Flocking-based live streaming of 360-degree video

Conference Paper · May 2020

DOI: 10.1145/3339825.3391856

5 authors, including:

Liyang Sun
New York University
7 PUBLICATIONS **40** CITATIONS
SEE PROFILE

Yixiang Mao
New York University
4 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Yao Wang
Dalian University of Technology
249 PUBLICATIONS **5,501** CITATIONS
SEE PROFILE

Flocking-based Live Streaming of 360-degree Video

Liyang Sun, Yixiang Mao, Tongyu Zong, Yong Liu and Yao Wang
New York University
{ls3817,yixiang.mao,tz1178,yongliu,yw523}@nyu.edu

ABSTRACT

Streaming of live 360-degree video allows users to follow a live event from any view point and has already been deployed on some commercial platforms. However, the current systems can only stream the video at relatively low-quality because the entire 360-degree video is delivered to the users under limited bandwidth. In this paper, we propose to use the idea of "flocking" to improve the performance of both prediction of field of view (FoV) and caching on the edge servers for live 360-degree video streaming. By assigning variable playback latencies to all the users in a streaming session, a "streaming flock" is formed and led by low latency users in the front of the flock. We propose a collaborative FoV prediction scheme where the actual FoV information of users in the front of the flock are utilized to predict of users behind them. We further propose a network condition aware flocking strategy to reduce the video freeze and increase the chance for collaborative FoV prediction on all users. Flocking also facilitates caching as video tiles downloaded by the front users can be cached by an edge server to serve the users at the back of the flock, thereby reducing the traffic in the core network. We propose a latency-FoV based caching strategy and investigate the potential gain of applying transcoding on the edge server. We conduct experiments using real-world user FoV traces and WiGig network bandwidth traces to evaluate the gains of the proposed strategies over benchmarks. Our experimental results demonstrate that the proposed streaming system can roughly double the effective video rate, which is the video rate inside a user's actual FoV, compared to the prediction only based on the user's own past FoV trajectory, while reducing video freeze. Furthermore, edge caching can reduce the traffic in the core network by about 80%, which can be increased to 90% with transcoding on edge server.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**.

KEYWORDS

Live 360-degree Video, Collaborative FoV Prediction, Edge Caching

ACM Reference Format:
Liyang Sun, Yixiang Mao, Tongyu Zong, Yong Liu and Yao Wang. 2020. Flocking-based Live Streaming of 360-degree Video. In *11th ACM Multimedia Systems Conference (MMSys'20), June 8–11, 2020, Istanbul, Turkey*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3339825.3391856

MMSys'20, June 8–11, 2020, Istanbul, Turkey

ACM ISBN 978-1-4503-6845-2/20/06…$15.00
https://doi.org/10.1145/3339825.3391856

1 INTRODUCTION

Live streaming of 360° video facilitates immersive view experience by allowing users dynamically choose their view directions in live events and is potential to be popular in many fields, e.g., concert and E-sport. However, the bandwidth requirement of 360° video is much higher than the traditional 2D-planar video. How to deliver high-quality 360° video with short playback latency over the global Internet has become a hot topic for both academia and industry. To address the bandwidth and latency challenges, there are two proven effective solutions: *Field-of-View (FoV) streaming* and *content caching*. Instead of streaming the whole 360° video, FoV streaming only streams a fraction of video within the predicted user FoV. It can significantly reduce the bandwidth requirement of 360° video streaming. However, the user Quality-of-Experience (QoE) of FoV streaming largely hinges on the accuracy of user FoV prediction. Meanwhile, without the global deployment of multicast, caching at different levels of the network infrastructure can reduce the redundant traffic resulting from streaming the same video to multiple users. While caching has been widely employed for video-on-demand, live video streaming can also benefit from caching if users can tolerate some streaming latency: if users A and B in the same local network are watching the same live event, and B can tolerate a longer playback latency than A, then a local network cache can temporally store each video segment downloaded by A and serve it to *B* after a short delay, resulting in faster video download on B, and reduced traffic in the core network between the server and the local network edge. Indeed, recent study has shown that the playback latencies in the commercial live streaming services on Over-the-Top (OTT) devices range from 10 to 30 seconds [26], presenting abundant opportunities for effective content caching for live video streaming.

In this paper, we investigate how the idea of "flocking" can be used to improve the efficiency of live 360° video streaming. We treat all users watching the same live event from the same local network as a "streaming flock". While all the users display the video at the same speed, they can be engineered to have different playback latencies within a short range, determined by their network conditions. The relative position of a user in a streaming flock is therefore determined by her playback latency. We explore the *flocking gain in both FoV prediction accuracy and live video caching*. At a high level, the view directions of users in the front of a flock, i.e., with shorter playback latency, serve as valuable inputs to predict the view directions of users behind them, i.e., with longer playback latency. Leveraging on this, we develop a novel light-weight FoV prediction algorithm that predicts a target user's view direction for a video scene based on her own past FoV trajectory as well as the actual view directions of "similar" users who have watched the same video scene very recently. Similarly, video segments downloaded by users in the front of a flock can be cached to serve users behind them. We develop different tile-based caching strategies

to maximize the caching gain by taking into account the relative flock positions and FoV consistency among users. We further study the bandwidth, computation and storage trade-off when the cache server is equipped with video transcoding capability.

2 RELATED WORK

In recent years, numerous solutions have been proposed to solve 360° video streaming related problems. In the following, we review works closely related to FoV prediction, live streaming, and edge caching.

User FoV prediction is one of the most important techniques for all types of 360° video streaming including video-on-demand (VoD) [9, 17, 29, 33], live 360° [14, 30] and interactive 360° video [5, 40]. Different FoV prediction methods are proposed to address different requirements. In [4], linear regression and deep neural network (DNN) based solutions are proposed to predict user future FoV center using historical FoV trajectory. Instead of only using the past FoV trajectory, video content features are also utilized to predict future FoV in [11]. In [19], the authors focused on FoV prediction over long time horizon, which is important for on-demand streaming with long buffers to smooth the network traffic, and multiple LSTM-based models are proposed. Auto Regressive Moving Average (ARMA) prediction and transition probability model are applied in [12] and [7], respectively. The study in [16] shows user perceptual quality is also affected by the relative moving speed of the objects, depth-of-field and luminance change. In [3, 19], collaborative FoV prediction based on other users' viewing directions are considered. However, these methods are proposed for VoD streaming and assume that there are always a large number of users who have watched the same video. Our proposed collaborative FoV prediction which is specifically designed for live streaming, can work with any variable number of available users and is light-weight.

Live 360° video streaming poses more challenges for both end users and video server than VoD type of 360° video streaming. In [20], the authors proposed a measurement platform to conduct measurement on the existing commercial live 360° streaming platforms, e.g., Facebook and YouTube. QoE metrics including video bitrate, duration and the number of video freezes, and user real-time latency are collected from a large group of viewers from different countries. Besides, [14] proposes a live 360° streaming system which strikes a trade-off between the bandwidth usage and video quality within user's FoV.

In live 360° streaming system, encoding efficiency is challenged by the stringent real-time requirement. As with VoD, tile-based video encoding and delivery is widely used to achieve FoV-adaptive video streaming [1, 42]. Based on this design, different encoding methods can be utilized. In [31], tiles with different resolutions are aggregated into one High Efficiency Video Coding (HEVC) bitstream on-the-fly allowing the usage of only a single decoder on the end device. GPU-based real-time HEVC encoding platform is developed in [2] and evaluated by the measurement framework proposed in [25] under 5G network environment. Layered coding scheme is applied in [27, 35] to reduce the occurrences of video freezes without compromising the quality and bandwidth efficiency. We focus on the system-level design for live streaming, taking advantage of the fact that viewers are often interested in similar regions in the 360° scope. By intentionally assigning varying playback latencies to users based on their network conditions, we can improve the accuracy of collaborative FoV prediction, while reducing the likelihood for video freezing. Standard tile-based coding and streaming are adopted in our system.

Edge caching for video streaming is widely studied in recent works [18, 21, 41]. In [13], two-dimensional caching algorithm which takes both content and network contexts into consideration for 5G networks are proposed. Besides, the trade-off between the gain of coded caching and delivery delay is investigated in [28].

For 360° video, edge caching also plays an important role to solve both network efficiency and video delivery problems. Different from legacy video caching, 360° users move their FoVs among tiles during the playback (considered as non-linear viewing), which can benefit from the caching technique proposed in [15] that is demonstrated to be beneficial for non-linear video content. In [22], FoV-aware caching strategy is proposed, and their results show that the proposed algorithms generate significant improvement compared with the traditional caching. However, they consider users start watching 360° video with large latency gap about 30*s*. Different from this, our work strategically assigns users varying latencies over an acceptable range (0-20 s) and consider both the FoV and latency factors to determine whether to cache a downloaded video tile. A novel HEVC transcoding scheme is proposed in [10]. Authors in [6] study the trade-off between caching and edge computing for immersive video delivery. Caching and computation offloading policy are jointly optimized to reduce the required transmission rate in [34]. Here we consider how to improve caching performance via real-time transcoding, at the expense of computation.

3 CHALLENGES AND OUTLINES OF PROPOSED SOLUTIONS

In this section, we provide an overview of the challenges and the proposed solutions for live 360° video streaming.

3.1 Online User FoV Prediction

A user's view direction for 360° video is affected by both the distribution of the attractive objects in a video scene and her personal preferences to them. FoV prediction for future frames based on the FoV trajectories for the past frames is hard because there could be newly appeared objects of interests that are not predictable from the past. In this challenging scenario, knowing which areas other users (earlier viewers) have focused on for those future frames could greatly help the FoV prediction for the current user (late viewer). Even in the situation when no new objects appear in the future frames, the distribution of the viewing areas of the earlier viewers can still help to predict the FoV of late viewer, especially if the distribution is non-uniform and has one or a few peaks. In [3, 19], FoV prediction based on multiuser trajectories was proposed. With the help of information from other users, the prediction accuracy for a target user can be improved. These studies were based on the user FoV traces from VoD streaming and assumed that an equal number of earlier viewers are always available and is relatively large. However, in live 360° video streaming, the number of earlier viewers for a target user is variable and dynamically changing.

Solution: In this paper, we employ a flocking-based real-time FoV information sharing strategy for live 360° video streaming. With the shared FoV trajectories from users in front of the target user in terms of latency, we develop a light-weight collaborative FoV prediction algorithm that adaptively combines the prediction from the user's own past trajectory and the prediction from the trajectories of users in the front of the flock. Details will be presented in Sec. 4.1

3.2 Temporal and Spatial Rate Adaptation

Similar to 2D-planar video streaming, to cope with dynamic network bandwidth, the streaming rate of live 360° video has to be adapted over time. It can be achieved by partitioning a 360° video into temporal segments with some unit time, e.g., one second, and dynamically changing the coding rate of segments. Additionally, to cope with user view direction changes, rate allocation over different directions within the same segment has to be adapted. In a tile-based design, each 360° video segment is spatially partitioned into multiple tiles in the Equirectangular Projection (ERP) format, and each tile is coded with multiple rates. The rate and consequently the quality chosen for a tile should be determined by its distance to the center of a user's FoV. The tiles within the predicted FoV should be allocated with more bits than the tiles around the boundary or outside of FoV. In addition, the impact of the rate difference between two spatially adjacent tiles on the user perceived video quality should also be considered.

Solution: We use segment+tile based design to achieve temporal and spatial rate adaption. Similar to DASH for 2D-planar video, we can adapt the total coding rate budget for a 360° video segment using buffer-based and/or rate-based algorithms. Each segment consists of multiple frames, and a user's view direction can change at the frame-level. A tile within a user's FoV at one frame may fall out of her FoV at the next frame. Instead of predicting one view direction for each segment, we predict the *tile attention distribution*, i.e., the fraction of time that a specific tile falls into the user's FoV over the whole duration of a segment. Given the predicted attention distribution, we proportionally allocate the total rate budget for a segment over all the tiles. Details will be presented in Sec. 4.3.

3.3 Massive Concurrent Requests and Redundant Network Traffic

One main challenge of live video streaming is bursty requests. Popular live events attract people around the world to join the live streaming service within a short time span. The so-called flash-crowd effect is further amplified by social cascading in online social networks and recommendation engines of video streaming platforms. The workload on the video streaming server would be extremely heavy. In the context of live 360° video streaming, this problem becomes more severe as each user request might be consisted of multiple tiles with different rates. Besides, the huge volume of video traffic imposed on network infrastructure should also be considered. As a large number of users watch the same live event, most of the traffic transmitted over the network would be redundant. So, serving bursty requests and eliminating redundant video traffic are critical to both Internet service providers (ISPs) and content providers.

Solution: Caching is an efficient solution to reduce the redundant network traffic and improve the QoS of user content retrieval. In live 360°, one video tile can be requested multiple times by users in the flock. If the popular tiles can be cached at a local server, it can reduce the workload on the original video server and the traffic in the core network significantly. Different from VoD, live video content only needs to be cached for a short time span corresponding to the maximum playback latency of the flock. The future popularity of a tile watched by a user is determined by how many other users behind her in the flock are likely to watch the tile, and how many of them are likely to request the same rate version. In Sec. 5, we will develop a novel live 360° video tile caching algorithm that takes into account the latency spread and FoV divergence of users in the flock.

4 FLOCKING-BASED LIVE 360° VIDEO STREAMING

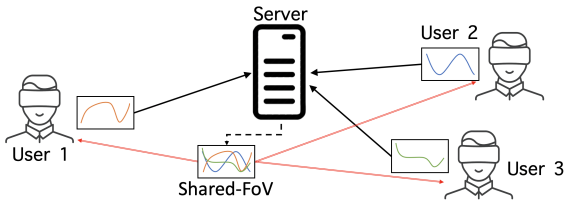


Figure 1: User FoV sharing in flock-based streaming.

In order to cope with the challenges mentioned in Sec. 3, we propose a flocking-based live 360° video streaming system. The detailed workflow is shown in Fig. 1. During the playback, each user will record FoV center direction for each watched frame and upload historical FoV trajectory along with the request for new tiles to video server. The server stores the received users' FoV information into a shared FoV table. A user can download the shared FoV information together with Media Presentation Description (MPD) file. The shared FoV table maintained by the server allows all users in a flock to fully exchange their up-to-date FoV information, which can be used to improve their FoV prediction accuracy.

4.1 Collaborative FoV Prediction

With shared FoV information, a user can conduct *collaborative FoV prediction*. Fig. 2 illustrates the detailed process of predicting future FoV for a target user based on the FoV information of users in front of her in the flock. It consists of the following steps:

① At a specific time, the target user u_t (user 1 in this case) is watching segment $i+1$ (shown by the black vertical line). She also has prefetched segment $i+2$ in the buffer and is going to download segment $i+3$.

② Before sending out the request to the server, u_t first predicts the center of FoV for segment $i+3$, denoted by $c_{t,\overline{i+3}}$ [1], based on

[1] $\overline{i+3}$ represents the frame in the middle of segment $i+3$. So only the FoV center of the frame in the middle of a segment is predicted.

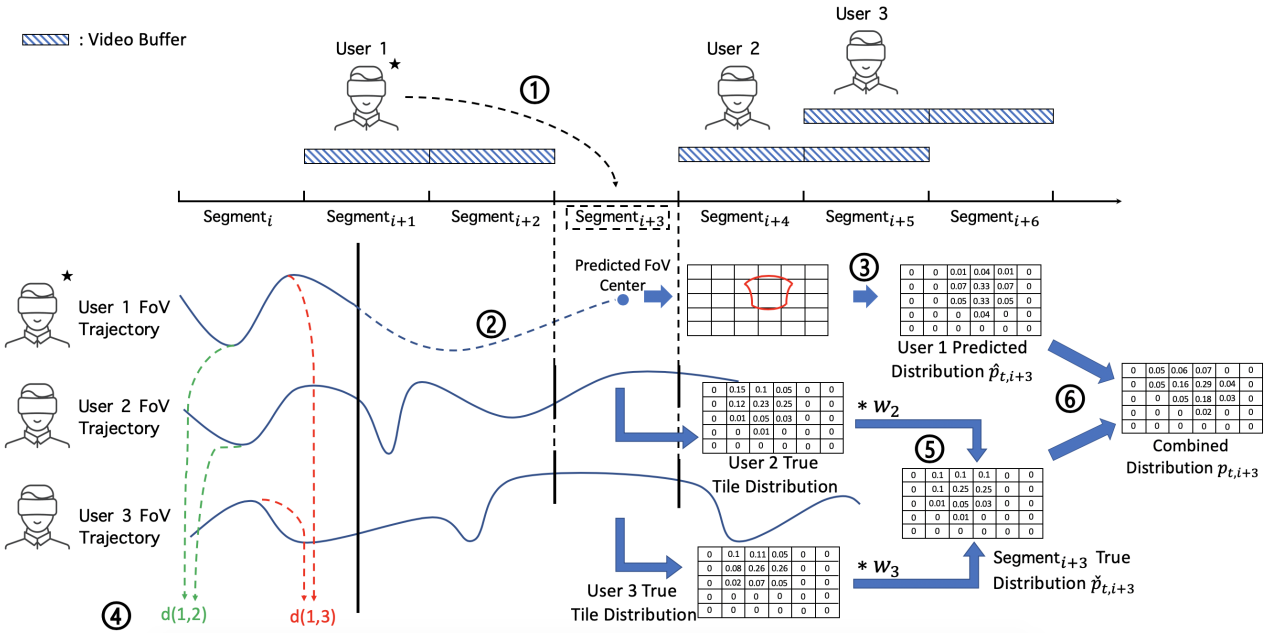


Figure 2: Collaborative FoV prediction: to predict FoV of user 1 for a future video segment $i+3$, instead of only using her past FoV trajectory, FoV information of other users are utilized in a collaborative filtering fashion.

her own frame-level FoV trajectory $\mathcal{T}_t$ for all frames she has viewed up to the first few frames of segment $i+1$. To filter the noise in FoV measurement, we use Kalman Filter [38] to process the data before inputting the data to FoV prediction module. Different time-series prediction methods can be used to make FoV prediction, e.g., linear regression, recursive least squares (RLS), and Long Short-Term Memory (LSTM). In this paper, we adopt truncated linear FoV prediction proposed in [32].

③ Based on the predicted FoV center $c_{t,\overline{i+3}}$, we can project the FoV on the sphere to the ERP plane, and further obtain the tile coverage (the area of the red envelop in Fig. 2) through the offline calculated mapping $\mathcal{M}(\cdot)$ based on the geometry mapping from the sphere to ERP [8]. The value for each tile is the percentage of pixels in this tile that falls into the mapped predicted FoV on the ERP. For example, if the tile is completely inside the mapped FoV, the value is 1. Then, the coverage map $\mathcal{M}(c_{t,\overline{i+3}})$ is normalized into the tile attention distribution $\hat{p}_{t,i+3}$. In this attention distribution, the value of each tile represents how much attention a tile can acquire from the user within the playback of the video segment. In our current implementation, the tile attention distribution for a segment is determined by the predicted FoV center of the middle frame of the segment. More generally, a user can also predict the FoV centers of all the frames in a segment and average the tile attention distribution for all the frames to derive the attention distribution for the segment. Alternatively, a user may directly predict this segment-level attention distribution from the past trajectory. These alternatives will be explored in the future.

④ So far, the prediction is only based on the past trajectory of the target user. To improve the prediction, the system further looks for users who have watched segment $i+3$, and uses a weighted average of their tile attention distributions to generate another prediction:

$$\check{p}_{t,i+3} = \sum_{j \in S_t} \beta_j p_{j,i+3}, \tag{1}$$

where S_t is the set of users who have already watched the video segment to be watched by the target user u_t. β_j and $p_{j,i+3}$ are the normalized weight of user j and user j's true attention distribution of segment $i+3$, respectively. We illustrate the calculation of β_j and $p_{j,i+3}$ using the following example.

In this example, user 2 and user 3 are watching segment $i+4$ and $i+5$, respectively. Their FoV historical trajectories of all the previous segments (including segment $i+3$) have already been uploaded to the server. We assume a *collaborative filtering* based hypothesis: a user with similar trajectory to the target user in the past is more likely to have similar viewing areas in the future, therefore should be given a larger weight in Eq. (1). To determine the weight, user u_t will calculate the distance between her own past FoV trajectory with each of other users' trajectories in the same past period (on the left side of the black vertical line). The distance between FoV trajectories is defined as the average great-circle sphere distance [36] of all the time-aligned FoV center pairs. We will use $d(j,t)$ to denote the distance between user j and the target user t. For example, as illustrated in Fig. 2, if distance $d(1,2)$ is less than $d(1,3)$, user 2 should be assigned a larger weight. We choose to use the reversed sigmoidal function defined by Eq. (2) to map the distance to the unnormalized weight:

$$w_j = \mathcal{H}(d(j,t)) = \frac{e^{-\gamma(d(j,t)-\phi)}}{1+e^{-\gamma(d(j,t)-\phi)}}, \tag{2}$$

where γ and ϕ are parameters to be optimized as explained in Sec. 4.1.1. In the special case when γ is very large, it will have a

thresholding effect so that when the distance is greater than ϕ, the weight is 0. The weights are finally normalized as below:

$$\beta_j = \frac{w_j}{\sum_{j \in S_t} w_j}. \quad (3)$$

⑤ Since user j has watched segment $i+3$, we can calculate her actual segment-level tile attention distribution $p_{j,i+3}$ based on its frame-level FoV trajectories within this segment. We follow the same process in step ③ to determine the frame-level attention distribution $\mathcal{M}(c_{j,f})$ for each frame f within segment i+3, and then average these frame-level distributions to obtain the segment-level attention distribution:

$$p_{j,i+3} = \frac{\sum_{f \in F_{i+3}} \mathcal{M}(c_{j,f})}{|F_{i+3}|}, \quad (4)$$

where F_{i+3} is the set of all the frames of segment $i+3$.

⑥ To combine the attention distribution $\hat{p}_{t,i+3}$ predicted from user u_t's own trajectory and the attention distribution $\check{p}_{t,i+3}$ from collaborative prediction, we propose to use a simple weighted average:

$$\tilde{p}_{t,i+3} = \alpha \hat{p}_{t,i+3} + (1-\alpha)\check{p}_{t,i+3}. \quad (5)$$

We hypothesize that the prediction based on others' attention distribution $p_{j,i+3}$ is more accurate when more users have viewed segment $i+3$ and their past trajectories are more similar to the target user. Therefore, we design the weight α as:

$$\alpha = \frac{1}{1 + \sum_{j \in S_t} w_j} \quad (6)$$

If there is no user in S_t, the target user will only use her own FoV prediction as the value of α in Eq. (5) is 1.

4.1.1 Parameters Tuning. Based on the definition of Eq. (3) and (6), the contribution of self-prediction $\hat{p}_{t,i}$ and collaborative prediction $\check{p}_{t,i}$ is controlled by the parameters γ and ϕ in the distance-to-weight function in Eq. (2). We perform offline parameter tuning through an exhaustive grid search to find the optimal setting. As the distance $d(j,t)$ between curves is defined by the sphere distance (in radian) and ranges from 0 to π, the value of γ and ϕ are chosen from set $\{1,2,3,4,5\}$ and $\{0.5,1,1.5,2,2.5\}$, respectively. We leverage the FoV traces from the dataset in [39] to perform offline optimization. The dataset has 48 user traces for each video. We randomly pick one user t as the target user and one video segment i for prediction and assume all other users have watched this segment. Then we use the proposed method to obtain the collaborative prediction. We use the KL divergence between the predicted attention distribution and the actual distribution to evaluate the accuracy of the prediction. For each candidate pair of γ and ϕ, we perform massive random selections of users and segments and determine the average KL divergence. The grid search shows that the combination of $\gamma = 3$ and $\phi = 0.5$ generates the lowest tile attention distribution prediction error. For all the following experiments, the value of γ and ϕ keep fixed at 3 and 0.5 respectively. The detailed evaluation results will be presented in Sec. 6.

4.1.2 System Overhead Analysis. To perform real-time collaborative FoV prediction within a flock, users have to upload their FoV trajectories. Assuming FoV sampling frequency is 30Hz, for each segment with duration of one second, 30 FoV data points (3D or 4D) will be uploaded. Users also have to download the shared FoV table of all the early users from the server. The amount of FoV data to be downloaded for each user depends on her location in the flock. The earlier, the less. Assuming there are 100 users in one flock, at most 100*30 FoV data points of the past trajectory and 100 tile distributions of the future segment should be downloaded. Both the upload and download of FoV information can be piggybacked with the video segment request and video segment, respectively, without introducing extra latency. Compared with the video content data at rate of tens or hundreds of Mbps, this overhead can be negligible. Furthermore, we will evaluate the overhead reduction and the performance impact if the target user only uses the average tile distribution of all the earlier users without distance weights.

4.2 Network Aware Latency and Buffer Upper Bound Assignment

When birds fly in a flock, the birds in the front have to fight harder against headwind. It is therefore wise to have stronger birds lead a flock. We will follow a similar strategy to place "strong" users with better network conditions in the front of a streaming flock. In this section, we will elaborate how this can be naturally realized by manipulating the target playback latency and maximum buffer length on all users.

At any given time t, if a user's target latency is l, and the maximum buffer length is $B^{(u)} \le l$, the user should be watching video generated at time $t-l$ and downloading video generated within $[t-l, t-l+B^{(u)}]$. l and $B^{(u)}$ are two critical parameters for live 360° video streaming. Before downloading segment $t-l+B^{(u)}$, the user should first estimate its FoV based on her FoV trajectory up to $t-l$. Therefore $B^{(u)}$ determines the FoV prediction temporal horizon, and the larger the $B^{(u)}$, the less accurate the trajectory-based self-prediction. On the other hand, for collaborative FoV prediction, the user can leverage FoV information of users with shorter playback latency who have watched segment $t-l+B^{(u)}$. Therefore, the larger the l, the more potential for collaborative FoV prediction. Meanwhile, streaming buffer is important to absorb network bandwidth oscillation, a larger $B^{(u)}$ is beneficial to achieve high quality.

In our proposed streaming flock, users at the front must have short playback latency, that means they have to assume small l and $B^{(u)}$. The immediate requirement is that they must have high bandwidth and stable network condition so that they don't run into video freeze or segment skip even with short streaming buffer. Additionally, since they have no/low chance to benefit from collaborative FoV prediction, they may have to download more tiles outside of the predicted FoV to accommodate FoV prediction errors. The good news for them is that since $B^{(u)}$ is small, the FoV prediction horizon is short, so that the trajectory-based FoV prediction is generally more accurate. Meanwhile, for users with unstable network conditions, to maintain smooth streaming, a large $B^{(u)}$ is necessary. This naturally pushes them to the back of the flock. The negative impact of long FoV prediction horizon resulted from large $B^{(u)}$ can be compensated by collaborative FoV prediction based on FoV information of users in the front. This cooperative flocking strategy can improve both the individual and overall user QoE.

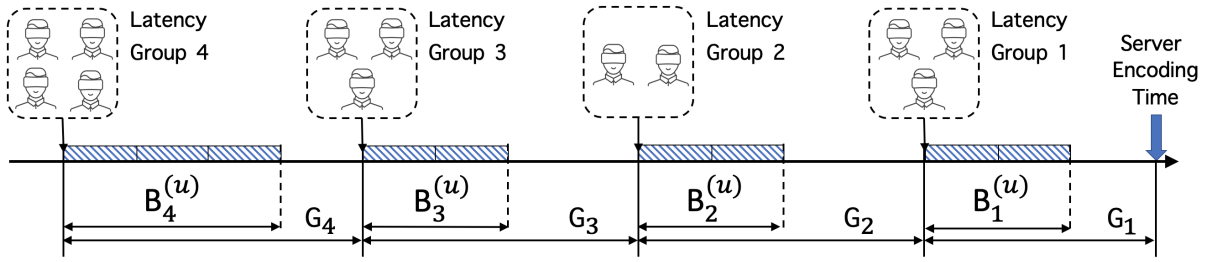


Figure 3: Latency and buffer upper bound for different groups.

In order to enable such network-aware assignment, before requesting for the first video segment, a user operates a short-term monitoring on her current network condition. To illustrate, in Fig. 3, users are divided into four groups with target latencies $l_1 = G_1$, $l_2 = G_1 + G_2$, $l_3 = G_1 + G_2 + G_3$, and $l_4 = G_1 + ... + G_4$. Since this is live streaming, the buffer upper bound of Group 1 $B_1^{(u)}$ should be less than G_1 [2]. Then, for latency Group 2, in order to benefit from Group 1 users' FoV information, the buffer length of the users in Group 2 should not exceed where users in Group 1 are watching. In another word, the buffer upper bound of Group 2, $B_2^{(u)}$ should not be greater than G_2. More generally, to enable Group k users to benefit from all the previous groups, its buffer upper bound should satisfy $B_k^{(u)} < G_k$.

4.3 Spatial and Temporal Rate Adaption

4.3.1 Bandwidth Prediction. Due to the dynamic network environment, adaptive rate control becomes crucial to video streaming especially for live video streaming with short buffer. Bandwidth prediction is one of the most important parts in adaptive rate control. In our system, bandwidth prediction is an independent component, and any effective bandwidth prediction algorithm can be applied. In the simulation results shown later, the harmonic mean of the download bandwidth of the past 10 video segments is calculated as the predicted bandwidth for the next segment.

4.3.2 Temporal Rate Adaption. Instead of video rate, user QoE is also affected by the rate fluctuation between two adjacent video segments. In order to solve this problem, temporal rate adaption should be optimized. For example, model predictive control (MPC) [24] can be applied to get the optimal temporal rate allocation for a given sequence of predicted bandwidth. In addition, model-free based solution, e.g., reinforcement learning (RL) [23], can also solve this problem efficiently through exploring the optimal rate allocation in the environment. In our current work, we operate temporal rate allocation based on one-step bandwidth prediction and will explore potential gain of different temporal rate adaption algorithms in future work.

4.3.3 Spatial Rate Adaption. As each 360° frame is spatially divided into multiple tiles, with a limited bandwidth budget, how much bandwidth is allocated to each tile might lead to different perceptual qualities. Therefore, given a predicted bandwidth, rate should be efficiently utilized on each tile. In our current work, we allocate the total rate among the tiles proportional to the predicted tile attention distribution. Based on the study in [33], the quality-rate model of tiles differs with tile position. For example, to obtain the same quality, the tiles near the equator and south pole require more bits than the tiles near the north pole. Therefore, the quality can be further improved by optimizing the rate allocation based on position-dependent quality-rate-models. This is a subject of future study.

[2]If the user finishes downloading the latest video segment and the following video segment is still under encoding or transcoding by the server, the user has to wait for the server processing to be completed

5 FLOCKING-BASED LIVE 360° VIDEO CACHING

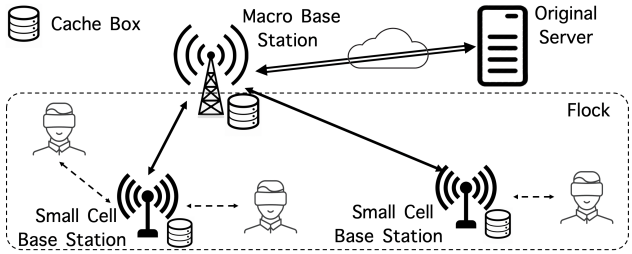


Figure 4: Edge caching for live 360° video streaming.

In live 360° video streaming, the workload of the core network can be heavy due to both the bursty video requests and the high rates of 360° video. One efficient solution to reduce the core network traffic is to cache the popular content on edge servers which are located close to the end users. In a 5G network, a hierarchy of cache servers can be deployed. For example, shown as Fig. 4, several small cell base stations (SBSs) are associated to one macro base station (MBS). Cache boxes can be collocated to the SBSs as well as the MBS. A video chunk request will be served in a hierarchical fashion, from SBS, MBS, finally back to the original server. All the users served by the same SBS/MBS cache can make up a flock and their FoV information can be shared. In this section, in addition to studying the benefit of naive caching, transcoding is also discussed as an important video technique which can further reduce the traffic in the core network.

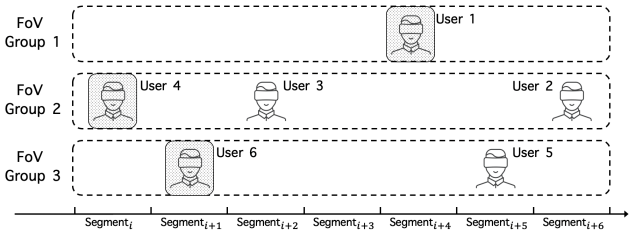


Figure 5: Latency-FoV based caching strategy. The users are divided into different FoV groups based on their past FOV trajectory similarity. A user with the largest latency in each group is marked (user 4 and 6), and so is a lone user in a group (user 1).

5.1 Latency-FoV based Caching Strategy

In real networks, due to the cost or space limit, cache can only work under some storage limits, and hence, it is important to utilize the limited caching space efficiently. Least Recently Used (LRU) is one simple and efficient method to decide if a video content should be cached or not. However, the pattern of the content data varies according to the context. In the case of live 360° video streaming, there are two unique properties. Firstly, even though users watch the live streaming with different latencies, the gap between the users with shortest and longest latency is not large. Normally, it ranges from several seconds to tens of seconds. That is to say, one video tile should be stored in the cache at most tens of seconds. Secondly, if there is an attraction point in the video, users are likely to request tiles around the point but not tiles far away from it. Hence tile popularity can be inferred by the user watching behavior.

Based on these two observations, we propose a latency-FoV (LF) based live 360° caching strategy. Illustrated by Fig. 5, at a particular time, users are positioned horizontally according to their latency. For example, user 2 is watching the segment $(i+6)$ with shortest latency. The users are also implicitly divided into different FoV groups based on their similarity in the FoV trajectories. More specifically, for each user, the FoV trajectory distance between herself and each of the other users is calculated at some frequency, e.g. every 5 seconds. If the distance is greater than a threshold, the two users are not considered as neighbors. In the proposed caching algorithm, users with long latency and few neighbors sharing the similar watching behaviors will be marked and the tiles requested by them will be ignored by the cache. For example, user 4 and user 6 are the two users with the longest latency in their respective groups, no other users request and download video tiles after them. So, these two users are marked (shaded in Fig. 5). In addition, as user 1 is clustered into FoV Group 1 and no other users are in the same FoV Group. Then user 1 is also marked. The tiles requested by the marked users are ignored and all the other tiles are cached following the LRU caching rule.

5.2 Transcoding and Enhanced Transcoding at Edge Server

Normally, a cache manager decides whether to store a content when the data passes through it. However, the capability of an edge server in the current network is not limited to storage only. In this section, we assume the edge server supports real-time video transcoding. For example, if a 360° video tile is requested and there is a corresponding tile with a higher rate stored in the cache, the edge server can operate real-time transcoding from the high rate to the requested rate. Through this way, the traffic transmitted over the core network can be saved.

Nevertheless, the method mentioned above is affected by the requested rate of users with shorter latency. In some cases, even though the same tile has been requested, if the rate is lower than the one being requested, the edge server cannot transcode from low rate to high rate. In order to solve this problem, we further propose an enhanced transcoding (E-Transcoding) strategy. In this method, as soon as a 360° video tile is requested, the highest rate for this tile is downloaded from the original server and cached in the edge. Then the edge server will transcode it to any requested lower rate. This way, each requested tile will be downloaded only once from the original server. To go even further, assuming abundant cache storage and abundant bandwidth between video server and edge cache, the video server can deliver all the tiles with the highest rate to the edge as long as one video segment is ready. In this case, all the requested tiles will hit the cache. And we call this ultimate transcoding (U-Transcoding).

6 PERFORMANCE EVALUATION

6.1 Experiment Setting

In order to demonstrate the efficiency of the proposed streaming scheme and caching algorithm, we perform trace-driven simulations. The user FoV traces are from the dataset offered by [39]. In this dataset, for each of the 9 videos for different live events, FoV trajectories of 48 users are provided. In the following experiments, we assume all the 48 users are served by the same MBS cache, but they may connect to different SBSs (WiGig base stations) under the same MBS. The duration of each video ranges from 170s to 520s and the spatial resolution ranges from 1920×960 to 2880×1440. The FoV traces of a video with duration of 360s are utilized in our experiments. User network condition is emulated using the WiGig bandwidth traces from [32]. The provided bandwidth traces are collected under simulated environments with different levels of interference, which leads to different bandwidth mean and variance. In addition to the original traces from [32], additional synthetic bandwidth traces are generated using Hidden Markov Model (HMM) models which are obtained by fitting to the original dataset. The average bandwidth of the synthetic traces ranges from 550 Mbps to 720 Mbps with the standard deviation ranging from 83 Mbps to 300 Mbps.

To match with the dynamic range of the bandwidth traces, we assume that the video server codes each video at multiple rates of $\{100, 500, 1000, 1500, 2000, 2500\}$ Mbps (for the entire 360° video). We further assume each video in the ERP format is divided into 6×5 tiles and coded independently, with an average rate that is 1/30 of the total video rate. Note that although the videos in [39] have a low resolution and hence do not require such high rates, much higher resolutions (e.g. $24K \times 12K$ to match the retina resolution) are required for high-quality rendering of 360° video, for which the assumed rate range is appropriate. Finally, we assume that

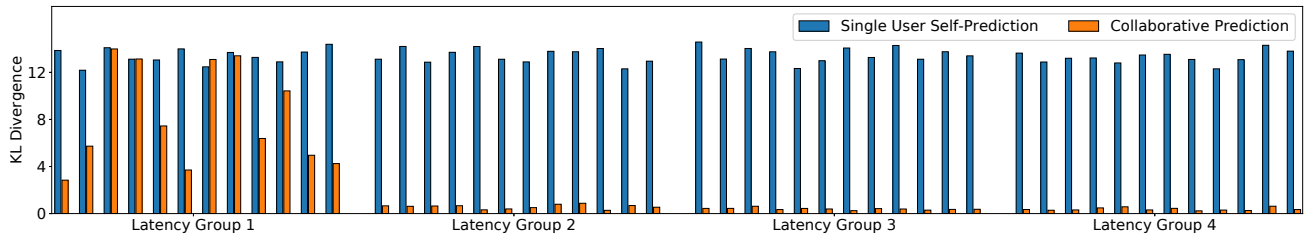


Figure 6: KL Divergence between predicted tile attention distribution and ground truth distribution of all users in different latency groups. Latencies and buffer upper bounds of group 1 to 4 are {3s, 2s}, {8s, 3s}, {13s, 3s} and {19s, 3s}, respectively.

users' viewing directions are mostly driven by the video content, but not the resolution of the video, and hence the FoV traces for the low-resolution videos in [39] can be used as the FoV traces for their high-resolution versions. To predict user FoV or calculate user watching behavior similarity, FoV information of frames in the past 1 to 2 seconds (including the previous segment of 1 second and the frames that have just been watched in the current segment) are utilized. For the trajectory-based self-prediction, truncated linear prediction [19, 32] is employed, because this simple method was found to perform as well as more complicated LSTM-based methods when the prediction horizon is short (1-2 seconds). Future work will explore the use of more accurate prediction methods.

6.2 Evaluation of Flocking-based Live 360° Video Streaming

6.2.1 QoE Metrics. In order to evaluate FoV prediction accuracy, we calculate KL Divergence [37] between the predicted attention distribution and the true attention distribution. In terms of video rate, the delivered rate is defined by the total rate of all the requested and downloaded tiles by a user. Then, we define the effective rate as the total rate watched by the user in the actual FoV of the user in each frame. In addition, the duration of video freeze is also recorded.

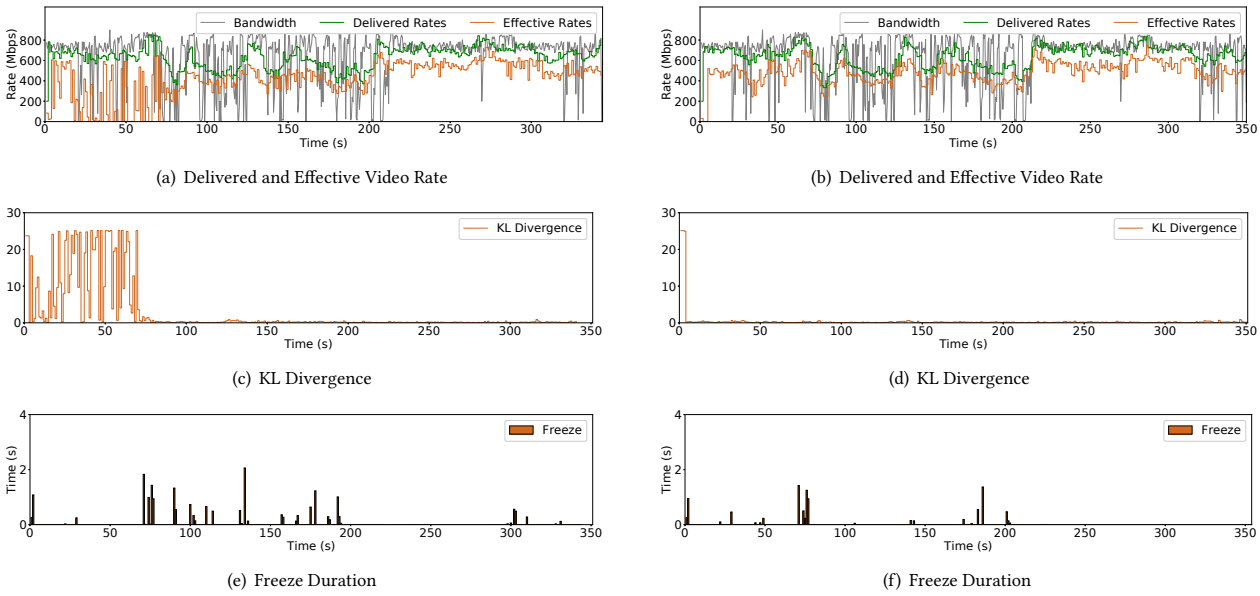
6.2.2 Gain from Collaborative FoV Prediction. First, we evaluate the flocking-based FoV prediction using the user FoV and WiGig network bandwidth traces discussed in Sec. 6.1. There are 48 users and each of them is assigned a unique FoV trajectory and a bandwidth trace. For this experiment, the latency assignment is not based on the network conditions as discussed in Sec. 4.2, rather users are assigned to the 4 latency groups randomly with 12 users in each group. Same assignment is used for the evaluation of different FoV prediction methods. We choose to have 4 latency groups, with latency of 3s, 8s, 13s and 19s, respectively. We set the actual initial latency of each user to be slightly different from the group average latency by adding a random noise. The buffer upper bounds for different groups from 1 to 4 are set as: 2s, 3s, 3s and 3s respectively, according to Fig. 3.

Fig. 6 illustrates the average KL Divergence of the tile attention distribution for all the video segments between the predicted distribution and the actual distribution. Single user self-prediction means that only the trajectory-based self-prediction in Eq. (5) is used. For this method, the average KL Divergence for all users are roughly the same, as expected. However, for multiuser collaborative prediction, the KL Divergences of some of the users within Group 1 are much smaller than the ones of single user self-prediction (hence more accurate prediction), and the improvement brought by collaborative prediction increases as the group latency increases. The results confirm our hypothesis that the users with long latency can benefit from users with shorter latency regarding the FoV prediction as they can gather information about which tiles in the segment to be downloaded are more popular at the time of prediction. As Group 2 (with 8s latency) can achieve a similar performance to the later groups, it implies that our system can perform well with a latency gap of 10s. While comparing the KL Divergence for each user, we find that collaborative prediction is always better than single user self-prediction, even for the users with short latency, which suggests that it is always beneficial to leverage other users' attention distributions to help predict your own.

6.2.3 Gain from Latency and Buffer Upper Bound Assignment. In this experiment, we compare random user group assignment with the proposed network aware assignment described in Sec. 4.2. Same as the first experiment, we still use 4 latency groups. More specifically, the relative standard deviation (RSD) of user's bandwidth is calculated. Through comparing it with the predefined RSD thresholds, a user's network condition is classified so that the user can be assigned to a latency group accordingly. In order to make a fair comparison, the bandwidth traces are pre-selected to guarantee that, even with network aware latency group assignment, the number of users assigned into each latency group is the same [3].

First, we compare the performance achieved for one user by the two different schemes. With random assignment, a sample user u_t is assigned to Group 1 with short initial latency of 3s and buffer upper bound of 2s; However, based on her network condition, user u_t should be assigned into Group 3 with 13s initial latency and 5s buffer upper bound. Fig. 7 compares these two assignment strategies in terms of QoE metrics including delivered video rate, effective video rate, FoV prediction KL Divergence and video freeze. The comparison between Fig. 7(a) and 7(b) show that the delivered video rates (the green curves) are roughly the same for both cases and close to the available bandwidth (the gray curves). However, in terms of the effective video rate, which is affected by FoV prediction accuracy, which in turn depends on the latency and buffer setting, the performance with random latency assignment is much worse. Especially for the period of time before 70s, due to the lack

[3] For all the experiments, the same set of selected FoV and bandwidth traces are utilized.



(a) Delivered and Effective Video Rate

(b) Delivered and Effective Video Rate

(c) KL Divergence

(d) KL Divergence

(e) Freeze Duration

(f) Freeze Duration

Figure 7: QoE metrics comparison of a sample user between the case when she is randomly assigned into Group 1 ($l_1 = 3s$, $B_1^{(u)} = 2s$) without considering her network condition (left) and when she is assigned into Group 3 ($l_3 = 13s$, $B_3^{(u)} = 5s$) based on her network condition (right).

Table 1: Comparison among three systems differing in FoV prediction and latency group assignment. Note that users in each group are the same among the first two schemes and differ from the last scheme.

Latency Groups		Self-Prediction	Collab-Prediction	Flocking Strategy
Group 1 (3s)	$B^{(u)}$	2	2	2
	Freeze	10.17	10.83	**4.74**
	Effective Rate	216.02	**303.06**	250.68
	KL	13.4	**8.28**	10.85
Group 2 (8s)	$B^{(u)}$	3	3	4
	Freeze	11.10	13.74	**9.6**
	Effective Rate	206.85	424.02	**450.29**
	KL	13.41	0.58	**0.5**
Group 3 (13s)	$B^{(u)}$	3	3	5
	Freeze	15.62	17.20	**12.16**
	Effective Rate	200.0	449.43	**458.25**
	KL	13.56	0.4	**0.39**
Group 4 (19s)	$B^{(u)}$	3	3	6
	Freeze	**13.37**	14.8	14.0
	Effective Rate	211.49	**469.71**	441.08
	KL	13.28	0.38	**0.34**
Overall	Freeze	12.56	14.14	**10.13**
	Delivered Rate	628.95	643.79	**645.4**
	Effective Rate	208.59	**411.56**	400.08
	KL	13.41	**2.41**	3.02

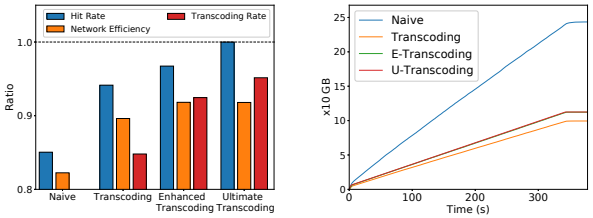
of other users' information, effective video rate is negatively affected by the inaccurate FoV prediction. The KL Divergence curve in Fig. 7(c) further confirms this point. After several video freezes, shown in Fig. 7(e), the latency of user u_t increases, which allows her to obtain more information about other users' FoV distribution and make more accurate FoV prediction after 70s. On the contrary, with the proposed latency assignment, this user can benefit from other users' FoV information to improve her FoV prediction accuracy throughout the entire video duration, as shown in Fig 7(d). Comparing Fig. 7(e) and 7(f), we find that video freeze is avoided in most cases if the buffer upper bound is adjusted to 5s from 2s.

Besides individual user evaluation, we also perform group-level evaluation. In Table 1, the average QoE metrics of each latency group are compared among three schemes with different FoV prediction and flocking strategies. The "self-prediction" and "collaborative prediction" systems both use random latency assignment but different FoV prediction methods, while "flocking strategy" uses collaborative FoV prediction and network aware latency assignment. The average effective rate of each group and the overall average further demonstrate that collaborative FoV prediction improve the FoV prediction accuracy and effective rate dramatically. Furthermore, while comparing collaborative prediction (with random latency and buffer upper bound assignment) and flocking strategy (with network aware assignment), we find that freeze can be reduced with negligible influence on the effective rate and KL Divergence. Note that with network aware assignment, users in the groups with long latency are assigned a longer buffer upper bound and hence will generally have longer FoV prediction horizon and slightly reduced FoV prediction accuracy. For example, the overall average freeze is reduced from 14.14s to 10.13s with effective rate reducing from

411.56 Mbps to 400.08 Mbps. So, we can draw the conclusion that both the individual and overall user experience can be improved with appropriate latency and buffer upper bound assignment.

6.3 Caching Performance for Live 360° Video Streaming

As discussed in Sec. 5.1, latency-FoV (LF) based caching strategy is proposed to improve the caching efficiency and in Sec. 5.2, several transcoding techniques are proposed to reduce the traffic delivered in the core network. In this section, we investigate the caching gain in terms of bandwidth saved in the core network and the cache hit rate of the proposed caching solutions. Recall that the proposed LF-based caching strategy (Sec. 5.1) marks users with the longest latencies or the least number of users with similar FoV trajectories, and ignores contents downloaded by these users. In our experiment, we assign the users into 4 latency groups based on their network condition as discussed in Sec. 6.2.3, and users are marked every 5s based on their real-time latency and past FoV trajectories.

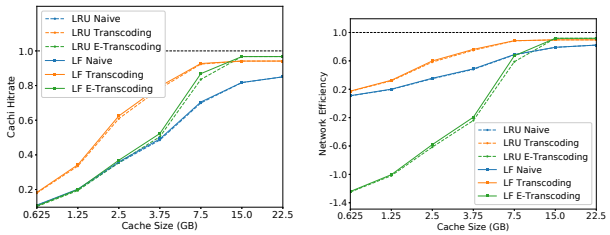


(a) Cache Performance under Different Strategies with LRU Applied (b) Cache Size (Assuming tiles are cached for the entire streaming session)

Figure 8: Hit rate, network efficiency, transcoding ratio and storage consumption comparison among different edge functionalities assuming the cache size is unlimited.

6.3.1 Caching Performance without Storage Capacity Limit. We first evaluate the cache performance assuming there is unlimited cache storage at the edge server. We also assume all the 48 users are in the same local network. The experiment setting remains the same as Sec. 6.2.3 in which the latency and buffer upper bound of all the 48 users are assigned based on their network conditions. We evaluate cache performance by the cache hit rate, network efficiency, transcoding ratio and the storage requirement. Network efficiency is defined as $\frac{V_r - V_d}{V_r}$ where V_r and V_d are the total volumes of traffic requested by the users and that delivered from the original server to the cache, respectively. It quantifies the reduction percentage of the core network traffic load by caching locally. Transcoding ratio is the ratio between the number of transcoded tiles and the total number of requested tiles. High transcoding rate is associated with higher computation cost.

As shown in Fig. 8(a), if the edge only operates naive caching, about 85% of all the requested tiles are cached by the edge server and 82% of the traffic of core network is saved. Furthermore, if transcoding is enabled on the edge server, the cache hit rate is increased to 95%. As transcoding can only be operated from a high-rate video version to a low rate version, the network efficiency, not as high as hit rate, reaches about 90%. With the enhanced transcoding, both the hit rate and network efficiency can be improved slightly, reaching 96.5% and 92%, respectively. For ultimate transcoding, the hit rate is 100% as all the requested tiles hit the cache. And the network efficiency is roughly the same as the enhanced transcoding. The transcoding ratio also increases from simple naive caching to the most advanced ultimate transcoding. The results in Fig. 8(b) illustrates, without transcoding, storage consumption increases linearly with a high slope. However, with transcoding, storage increases at a significantly lower slope. With the enhanced and ultimate transcoding, because all the tiles are delivered at the highest rates, their storage requirements are higher than the transcoding approach. Note that Fig. 8(b) is generated assuming any cached tiles will be kept in the cache until the end of the streaming session. In reality, any tile that has been cached for more than a certain duration can be removed, where the duration depends on the expected dynamic range in the latency of all the users. From Fig. 8(b), we can estimate that the cache size needed by the algorithms will be approximately 30 GB, 10 GB, 13 GB and 13 GB if the cache lifetime is 30 seconds.



(a) Hit rate Curve (b) Network Efficiency Curve

Figure 9: Hit rate and network efficiency performance of LRU and latency-FoV (LF) based caching algorithms at different cache sizes (in GB).

6.3.2 Caching Performance under Different Storage Capacities. In order to evaluate how the caching performance is affected by the cache size, we conduct experiments with different cache sizes ranging from 0.625 to 22.5 GBytes. For LRU-based caching, when the cache reaches its storage limit, and a new tile request cannot be served from the cache, the earliest cached tiles will be removed until there is enough space to store the new tile. Note that as with the previous experiment, here we do not remove the cached tiles based on their lifetime. Results in Fig. 9(a) show that transcoding can achieve the highest hit rate when cache size is less than 15 GB and naive caching algorithm performs the worst among the three. When cache size reaches 15 GB, both transcoding and enhanced transcoding reach upper bound. With further analysis, 15 GB can cover 48s video content if the highest rate for a video segment is 2500 Mbps. And 48s is larger than the latency gap between the users with the shortest and the longest playback latencies in our setting. Theoretically, as long as the cache size is large enough to cover all the tiles that are actively be requested by some users, the hit rate can reach its upper bound.

The network efficiency results in Fig. 9(b) shows that the enhanced transcoding is not effective unless the cache size is sufficiently large. For example, when the cache size is less than 7.5 GB, the network efficiency is negative, which means the total volume of data delivered in the core network is actually greater than the total volume of tiles requested. The reason is that the same tile might be delivered in the highest rate multiple times from original server to the edge server no matter what the requested rates are. However, when the cache size is relatively large, its network efficiency improves dramatically and reaches the upper bound when cache size is 15 GB. The results of both hit rate and network efficiency further demonstrate LF outperforms LRU in all the cases by about 5%. We expect the performance improvement of LF over LRU will increase when the number of users in a streaming flock increases.

7 CONCLUSION

In this paper, we investigate how the idea of "flocking" can be used to improve the efficiency of live 360° video streaming from the aspects of both FoV prediction and live video caching. By assigning users to different latency groups and making use of the actual FoV attention distributions of the front users who have watched the same video segment, the FoV prediction accuracy for a latter user can be improved, leading to significant increase of the effective video rate (approximately doubled when compared with self-trajectory based FoV prediction). In addition, by assigning users into latency groups with different latencies and buffer upper bounds based on their network conditions, the seemly conflicting goals of low video freeze ratio (requiring long streaming buffer) and high FoV prediction accuracy (requiring short streaming buffer) on individual users can be simultaneously achieved at the flock-level. We further propose a latency-FoV based live 360° video caching strategy and investigate the potential gain of applying transcoding on the edge server. We show that caching at the edge can reduce the core network traffic by about 80% under reasonable cache sizes, which can be increased to 90% with real-time transcoding. The proposed latency-FoV based strategy can further improve the caching performance over the conventional LRU caching by about 5%. For deployment in real systems, whether to use transcoding depends on the desirable trade-off between the bandwidth cost inside the core network and computation cost on the edge servers.

ACKNOWLEDGEMENT

We thank our shepherd Carsten Griwodz and the anonymous reviewers for the valuable comments. This work was supported in part by NSF under contract number CNS-1816500.

REFERENCES

[1] Patrice Rondao Alface, Jean-François Macq, and Nico Verzijp. 2012. Interactive omnidirectional video delivery: A bandwidth-effective approach. *Bell Labs Technical Journal* 16, 4 (2012), 135–147.

[2] Trevor Ballard, Carsten Griwodz, Ralf Steinmetz, and Amr Rizk. 2019. RATS: adaptive 360-degree live streaming. In *Proceedings of the 10th ACM Multimedia Systems Conference*. 308–311.

[3] Yixuan Ban, Lan Xie, Zhimin Xu, Xinggong Zhang, Zongming Guo, and Yue Wang. 2018. Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[4] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu. 2016. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 1161–1170.

[5] Matthias Berning, Takuro Yonezawa, Till Riedel, Jin Nakazawa, Michael Beigl, and Hide Tokuda. 2013. pARnorama: 360 degree interactive video for augmented reality prototyping. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. ACM, 1471–1474.

[6] Jacob Chakareski. 2017. VR/AR immersive communication: Caching, edge computing, and transmission trade-offs. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*. ACM, 36–41.

[7] Gene Cheung, Zhi Liu, Zhiyou Ma, and Jack ZG Tan. 2017. Multi-stream switching for interactive virtual reality video streaming. In *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2179–2183.

[8] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. 2018. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 518–533.

[9] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. 2017. Viewport-adaptive navigable 360-degree video delivery. In *2017 IEEE international conference on communications (ICC)*. IEEE, 1–7.

[10] Fanyi Duanmu, Zhan Ma, Wei Wang, Meng Xu, and Yao Wang. 2016. A novel screen content fast transcoding framework based on statistical study and machine learning. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 4205–4209.

[11] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 67–72.

[12] Vamsidhar Reddy Gaddam, Michael Riegler, Ragnhild Eg, Carsten Griwodz, and Pål Halvorsen. 2016. Tiling in interactive panoramic video: Approaches and evaluation. *IEEE Transactions on Multimedia* 18, 9 (2016), 1819–1831.

[13] Chang Ge, Ning Wang, Severin Skillman, Gerry Foster, and Yue Cao. 2016. QoE-driven DASH video caching and adaptation at 5G mobile edge. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. ACM, 237–242.

[14] Carsten Griwodz, Mattis Jeppsson, Håvard Espeland, Tomas Kupka, Ragnar Langseth, Andreas Petlund, Peng Qiaoqiao, Chuansong Xue, Konstantin Pogorelov, Micheal Riegler, et al. 2018. Efficient Live and on-Demand Tiled HEVC 360 VR Video Streaming. In *2018 IEEE International Symposium on Multimedia (ISM)*. IEEE, 81–88.

[15] Carsten Griwodz, Frank T Johnsen, Simen Rekkedal, and Pål Halvorsen. 2006. Caching of interactive multiple choice MPEG-4 presentations. In *2006 IEEE International Performance Computing and Communications Conference*. IEEE, 7–pp.

[16] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. 2019. Pano: Optimizing 360 video streaming with a better understanding of quality perception. In *Proceedings of the ACM Special Interest Group on Data Communication*. 394–407.

[17] Mohammad Hosseini and Viswanathan Swaminathan. 2016. Adaptive 360 VR video streaming: Divide and conquer. In *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 107–110.

[18] Chenglin Li, Laura Toni, Junni Zou, Hongkai Xiong, and Pascal Frossard. 2017. QoE-driven mobile edge caching placement for adaptive video streaming. *IEEE Transactions on Multimedia* 20, 4 (2017), 965–984.

[19] Chenge Li, Weixi Zhang, Yong Liu, and Yao Wang. 2019. Very Long Term Field of View Prediction for 360-degree Video Streaming. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 297–302.

[20] Xing Liu, Bo Han, Feng Qian, and Matteo Varvello. 2019. LIME: understanding commercial 360° live video streaming services. In *Proceedings of the 10th ACM Multimedia Systems Conference*. ACM, 154–164.

[21] Ge Ma, Zhi Wang, Miao Zhang, Jiahui Ye, Minghua Chen, and Wenwu Zhu. 2017. Understanding performance of edge content caching for mobile video streaming. *IEEE Journal on Selected Areas in Communications* 35, 5 (2017), 1076–1089.

[22] Anahita Mahzari, Afshin Taghavi Nasrabadi, Aliehsan Samiei, and Ravi Prakash. 2018. Fov-aware edge caching for adaptive 360 video streaming. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 173–181.

[23] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 197–210.

[24] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. 2000. Constrained model predictive control: Stability and optimality. *Automatica* 36, 6 (2000), 789–814.

[25] Cise Midoglu, Özgü Alay, and Carsten Griwodz. 2019. Evaluation Framework for Real-Time Adaptive 360-Degree Video Streaming over 5G Networks. In *Proceedings of the 2019 on Wireless of the Students, by the Students, and for the Students Workshop*. 6–8.

[26] MUX. 2019. *The Low Latency Live Streaming Landscape in 2019*. https://mux.com/blog/the-low-latency-live-streaming-landscape-in-2019/

[27] Afshin Taghavi Nasrabadi, Anahita Mahzari, Joseph D Beshay, and Ravi Prakash. 2017. Adaptive 360-degree video streaming using scalable video coding. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 1689–1697.

[28] Urs Niesen and Mohammad Ali Maddah-Ali. 2015. Coded caching for delay-sensitive content. In *2015 IEEE International Conference on Communications (ICC)*. IEEE, 5559–5564.

[29] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. ACM, 1–6.

[30] Rodrigo Silva, Bruno Feijó, Pablo B Gomes, Thiago Frensh, and Daniel Monteiro. 2016. Real time 360 video stitching and streaming. In *ACM SIGGRAPH 2016 Posters*. ACM, 70.

[31] Robert Skupin, Yago Sanchez, Cornelius Hellge, and Thomas Schierl. 2016. Tile based HEVC video for head mounted displays. In *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 399–400.

[32] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. 2018. Multi-path multi-tier 360-degree video streaming in 5G networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 162–173.

[33] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. 2019. A two-tier system for on-demand streaming of 360 degree video over dynamic networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 43–57.

[34] Yaping Sun, Zhiyong Chen, Meixia Tao, and Hui Liu. 2018. Communication, computing and caching for mobile VR delivery: Modeling and trade-off. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[35] Afshin TaghaviNasrabadi, Anahita Mahzari, Joseph D Beshay, and Ravi Prakash. 2017. Adaptive 360-degree video streaming using layered video coding. In *2017 IEEE Virtual Reality (VR)*. IEEE, 347–348.

[36] Wikipedia contributors. 2019. Great-circle distance — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Great-circle_distance&oldid=913029918 [Online; accessed 8-January-2020].

[37] Wikipedia contributors. 2019. Kullback–Leibler divergence — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Kullback%E2%80%93Leibler_divergence&oldid=930573126 [Online; accessed 13-January-2020].

[38] Wikipedia contributors. 2020. Kalman filter — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Kalman_filter&oldid=934461652 [Online; accessed 10-January-2020].

[39] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A dataset for exploring user behaviors in VR spherical video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 193–198.

[40] Xiufeng Xie and Xinyu Zhang. 2017. Poi360: Panoramic mobile video telephony over lte cellular networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. ACM, 336–349.

[41] Yu-Ting Yu, Francesco Bronzino, Ruolin Fan, Cedric Westphal, and Mario Gerla. 2015. Congestion-aware edge caching for adaptive video streaming in information-centric networks. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 588–596.

[42] Alireza Zare, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. 2016. HEVC-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 601–605.