# Tile-based edge caching for live 360$^o$ video streaming

Pantelis Maniotis, Nikolaos Thomos, *Senior Member, IEEE*

***Abstract*—360$^o$ video is an increasingly popular technology on commercial social platforms and is a vital part of emerging Virtual Reality (VR) and Augmented Reality (AR) applications. However, the delivery of 360$^o$ video content in mobile networks is challenging because of the size of 360$^o$ video files. As a remedy to the excess bandwidth requirements of 360$^o$ video delivery systems, encoding into multiple quality layers and tiles and caching of the most popular tiles at the wireless edge have been previously proposed. Existing works shown promising performance for Video on Demand (VoD) 360$^o$ video delivery, but they cannot be straightforwardly extended in a live streaming setup. Motivated by the above, we study the optimal cache content placement problem at the Small Base Stations (SBSs) for a live streaming setup with the aim to increase the overall quality of the delivered 360$^o$ videos to the users and reduce the service cost. To solve the cache optimization problem, we propose a novel framework that is able to accommodate multiple simultaneous requests for the same or different content. Our framework employs Long Short-Term Memory (LSTM) networks, which have been be shown to be efficient for time series forecasting. To further enhance the delivered video quality, users located in the overlap of the coverage areas of multiple SBSs are allowed to receive data from any of these SBSs. We evaluate and compare the performance of our algorithm with that of the Least Frequently Used (LFU), Least Recently Used (LRU), and First In First Out (FIFO) algorithms, which are commonly used in the literature. The results show that our framework is able to cache the most popular content at the SBSs. This leads to an increase in the overall delivered quality, while the cost related to backhaul links is reduced.**

***Index Terms*—Tile-encoding, 360$^o$ video, live streaming, edge-caching.**

## I. Introduction

In recent years, we have witnessed the emergence of many live streaming platforms such as YouTube Live, Facebook Live, Twitch, Periscope and Meerkat [1], which has been fostered by the proliferation of mobile devices that provide access to the Internet from "everywhere". These platforms allow attendands of events, e.g., games, shows, conferences, etc., to capture them using affordable 360$^o$ cameras, and then broadcast them to potentially millions of viewers.

To provide users an immersive experience, 360$^o$ videos may be watched with the help of Head Mounted Displays (HMDs), e.g., Oculus Rift, Samsung Gear VR, and HTC Vive [2]. In HMDs, each 360$^o$ scene is projected in the internal part of a spherical surface [3]. A user wearing an HMD watches only a Field of View (FoV) of the spherical scene, known as viewport. The projected viewport is controlled from the orientation of the viewer's head in the x, y and z axes, as shown in Fig. 1. These axes are called pitch, yaw, and roll, respectively [4].

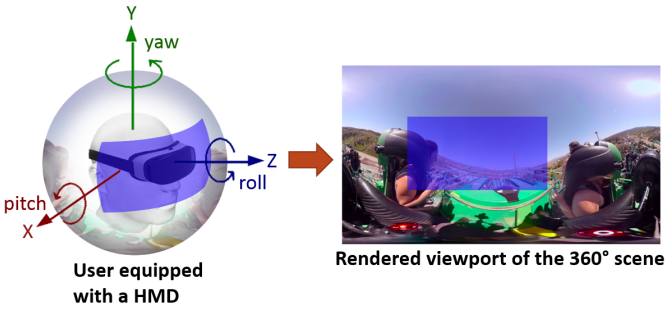


Fig. 1. User equipped with a Head Mounted Display (HMD), and the corresponding video scene.

To prevent users from experiencing motion sickness, the response of the mobile network to the users' head movements should be as fast as the HMD refresh rate [5]. Considering that the refresh rate is commonly 120Hz, the mobile network should deliver the requested viewport to the users in less than 10msec. The end-to-end delivery delay imposes a tight constraint that challenges mobile networks to respond to such demands by tracking and rendering only the requested viewport. Transmitting the whole scene could help to overcome the above limitation, as there would be no need for real-time respond to head movements. However, this is not an efficient strategy as it requires significant bandwidth resources due to the high resolution of 360$^o$ videos (e.g., 4K, 8K or higher) [6]. In addition, delivering the entire scene leads to significant bandwidth waste since only a part of the 360$^o$ video will be displayed. The problem deteriorates when the network infrastructure must support a large number of users which may want to access the requested content from various locations.

Tile-based streaming has been proposed in [3], [7], [8] to reduce bandwidth requirements for delivering 360$^o$ videos. In tile-based video streaming systems, 360$^o$ videos are encoded into independently encoded segments (see Fig. 2), known as tiles. Encoding in tiles is supported by modern video encoding standards such as the H.265/HEVC [9], AV1, and VP9. Tile-encoding permits to stream only the requested viewport in high quality to a user. This viewport is predicted based on the users' navigation patterns, which allows its prefetching to the user in order to meet the delivery deadlines. The rest of the scene is still transmitted to the user, but at a lower quality, in order to prevent the rendering of black areas (i.e., areas without content) in case of an erroneous estimation of the demanded viewport [10].

P. Maniotis and N. Thomos are with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom (e-mail: {p.maniotis, nthomos}@essex.ac.uk).

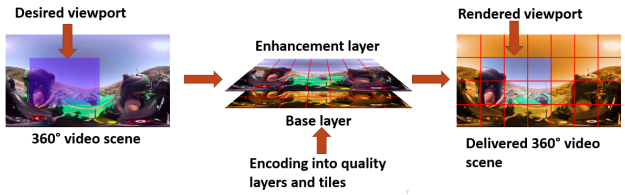


Fig. 2. Desired viewport, encoding into multiple quality layers and tiles, and delivered tiles to a user.

When multiple users request $360^o$ videos, independently streaming the requested content to the users leads to waste of valuable bandwidth in the core network and the backhaul links. To avoid such inefficient use of bandwidth resources, edge caching has shown to be an efficient method to accommodate the demands of multiple users in cellular networks. In edge caching systems, Small Base Stations (SBSs), e.g., pico-cells and femto-cells, are equipped with a cache where they can store popular content files [11], [12]. The use of caches permits to serve the content from the SBSs and limits the need to retrieve it from distance servers through backhaul links. Content retrieval from caches also reduces the experienced latency. However, existing works [13]–[15] that exploit $360^o$ video properties to build edge caching systems are appropriate for Video on Demand (VOD) systems, and they cannot be trivially used for $360^o$ video live streaming. Specifically, the works in [13]–[15] belong to the family of offline-caching schemes, and require the knowledge of the content popularity distribution.

In this paper, we propose a novel online caching framework to support live $360^o$ video streaming. To the best of our knowledge, this is the first caching scheme for $360^o$ videos that is appropriate for live streaming scenarios. Our system aims at maximizing the overall quality rendered by the users, and decrease the usage of the backhaul links. To achieve this, our algorithm uses limited observations of the users' requests for the various $360^o$ videos and viewports in order to decide the optimal cache eviction/placement strategy. Similarly to [13]–[15], we exploit encoding of the $360^o$ videos into multiple quality layers and tiles. When our system decides to cache a $360^o$ video, it caches all the tiles of that video encoded in base quality to ensure interactivity, and a number of tiles in high quality to enhance the quality of the delivered video quality. The latter decision is based on videos' popularity, i.e., for the most popular videos more tiles are cached in high quality, as well as the tiles' popularity, which determines which tiles are most likely to be requested.

To determine which tiles of the $360^o$ videos will be cached and in what quality, we use LSTM networks. These networks have been efficiently used in the literature for time series forecasting [16]. Through the use of the LSTM networks, the popularity of the $360^o$ videos and tiles for the next Group of Pictures (GOP) can be predicted. This permits the predicted content to be prefetched at the caches of the SBSs, and facilitates on time delivery of the content to the users. To evaluate the performance of our solution we use both real and synthetic $360^o$ video traces, and compare our solution with that of the Least Frequently Used (LFU), Least Recently Used (LRU), and First In First Out (FIFO) algorithms, which are common cache update solutions used to support live streaming applications. To further enhance the performance of all methods, we allow the association of users with multiple SBSs, when the users are located in the communication range of these SBSs. In this way, users have access to content stored in multiple caches from where they can satisfy their requests.

In summary, the contributions of our work can be summarized as follows:

- The introduction of a novel online caching framework to support live $360^o$ video streaming. Our framework takes into account both the popularity of each $360^o$ video and the popularity of each viewport to decide which tiles to cache at the SBSs.
- The use of LSTM networks to predict the evolution of $360^o$ video content popularities. . This permits to prefetch the caches of the SBS with content that is likely to be requested in the future, and allows SBSs to serve it to the users in timely manner.
- The extensive evaluation of the proposed solution with respect to various parameters and the comparison of it with several state-of-the-art schemes in order to showcase the benefits coming from our LSTM empowered framework.

The rest of this paper is organized as follows. In Section II, we overview works related to $360^o$ videos, $360^o$ video live streaming, and edge caching. Then, in Section III we describe the system setup, and afterwards, in Section IV we provide the system model. Next, in Section V we evaluate the performance of the proposed scheme. Finally, conclusions are drawn in Section VI.

## II. Related Work

During the past decade, $360^o$ video attracted significant attention from the academic community. Different aspects of $360^o$ video have been studied and systems for processing and streaming such content have been presented. For example, authors in [17] designed subjective experiments in order to analyze users' navigation patterns when viewing $360^o$ videos. It is observed that for a short time period in the beginning of the $360^o$ video streaming, e.g., first 30 seconds, users tend to look around and their attention is very low. However, after about 1 minute, users' attention becomes more focused. Authors in [18] propose a solution based on MPEG DASH, in order to cope with the bandwidth limitations when streaming $360^o$ videos at resolution of 16K with a FoV of 4K. A Hadoop/Spark based transcoding solution is presented in [19], to facilitate the delivery of high resolution $360^o$ videos, i.e., 8K and above. The evaluation shows that real-time transcoding of $360^o$ videos with resolution of 8K, split in 8x8 tiles, can be achieved at a rate of 99 fps. To predict the demanded viewports by the users in the near future, authors in [20] present a contextual bandit algorithm. Differently from [20], authors in [21] propose a trajectory-based viewport prediction algorithm, aiming to predict the users' requested viewports in the long-term.

Tile-encoding has been exploited for $360^o$ live video streaming [22] in order to facilitate the delivery in high quality of

only the requested FoV. The presented live streaming platform is supported by an architecture based on RTP and DASH. The results made apparent the trade-off between video quality and bandwidth usage and shown bandwidth savings of about 50%. Differently from [22], authors in [23] proposed a multicasting system to support $360^o$ video live streaming. Simulations shown that users with low bandwidth capacities experienced a quality gain of 3 dB, while users with high bandwidth capacities saw a gain of 3.5-4 dBs. To guarantee the delivery of high-quality $360^o$ videos to the users, a live streaming system based on MPEG media transport (MMT) is proposed in [24]. A generic measurement system is presented in [25] for the collection of key performance statistics, e.g., video quality change, rebuffering events, for evaluating the performance of live streaming platforms for $360^o$ videos. Authors in [2] proposed a mobile video telephony system over LTE cellular networks that dynamically adjusts the compression strategy, in order to provide a high Quality of Experience (QoE) to the users.

Edge caching has been proposed as an enabling technology to support streaming of both standard videos [26], and $360^o$ videos [13]–[15], [27]. In edge caching architectures, SBSs cache popular content, from where users can retrieve the requested content directly. Caching is beneficial in terms of reducing the usage of the pricey backhaul links [12], [28], and helps users enjoy services with less latency. The advantages of using edge caches to store $360^o$ videos is demonstrated in [13], [14]. The decisions regarding where to cache a content and from where to retrieve it are made jointly a on per tile and per layer basis. Differently to [13], [14], authors in [15] examine $360^o$ video caching when each tiles are encoded at different resolutions, and in multiple layers.

## III. System Setup

In this section, we first introduce the considered live streaming architecture. Then, we discuss the transcoding of $360^o$ videos so that they have multiple quality layers and tiles. Finally, we describe the considered mobile edge network architecture, the users' requests model and the end-to-end delay in the mobile edge network.

*1) Live Streaming Architecture:* A high level representation of the considered live $360^o$ video streaming architecture is depicted in Fig. 3. We assume multiple users (broadcasters) that capture a $360^o$ FoV of a scene, using omnidirectional cameras. The captured $360^o$ videos are first transmitted to the Live Stream server using the Real-Time Messaging Protocol (RTMP) [29]. RTMP is selected as it can ensure low end-to-end latency between the broadcaster and viewers. The Live Stream server transcodes the $360^o$ videos so that they consist of multiple multiple quality layers and tiles. This is because the captured $360^o$ videos from the broadcasters may not be necessarily encoded in that format. The transcoded video streams are transmitted to the Content Delivery Network (CDN) using HTTP. The mobile edge servers are populated with content from the CDN according to the proposed cache optimization algorithm, which will be presented in Section IV. We would like to note that the focus of this work is on the cache optimization of the mobile edge caches.

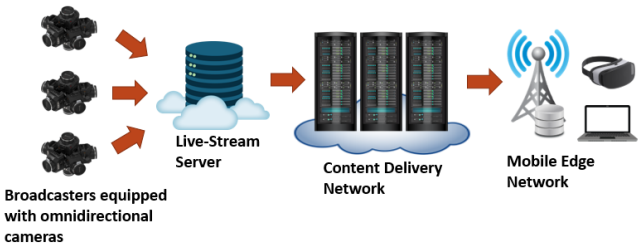


Fig. 3. Considered live streaming architecture.

*2) Transcoding of $360^o$ videos:* We assume that the total number of the captured $360^o$ videos is $V = |\mathcal{V}|$, where $\mathcal{V} = \{1, \dots v, \dots, V\}$ is the set of $360^o$ videos that comprise the content library. The video transcoding at the Live Stream servers is performed using H.265/HEVC [9], but our scheme is compliant with other video codecs. Each video stream is encoded into a number of $L$ quality layers and $M$ tiles. The first quality layer is the base layer, while the rest $L-1$ layers are known as enhancement layers. The acquisition of a tile at the base quality layer offers reconstruction of the tile at the lowest quality, while the acquisition of up to the $l$-th quality layer gradually enhances the quality of that tile. As the duration of each video may vary, each video consists of a number of GOPs, that depends on the duration of each video.

*3) Content Delivery Network:* CDN is comprised of geographically distributed servers that collaboratively cache and distribute popular content to a global reach, using high speed links. Each CDN server has the capacity to cache a number of video files. Using a CDN, the requested content in a geographic area may be served from the cache of the CDN server that is closer to the users. In this way, the demanded content is retrieved by the users with less latency, as their requests do not have to be routed to the Live Stream server. The use of CDNs reduces significantly the traffic reaching the Live Stream server.

*4) Mobile Edge Network:* We consider a mobile edge network architecture as the one depicted in Fig. 4. This network consists of $N$ Small Base Stations (SBSs), i.e., microcells, and a Macrocell Base Station (MBS). Let $\mathcal{N} = \{1, \dots n, \dots, N\}$ be the set of the $N$ SBSs, and $N+1$ represent the MBS. The MBS is connected with the CDN through a high capacity backhaul link, i.e., optical fiber, while the connection of the SBSs with the CDN is established through the MBS by millimeter wave links. The communication ranges of the SBSs are represented by the set $\mathcal{P} = \{p_1, \dots p_n, \dots, p_N\}$, with $p_n$ being the communication range of the $n$th SBS. We denote the communication range of the MBS by $p_{N+1}$; all SBSs are within range of the MBS otherwise they would not have access to backhaul of the MBS. The cache capacity of the $n \in \mathcal{N}$ SBS is denoted by $C_n \geq 0$, $\forall n \in \mathcal{N}$.

*5) Users:* In the considered network architecture, we consider $U$ users that form the set $\mathcal{U} = \{1, \dots u, \dots, U\}$. Users may be located in the overlap of the coverage areas of multiple SBSs, as shown in Fig. 4. When this happens, users may be associated with any of these SBSs. The primary SBS for each user is the one that has the maximum signal-to-

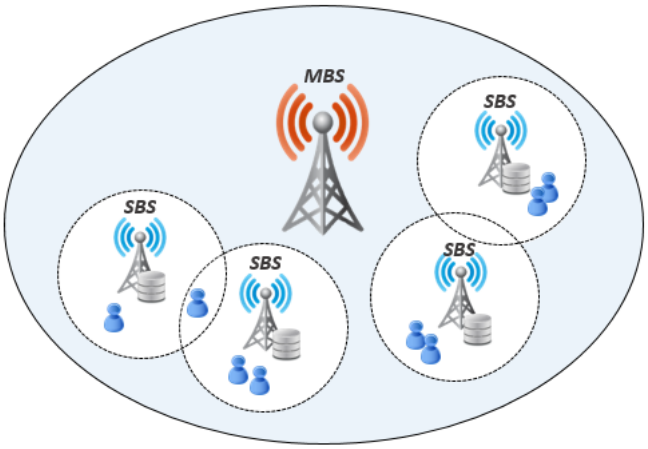


Fig. 4. Considered mobile-edge architecture.

interference-plus-noise ratio (SINR). The association of a user with multiple SBSs allows requested tiles that are not cached at the primary SBS, but are stored in the cache of one (or more) of the other SBSs the user resides, to be delivered to the user from these caches. We assume that time is slotted in $T$ time slots. In each time slot $t \in \mathcal{T}$, we denote the request of user $u \in \mathcal{U}$ for GOP $g \in \mathcal{G}$ of a $360^o$ video $v \in \mathcal{V}$ by $w_u^t$. Let $W_u = \{w_u^1, \ldots, w_u^t, \ldots, w_u^T\}$ be the set which has $T$ consecutive requests from user $u \in \mathcal{U}$.

*6) End-to-end delivery constraint:* The end-to-end delay captures the overall delay a user experiences from when they request the data until the data is delivered to them. This delay depends on from where the data is retrieved. Let $d_n$ be the delay needed to transmit one Mbit from the cache of the $n$th SBS to a user within the SBS communication range. Similarly, let $d_{N+1}$ be the delay needed to transmit one Mbit that is fetched from the backhaul of the MBS to a user. In order to guarantee the timely delivery of the tiles of each GOP to the users the following constraint should be met:

$$\sum_{n \in \mathcal{N}_B} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} o_{vglm} \cdot d_n \cdot y_{vglm}^n \leq t_{disp}, \forall v \in \mathcal{V}, g \in \mathcal{G} \quad (1)$$

The parameter $o_{vglm}$ in (1) denotes the size (in Mbits) of the $m$th tile of the $l$th quality layer of the $g$th GOP of the $v$th $360^o$ video. The variable $y_{vglm}^n$ takes the value 1 when the the $m$th tile of the $l$th quality layer of the $g$th GOP of the $v$th $360^o$ video is delivered on time to the $u$th user from the $n$th SBS ($n \in \mathcal{N}$) or the MBS ($n = N+1$), and 0 otherwise. The parameter $t_{disp}$ corresponds to the time needed for each GOP to be displayed.

## IV. System Model

*1) Caching Entity (CE):* We consider that each SBS is equipped with a CE, as shown in Fig. 5. This entity is responsible for deciding which $360^o$ videos and tiles should be cached at each SBS. Each caching entity is composed of a number of modules, e.g., User Requests Processor, User Requests Forecasting, Feature Updater, etc., that their operation will be discussed later.

*2) Users Request Processor (URP):* The URP module is responsible for decomposing the user requests $w_u^t, t \in \mathcal{T}, u \in \mathcal{U}$. Specifically, if a viewport consists of $k$ tiles, each user request $w_u^t$ is decomposed into $k+1$ requests $w_{u,i}^t$, as shown in Fig. 6. The first request $w_{u,0}^t$ is for receiving all the tiles of the requested $360^o$ video at the base quality. The rest $k$ requests $\{w_{u,1}^t, \ldots, w_{u,i}^t, \ldots, w_{u,k}^t\}$ are for receiving each of the $k$ tiles of the requested viewport in high quality. This decomposition gives our system the flexibility to decide which $360^o$ videos (all tiles at the base quality) should be cached to ensure interactivity, and which tiles of these videos should be cached in high quality. Let the set $\mathcal{W}_u^t = \{w_{u,0}^t, w_{u,1}^t, \ldots, w_{u,i}^t, \ldots, w_{u,k}^t\}$ describe these $k+1$ requests.

*3) Feature Updater (FU):* The FU module is responsible for updating features regarding the requests for the various $360^o$ videos and tiles. Specifically, this module calculates in each time slot $t \in \mathcal{T}$, the number of times each request (for receiving either a $360^o$ video at the base quality, or a tile of a viewport in high quality) was encountered. The computed features are transferred to the Feature Database (FD) module, where this information is stored.

*4) Feature Database (FD):* The FD module stores the features computed by the FU module regarding the number of times the various $360^o$ videos (all tiles at base quality) and tiles (in high quality) were requested at an SBS.

*5) Users' Requests Queue (URQ):* After the decomposition of each user request into multiple requests by the URP module, the decomposed requests are directed to the URQ module. This module applies a technique called request coalescing [30]. According to this technique, when multiple requests for the same content arrive simultaneously at an SBS, the first request is prioritized for processing, while the rest of the user requests are hold in a queue. This mechanism is needed because in live streaming scenarios, many people are watching the same content almost simultaneously. Without such mechanism, in case a requested content is not cached at the SBS, a cache miss will occur for all the users' requests for that content. This would cause all the traffic related to that content to be redirected to the origin CDN or even the Live Stream server, causing the crashing of these servers.

*6) Content Prefetcher (CP):* Due to the end-to-end delay, SBSs are not able to respond instantly to the users' head movements and transmit the demanded tiles by the users. To overcome this problem, the CP module is used. Specifically, when the CP module receives requests $\mathcal{W}_u^t = \{w_{u,0}^t, w_{u,1}^t, \ldots, w_{u,i}^t, \ldots, w_{u,k}^t\}$ for the various $360^o$ videos (in base quality) and tiles (in high quality) from the URP at the time slot $t$, it decides what content should be prefetched for the various $360^o$ videos and tiles for the next time slot $t+1$. Let the set $\mathcal{Z}_u^{t+1} = \{z_{u,0}^{t+1}, z_{u,1}^{t+1}, \ldots, z_{u,i}^{t+1}, \ldots, z_{u,k}^{t+1}\}$ denote the content that will be prefetched to the users regarding the time slot $t+1$. Enabling prefetching allows the timely delivery of the content to the users. To decide which content should be prefetched, similarly to [27], we use the Last Sample Replication (LSR) [10] algorithm. According to this algorithm, when the CP module receives a user request $w_{u,i}^t$ at time slot $t$, the content $z_{u,i}^{t+1}$ that is decided to be prefetched for the time slot $t+1$ is considered to be the same with the request

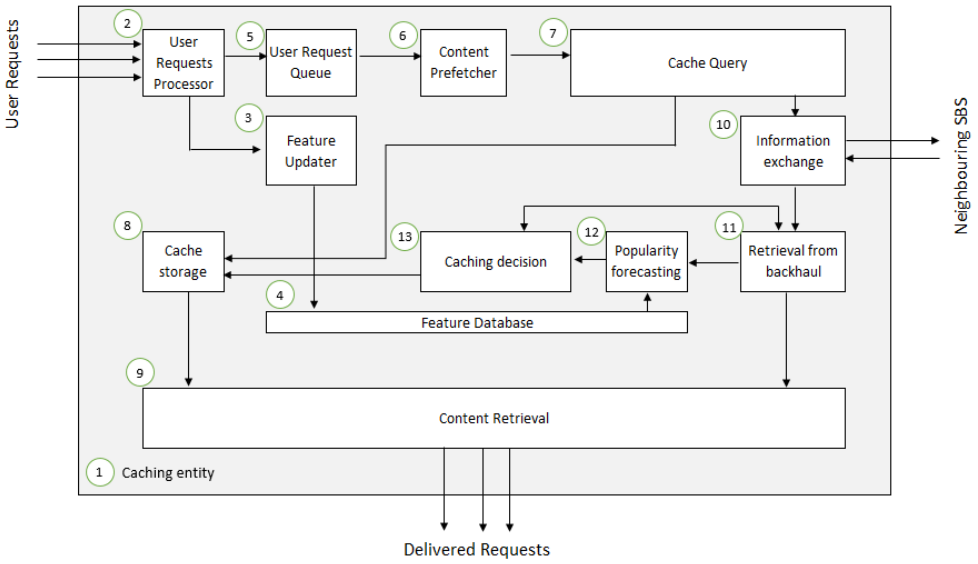


Fig. 5. Flow of operations in a caching entity.

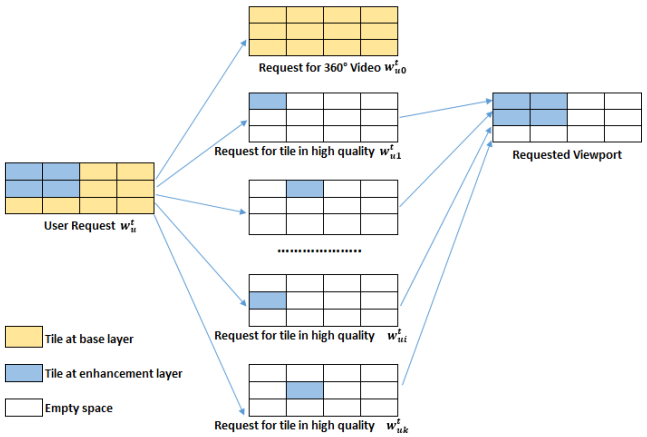


Fig. 6. Decomposition of user request $w^t_u$ into $k+1$ requests.

$w^t_{u,i}$. For the sake of simplicity, we assume that for the first time slot, the content that will be prefetched to the users is the requested content.

*7) Cache Query (CQ):* The CQ module examines whether the content indicated by the CP module is already cached at the SBS. When this content is cached at the SBS, it is served to the user locally from the Cache Storage (CS) of the SBS. When the content indicated by the CP module is not available at the SBS, the Information Exchange (IE) module is activated to check whether it can be served by a neighbouring SBS. This happens when the user is in the communication range of the neighbouring SBS. In case the content is cached at a neighbouring SBS and the user can be associated with that SBS, the content is delivered to the user from that SBS. Differently, when the content indicated by the CP module is not cached at any neighbouring SBS the user may be associated with, the content is fetched at the SBS through the backhaul of the MBS, and served from there to the user.

*8) Cache Storage (CS):* The CS module is responsible for storing the $360^o$ videos at the base quality and the tiles of the cached videos in high quality. Each SBS has a separate cache storage module with capacity $C_n$, where $n$ is the index of the SBS node.

*9) Content Retrieval (CR):* The CR module is responsible for the delivery of the content to the users. This module retrieves the content either through: a) the CS module, when it is cached at the SBS, or b) the Retrieval from Backhaul (RFB) module that retrieves the content from the backhaul, when it is not cached at the SBS.

*10) Information Exchange (IE):* The IE module is responsible for the communication of an SBS with its neighbouring SBSs. Specifically, in case of a cache miss for a content at an SBS, the SBS checks whether that content may be served to the user from a neighbouring SBS. The communication between the SBSs is accomplished by millimeter wave links through the MBS. The delay needed for the above communication is captured by the delay parameter $d_n$. Recall, that this parameter denotes the delay needed to deliver one Mbit from the $n$th SBS to a user, in the constraint (1).

*11) Retrieval from Backhaul (RFB):* The RFB module is responsible for the retrieval of the content that will be prefetched to caches of the SBSs through the backhaul. After the retrieval of that content at the RFB module, its popularity is estimated by the Popularity Forecasting (PF) module, and a decision is made at the Caching Decision (CD) module about whether to cache that content.

*12) Popularity Forecasting (PF):* The PF module forecasts at each time $t$, the popularity of the content that will be prefetched for the time slot $t+1$. Specifically, the PF module uses a window of $h$ time slots, in order to capture the trends in the popularity of the content that will be prefetched to the users, and cache at the SBSs content that will be popular. To this aim, the features (number of requests for videos and tiles) stored at the FD module regarding the previous $h-1$ time

slots along with the current time slot $t \in \mathcal{T}$ are used. Let us denote by $\lambda^t_{n,v,0}$ the popularity of the $360^o$ video $v \in \mathcal{V}$ (base quality) at the SBS $n \in \mathcal{N}$ in the time slot $t \in \mathcal{T}$. Similarly, let $\lambda^t_{n,v,m}$ stand for the popularity of the tile $m \in \mathcal{M}$ of the video $v \in \mathcal{V}$ at the SBS $n \in \mathcal{N}$, where $\mathcal{M} = \{1, \ldots, m, \ldots M\}$.

One way to predict these popularities would be to use Recurrent Neural Networks (RNNs), as they have been shown to be effective for time series data forecasting [31]. However, simple RNNs cannot capture long-term dependencies, as they lack control structures, which causes the norm of gradients to decay or explode during training [32]. To overcome this problem, LSTM networks may be used [33]. LSTMs are a special type of RNNs able to learn long-term dependencies. Inspired by [34], we use an LSTM network for the forecasting of the popularity of the content retrieved by the RFB module. The LSTM network takes as input the features of a content regarding the previous $h-1$ time slots along with the current time slot $t$, and outputs the estimated popularity for the content for the time slot $t+1$. The LSTM is initially pre-trained offline (warm-up phase) with historic data profiles using the backpropagation through time method, in order to find a good starting point for its weights. Then, these weights are used for the popularity prediction of the content retrieved by the RFB module.

*13) Caching Decision (CD):* The CD module makes decisions regarding whether to cache the retrieved content by the RFB module. To this aim, it uses the popularities predicted by the PF module.

Let us denote the total number of cached $360^o$ videos (at the SBS $n \in \mathcal{N}$) at the base quality as $b_n$, and the total number of cached tiles in high quality as $f_n$. In addition, let the forecast popularities of the cached $360^o$ videos at the base quality at the time slot $t+1$ be described by the set $\mathcal{B}^{n,t+1} = \{B^{n,t+1}_1, \ldots, B^{n,t+1}_i, \ldots, B^{n,t+1}_{b_n}\}$, and the forecast popularities of the cached tiles in high quality by the set $\mathcal{F}^{n,t+1} = \{F^{n,t+1}_1, \ldots, F^{n,t+1}_j, \ldots, F^{n,t+1}_{f_n}\}$.

When the content that will be prefetched to a user is cached locally or at a neighbouring SBS the user resides, it is delivered to the user from the SBS it is cached. In such case, no decision is made at the CD module. In different case, the content is retrieved at the RFB module, and a decision is made about whether to cache it. Specifically, a decision is first made about whether to cache at the SBS the request $z^{t+1}_{u,0}$ regarding the $360^o$ video indicated by the CP module (when it is not cached). If the predicted popularity by the PF module for the $z^{t+1}_{u,0}$ is $\lambda^{t+1}_{n,v,0}$, the $z^{t+1}_{u,0}$ will be cached at the SBS in the place of the $i$th cached $360^o$ video at the base quality if $\lambda^{t+1}_{n,v,0} > B^{n,t+1}_i$ and $\min(\mathcal{B}^{n,t+1}) = B^{n,t+1}_i$. Next, in case the $360^o$ video is cached at the base quality, a decision is made for each one of the tiles in high quality $\{z^{t+1}_{u,1}, \ldots, z^{t+1}_{u,i}, \ldots, z^{t+1}_{u,k}\}$ that are not cached about whether they should be stored at the CS module. Specifically, if the predicted popularity by the PF module for the tile $z^{t+1}_{u,i}$ is given by $\lambda^{t+1}_{n,v,m}$, the tile $z^{t+1}_{u,i}$ will be cached in the place of the $j$th cached tile in high quality if $\lambda^{t+1}_{n,v,m} > F^{n,t+1}_j$ and $\min(\mathcal{F}^{n,t+1}) = F^{n,t+1}_j$. However, if the initial decision regarding the request $z^{t+1}_{u,0}$ was not to cache it, no further decision is made, and none of the content requests $\{z^{t+1}_{u,1}, \ldots, z^{t+1}_{u,i}, \ldots, z^{t+1}_{u,k}\}$ regarding the tiles in high quality are cached. The aforementioned decision process (cache updates) made by the CD module using the forecast popularities by the PF module is summarized in Algorithm 1.

**Algorithm 1** Caching decisions using forecast popularities

**Offline Phase**
Pre-train the LSTM network with historic transition profiles, using the backpropagation method
**Online Phase**
**for** each time slot $t$ **do**
  **for** each user $u$ **do**
    **for** each user request $w^t_{u,i}, i \in \{0, 1, \ldots, k\}$ **do**
      **if** $z^{t+1}_{u,0}$ (all tiles at base quality) are not cached **then**
        **if** $\lambda^{t+1}_{n,v,0} > B^{n,t+1}_i$ and $\min(\mathcal{B}^{n,t+1}) = B^{n,t+1}_i$ **then**
          Cache $z^{t+1}_{u,0}$ in place of the $i$th cached $360^o$ video at base quality
        **end if**
      **end if**
      **if** $z^{t+1}_{u,0}$ (all tiles at base quality) are cached **then**
        **if** $z^{t+1}_{u,i}$ (tile in high quality) is not cached **then**
          **if** $\lambda^{t+1}_{n,v,m} > F^{n,t+1}_j$ and $\min(\mathcal{F}^{n,t+1}) = F^{n,t+1}_j$ **then**
            Cache $z^{t+1}_{u,i}$ in place of the $j$th cached tile in high quality
          **end if**
        **end if**
      **end if**
    **end for**
  **end for**
**end for**

Following the above workflow, the cache is populated with the most popular $360^o$ videos at the base quality. Tiles in high quality are cached only for the videos with cached base layer tiles. The number of cached tiles in high quality for each cached $360^o$ video depends on videos' popularity, i.e., the more popular is a $360^o$ video, the more tiles are cached at the SBSs. Hence, for the least popular videos a small number of tiles or even no tiles may be cached at the SBSs. This provides our scheme greater flexibility in deciding how many tiles to cache per video, helps increase the cache hits, and enhance the quality of the displayed video, as we see in the next section.

The key notation of our problem is summarized in Table I.

TABLE I
NOTATION

| Symbol | Physical Meaning |
|---|---|
| $\mathcal{N}$ | Set of Small Base Stations |
| $\mathcal{N_B}$ | Set of SBSs and MBS |
| $\mathcal{V}$ | Set of $360^o$ videos |
| $\mathcal{G}$ | Set of GOPs |
| $\mathcal{L}$ | Set of quality layers |
| $\mathcal{M}$ | Set of tiles |
| $\mathcal{U}$ | Set of users |
| $\mathcal{T}$ | Set of time slots |
| $\delta vglt$ | Distortion reduction from obtaining the tile $vglt$ |
| $o_{vglt}$ | Size of the tile $vglt$ |
| $C_n$ | Cache capacity of the SBS $n$ |
| $t_{disp}$ | Time duration of a GOP |

## V. Performance Evaluation

In this section, we evaluate the performance of the proposed framework for enabling live $360^o$ video streaming. For all the schemes under comparison, users can be associated with multiple SBSs, when they reside in the transmission range of these SBSs. Hence, they can obtain their data from one of the neighboring SBSs when the data is not found in the primary SBS. We should note that when the users' requests arrive at an SBS, the cache update decisions are made at that SBS regardless if the users obtained their data from a neighboring SBS, or the backhaul.

### A. Simulation Setup

The schemes under comparison, as well as the proposed scheme, are described below:

1) *Least Frequently Used (LFU):* In this scheme, the network operator keeps track of the number of requests that occurred for each GOP of each cached $360^o$ video. When a request for the $g$th GOP of a $360^o$ video arrives at an SBS, the LSR algorithm is used to decide which content will be prefetched as follows: a) if no tiles of the GOP $g+1$ are cached at the SBS, all the tiles of the GOP $g+1$ of the $360^o$ video that was requested the least frequently will be evicted from the cache at the SBS. Next, the evicted tiles will be replaced by the tiles indicated by the LSR algorithm; b) if the tiles of the GOP $g+1$ are cached at the base quality, but some (or all) of the cached tiles in high quality are different from the tiles of the viewport indicated by the LSR, these tiles will be evicted from the cache of the SBS and the tiles of the viewport indicated by the LSR will be cached in their place.
2) *Least Recently Used (LRU):* In this scheme, the network operator keeps track of how recent are the requests that occurred for each GOP of each cached $360^o$ video. When a user request for a GOP $g$ of a $360^o$ video arrives at an SBS, according to the LSR algorithm: a) if no tiles of the GOP $g+1$ are cached at the SBS, all the tiles of the GOP $g+1$ of the $360^o$ video that was requested the least recently will be evicted from the cache at the SBS. Then, the tiles indicated by the LSR algorithm will be cached in the place of the evicted tiles; b) if the tiles of the GOP $g+1$ are cached at the base quality, but some (or all) of the cached tiles in high quality are different from the tiles of the indicated by the LSR viewport, these tiles will be replaced by the tiles that form the indicated by the LSR viewport.
3) *First In First Out (FIFO):* In this scheme, the network operator keeps track of when the requests for each GOP of each cached $360^o$ video occurred. For a user request for the GOP $g$ of a $360^o$ video, according to the LSR algorithm: a) if no tiles for the GOP $g+1$ are cached at the SBS, all the tiles of the GOP $g+1$ of the $360^o$ video that was requested the earliest will be evicted from the cache at the SBS. Next, all tiles of the output of the LSR algorithm for the GOP $g+1$ will be cached at the SBS; b) if the tiles of the GOP $g+1$ are cached at the base quality, but some (or all) of the cached tiles in high quality are different from the tiles of the viewport indicated by the LSR, these tiles will be evicted from the cache of the SBS and the tiles forming the indicated by the LSR viewport will be cached in their place.
4) *Proposed Scheme:* In the proposed scheme, the caching decisions are performed following the cache update framework described in Section IV. The proposed scheme uses popularity forecasting to decide with what content to populate the SBSs caches and how to update them. For each cached video, all the tiles in base quality are cached at the SBS. In addition, for each GOP, a number of tiles for the most popular videos are cached in high quality.

For all the conducted experiments, unless otherwise specified, we assume a cellular network that consists of $N = 3$ SBSs along with an MBS. The coverage range of each SBS is $p_n$ = 200m, and the coverage range of the MBS is $p_{N+1}$ = 2000m. The cache capacity of the SBSs is set to be enough to cache 10% of the $360^o$ videos content libary. This space is calculated assuming that for each GOP of each cached $360^o$ video, all the tiles at the base quality along with the tiles of one viewport in high quality are cached. However, as we have already mentioned, in the proposed scheme the number of tiles in high quality that will be cached for each $360^o$ video depend on the popularity of each $360^o$ video and viewport. Furthermore, we consider that the total number of users is $U = 540$, who are randomly placed in the coverage area of the $N = 3$ SBSs. The bit-rate (transmission delay) at which data is delivered from the cache of the users' primary SBS is $1/d_n = 14$ Mbit/sec. The bit-rate at which case data is delivered to a user from a non primary SBS equals to $1/d_n = 13$ Mbit/sec. When a request for a $360^o$ video at base quality or tile in high quality is not cached at any of the SBSs the user resides, it is delivered from the backhaul with bit-rate equal to $1/d_{N+1} = 2.9$ Mbit/sec.

The content library is comprised of $V = 100$ videos. Each $360^o$ video has a duration of 300 GOPs, with each GOP lasting $t_{disp} = 1$ sec. Each GOP of the $360^o$ videos is encoded in $M = 12$ tiles and $L = 2$ quality layers, while the size of each viewport is 4 tiles. The considered viewports are depicted in Fig. 7. The bitrate of the base layer is 2 Mbps, while the bitrate of the enhancement layer is 12 Mbps. The distortion reduction achieved by acquiring a tile at the base quality layer is $\delta_{v,g,1,m} = 30$ dB, while the distortion reduction achieved by

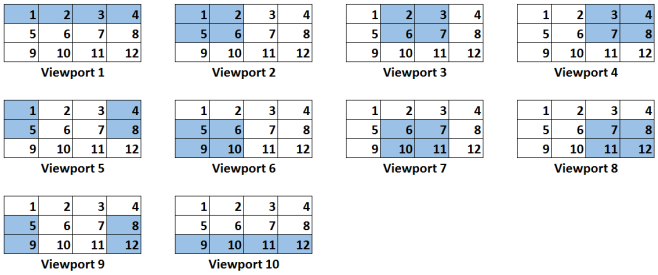


Fig. 7. Illustration of the considered viewports constructed from various tiles.

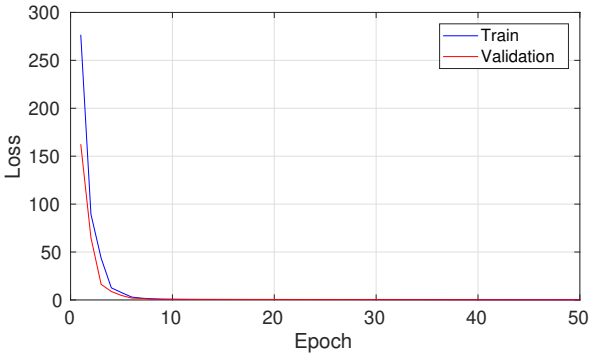


Fig. 8. Popularity estimation of a $360^o$ video, using LSTM network.

obtaining a tile at the enhancement quality layer is $\delta_{v,g,2,m} = 10$ dB.

We assume that the popularity of the $360^o$ videos through the GOPs is time-varying, and at any given moment, users may stop watching a $360^o$ video to view another one. To this aim, we assume that for the first GOP, the users' requests for the various $360^o$ videos follow the Zipfian distribution [35], with shape parameter parameter $\eta_v = 1$. Hence, the probability of a $360^o$ video to be selected is given by:

$$p_v = \frac{1/v^{\eta_v}}{\sum_{v \in \mathcal{V}} 1/v^{\eta_v}} \tag{2}$$

To capture the evolution of the popularity of the $360^o$ videos with the time, inspired by [36], we assume that the probability of a user to stop watching a $360^o$ video at the GOP $g \in \{2, \ldots, G\}$ follows the Weibull distribution. Specifically, we assume that each one of the $360^o$ videos that comprise our content library falls with equal probability to one of the 14 video categories, e.g., News, Sports, Education, etc., presented in [36]. Then, according to the video category each $360^o$ video falls into, we use the corresponding Weibull distribution parameters presented in [36], to estimate for each user, the probability of dropping a $360^o$ video at the GOP $g \in \{2, \ldots, G\}$. For each GOP, when users stop watching a $360^o$ video, the probability to select a different one to watch follows again the Zipfian distribution.

In terms of the viewports' requests, we simulated them with realistic navigation patterns obtained from the dataset described in [37]. To this aim, we initially sampled 10 different videos from this dataset. For each sampled video we obtained 30 trajectories. To assign these trajectories to the considered user requests, we mapped with uniform probability, the 100 videos that comprise the content library to one of the 10 sampled videos. Then, for each user request for a specific $360^o$ video, according to its mapped index, we assigned with uniform probability one of the 30 available trajectories for that video.

### B. LSTM Neural Network training

We consider a Deep LSTM network comprised of four layers. The input layer gets as input a 3D tensor with shape (samples, time-steps, 1). Each of the two hidden layers is an LSTM layer with 100 LSTM cells. The output layer is a Dense layer comprised by 1 unit. The Deep LSTM network is implemented with the open source library Keras in Python. The hidden layers have as activation function the ReLu function. The optimizer used is Adam. The LSTM network is initially pre-trained (warm-up phase) with 2000 samples of historic data profiles, as explained in Section IV. Each sample represents the evolution in the popularity (of a $360^o$ video or tile in high quality) over $h = 10$ consecutive time slots (time-steps). The LSTM is pre-trained with a batch size of 300 for 50 epochs, using the MSE loss function between the outputs of the LSTM and the actual popularities. The historic data profiles are split into a training set and validation set, where the training set accounts for the 90% of the historic data profiles, and the validation set the rest 10%. After the pre-training of the LSTM, the trained weights are used by the PF module for the forecasting of the future content popularities (online phase). Specifically, during the time slot $t$, for each content retrieved by the RFB module, the features regarding the $h - 1 = 9$ previous time slots along with the current time slot $t$ are provided as an input to the LSTM network, while its output is the predicted popularity at the time slot $t + 1$. During the online phase, the weights of the LSTM network are updated every 20 time slots by randomly sampling 100 samples from the FD module, and training the LSTM network with that samples for 20 epochs. As we can note from Fig. 8, the loss decreases with the number of epochs for both training and validation sets. We can also see that the loss function converges to zero which means that the LSTM network is able to predict the popularity of the content that is prefetched to the caches of the SBSs with a small error.

### C. Parameter Analysis

*1) Cache Size:* We first study the impact of the cache size on the overall quality of the rendered viewports. To this end, we vary the cache capacity $C_n$ in the range [5, 25]% of the size of the content library. As we can note from Fig. 9, the proposed scheme achieves a better performance in terms of the overall quality of the rendered viewports, for all cache sizes. Specifically, for small cache sizes, i.e., 5%, the performance gap between the proposed scheme and the LFU, LRU, and FIFO is approximately 1.4 dB, 1.8 dB, and 1.9 dB, respectively. For large cache size values, i.e., 25%, this gap increases to about 2 dB, 2.2 dB, and 2.3 dB, respectively. This performance gap is attributed to the fact that

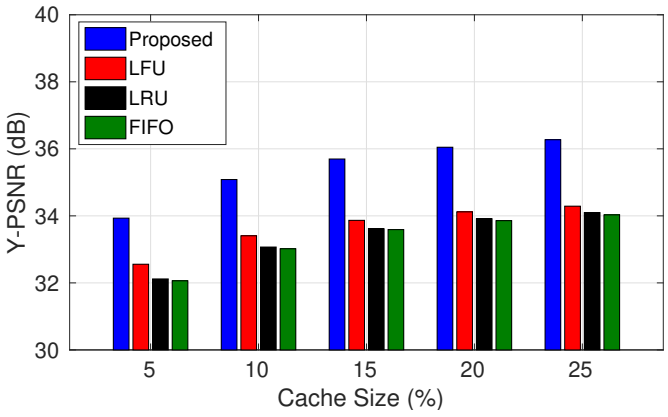


Fig. 9. Y-PSNR of the rendered viewports with respect to the cache size for all the schemes under comparison.

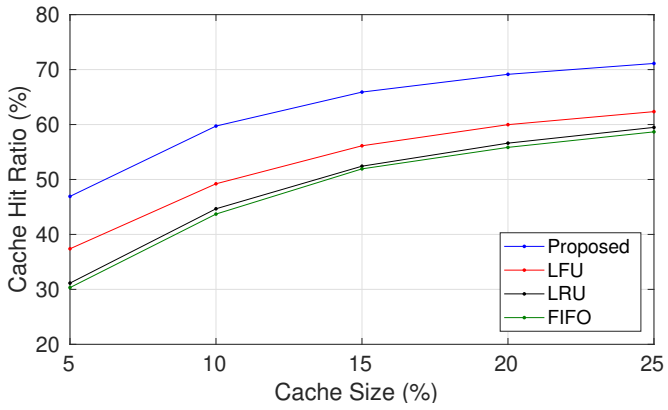


Fig. 10. Cache Hit Ratio with respect to the cache size for all the schemes under comparison.

the proposed scheme achieves a better cache hit ratio compared to its counterparts, as is evident from Fig. 10. Specifically, the performance gap between the proposed scheme and the LFU, which is the second best performing scheme in terms of the cache hit ratio, for small cache sizes (e.g., 5-10%) is about 10%. When the cache size takes large values, i.e., 25%, this gap closes to about 8.5% . This is because as the cache size increases, more content can be cached at the SBSs for all schemes. The increased cache hit ratio of the proposed scheme is attributed to the use of the LSTM network, which is able to predict the popularity of the content that will be prefetched to the caches of the SBSs, as described in Section IV. In addition, the proposed scheme exploits better the increased cache space which leads to an increased overall quality of the rendered viewports. This is because in contrast to the comparison schemes which cache for the popular videos only a viewport in high quality, the proposed scheme may cache a varied number of of tiles in high quality which depends on the popularity of both the videos and the viewports. Specifically, for the most popular $360^o$ videos, most (or all) of the tiles of the $360^o$ scene may be cached in high quality at the SBS. For the less popular $360^o$ videos, only a few (or none) of the tiles in high quality are cached at the SBS. This provides to the proposed scheme higher flexibility in terms of the caching decisions regarding the tiles that may be cached in high quality. As a result, more tiles will be delivered in high quality to the users from the caches of the SBSs at small delay. Thus, it is more likely for a greater number of tiles to be delivered in high quality in total from the caches of the SBSs along with the backhaul of the MBS to the users, under the tight end-to-end delivery constraint.

*2) Video popularity distribution:* In Fig. 11, we investigate the impact of the users' requests for the various $360^o$ videos on the overall quality of the rendered viewports. To this aim, we vary the Zipf shape parameter $\eta_v$ in the range $[0.8, 1.6]$. As we can see, an increase in the Zipf shape pareameter $\eta_v$ leads to an increase in the overall quality of the rendered viewports, for all the schemes. This is because as the Zipf shape parameter $\eta_v$ increases, the video popularity distribution gets steeper, and a smaller number of $360^o$ videos becomes more popular. As a result, the overall caching efficiency increases, as shown

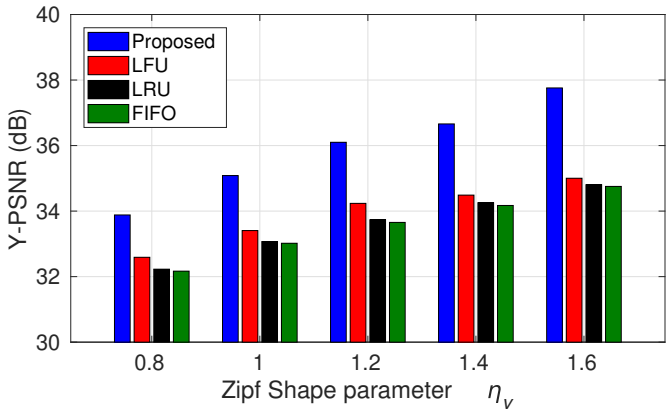


Fig. 11. Y-PSNR of the rendered viewports with respect to the Zipf shape parameter of the $360^o$ videos for all schemes under comparison.

in Fig. 12. Thus, more tiles will be served directly from the caches of the SBSs at a small delay, allowing more tiles to be served in total to the users under the end-to-end time constraint. Similarly with the previous comparison, the superiority of the proposed scheme regarding the overall cache hit ratio is attributed to the prediction of the popularities of the content that is prefetched to the caches of the SBSs, and the increased flexibility regarding the tiles that may be cached in high quality. In addition, due to the increased flexibility in terms of the caching decisions in the proposed scheme, the increase in the shape parameter $\eta_v$ is more beneficial for the proposed scheme, in terms of the overall quality of the rendered viewports. Speficially, as the shaper parameter raises from $0.8$ to $1.6$, the performance gap between the proposed scheme and the LFU raises from about $1.4$ dB to about $2$ dB. Similar observations can be made from the comparison of the proposed scheme with the LRU and FIFO schemes.

*3) Viewports popularity distribution:* To examine the impact of the viewports' popularity on the overall quality of the rendered viewports, we assume that the viewports popularity follows a Zipf distribution with shape parameter $\eta_p$. Considering that the cache capacity is $C_n = 10\%$, we vary the shape parameter $\eta_p$ in the range $[0.5, 2.5]$, and measure its impact on the overall quality of the rendered viewports. The performance of the under-comparison schemes is presented in Fig. 13. From

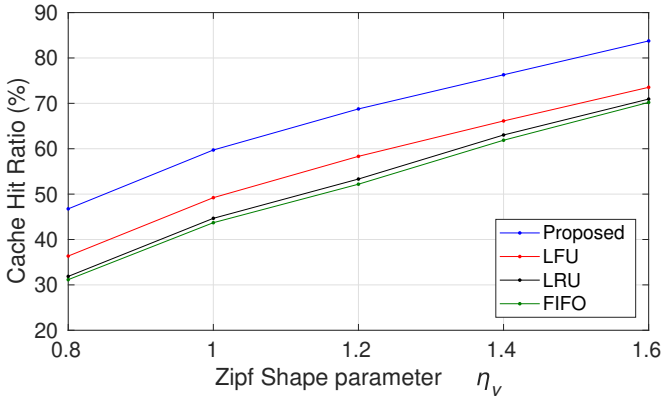


Fig. 12. Cache Hit Ratio with respect to the Zipf shape parameter of the $360^o$ videos for all schemes under comparison.

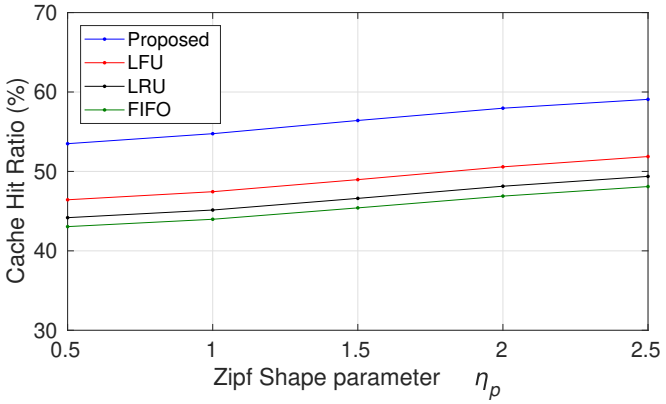


Fig. 14. Cache Hit Ratio with respect to the Zipf shape parameter of the viewports for all schemes under comparison.

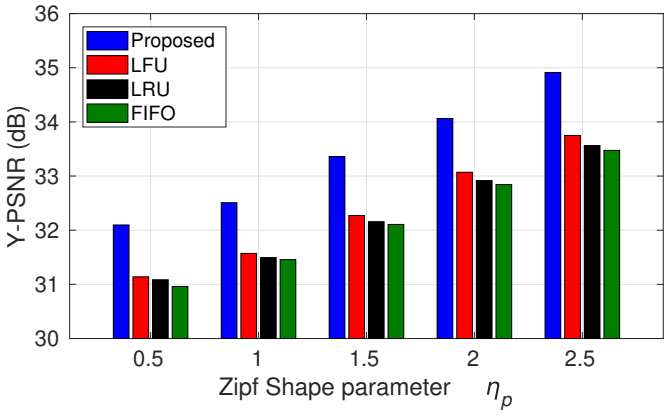


Fig. 13. Y-PSNR of the rendered viewports with respect to the Zipf shape parameter of the viewports for all schemes under comparison.

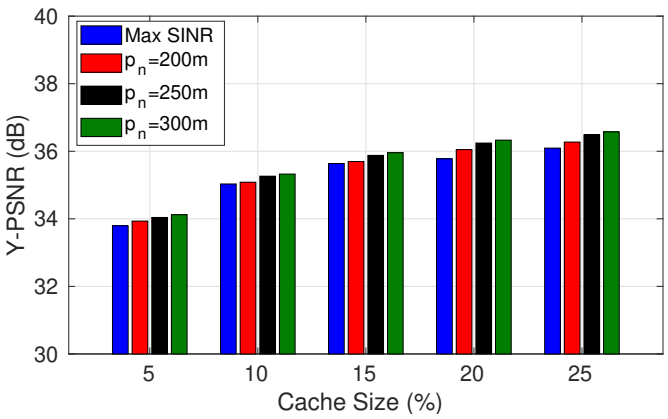


Fig. 15. Y-PSNR of the rendered viewports with respect to the cache size for all schemes under comparison.

this figure, we can note that an increase in the shape parameter $\eta_p$ leads to an increase in the overall quality of the rendered viewports for all the examined schemes. This is attributed to the fact that as the shape parameter $\eta_p$ increases, the requests for the viewports become less diverse, and a smaller number of viewports becomes more popular. Thus, the effectiveness of the cache is increased, as shown in Fig. 14. This results in more tiles in high quality to be served to the users directly from the caches of the SBSs at a small delay. Thus, more tiles in high quality will be served to the users in total under the end-to-end time constraint. For small values of $\eta_p$, i.e., $\eta_p = 0.5$, the performance gap between the proposed scheme and the LFU, LRU, and FIFO is about 0.9 dB, 1 dB, and 1.1 dB, respectively. Similarly, when the shape parameter $\eta_p$ becomes large, i.e., $\eta_p = 2.5$, the performance gap between the proposed scheme and the LFU, LRU, and FIFO becomes about 1.1 dB, 1.3 dB, and 1.4 dB, respectively.

*4) SBS Radius:* To understand the impact of users' association with multiple SBSs on the overall quality of the rendered viewports, we vary the SBSs radius $p_n$ in the range $[200, 300]$ m. As the radius of the SBSs increases, the overlap between the coverage area of the SBSs also increases. This results in more users being in the transmission range of multiple SBSs, and hence be able to be associated with multiple SBS. For the sake of completeness, we also examine the case where users located in the overlap of the coverage areas of the SBSs are assigned to the SBS with the maximum SINR. In that case, the increase of the transmission range of the SBSs from 200 m to 300 m does not affect the overall rendered quality of the viewports. This is because users are always assigned to the same SBSs, regardless of the increase in the transmission range of the SBSs. The simulation results are depicted in Fig. 15. As we can see, an increase in the radius of the SBSs leads to an increase in the overall quality of the rendered viewports due to the increase of the cache hit ratio achieved from the fact that more users will be associated with multple SBSs (see Fig. 16), for all cache sizes. As expected, when users are assigned to the SBS with the maximum SINR, the overall quality of the rendered viewports is lower compared to its counterparts. This is due to the fact that users are associated with only one SBS from which they can download their data.

*5) Backhaul Usage:* In Fig. 17, we examine the usage of the backhaul of the MBS with respect to the cache size of the SBSs, for all the schemes under comparison. Reducing the backhaul usage is of high importance, as this leads to the reduction of the network operating cost [38]. To this aim, we vary the cache capacity of the SBSs in the range $[5, 25]\%$ of the content library. As we can observe, an increase in the cache size of the SBSs leads to a decrease in the usage of the backhaul of the MBS for all schemes. This is because

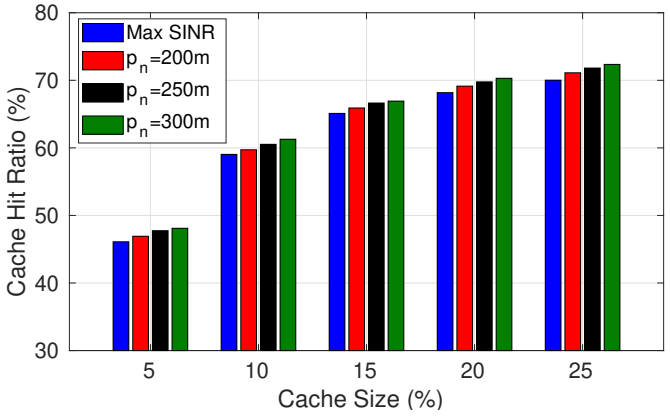


Fig. 16. Cache Hit Ratio with respect to the cache size for all schemes under comparison.

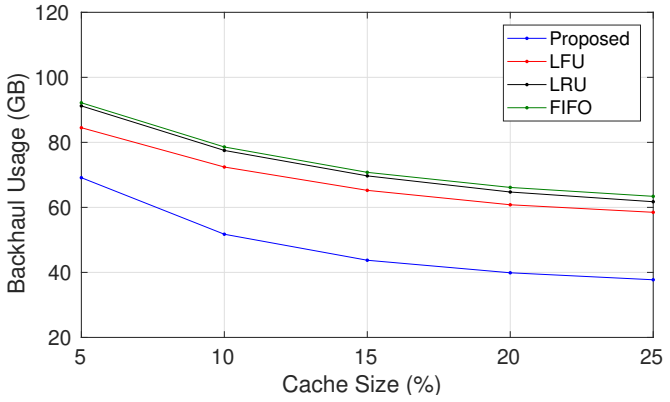


Fig. 17. Backhaul usage with respect to the Cache Size for all schemes under comparison.

as the cache size increases, more content will be cached at the SBSs. As a result, more user requests will be served directly from the caches of the SBSs without the need of the content to be fetched from the core network through the backhaul of the MBS. We can further note that as the cache size of the SBSs increases from 5% to 25%, the performance gap in terms of the backhaul usage between the proposed method and the LFU scheme raises from about 15 GB to about 23 GB. This is because the increased cache size of the SBSs is exploited better by the proposed scheme, as it takes advantage of the increased flexibility in deciding which should be cached in high quality. Similar conclusions can be drawn when comparing the proposed scheme with the LRU and FIFO schemes.

## VI. Conclusion

In this paper, we studied the problem of online caching to support live streaming of $360^o$ videos in mobile networks. The proposed framework used LSTM networks to predict the evolution of the popularity of video tiles in future GOPs. These estimates were used to update the cached content at the SBSs, and allowed prefetching of it to the users on time. In addition, it permitted the backhaul usage to be minimized, and the overall quality of the rendered videos to be increased. To further enhance the performance of our proposed method, we exploited the potential association of users with multiple SBSs. Hence, users located in the overlapping coverage areas of the SBSs had access to all the caches of these SBSs. We tested our scheme for both real and synthetic navigation patterns and compared it with the LFU, LRU, and FIFO schemes. From the results, it was showed that our method outperformed its counterparts significantly, in terms of the overall quality of the rendered viewports, the cache hit ratio and the backhaul usage. In addition, it made apparent the benefits of using LSTM networks to predict the evolution of the content popularity, and make the most from content prefetching and caching at the SBSs.

## References

[1] K. Bilal and A. Erbad, "Impact of multiple video representations in live streaming: A cost, bandwidth, and qoe analysis," in *Proc. of IEEE International Conference on Cloud Engineering (IC2E'17)*, Vancouver, BC, Canada, Apr. 2017, pp. 88–94.

[2] X. Xie and X. Zhang, "Poi360: Panoramic mobile video telephony over lte cellular networks," in *Proc. of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: ACM, 2017, pp. 336–349. [Online]. Available: http://doi.acm.org/10.1145/3143361.3143381

[3] A. TaghaviNasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, "Adaptive 360-degree video streaming using layered video coding," in *Proc. of IEEE Virtual Reality (VR'17)*, Los Angeles, CA, USA, Mar. 2017, pp. 347–348.

[4] A. Ghosh, V. Aggarwal, and F. Qian, "A rate adaptation algorithm for tile-based 360-degree video streaming," *CoRR*, vol. abs/1704.08215, 2017. [Online]. Available: http://arxiv.org/abs/1704.08215

[5] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. of IEEE International Conference on Communications (ICC'17)*, Paris, France, May 2017, pp. 1–7.

[6] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "360 video viewing dataset in head-mounted virtual reality," in *Proc. of the 8th ACM on Multimedia Systems Conference (MMSys'17)*, Taipei, Taiwan, 2017, pp. 211–216.

[7] S. Rossi and L. Toni, "Navigation-aware adaptive streaming strategies for omnidirectional video," in *Proc. of IEEE 19th International Workshop on Multimedia Signal Processing (MMSP'17)*, Luton, UK, Oct. 2017, pp. 1–6.

[8] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Hevc-compliant tile-based streaming of panoramic video for virtual reality applications," in *Proc. of the 2016 ACM on Multimedia Conference*, Amsterdam, Netherlands, 2016, pp. 601–605.

[9] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[10] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai, "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*, vol. 9, no. 1, pp. 43–57, March 2019.

[11] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Edge-caching wireless networks: Performance analysis and optimization," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2827–2839, Apr. 2018.

[12] J. Poderys, M. Artuso, C. M. O. Lensbøl, H. L. Christiansen, and J. Soler, "Caching at the mobile edge: A practical implementation," *IEEE Access*, vol. 6, pp. 8630–8637, 2018.

[13] P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of $360^o$ videos in heterogeneous networks," in *Proc. of IEEE 21st Int. Workshop on Multimedia Signal Processing (MMSP'19)*, Kuala Lumpur, Malaysia, Malaysia, Sep. 2019.

[14] ——, "Tile-based joint caching and delivery of $360^o$ videos in heterogeneous networks," *IEEE Trans. on Multimedia*, in press.

[15] 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*, ser. Mobihoc '19. New York, NY, USA: ACM, 2019, pp. 171–180. [Online]. Available: http://doi.acm.org/10.1145/3323679.3326515

[16] J. Brownlee, "How to develop lstm models for time series forecasting," 2020. [Online]. Available: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting

[17] H. Huang, J. Chen, H. Xue, Y. Huang, and T. Zhao, "Time-variant visual attention in 360-degree video playback," in *Proc. of IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE'18)*, Dalian, China, Sep. 2018, pp. 1–5.

[18] L. Bassbouss, S. Pham, and S. Steglich, "Streaming and playback of 16k 360 videos on the web," in *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, Jounieh, Lebanon, April 2018, pp. 1–5.

[19] Y. Kim, J. Huh, and J. Jeong, "Distributed video transcoding system for 8k 360 vr tiled streaming service," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, South Korea, Oct 2018, pp. 592–595.

[20] J. Heyse, M. T. Vega, F. de Backere, and F. de Turck, "Contextual bandit learning-based viewport prediction for 360 video," in *Proc. of IEEE Conference on Virtual Reality and 3D User Interfaces (VR'19)*, Osaka, Japan, Japan, March 2019, pp. 972–973.

[21] S. Petrangeli, G. Simon, and V. Swaminathan, "Trajectory-based viewport prediction for 360-degree virtual reality videos," in *Proc. of IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR'18)*, Taichung, Taiwan, Taiwan, Dec. 2018, pp. 157–160.

[22] M. Jeppsson, H. Espeland, C. Griwodz, T. Kupka, R. Langseth, A. Petlund, P. Qiaoqiao, C. Xue, K. Pogorelov, M. Riegler, D. Johansen, and P. Halvorsen, "Efficient live and on-demand tiled hevc 360 vr video streaming," in *Proc. of IEEE International Symposium on Multimedia (ISM'18)*, Taichung, Taiwan, Dec. 2018, pp. 81–88.

[23] R. Aksu, J. Chakareski, and V. Swaminathan, "Viewport-driven rate-distortion optimized scalable live 360 video network multicast," in *Proc. of IEEE International Conference on Multimedia Expo Workshops (ICMEW'18)*, San Diego, CA, USA, Jul. 2018, pp. 1–6.

[24] Y. Hu, S. Xie, Y. Xu, and J. Sun, "Dynamic vr live streaming over mmt," in *Proc. of IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB'17)*, Cagliari, Italy, Jun. 2017, pp. 1–4.

[25] X. Liu, B. Han, F. Qian, and M. Varvello, "Lime: Understanding commercial 360° live video streaming services," in *Proc. of the 10th ACM Multimedia Systems Conference*, ser. MMSys '19. New York, NY, USA: ACM, 2019, pp. 154–164. [Online]. Available: http://doi.acm.org/10.1145/3304109.3306220

[26] C. Ge, N. Wang, W. K. Chai, and H. Hellwagner, "Qoe-assured 4k http live streaming via transient segment holding at mobile edge," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1816–1830, Aug. 2018.

[27] P. Maniotis and N. Thomos, "Viewport-aware deep reinforcement learning approach for 360 video caching," *ArXiv*, 2020.

[28] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, Sept. 2016.

[29] X. Lei, X. Jiang, and C. Wang, "Design and implementation of streaming media processing software based on rtmp," in *Proc. of 5th Int. Congress on Image and Signal Processing*, Chongqing, China, Oct. 2012, pp. 192–196.

[30] [Online]. Available: https://engineering.fb.com/ios/under-the-hood-broadcasting-live-video-to-millions/

[31] T. Ergen and S. S. Kozat, "Online training of lstm networks in distributed systems for variable length data sequences," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 5159–5165, Oct 2018.

[32] N. M. Vural, S. Ergut, and S. S. Kozat, "An efficient and effective second-order training algorithm for lstm-based adaptive learning," 2019.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[34] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "Making content caching policies "smart" using the deepcache framework," *SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 5, p. 64–69, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3310165.3310174

[35] L. Alexander, R. Johnson, and J. Weiss, "Exploring zipf's law," *Teaching Mathematics and Its Applications: International Journal of the IMA*, vol. 17, no. 4, pp. 155–158, Dec 1998.

[36] E. Baccour, A. Erbad, A. Mohamed, K. Bilal, and M. Guizani, "Proactive video chunks caching and processing for latency and cost minimization in edge networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'19)*, Marrakesh, Morocco, Morocco, 2019, pp. 1–7.

[37] F. Duanmu, Y. Mao, S. Liu, S. Srinivasan, and Y. Wang, "A subjective study of viewer navigation behaviors when watching 360-degree videos on computers," in *Proc. of IEEE International Conference on Multimedia and Expo (ICME'18)*, San Diego, CA, USA, July 2018, pp. 1–6.

[38] B. Perabathini, E. Baştuğ, M. Kountouris, M. Debbah, and A. Conte, "Caching at the edge: A green perspective for 5g networks," in *Proc. of IEEE International Conference on Communication Workshop (ICCW'15)*, London, UK, Jun. 2015, pp. 2830–2835.