order to improve the immersion of the viewing experience, the MTP delay should be kept very small, usually within 20 milliseconds, otherwise it will cause physiological dizziness [3].

Early transmission models of 360-degree video simply deal with the panorama as a high resolution video to be encoded with uniform quality. However, since a user's Field of View (FoV) only covers a small portion of the entire sphere, e.g., 110 degrees horizontally and 90 degrees vertically, streaming the entire 360-degree video in full resolution is costly and wasteful. It has been shown that Linear Regression(LR) approach can accurately predict user's future FoV within one second [4]. Tiling [5] [6] [7]is an effective technique towards bandwidth reduction, since the parts of the panorama outside the FoV do not need to be transmitted to the client. But when the user's FoV changes suddenly, tiling may cause more video rebuffering events. In [8] [9] [10], Scalable Video Coding (SVC) encoding method is applied to tiles. SVC can encode the video into one base layer and several enhancement layers. The base layer can be decoded independently, while the enhancement layer must rely on base layer decoding. No matter what the user's FoV is, the base layer can be prefetched and buffered at the client for a long time, which greatly reduces rebuffering. Although tile-based approach and SVC encoding for 360-degree video transmission can effectively reduce bandwidth requirements, it is still challenging to transmit 360-degree video streams from remote content servers due to network latency.

Caching is another powerful solution. Deploying a cache between clients and content servers can effectively decrease bandwidth consumption and network latency. In the traditional TCP/IP, content is bound to the server's address, and IP-centric addressing characteristic fundamentally limit the network's response to large-scale video distribution. Therefore, in order to break the above restrictions, an information-centric network (ICN) [11] came into being. Its name-based routing, universal in-network caching, hop-by-hop transmission, and

built-in multicast support provide opportunities for optimized SVC-encoded 360-degree video delivery. With the continuous increase of mobile data traffic, recent studies have tried to deploy cache at the edge of the network closer to the user [12]. This approach can satisfy some customers' requests for video tiles without contacting the remote server, thereby reducing core network load, response time, and maintaining user QoE [13]–[16]. Tagami et al. [17]proposed a 360-degree panoramic live video streaming transmission scheme based on tiling transmission in ICN. This solution performed transcoding at the edge node close to the client, avoiding repeated transmission of multiple different quality versions of videos on the network.

The difference between 360-degree video caching and conventional video caching is that each tile may need to be cached simultaneously at multiple resolutions, since it may appear in different positions at different viewports [18]. There are two key problems in this case. One is how to choose an appropriate video representation in the local cache of an edge server with a limited cache size, and the other is which quality representation of the chunk should be replaced when the cache is insufficient. Therefore, this paper aims at improving the cache hit rate and maximizing user QoE by studying the edge cache replacement strategy of tile-based 360-degree video encoded with SVC.

Numerous attempts have dedicated to study 360-degree video caching technology. Maniotis et al. [19] considered cooperative caching of tile-based 360-degree video using SVC between one MBS and multiple SBS. Yeh et al. [20]used a real-time image-based generation scheme to synthesize the required views in the 360-degree video, thereby reducing the storage space required for caching the video. Papaioannou et al. [18]proposed a static caching scheme for 360-degree videos. This scheme regards the multi-resolution situation as the K-Medoids problem and the layered coding situation as the multi-choice knapsack problem, in order to seek the optimal solution for 360-degree video caching. For dynamic networks, the computational complexity of finding the best solution is unacceptable. The work of Mahzari et al. [21]considered dynamic caching of 360-degree videos, paying attention to the user's FoV and using maximum likelihood estimation to calculate the probability of requesting video tiles with a certain quality, and then evicting specific quality video tiles with the lowest prediction value from the cache. However, it only considers two quality level's caching probability. Different from the above caching strategy, this paper not only considers the FoV characteristics of 360-degree video, but also explicitly considers the layer information, the size information, and the popularity of the video chunks which encoded with SVC.

In this paper, we propose a cache replacement strategy, namely Size-Popularity-Layer-FoV (SPLF), for tile-based 360-degree video stream using SVC encoding. SPLF calculates a cache value for video chunks depending on the following factors. The first one is SVC layer of the video chunk. Since lower-layer video chunks have higher volume of requests, they should be evicted with a lower probability than the higher-layer ones when the cache capacity is exceeded. The second

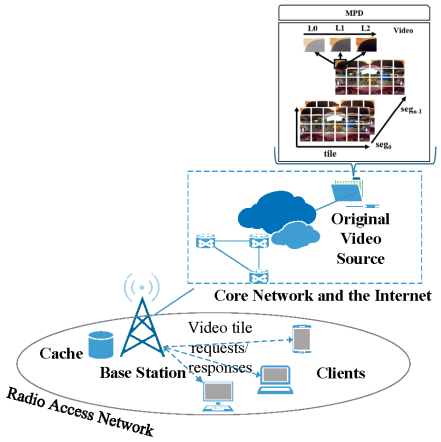

Fig. 1. Architecture of edge caching.

factor is FoV information, if the chunk is in user's FoV, it will have lower priority for evicting. The third factor is the popularity of video chunks. Chunks with higher popularity should be cached with higher probability. Moreover, when the cache is full, it was found that replacing a large content chunk can improve the hit rate more than replacing multiple small ones [22]. therefore, video chunk' size is considered as the fourth cache value factor in SPLF. An aging factor is also retained to measure the freshness of the cached chunks. We evaluate SPLF cache replacement strategy with real users' head movement dataset. Experimental results show that compared with LRU, LFU and GDSF [22], SPLF significantly improves cache hit rate and reduces bandwidth requirements. Furthermore, in terms of the number of rebuffering events, duration of rebuffering events and video playback quality, SPLF considerably outperforms LRU, LFU and GDSF.

II. SYSTEM ARCHITECTURE

A. Network Scenario

As depicted in Fig. 1, we consider a scenario that has a cache-enabled base station (BS), deployed in the edge of clients. The cache capacity of cache-enabled BS is limited. When the cache is full, the existing content in the cache needs to be replaced to make room for the new content. Therefore, cache replacement strategy plays an important role in the practicability of the video transmission system.

When the requested packet initiated by the client arrives at the BS, the caching module of the BS first checks whether the requested content has been cached in the cache. If there is, it is a cache hit. The BS sends back the content immediately; otherwise, it is a cache miss, and the request packet needs to be redirected to the remote Internet CDNs to obtain the content and then sends it back to the client. Therefore, increasing cache hit will not only decrease the download latency experienced by the client but also greatly reduce bandwidth consumption.

From the point of view of client and network, increasing the hit ratio is an important goal for caching systems.

tile_1	tile_2	tile_3	tile_4	tile_5	tile_6
tile_7	tile_8	tile_9	tile_10	tile_11	tile_12
tile_13	tile_14	tile_15	tile_16	tile_17	tile_18
tile_19	tile_20	tile_21	tile_22	tile_23	tile_24

Fig. 2. Spatial Organization of Tiles.

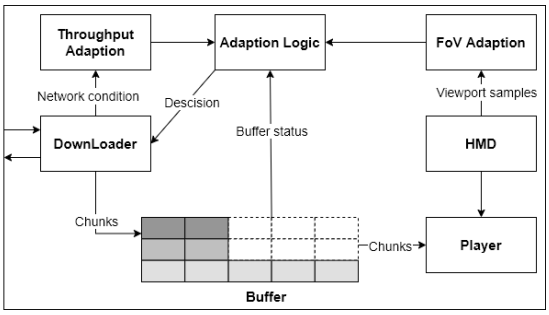

Fig. 3. Client components.

B. 360-degree Video Model

Each video is segmented into N segments in the time dimension, each of duration D seconds(N= video duration / D). Let s_t denote the t^{th} segment, where $t \in \{0, ..., N-1\}$. To simplify the analysis, we assume that the video frame rate is 30 frames per second. We use the most common projection method, equirectangular projection(ERP) [8] [23], to map a 360-degree video frame into a rectangle before encoding. Then the rectangle frame is further divided into M tiles. As illustrated in Fig. 2, we equally divide the sphere into M = 4 × 6 = 24 tiles. Each tile is encoded into L different quality layers based on SVC, and the encoded tile is called chunk. All the chunks along with an MPD file are stored at a server.

Fig. 3 depicts the internal components of a 360-degree video streaming client. A client consists of the following modules: MPD parser, head movement predictor, downloader, player, throughput predictor. Adaptation is performed by the adaptation logic unit. the inputs to this unit are:

- Video information: number of chunks, quality layers and their bitrates, and tiling configuration which are sent to the client at the beginning of a session.
- Viewport: Samples from the user's previous viewports.
- Network throughput: Measured throughput samples from previously downloaded chunks.
- Buffer status: Buffer occupancy in seconds.

The output is the scheduling decision. The scheduler at the client side performs both viewport and quality adaptation according to user's instantaneous head movement and network throughput variations [8].

III. PROPOSED METHOD

A. Cache value of Video Chunks

Chunks are the smallest downloadable units in our system. Each chunk can be represented as $C_{v,k,i,q}$ where v is the video index, k is the segment index, i is the tile number, and $q \in \{0, ..., L-1\}$ is the quality layer. To achieve quality level q for a tile, all co-located tiles from layer 0 to q are required. We assigned a cache value of a video chunk $C_{v,k,i,q}$ at time t , which is the possibility of the chunk being requested again, to each cached each video. The larger the cache value, the more likely the chunk will be future requested. Hence, the video chunk with maximum cache value is less likely to be replaced when the cache capacity is exceeded. The cache value $H(t, C_{v,k,i,q})$ is given by:

$$H(t, C_{v,k,i,q}) = clock + P_{C_{v,k,i,q}}(t) \times \frac{1}{size_{C_{v,k,i,q}}} \times \frac{L-q}{L} \times \frac{1 + isfov_{C_{v,k,i,q}}}{2} \quad (1)$$

- $P_{C_{v,k,i,q}}(t)$ represents the popularity of $C_{v,k,i,q}$ at time t, it is defined as:

$$P_{C_{v,k,i,q}}(t) = \begin{cases} 1, & \text{if } n = 1 \\ 1 + \sum_{j=1}^{n-1} \frac{1}{t - t_j}, & \text{if } n > 1 \end{cases} \quad (2)$$

  Where n is the number of times the chuck $C_{v,k,i,q}$ has been accessed so far, t_j is the j^{th} arrival time of the chunk. Therefore, if the chuck is frequently accessed and recently accessed, it will have higher popularity.
- $Size_{C_{v,k,i,q}}$ represents the byte size of $C_{v,k,i,q}$. We take the reciprocal of it as a factor, which makes larger video chunk probably to be replaced to increase the diversity of cached content.
- L is the SVC levels a video is encoded, and $q \in \{0, ..., L-1\}$ is the quality layer of the chunk. The higher-layer chunk has the high probability to be replaced.
- $isfov_{C_{v,k,i,q}}$ indicates whether the currently requested tile is in the user's viewport or not. If the current tile is in the user's viewport, the value is 1; otherwise, it is 0.
- Clock is a cache aging factor. When the cache space is insufficient to store a new chunk, one or more cached video chunks have to be replaced to make room for the new ones. If only one chunk needs to be replaced, the clock value of the new video chunk is set to the cache value H of the replaced video chunk. Otherwise, it is set to the largest cache value among all of replaced video chunks.

B. SPLF (Size-Popularity-Layer-Fov) cache replacement strategy

An overview of SPLF caching strategy in pseudo code is shown in the algorithm 1. The actions of the cache module takes upon receiving a request for chunk $C_{v,k,i,q}$ is described. At first, function update_popularity is called to calculate the popularity of chunk $C_{v,k,i,q}$. Then, cache_lookup is called to check the cache. If the video chunk has been cached before, its cache value H is updated by cache_updatestore before the chunk is sent it to client; Otherwise, download module is called to fetch the chunk from remote content server firstly, then H value is calculated before the chunk is stored into the cache by cache_store function. Whenever the cache capacity is exceeded, cache_remove_min removes chunks with the lowest H value.

Algorithm 1 SPLF caching replacement algorithm

- $cache$ is a min-heap maintains cached chunks ordered on their H as key.
- $remove\_min()$ is a function removing chunks with lowest H from cache and returning lowest H.
- $update\_popularity()$ is a function that updates popularity of the chunk.
- $cache\_capacity$ is the total amount of space for caching.
- $cache\_used\_size()$ is a function returning the amount of space used so far in cache.
- $is\_in\_FoV()$ is a boolean function that returns whether the requested tile is part of the client's FoV or not, using the flag sent in the client's request.
- $simTime()$ is a function returning the current system running time.
- $chunk\_arriveTime$ is a two-tuple that records each arrival time of the chunk.

function UPON_REQUEST($C_{v,k,i,q}$)
  $P_{C_{v,k,i,q}} \leftarrow update\_popularity(C_{v,k,i,q})$
  if $is\_in\_FoV(C_{v,k,i,q})$ **then**
    $isfov_{C_{v,k,i,q}}$=1
  else
    $isfov_{C_{v,k,i,q}}$=0
  end
  if $cache\_lookup(C_{v,k,i,q})$ **then**
    $\mathrm{H}_{C_{v,k,i,q}} = P_{C_{v,k,i,q}}(t) \times \frac{1}{size_{C_{v,k,i,q}}} \times \frac{L-q}{L} \times \frac{1+isfov_{C_{v,k,i,q}}}{2}$
    $cache\_updatestore(C_{v,k,i,q}$,key=$H_{C_{v,k,i,q}})$
    chunk $\leftarrow cache\_get(\mathrm{C}_{v,k,i,q}$)
    send(chunk)
  else
    chunk $\leftarrow download(\mathrm{C}_{v,k,i,q})$
    clock=0
    while $cache\_used\_size()$+$chunk\_size_{C_{v,k,i,q}}$ >$cache\_capacity$ **do**
      H_{min}=$remove\_min()$
      clock $\leftarrow max$(clock,H_{min})
    end
    $\mathrm{H}_{C_{v,k,i,q}} = clock + P_{C_{v,k,i,q}}(t) \times \frac{1}{size_{C_{v,k,i,q}}} \times \frac{L-q}{L} \times \frac{1+isfov_{C_{v,k,i,q}}}{2}$
    $cache\_store(C_{v,k,i,q}$,key=$H_{C_{v,k,i,q}})$
    send(chunk)
  end
end function
function UPDATE_POPULARITY($C_{v,k,i,q}$)
  t=$simTime()$
  // $C_{v,k,i,q}$ is visited before
  if *chunk_arriveTime.find*($C_{v,k,i,q}$) **then**
    for *each arriveTime* t_j *of* $C_{v,k,i,q}$ **do**
      $P_{C_{v,k,i,q}}$=$P_{C_{v,k,i,q}}$+$\frac{1}{(t-t_j)}$
    end
    $P_{C_{v,k,i,q}}$=$P_{C_{v,k,i,q}}$+1
  else
    $P_{C_{v,k,i,q}}$=1
  end
end function

In order to verify the feasibility and effectiveness of the caching strategy proposed in Section 3, we use ccnSim [24], a simulation module based on OMNet++, to implement the system model described in Fig. 1 and carry out simulation experiments. SVC encoded 360-degree video chunks along with the MPD file are stored at a video server. As shown in Fig. 3, main modules including MPD parser, FoV adaption, downloader, player, throughput adaption are deployed in client.

Five 360-degree videos are used in the simulation, which are Alaska Aurora, Run Live, Diving, Roller Coaster and Google Animation Spotlight Story [25], with the total amount about 5.17GB. They are encoded into three layers using the SHVC reference software [26] with the QP parameters 32-28-24 [8] respectively. The bandwidth between the client and the BS is set to be 15Mbps-50Mbps, and the delay is 1-2ms. The bandwidth between the BS and the video server is set to 1Gbps, and the delay is 10ms [27]. The cache capacity is set to 1%, 3%, and 5% of the total video size.

The head movement trajectories of 150 users are recoded when watching 5 videos with Oculus Rift DK2. In our simulation, 50 FoV traces are selected. They are from 10 random users, each user watches the above five 360-degree videos. Assumed video request sessions that created by clients are based on Poisson distribution with arriving rate of λ=$\frac{1}{30}$, which makes clients to start watching 360-degree video every 30 seconds on average [21].

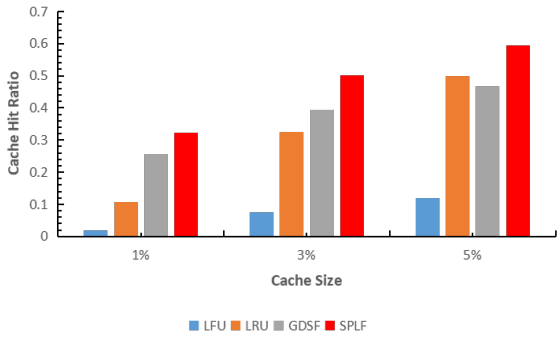

Fig. 4. cache hit ratio vs different cache capacities.

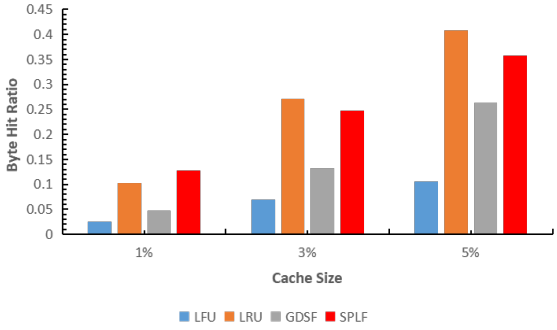

Fig. 5. byte hit ratio vs different cache capacities.

We compare the performance of our proposed SPLF cache replacement strategy with the widely used cache replacement strategies, including least used frequency (LFU) , least recently used (LRU), and GDSF strategy [15]. In each experiment, the LCE caching strategy is used.

The following performance metrics are used for comparison: *Cache Hit Ratio*, *Byte hit ratio*, *Average Access Latency Ratio*, *Bandwidth Saving*, *Average Number of Stalls*, *Average Rebuffering Time* and *Average playback layer*.

We use *Cache Hit Ratio* and *Byte hit ratio* to measure the cache usage in the edge cache(BS). *Cache Hit Ratio* is the ratio between request hit, the number of requests that are satisfied in the BS, to request all, the number of total requests BS received. Similarly, *Byte hit ratio* is the ratio between hit bytes, the number of chunk bytes that are provided in the BS, to all bytes, the number of total chunk bytes that requested.

Fig. 4 shows the *cache hit ratio* for different cache replacement strategies with respect to different cache size. As expected, the hit ratio for all strategies increases as the cache size increases. The hit rate of is SPLF strategy is about 40%, 16% and 10% higher than that of LFU, LRU and GDSF respectively. Especially when the cache space is very small (1%), the cache hit rate is still highest among the compared strategies. Since smaller chunks are the most popular requested ones, and SPLF strategy is SVC and FoV awareness, it gives a higher cache value to the chunk with the lower SVC layer or in user's FoV, which are also have a lager probability to be requested. Therefore, SPLF can remove chunks with least likelihood of being requested in the future and get higher cache hit rate.

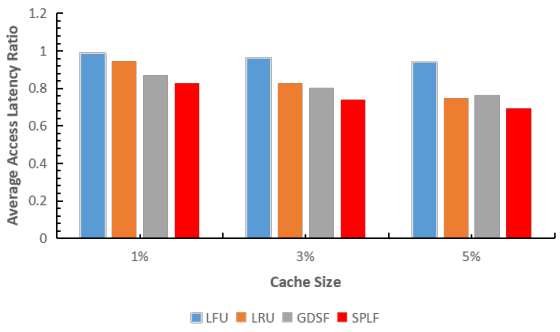

Fig. 6. average access latency ratio vs different cache capacities.

Fig. 5 compares the *byte hit rates* of the four strategies. Since small video chunks have greater cache value in GDSF and SPLF strategies, larger video chunks are likely replaced when the cache is full. As the cache space is very small, SPLF outperforms other strategies. When the cache space is larger, it is not the highest one. However, because the SPLF algorithm takes into account the popularity of video chunks and viewport information, its byte hit rate is about 9% higher than GDSF and about 18% higher than LFU.

We use *Average Access Latency Ratio* to measure how quickly a user can fetch video content while applying different cache replacement strategies. The smaller the *average access latency ratio*, the shorter the distance between the cache and the user, and thus the shorter of request delay. The *Average Access Latency Ratio* is given by :

$$Average\_Access\_Latency\_Ratio = \frac{\sum_{i=1}^{n} hop\_from\_client_i}{\sum_{i=1}^{n} hop\_client\_to\_server_i} \tag{3}$$

Where n is the total amount of data chunks requested by all clients, $hop\_from\_client_i$ is the number of hops from client i to the hit content cache, and $hop\_client\_to\_server_i$ is the number of hops from client i to the content source.

Fig. 6 shows the *average access latency ratio* of the four strategies. It can be seen that the *average access latency ratio* for all strategies decreases as the cache size increases. This is because more video chunks are provided by BS when the cache size is larger. SPLF have the lowest *average access latency ratio*, thus implying the best user experience. This is because that the cache in the edge of network evicts chunks with low cache value, thus stores more popular chunks.

Bandwidth Saving (in percentage) is used to measure network bandwidth consumption. It is the amount of saved bandwidth when client's requests result in cache hit. The larger the *bandwidth saving*, the more video chunks are provided

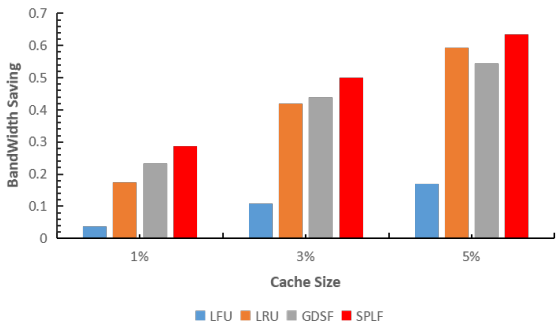

Fig. 7. bandwidth saving vs different cache capacities.

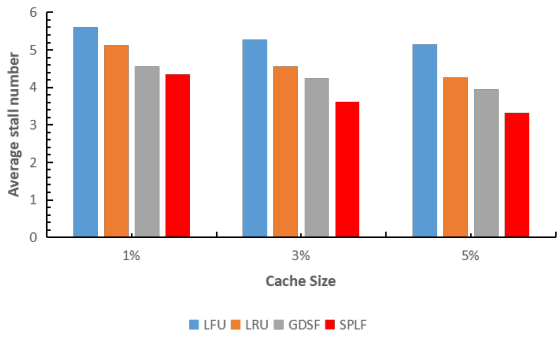

Fig. 8. average stall number vs different cache capacities.

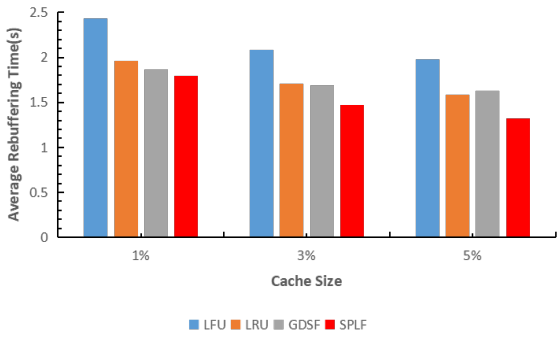

Fig. 9. average rebuffering time vs different cache capacities.

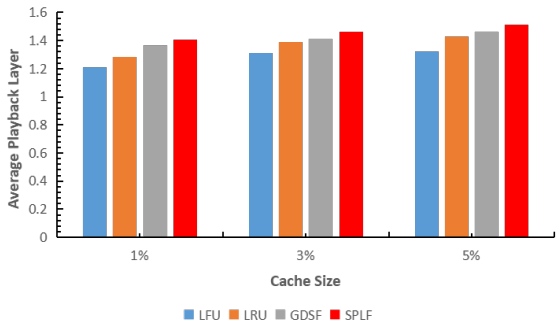

Fig. 10. average playback layer vs different cache capacities.

by the cache, thus saving more network bandwidth. As seen from Fig. 7, the SPLF is obviously superior to other strategies in terms of *bandwidth saving* due to its higher cache hit ratio.

When there is no segment to be played in the buffer of the client, the video playback will stop, This is called rebuffering. Number of Stalls is the number of occurrences of rebuffering and Rebuffering Time is the total duration of time that a client experiences rebufferings. We use $Average\ Number\ of\ Stalls$ and $Average\ Rebuffering\ Time$ to measure streaming performance in the simulation. $Average\ playback\ layer$ is used to reflect playback quality of videos, which is the average video playback layer of all users. The higher the playback layer, the better the user experience. Suppose the viewport at time t is $FoV(t)$, and the tiles covered by $FoV(t)$ is $FoV_{tiles}(t)$. Let $x_{v,k,i,q}$ be a binary decision variable denoting whether chunk $C_{v,k,i,q}$ is chosen for playback. The $Average\ playback\ layer$ is given by:

$$Average\_Playback\_Layer = \frac{\int_{t=0}^{ND} \sum_{i \in FoV_{tiles}(t)} \sum_{q=0}^{L-1} x_{v,\lfloor \frac{t}{D} \rfloor,i,q} dt}{ND} \tag{4}$$

Fig. 8 - Fig. 10 compares the performance of the above four strategies in terms of the $average\ number\ of\ stalls$, $average\ rebuffering\ time$, and $average\ playback\ layer$. These three evaluation indicators are closely related to the user's QoE. Fewer number of playback stalls, shorter rebuffering time and higher average playback layer, will result in client's better experience. It is easy to see from the figures, SPLF strategy outperforms other strategies as expected. The adaptive logic of the client and the calculation method of cache value contribute to this. Clients prefer to download the base layer of the entire spherical view at first. The higher layers are requested according to the current throughput prediction and head motion prediction. As mentioned earlie, SPLF strategy keeps as many low-quality video chunks in FoV as possible, which makes the video stream smoother. Also, after the low-quality layer video chunk is obtained in the cache, the enhancement layer can be pre-fetched in a short time, thereby enhancing the quality of the video in the user's viewport.

IV. CONCLUSION

With the increase of 360-degree video streaming in popularity, it faces the challenges of high bandwidth and low Motion-to-Photon latency. Edge caching can reduce network latency and bandwidth consumption, and make 360-degree video transmission smoother, especially in dynamic network scenario such as mobile networks. In this paper, we propose an edge cache replacement strategy for SVC tile-based 360-degree panoramic streaming named SPLF. Each cached video chunk has a cache value which is calculated based on the size, popularity, FoV information and SVC layer. It can be used to measure the likelihood that a video block will be accessed in the future. Our simulation results show that SPLF outperforms the existing solutions, and can significantly increase user's QoE and reduce network and server resources.

ACKNOWLEDGMENT

This work is partly supported by the National Natural Science Foundation of China under Grant Nos: 61772279.

REFERENCES

[1] Cisco Visual Networking Index 2017–2022 White Paper. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[2] M. Zink, R. Sitaraman, and K. Nahrstedt, "Scalable 360° Video Stream Delivery: Challenges, Solutions, and Opportunities," in Proceedings of the IEEE, vol. 107, pp. 639–650, April 2019.

[3] P. Fuchs, "Virtual Reality Headsets—A Theoretical and Pragmatic Approach," Boca Raton, FL, USA: CRC Press, 2017.

[4] Q. Feng , J. Lusheng, H. Bo , and G. Vijay, "Optimizing 360 video delivery over cellular networks," In Proceedings of the 5th Workshop on AllThings Cellular: Operations, Applications and Challenges. ACM, 2016, pp. 1–6.

[5] C. Zhou, M. Xiao and Y. Liu, "ClusTile: Toward Minimizing Bandwidth in 360-degree Video Streaming," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 962–970.

[6] V. R. Gaddam, M. Riegler, R. Eg, C. Griwodz and P. Halvorsen, "Tiling in Interactive Panoramic Video: Approaches and Evaluation," in IEEE Transactions on Multimedia, vol. 18, pp. 1819-1831, Sept. 2016.

[7] T. C. Nguyen and J. Yun, "Predictive Tile Selection for 360-Degree VR Video Streaming in Bandwidth-Limited Networks," in IEEE Communications Letters, vol. 22, pp. 1858–1861, Sept. 2018.

[8] A. T. Nasrabadi, A. Mahzari, J. D. Beshay and R. Prakash, "Adaptive 360-degree video streaming using layered video coding," 2017 IEEE Virtual Reality (VR), Los Angeles, CA, 2017, pp. 347–348.

[9] F. Duanmu, E. Kurdoglu, S. A. Hosseini, Y. Liu and Y. Wang, "Prioritized Buffer Control in Two-tier 360 Video Streaming," ACM VR/AR Network'17, 2017, pp. 13–18.

[10] L. Xing , X. Qingyang, G. Vijay ,H. Bo, Q.Feng , and V. Matteo, "360° Innovations for Panoramic Video Streaming," ACM Hot Topics in Networks (HotNets'17), 2017, pp. 50–60.

[11] V. Jacobson, DK. Smetters, JD. Thornton, P. Michael, N. Briggs, and R. Braynard, "Networking named content,in Commun," ACM, vol. 107, 2009, pp. 117–124.

[12] H. A. Pedersen and S. Dey, "Enhancing Mobile Video Capacity and Quality Using Rate Adaptation, RAN Caching and Processing," in IEEE/ACM Transactions on Networking, vol. 24, pp. 996-1010, April 2016.

[13] W.Cedric, "Challenges in networking to support augmented reality and virtual reality," In Int. Conf. on Computing, Networking and Communications, 2017, pp. 26–29.

[14] K. Bilal and A. Erbad, "Edge computing for interactive media and video streaming," In 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, 2017, pp. 68–73.

[15] X. Hou, Y. Lu and S. Dey, "Wireless VR/AR with Edge/Cloud Computing," 2017 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, 2017, pp. 1-8.

[16] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. Dias Silva, "VR is on the Edge: How to Deliver 360-degree Videos in Mobile Networks," In Proceedings of the Workshop on Virtual Reality and Augmented Reality Network. ACM, 2017, pp. 30–35.

[17] A. Tagami et al., "Tile-Based Panoramic Live Video Streaming on ICN," 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 2019, pp. 1–6.

[18] G. Papaioannou, and I. Koutsopoulos, "Tile-based Caching Optimization for 360° Videos," In Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2019, pp. 171–180.

[19] P. Maniotis, E. Bourtsoulatze and N. Thomos, "Tile-Based Joint Caching and Delivery of 360°Videos in Heterogeneous Networks," 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), Kuala Lumpur, Malaysia, 2019, pp. 1–6.

[20] M. Yeh, C. Wang, D. Yang and W. Liao, "Efficient Caching for 360°Videos with Dynamic View Selection," 2019 IEEE/CIC International Conference on Communications in China (ICCC), Changchun, China, 2019, pp. 225–230.

[21] A. Mahzari, A. Taghavi Nasrabadi, A. Samiei, and R. Prakash, "FoV-Aware Edge Caching for Adaptive 360° Video Streaming," In Proceedings of the 26th ACM International Conference on Multimedia (MM '18), 2018, pp. 173–181.

[22] L. CHERKASOVA, "Improving WWW proxies performance with Greedy-Dual-Size-Frequency caching policy," Hp Laboratories Technical Report, 1998, pp. 1–14.

[23] S. Liyang, D. Fanyi and L. Yong, "Multi-path Multi-tier 360-degree Video Streaming in 5G Networks," ACM MMSys'18, 2018, pp. 162–173.

[24] V. Jacobson, DK. Smetters, JD. Thornton, MF. Plass, NH. Briggs and RL. Braynard, "Networking named content," In Proceedings of the 5th Int'l Conference on Emerging Networking Experiments and Technologies (CoNEXT 2009), 2009, pp. 1–12.

[25] X. Corbillon, F. De Simone and G. Simon, "360-Degree Video Head Movement Dataset," ACM, 2017, pp. 199–204.

[26] SHVC. HEVC Scalability Extension reference software (2016).https://hevc.hhi.fraunhofer.de/shvc.

[27] C. Ge, N. Wang, G. Foster and M. Wilson, "Toward QoE-Assured 4K Video-on-Demand Delivery Through Mobile Edge Virtualization With Adaptive Prefetching," in IEEE Transactions on Multimedia, vol. 19, pp. 2222–2237, Oct. 2017.