Tile-based Caching Optimization for 360° Videos

Georgios Papaioannou
Athens University of Economics and Business
Department of Informatics
Athens, Greece
gepap@aueb.gr

Iordanis Koutsopoulos
Athens University of Economics and Business
Department of Informatics
Athens, Greece
jordan@aueb.gr

ABSTRACT

Panoramic or 360° video streaming and playback offer an immersive viewing experience and become increasingly popular, albeit very resource-demanding. Caching of 360° video content close to the user reduces content delivery delay and bandwidth consumption. Contrary to monolithic single-stream transmission, the recently proposed tiling approach allows the streaming of different parts (tiles) of the 360° content with different resolutions.

We study the problem of optimal caching of 360° video streams, where the problem is to decide which tiles and tile resolutions to cache. The difference from conventional video is that each tile may need to be cached with multiple resolutions since it may appear in different positions in different viewports, and the proportions of these positions are dictated by viewing statistics through e.g. Head-Mounted Display units. This observation leads to a novel caching objective: the cached resolutions for each tile should be *as close as possible* to the required ones for each tile when viewed by the user. We study the cases of caching tile streams either at different resolutions or in a layered encoding fashion. We seek to optimize an objective that combines (*i*) an error metric between requested and cached tile resolutions across different viewports, and (*ii*) coverage of the tile set. For the case of multiple resolutions, we show that the problem of selection of tile resolutions to cache so as to minimize the error metric above is equivalent to the K-Medoids problem. For layered encoding, the problem is to find the maximum-resolution layer to cache for each tile stream, and we show that this is equivalent to a Multiple-Choice Knapsack problem. We present an implementation of the cache optimization scheme, evaluate our model on a tiled 360° video distribution simulator and demonstrate a significant increase in cache hit ratio over state-of-the-art caching strategies.

KEYWORDS

Video caching, 360° video, video encoding, optimization.

ACM Reference Format:
Georgios Papaioannou and Iordanis Koutsopoulos. 2019. Tile-based Caching Optimization for 360° Videos. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2019).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.475/123_4

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MobiHoc 2019, July 2019, Catania, Italy
© 2019 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

1 INTRODUCTION

The latest development in video technology is 360° (or *spherical* or *panoramic*) video streaming. The 360° video is projected to end-users either through Head-Mounted Displays (HMDs) or through smartphones and personal computers, usually as a panoramic movie and less frequently as part of a mixed-reality experience[1]. 360° video is recorded through specialized omni-directional cameras or through a certain arrangement of multiple cameras, where each camera records an angle of the action, and the combined streams lead to a reconstructed 360° spherical panorama. Next, the spherical image domain is divided into sub-domains, the *tiles*. Each tile is a separate video stream which is encoded and decoded independently from others. The collection of tile streams is the entire 360° video. Each tile stream may consist of multiple temporal video segments of certain duration.

The currently active field of view (FoV) or *viewport* of a user i.e. the part of the spherical domain that is mapped to the display's projection to the user is determined dynamically, through the tracked head motion of an HMD or another orientation-defining user interface mechanism of a desktop or a mobile platform (see also illustration in Fig. 1). The subset of tiles for the current video segment that are intersected by the active FoV, i.e. the current *frame*, are streamed to the client. A change in the viewing direction naturally warrants an update to the tile set.

[1]We avoid the term *virtual reality* (VR) as by definition it involves interaction with client-side *real-time* image synthesis and is not relevant to the context of (static) video streaming and caching.

Panoramic video requires substantially more bandwidth than conventional video, primarily due to the high aggregate resolution of tiles that comprise the displayed frame. For example, HMDs easily demand resolutions higher than 2MPixels per eye pair for comfort and immersive viewing [9]. Recent 5G wireless transmission technologies attempt to fill the gap between provisioned bandwidth and required rate and latency in video delivery. However, the projected massive adoption of 360° video applications will dramatically increase the bandwidth demand both on the wireless edge where video delivery to users takes place and at the back-end/backhaul network, while the requirements for short round-trip times and optimal Quality of Experience (QoE) will be challenged even more [1].

Tiling facilitates bandwidth reduction, since the parts of the panorama outside the FoV need not be transmitted. Tiling also makes *foveated* display possible and efficient. Foveated display is a quality (and bandwidth) reduction approach that does not affect perceived visual clarity, due to the diminished capability of the human visual system to discern details as it moves away from the central gaze axis toward the periphery of the eyes' field of view. We exploit this to further improve our caching efficiency, as detailed in the sequel.

Caching of conventional video content has been extensively researched in recent years, starting from the seminal paper [7]. The idea is that popular content is prefetched at points at the network edge such as base stations, small cells or user devices, from where it can be retrieved faster when requested by users. Typical performance goals of caching are low average content delivery delay and low bandwidth consumption at the back-end where the entire video catalogue resides; these follow from increased cache hit ratio. Migrating to 360° video, caching content at HMD devices or small cells that cover video demand locations is expected to result in small content delivery time and significant bandwidth reduction at the back-end.

Caching of 360° video is fundamentally different from caching conventional video. First, assuming a human perception driven bandwidth (and quality) allocation as the tile distance from the gaze direction changes, different tile streams come with different resolution requirements. For example, a tile near the gaze direction (i.e. near the *fovea*) is requested at a high resolution, while the *same* tile, located close to the edge of another viewport, can be encoded with lower resolution. This implies that the same tile steam needs to be cached in several different resolutions or that layered encoding should cater for these different resolution requirements. When caching conventional video content, maximizing the cache hit ratio is a natural objective that in turn minimizes average delivery delay. In 360° video though, the user QoE goal dictates a novel objective. The cached tiles and their cached resolutions or layers should be as close as possible to the required ones for each tile when viewed by the user. This mismatch between required and cached resolutions should be expressed through an error metric that captures either both over/under-provisioning of cached resolutions or just under-provisioning, where the intuition is that having cached resolutions higher than the required ones should be avoided due to unnecessarily high consumption of transmission bandwidth, and in any case cached resolutions must suffice to cover the required ones as much as possible.

1.1 Our contribution

We cast the problem of optimal caching of 360° video streams that need to be prefetched to caches before actual video streaming to users takes place. The dynamics of user transitions between different viewports are found through historical data and give rise to the statistics of tile occurrences and corresponding needed resolutions. Each tile comes with different resolution requirements, i.e. different resolution levels, and with different frequencies of occurrence of each level, depending on viewport statistics and the relative tile positions in different viewports at different times. We are interested in caching tile resolutions that are as close as possible to the required ones for video streams.

To the best of our knowledge, caching has not been considered for 360° video. Multiple resolutions arise also in Scalable Video Coding (SVC), albeit not in the context of 360° video and the caching objective we consider here. The problem we address with the caching objective and its solution are novel and pave the way for several interesting directions. The specific contributions of our work are as follows.

- We formulate the optimization problem of caching 360° videos, which entails the decision on which tiles and tile resolutions to cache. We seek to minimize an objective that combines (*i*) an error metric between requested and cached tile resolutions across different viewports, and (*ii*) coverage of the tile set.
- For one tile stream and for squared error metric, we show that the problem of deciding which resolutions to cache so as to minimize the error metric above is equivalent to the K-Medoids problem that arises in data clustering, and thus it is NP-Hard.
- We formulate the same problem for the case of layered encoding, where the question is to find the maximum layer to cache for each tile stream. We show that this problem is equivalent to a Multiple-Choice Knapsack one, and thus it is also NP-Hard.
- We implement a complete simulation of processes and delay lines involved in caching, transmission and reproduction of 360° video content at the clients, including a stack of realistic behavioral models for the viewport change and video playback control. We introduce

perceptual criteria during quality optimization at the cache, inspired by foveated real-time rendering.
- We demonstrate a significant increase in cache hit ratio for our tile-based caching optimization scheme over state-of-the-art caching strategies.

2 RELATED WORK

2.1 Wireless video caching and encoding

Wireless video caching is at the forefront of research in caching in the last few years, following the seminal paper [7], and it is a technology of great potential for involved stakeholders [20]. A recent comprehensive survey on wireless caching with a view towards future research directions appeared in [21]. In a typical network architecture, users are served by multiple caches, and each cache is attached to a small cell. In [25], the authors study the joint problem of cache content retention and user-cache association so as to minimize a metric that captures a content storage cost that is convex on retention time, and an access cost for content unicast and multicast. The authors in [24] study joint video caching and routing in heterogeneous multi-cache networks, with the aim to reduce a cost term that includes delivery delay and network expenditure. In [5], the authors study the joint caching and recommendation problem, where the latter is used as means to control content demand. The work [14] considers caching at user devices and point-to-point content transmission and derives a scaling law of throughput that is agnostic to the number of users. The joint problem of routing and caching in arbitrary network graph topologies is studied in [13], with the goal to minimize transfer cost over the path for different routing regimes.

Another line of research in caching is the consideration of different video encoding options to cache. The fundamental dilemma in [10] is whether to cache different versions of each video file, where each version has different encoding rate, or to perform what is called *layered* video coding. In the latter, each video file is encoded into a base layer and some enhancement layers. The base layer contains the most essential information, and enhancement layers provide quality enhancements. A layer is decoded only if all lower-quality layers are available. The paper concludes that a mixed caching strategy of video file versions and layered coding provides the best performance in terms of long-run throughput i.e. rate at which user requests are satisfied.

Layered video caching in small cells (caches) that serve users is studied in [23] with the aim to minimize average retrieval delay. Scalable Video Coding (SVC) is used, whereby a video file is encoded in layers, and a certain quality video requires the delivery of multiple layers, from the basic one up to the one necessary to achieve that quality. The novel challenge is that video retrieval delay depends on the largest delivery delay for a required layer. A different architecture is considered in [6], where devices cache files at different qualities, and a central base station determines quality allocation to caches. The different devices may associate with nearby devices to obtain the video at the required quality. A stochastic optimization framework is employed, with the aim to maximize total received video quality.

2.2 360° video transmission

360° video delivery technology has emerged in the last couple of years. Head motion prediction may discern the direction (viewport) that the user looks at, and it renders to the user only the portion of video that is relevant to the viewport [2]. In a multi-user setting, the work of Bao et al. [3] performs motion prediction and then selects multicast or multiple unicasts to conserve bandwidth. Ozcinar et al. [19] present an enhancement to the MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard for varying the bitrate and quality over the set of tiles of the 360° video stream. The bitrate of the inner tiles of the viewport is higher, while that of outer tiles of the viewport is gradually reduced. Each tile is weighted according to the number of common pixels between that tile and all tiles in the viewport. Similar in spirit, Hosseini and Swaminathan [12] consider tile prioritization and rate allocation to tiles subject to Knapsack-like constraints that stand for limited link bandwidth. In [4], the authors study rate allocation across tiles so as to minimize the expected tile reconstruction error subject to link bandwidth capacity constraints, where the expectation is taken over the likelihood that a certain tile is requested. Further, the work of Zhou and Liu [26] studies a tiling solution that minimizes required bandwidth for streaming 360° videos.

An implementation of an architecture that involves tiling and adaptive tile-based streaming in order to conserve bandwidth is the topic of [8], while the holistic architecture in [18] involves big-data-enabled head motion prediction, multipath transmission and tile rate adaptation using SVC encoding that caters for motion prediction error. Dynamic caching has been considered in the context of multi-view 3D videos [16]. The novel modeling assumption here is that a video view can be reconstructed even if it does not reside in the cache, provided that some other cached views aid in reconstruction of the original view. The cache replacement problem so as to minimize view fetching and reconstruction costs is modeled as a Markov Decision Process.

Different from the existing literature, our work delves into the use of caching for 360° video transmission by considering either multiple cached resolutions for tile streams or layered coding for them. The formulated novel caching optimization problem combines accurate rate representation for requested tile streams, and tile coverage.

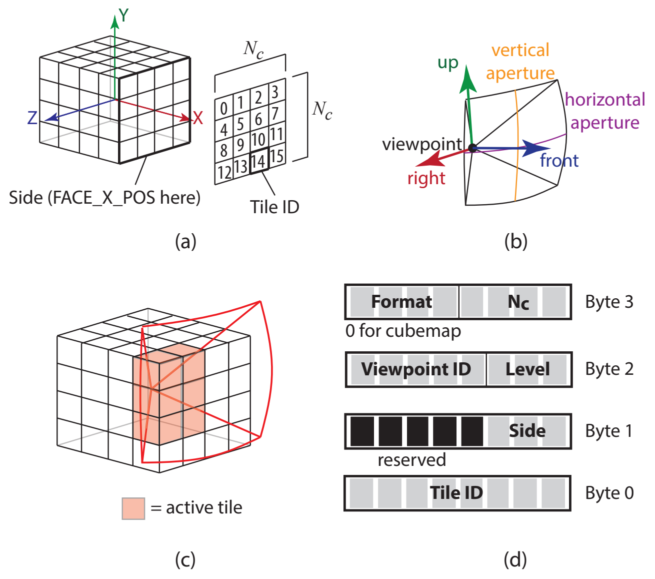

Figure 1: Panoramic (360°) video specification. (a) The spherical domain is split into rectangular tiles on the six sides of a cube. (b) The client's view frustum has its own reference frame and a device-specific horizontal and vertical aperture. (c) For a given video frame, the client requests those tiles that intersect the view frustum. (d) The tile identifier (tile code).

2.3 Perception-based quality allocation

Studies on peripheral vision reveal that a steady perceptual response requires increasing stimuli as eccentricity increases. This important *cortical magnification factor* M is strongly correlated to the neural volume at a particular area in our FoV. Factor M is an inverse function of distance, although inverse nonlinear functions have also been used. The notion of degrading-quality image generation as a function of retinal eccentricity was exploited early in the 90's to radially vary the sampling rate and image reconstruction filter kernel size for interactive volume rendering [17]. Guenter et al. [9] established the appropriate metrics and blending functions for *foveated rendering*, i.e. for smooth degradation of the rendered image across the FoV in order to accelerate real-time image synthesis without perceivable loss of quality. A comprehensive and recent practical study of the factors affecting the perception of a displayed image appears in [22]. We use a similar factor in our cache optimization model to assign an importance to each visible tile in a viewport, thus taking into account foveated display of 360° video content.

3 MODEL

We consider a single 360° video stream of certain duration, which is divided into a set $\mathcal{S}$ of temporal segments. The stream is spatially separated into a set $\mathcal{T}$ of $6N_c^2$ tiles, so that each video frame corresponds to a *cubemap*. Please refer to Fig. 1-(a),(c) for the definition of N_c and of a cubemap. For each video segment s, each tile i constitutes a video stream with the segment duration. For simplicity, we assume that the index i of a tile captures the number of the side of the cubemap and the tile number in the specific side. Each tile is encoded, cached and transmitted separately to the client. Although for clarity of presentation in the figure we use non-overlapping tiles, in a realistic scenario, tiles would form overlapping zones for smooth blending of the differently encoded image tiles.

Let p_i^s be the occurrence probability of tile i of segment s in the cubemap; this is the probability that tile i is requested by a user for segment s. For each segment s, it is $\sum_{i\in\mathcal{T}} p_i^s = 1$. Since arbitrary viewports are generated due to the multiple degrees of freedom and a continuous head motion domain, especially for head-tracked devices, we may directly measure and gather statistics $\{p_i^s\}$, for $i \in \mathcal{T}$ and $s \in \mathcal{S}$ for the tiles of the cubemap, and we do not need to do so for the viewport directions that generate these statistics (Fig. 2). This flexibility allows the use of arbitrary viewing angles and of viewing equipment with significantly different viewport parameters, such as aspect ratio, aperture or even shape.

For simplicity in the sequel we assume that the video stream has only one segment, hence we drop index s and define the probability p_i that tile i is requested, so that $\sum_{i\in\mathcal{T}} p_i = 1$. Multiple different segments may be treated without loss of generality as separate videos, whose popularity is determined by the joint probability of selecting the particular video file and of deciding to play the current segment, given the popularity of the current time interval in the video.

Depending on the position of a tile i in a viewport v, there exists a certain acceptable video resolution or encoding rate (level) with which the tile should be projected to the user. For example, if tile i is close to the axis of projection of viewport v to the user (or the gaze direction, if eye tracking is utilized), then it has to be encoded with higher resolution compared to a tile at the periphery of the FoV. Let q_{iv} be the desirable resolution associated with tile i when at viewport v. Note that q_{iv} may emerge through a continuous tile *importance* function in viewport v, $h_{i,v}(\cdot)$ and takes values in a discrete set of quality levels $\mathcal{R} = \{q_1, \dots, q_R\}$, with q_R representing the highest (reference) quality level. Let p_{ir} denote the conditional probability that level q_r is requested given that tile i is requested. This is in effect the occurrence rate of quality level q_r recorded on the cache for tile i.

We consider two modes for video encoding. In the first one, which we refer to as the *multiple-versions* one, several different versions of a tile can be maintained in the cache, each with a different resolution (encoding level) q_r. Each level q_r of a video tile i for the given video segment is associated

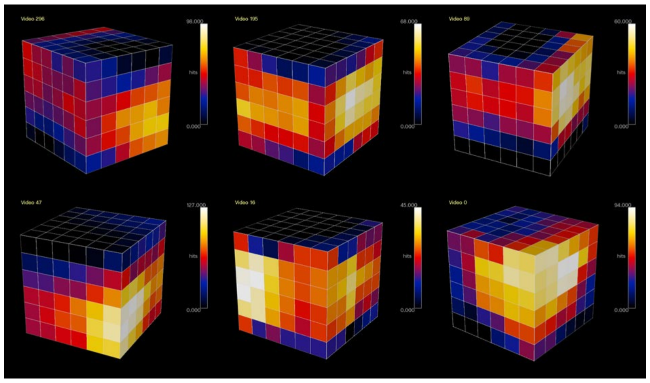

Figure 2: Indicative tile request statistics gathered by the cache for 6 different videos. They are used during optimization to estimate p_i. The above are screenshots from the simulator video statistics pane.

with caching space requirement β_{ir}, with $r \in \{1, \ldots, R\}$ so that β_{ir} is increasing in r, i.e. $\beta_{i1} \leq \beta_{i2} \leq \ldots \leq \beta_{iR}$.

In the second mode, the *layered-encoding* one, a set of layers $\mathcal{L} = \{\ell_1, \ldots, \ell_R\}$ exists. Quality level q_1 is guaranteed by layer ℓ_1 itself, quality level q_2 is achieved through layers ℓ_1 and ℓ_2 together, and so on. The highest quality level q_R is guaranteed through all layers $\ell_1, \ell_2, \ldots, \ell_R$ together. For tile i and layer ℓ, let $\gamma_{i\ell}$ denote the caching space requirements, with $\ell \in \mathcal{L}$. Note that it is $\gamma_{i\ell_1} \geq \gamma_{i\ell_2} \ldots \geq \gamma_{i\ell_R}$.

4 PROBLEM STATEMENT

Given a limited cache capacity C, we are interested in finding a tile and tile resolution caching policy that are optimal in the sense outlined below.

For the multiple-versions case, the tradeoff in caching one or more resolutions (encoding rates) for a tile is as follows. Caching more resolutions q_r for a tile i in general translates to better match between the requested and the cached resolutions for the tile. Recall that a tile may require different resolutions, depending on its importance $h_{i,v}(\cdot)$ in different viewports v, which in turn depends on its position in different viewports. For a tile that appears in different positions in several frequently arising viewports, we would need to cache several resolutions, so that it is delivered to the user with the appropriate resolution for each viewport. However, caching multiple resolutions for a given tile limits the residual caching capacity for other tiles. On the other hand, certain tiles may require fewer or even only one resolution. These are tiles that appear at about the same position across the different viewports they are part of, or tiles that are requested with a significant preference for a particular position/importance $h_{i,v}(\cdot)$. In these cases, it would make sense to conserve cache capacity by caching fewer or even one resolution for the tile.

A similar tradeoff arises for the case of layered encoding in caching multiple consecutive layers for tile streams. Again, a certain tile may be requested with different resolutions. We seek to determine the maximum-quality layer to place in the cache (together with lower-quality layers) for each tile stream so that tile requests targeting this maximum-quality resolution or lower-quality resolutions are satisfied through the layers present in the cache.

There exist fundamental differences in caching 360° videos compared to caching conventional videos. First, the tiling approach makes sense only for 360° videos. Furthermore, for each tile, the cached resolutions should be as close as possible on average to the resolutions needed for each tile, according to the tile presence in different viewports and viewport statistics. We attempt to capture this requirement in the sequel through different metrics for the multiple-versions and for the layered-encoding case.

4.1 Multiple-versions case

For each tile, we need to decide which resolution (encoding rate, or version) to cache. For each tile i and resolution $q_r \in \mathcal{R}$, let $x_{ir} = 1$ if we cache the r-th resolution for that tile, and 0 otherwise. Let $\mathbf{x} = (x_{ir} : i \in \mathcal{T}, r \in \mathcal{R})$ be the caching policy.

In order to capture the distance between a required resolution q_r and the corresponding cached resolution $q_{\tilde{r}}$, we define a normalized squared error metric,

$$\operatorname{dist}(q_r, q_{\tilde{r}}) = \frac{1}{D}(q_r - q_{\tilde{r}})^2\,, \tag{1}$$

where $D = \max_{r,r' \in \mathcal{R}} (q_r - q_{r'})^2$ is a normalization factor that denotes the maximum distance between two resolutions. According to this metric, not having a high enough resolution in the cache so as to support a required resolution incurs an error that negatively affects user QoE. On the other hand, using a cached resolution that is higher than the required one is not desirable as well, since it causes unnecessary bandwidth consumption during transmission.

Fix attention to a single tile i and assume that a subset $\mathcal{R}_i \subseteq \mathcal{R}$ of K_i resolutions need to be cached for that tile. Assume for now that the number of resolutions, K_i is given. A required resolution will be supported through the closest cached resolution. We are interested in finding the subset $\mathcal{R}_i$ of resolutions to cache so as to solve

$$\min_{\mathcal{R}_i} \sum_{r \in \mathcal{R}} p_{ir} \min_{j \in \mathcal{R}_i} \operatorname{dist}(q_r, q_j)\,. \tag{2}$$

For a single tile and for fixed K_i, the optimization problem above is equivalent to the K-Medoids clustering one, with $K = K_i$, and therefore it is NP-Hard. The K-Medoids problem seeks to select K cluster centers among a set of data points so as to minimize the distance between data points assigned to a cluster and the point designated as the center of that cluster. In our case, the set of data points is the set of possible resolutions, the chosen cluster centers are the cached resolutions, and the assignment of a data point to a cluster corresponds to the representation of a requested resolution by the cached one that is closest to it.

Taking all tiles into account, the problem is clearly more difficult. Our aim is to find for each tile i, the sizes K_i and the elements of the subsets $\mathcal{R}_i$ of cached resolutions so as to minimize the average squared error metric, that is

$$\min_{\mathbf{x}} Q(\mathbf{x}) = \min_{\mathcal{R}_1,\dots,\mathcal{R}_n} \frac{1}{n} \sum_{i\in\mathcal{T}} \sum_{r\in\mathcal{R}} p_{ir} \min_{j\in\mathcal{R}_i} \text{dist}(q_r, q_j)\,, \tag{3}$$

or equivalently,

$$\min_{\mathbf{x}} Q(\mathbf{x}) = \frac{1}{n} \sum_{i\in\mathcal{T}} \min_{\mathcal{R}_i} \sum_{r\in\mathcal{R}} p_{ir} \min_{j\in\mathcal{R}_i} \text{dist}(q_r, q_j)\,, \tag{4}$$

where finding the caching policy $\mathbf{x}$ is equivalent to finding the subsets $\mathcal{R}_i$ of resolutions (versions) to cache for each tile i, since it is $x_{ir} = 1$ if and only if $r \in \mathcal{R}_i$, and 0 otherwise. Further, it is $n = R|\mathcal{T}|$, and $0 \le Q(\mathbf{x}) \le 1$. The following cache capacity constraint needs to be satisfied:

$$\sum_{i\in\mathcal{T}} \sum_{r\in\mathcal{R}} \beta_{ir} x_{ir} = C\,, \tag{5}$$

and further, it should be

$$0 \le \sum_{r\in\mathcal{R}} x_{ir} \le R \text{ , for each tile } i\,. \tag{6}$$

This constraint says that it is $0 \le K_i \le R$ for each tile i, and $K_i = 0$ corresponds to not caching tile i at all.

4.2 Layered-encoding case

In the case of layered encoding, we need to decide on the highest layer to cache for each tile, together with all lower-quality layers. For each tile i and each layer $\ell \in \mathcal{R}$, let $x^L_{i\ell} = 1$ if we cache layers $\ell, \ell-1, \dots, 1$ for tile i, and $x^L_{i\ell} = 0$ otherwise. Let $\mathbf{x}^L = (x^L_{i\ell} : i \in \mathcal{T}, \ell \in \mathcal{R})$ be the caching policy for the layered-encoding case. We define the following normalized error between a required resolution (encoding rate) q_r and the resolution q_ℓ that can be provisioned by the cache if cached layers are $\ell, \ell-1, \dots, 1$:

$$\text{dist}_L(q_r, q_\ell) = \frac{1}{D}(q_r - q_\ell)^2_+\,, \tag{7}$$

where $y_+ = y$, if $y > 0$ and 0 otherwise. Compared to (1), this error metric is different; here it matters only when the cached layers are not enough to provision the required resolution q_r. On the other hand, because a cached tile layer of higher quality can always contribute layers *up to* that quality due to its layered structure, having layers in the cache such that the provisioned resolution is higher than the required one is not an issue in terms of the error metric.

For a single tile i, the most appropriate highest layer ℓ^*_i to cache is

$$\ell^*_i = \arg\min_{\ell\in\mathcal{R}} \sum_{r\in\mathcal{R}} p_{ir} \text{dist}_L(q_r, q_\ell)\,. \tag{8}$$

Taking all tiles into account, the optimization of the average squared error metric now is written as

$$\min_{\mathbf{x}^L} Q(\mathbf{x}^L) = \frac{1}{n} \sum_{i\in\mathcal{T}} \min_{\ell_i\in\mathcal{R}} \sum_{r\in\mathcal{R}} p_{ir} \text{dist}_L(q_r, q_{\ell_i})\,, \tag{9}$$

where the caching policy $\mathbf{x}^L$ is such that $x^L_{i\ell_i} = 1$ when the highest layer to cache for tile i is ℓ_i. Further, the cache capacity constraint to satisfy is

$$\sum_{i\in\mathcal{T}} \sum_{\ell_i\in\mathcal{R}} \left(\sum_{r=1}^{\ell_i} \gamma_{ir} \right) x^L_{i\ell_i} = C\,, \tag{10}$$

and also it should be

$$\sum_{\ell\in\mathcal{R}} x^L_{i\ell} \le 1, \quad \text{for each tile } i\,. \tag{11}$$

The formulation above is reminiscent of the Multiple-Choice Knapsack problem (MCKP) [15], if we make the following observations: the set of items to pick from in the MCKP problem are the layers of all tiles. A class of items in MCKP corresponds to the set of layers of a single tile. In MCKP, we need to select one item from each class; here we need to select one layer for each tile to cache. MCKP is NP-Hard.

4.3 Optimization objective

In the multiple-versions or the layered-encoding case, another metric we may seek to optimize is a coverage factor that takes into account the proportion of occurrence of each tile and the *coverage* of the tile set, namely the requirement that each tile needs to have at least one cached resolution, if it is frequently requested. For instance, for the multiple-versions case, the coverage factor is defined as

$$B(\mathbf{x}) = \sum_{i\in\mathcal{T}} p_i \min\{\sum_{r\in\mathcal{R}} x_{ir}, 1\}\,. \tag{12}$$

Since $0 \le Q(\mathbf{x}) \le 1$ and $0 \le B(\mathbf{x}) \le 1$, we can combine them in the following objective function $f(\mathbf{x})$ with a balancing weight $\alpha \in [0,1]$ as follows:

$$\max_{\mathbf{x}} f(\mathbf{x}) = \alpha B(\mathbf{x}) + (1-\alpha)(1-Q(\mathbf{x}))\,, \tag{13}$$

subject to (5) and (6). A similar optimization objective with respect to $\mathbf{x}^L$ may be formulated for the layered-encoding case, subject to (10) and (11). We have experimentally validated that $\alpha = 0.5$ yields the optimal balance between the two objectives.

5 IMPLEMENTATION AND EVALUATION

5.1 Caching Optimization

The measured tile importance of each video segment over the cubemap exhibits several local minima since several directions per panoramic video receive focused attention of the user during playback (see Fig. 2). Further, the optimal values of K_i for the case of multiple-versions videos are not known and are subject to optimization. We solicit the solution of the optimization problem through a caching policy $\mathbf{x}$ by using a Markov Chain Monte Carlo (MCMC) approach, a parallel Metropolis-Hastings sampling method [11] of the solution space of $\mathbf{x}$ with a method-dependent state transition (mutation) function, and the objective function in (13). Simulated annealing would be another option, but we choose here not to introduce additional parameters (e.g. temperature). We have also experimented with a genetic algorithm with random splicing points but have not registered improvements in performance so as to justify the additional operations.

The policy vector $\mathbf{x}$ is represented in the optimizer as a string of n bytes whose most significant bit signifies the inclusion of tile i in the policy regardless of level, and whose remaining bits correspond to active quality levels for this tile. For the layered-encoding case, only one of these bits is on, while for multiple-version case it may include up to 7 versions for the same tile i.

At each iteration k, the new policy vector $\mathbf{x}_k$ may either exceed the cache capacity C or under-populate the cache. To account for this, we measure the memory overhead incurred by $\mathbf{x}_k$ as

$$\text{Ov}(\mathbf{x}_k) = \frac{\text{mem}(\mathbf{x}_k) - C}{C}\,, \tag{14}$$

where $\text{mem}(\mathbf{x}_k)$ is calculated through (5) and (10) respectively for the two cases. We use $\text{Ov}(\cdot)$ to uniformly prune i.e. disable tiles with probability p_{prune}, where

$$p_{prune} = 1 - \frac{1}{1 + \text{Ov}(\mathbf{x}_k)}\,. \tag{15}$$

The overhead $\text{Ov}(\cdot)$ is used for determining the current mutation rate at iteration k, $r_m(k)$ to accelerate the inclusion of more tiles, if the cache is underpopulated. This is given by

$$r_m(k) = r_m \cdot \left(1 + \max\{0, -\text{Ov}(\mathbf{x}_{k-1})\}\right)\,, \tag{16}$$

where r_m is the overall mutation rate. At each iteration k, $r_m(k) \cdot |\mathcal{T}|$ tiles i are enabled with probability $p_{add} = p_i$ to promote them in the policy proportionally to occurrence. Note that p_i is derived from the measured occurrence of i-th tile in the cache statistics after normalization.

For the multiple-versions case, we select a new state for tile i with respect to present quality levels as follows. We first select the tile with probability $r_m(k)$, then randomly select the new $K_i \in \{0, \dots, R\}$, and we enable K_i random bits in the tile byte. For the layered-encoding case, instead of randomly turning on a quality level, we shift the current quality level up or down by one bit, according to the sign of $\text{Ov}(\cdot)$.

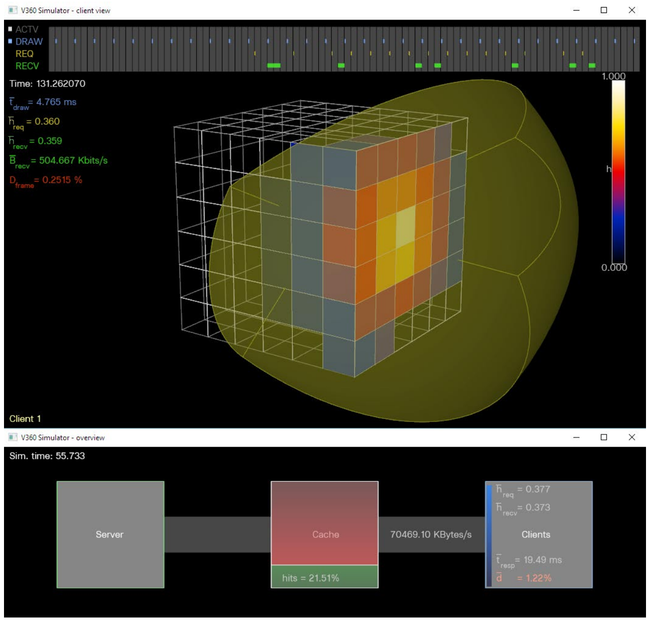

Figure 3: Screenshots from the simulator showing the detailed view of each client and the simulation overview window.

5.2 Simulator Implementation

Video streams for each tile and video were decomposed into chunks of 24-120 frames each depending on the video category we examine; these correspond to multiples of 1 sec or 24 frames per second video playback. Larger chunks were not deemed appropriate for 360° video playback, since any change in the viewing direction causes several client-side tiles to become invalid. In order to optimize requests and response times, only incremental updates are requested by the client. Chunks of tiles that existed in a previous frame and are still within the current view after a viewport change are only updated if (*i*) they become stale, or (*ii*) their quality level is insufficient for the perceptual importance currently designated to them. Namely, if a tile at the central region of a viewport becomes a peripheral one, i.e. one with lower importance, the existing higher-quality tile is used instead in both the multiple-versions and the layered-encoding cases, provided that the time stamp of the loaded chunk remains valid.

A unique 32-bit tile code is designated to each tile of a given frame, according to Figure 1(d). The tiling factor N_c is also stored in the tile code. The viewpoint selection, video stream ID and tile ID are jointly considered as a single tile index i for optimization purposes. For every frame currently under reconstruction and displaying at a client, a variable number of tiles must be requested, depending on the particular orientation and view frustum characteristics of the client at the given time. *All* tiles that are intersected with the user's view frustum are identified, and corresponding request packets are asynchronously sent to the cache for those tiles that are either unavailable at the client at the desired quality or that are stale. Migrating tile selection to the client helps to establish arbitrarily complex and generic visor or screen configurations and arbitrary orientations of the viewer's reference frame.

The simulator models the following delay lines: request packet preparation, tile decoding, frame reconstruction from received tiles, client-to-cache round-trip time (RTT) and cache-to-server RTT. We also model the cache and server processing throughput (bandwidth), which affects response times as the volume of requests increases. We keep track of various statistics at all three node types (client, cache, server), including instantaneous and average RTT to the client, cache throughput and *hits and misses per tile and per quality level*, which are normalized and used for cache optimization.

For client modeling, we use a stack of *behaviors*, i.e. algorithms that handle a different aspect of video playback. We have separate behaviors for choosing and playing a video, for pausing and resuming it, and for determining the current viewing coordinate system (Fig. 1(b)). For the latter, the head motion behavior eases motion in and out of different viewing directions using a cosine interpolation so as to avoid unnatural, sudden movements. Directions are drawn from a Normal distribution around fixed directions generated separately for each video at specific time points and stored in the video catalogue index so as to represent recorded preference of the users' gaze directions while watching a video.

The simulator is implemented from scratch in C++, using an OpenGL front-end for visualization (see Fig. 3). For all implemented algorithms (optimizer, behaviours, etc.), we heavily exploit parallelism via the OpenMP API.

5.3 Experimental Setup

In order to evaluate our model, we gather statistics for both the multiple-versions and the layered-encoding case using different ratios of cache-to-catalogue size, as shown in Fig. 4. For a fair comparison, we test our tile-based caching approach against a popularity-based caching policy. We use the same global optimization algorithm as in our method, but we optimize the caching policy $\mathbf{x}$ by first sampling the video catalogue according to the popularity distribution and then uniformly choosing tiles and quality levels for selected video segments. The population is regulated through the mutation rate and pruning mechanism described in Section 5.1.

We conducted experiments using two types of video libraries: a) a large collection of $4,000$ videos of small clips with an average duration of 300 sec that represents uploaded user content, and b) a smaller video catalogue of 800 videos with average video duration of $5,000$ sec that may stand for a movie and series episode catalogue. The two catalogues were built for both layered-encoding and multiple-versions stream playback. The following table summarizes the experiment characteristics.

Table 1: Experiment profiles.

Catalogue	Videos	Avg. length	Format	Size	Clients
Movies	800	5000 sec	Layered	2.3TB	30
Movies	800	5000 sec	Multiple versions	4.4TB	30
Video clips	4000	300 sec	Layered	670GB	40
Video clips	4000	300 sec	Multiple versions	1.3TB	40

For all experiments we use a tiling factor $N_c = 6$, a simulation time step of $1/60$ sec, and video playback at 24 frames per second, unless stated otherwise. The number of concurrent clients selected for the experiment profiles in Table 1 was chosen according to the specific delay line and bandwidth characteristics of the simulator so as to have a dropped frame rate below 2% throughout all tested cache sizes. Clients choose videos according to their popularity. For all experiments, we used at most $4 \cdot 10^4$ iterations for the optimizer and a base mutation rate $r_m = 0.4$, including the caching scheme we compare our method against.

5.4 Experimental Results

Figure 4 shows results of our experiments with the two classes of video catalogues and two video formats. In all cases, the caching policy that prioritizes tile occurrence and requested tile quality is superior. The only case when the popularity-based caching scheme gives better hit ratio than our tile-based approach is when cache size is too small and the video segments are very large, as in the case of the movie/series catalogue with multiple versions (where the cache-to-catalogue size ratio is below 2%). Video streams with layered encoding benefit more both from better memory utilization and the simpler convergence of the optimization process, as the latter does not need to optimize an extra parameter, the number of versions cached. In the same figure, we report the overall RTT at the client. Despite being only indicative due to its dependence on simulation parameters irrelevant to the caching policy, this time demonstrates the impact of the cache hit ratio in the particular scenario. Again, big savings are seen in the layered-encoding case.

During the optimization process, the quality factor $Q(\cdot)$ in the objective (13) attained minimum values consistently below 0.03. On the other hand, the coverage factor $B(\cdot)$ took

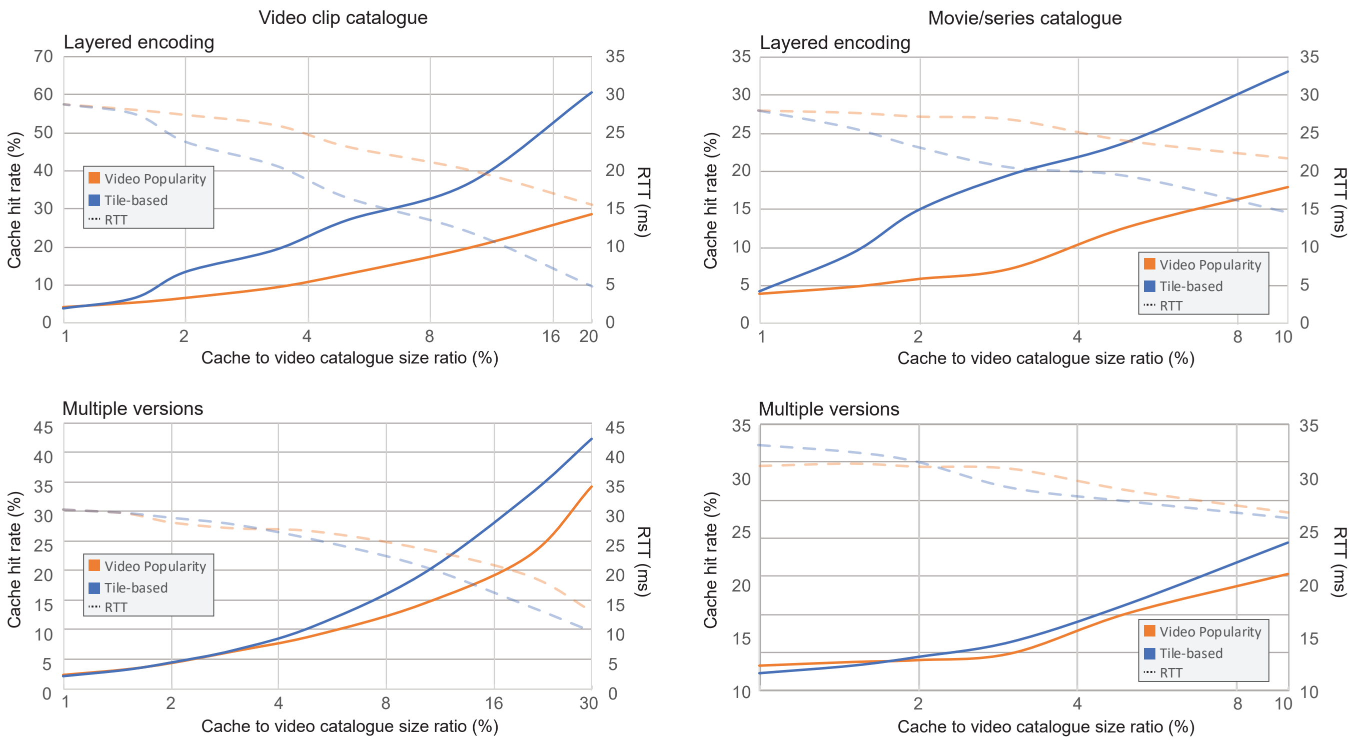

Figure 4: Cache hits and round-trip time for different cache sizes ($\log_2$ scale), for the two experiment categories: video clips (left - $4,000$ videos, 300 sec average duration), and movie/series catalogue (right - 800 videos, $5,000$ sec average duration).

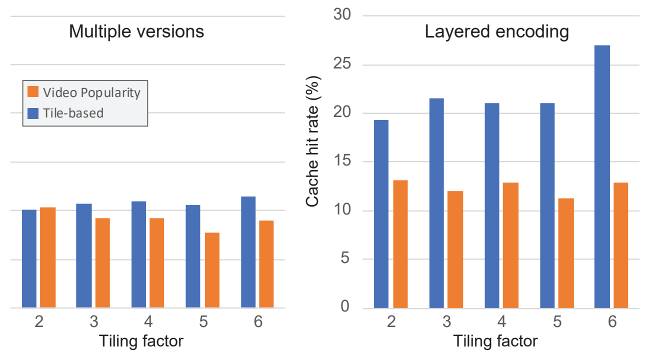

Figure 5: Impact of tiling factor to cache hits, for fixed cache size ratio (5%).

Table 2: Optimization time for $1,000$ iterations.

Video clips		Movies / episodes	
Layered	Multiple versions	Layered	Multiple versions
70 sec	145 sec	14 sec	26 sec

values proportional to the cache-to-catalogue size ratio; usually, when $B(\cdot)$ was significantly higher than this ratio, the performance of the cache policy was significantly improved compared to the popularity-based one.

As shown in Fig. 5, the choice of tiling factor N_c is not critical for the performance of the cache, although finer tessellations of the cubemap sample the tile occurrence distribution p_i better and allow for tighter fitting of the policy to the clients' demand, as demonstrated by the slight increase of hit rate for larger N_c. The optimization and objective function reflect the fact that the cache population is static and $\mathbf{x}$ can be determined offline.

Due to the fact that the entire policy string is examined in each iteration, the optimization time depends on the size of the catalogue. In Table 2, we report the optimization time per $1,000$ iterations for the test cases of Fig. 4 on an Intel i7-4930K 6-core processor (with 12 threads) with 32GB of RAM.

6 CONCLUSIONS

We studied the problem of optimal caching of 360° videos with the aim to optimize an objective different from the one in conventional video caching. This objective combines (*i*) an error metric between requested and cached tile resolutions across different viewports, and (*ii*) coverage of the tile set. Our experiments showed improvements in cache hit ratio when a caching policy based on tile statistics is enforced, compared to a popularity-driven one, especially in the layered-encoding case.

We believe that this work opens an interesting field with several directions for future study. A model extension could consider constraints on co-occurrences of tile streams within the same viewports. In that case, respective video segments all need to be delivered at very low latency and therefore need to be considered jointly for caching. A multi-user 360°-video transmission scenario could also be studied, where users have different viewport and tile occurrence statistics, and the objective is to conserve transmission bandwidth through tile stream multicasting to users that have the same or similar viewports. In this work, we adhered to a single cache scenario. In a scenario with multiple caches, user-cache association would also be needed.

Finally, in this work we studied a static caching scenario in the sense that the caching decision is made prior to transmission based on tile resolution demand statistics. A different model could make caching decisions i.e. tile/segment resolution insertions and evictions from the cache in a dynamic fashion using information available on the spot, such as the user instantaneous viewport direction or wireless channel conditions, and statistical knowledge about the future.

REFERENCES

[1] 2017. *Augmented and Virtual Reality: The First Wave of 5G Killer Apps.* white paper. ABI Research, Qualcomm.

[2] Y. Bao, H. Wu, A. A. Ramli, B. Wang, and X. Liu. 2016. Viewing 360 degree videos: Motion prediction and bandwidth optimization. In *2016 IEEE 24th International Conference on Network Protocols (ICNP).*

[3] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu. 2017. Motion-Prediction-Based Multicast for 360-Degree Video Transmissions. In *IEEE International Conference on Sensing, Communication, and Networking (SECON).* 1–9.

[4] J. Chakareski, R. Aksu, X. Corbillon, G. Simon, and V. Swaminathan. 2018. Viewport-Driven Rate-Distortion Optimized 360° Video Streaming. *arXiv:1803.08177* (2018).

[5] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos. 2017. Caching-aware recommendations: Nudging user preferences towards better caching performance. *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications* (2017).

[6] M. Choi, J. Kim, and J. Moon. June 2018. Wireless Video Caching and Dynamic Streaming under Differentiated Quality Requirements. *IEEE Journal on Selected Areas in Communications* 36, 6 (June 2018), 1245–1257.

[7] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire. 2013. Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution. *IEEE Communications Magazine* 51, 4 (2013), 142–149.

[8] M. Graf, C. Timmerer, and C. Mueller. 2017. Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over HTTP: Design, Implementation, and Evaluation. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17).* 261–271.

[9] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. 2012. Foveated 3D Graphics. *ACM Transactions on Graphics,* 31, 6, Article 164 (Nov. 2012), 164:1–164:10 pages.

[10] F. Hartanto, J. Kangasharju, M. Reisslein, and K. Ross. 2006. Caching video objects: layers vs versions? *Multimedia Tools and Applications* (Nov 2006).

[11] W.K. Hastings. 1970. Monte Carlo sampling methods using Markov chains and their application. *Biometrika* 57, 1 (1970), 97–109.

[12] M. Hosseini and V. Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer. *IEEE International Symposium on Multimedia (ISM)* (2016), 107–110.

[13] S. Ioannidis and E. Yeh. 2017. Jointly Optimal Routing and Caching for Arbitrary Network Topologies. In *Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN '17).* 77–87.

[14] M. Ji, G. Caire, and A. F. Molisch. 2015. The Throughput-Outage Tradeoff of Wireless One-Hop Caching Networks. *IEEE Transactions on Information Theory* 61, 12 (2015), 6833–6859.

[15] H. Kellerer, U. Pferschy, and D. Pisinger. 2004. *Knapsack Problems.* Springer.

[16] J. T. Lee, D. N. Yang, and W. Liao. 2016. Efficient Caching for Multi-View 3D Videos. In *2016 IEEE Global Communications Conference (GLOBECOM).* 1–7.

[17] M. Levoy and R. Whitaker. 1990. Gaze-directed Volume Rendering. *SIGGRAPH Comput. Graph.* 24, 2 (Feb. 1990), 217–223.

[18] X. Liu, Q. Xiao, V. Gopalakrishnan, B. Han, F. Qian, and M. Varvello. 2017. 360° Innovations for Panoramic Video Streaming. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks (HotNets-XVI).* 50–56.

[19] C. Ozcinar, A. De Abreu, and A. Smolic. 2017. Viewport-aware adaptive 360° video streaming using tiles for virtual reality. In *2017 IEEE International Conference on Image Processing (ICIP).* 2174–2178.

[20] G.S. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah. 2016. Wireless caching: technical misconceptions and business barriers. *IEEE Communications Magazine* 54, 8 (2016), 16–22.

[21] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire. 2018. The Role of Caching in Future Communication Systems and Networks. *arXiv:1805.11721* (2018).

[22] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn. 2016. Towards Foveated Rendering for Gaze-tracked Virtual Reality. *ACM Transactions on Graphics* 35, 6, Article 179 (Nov. 2016), 179:1–179:12 pages.

[23] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas. 2016. Caching and operator cooperation policies for layered video content delivery. In *IEEE INFOCOM 2016 - IEEE International Conference on Computer Communications.* 1–9.

[24] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas. 2014. Video delivery over heterogeneous cellular networks: Optimizing cost and performance. In *IEEE INFOCOM 2014 - IEEE International Conference on Computer Communications.* 1078–1086.

[25] S. Shukla, O. Bhardwaj, A.A. Abouzeid, T. Salonidis, and T. He. 2017. Hold'Em Caching: Proactive Retention-Aware Caching with Multi-path Routing for Wireless Edge Networks. In *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '17).* 24:1–24:10.

[26] C. Zhou, M. Xiao, and Y. Liu. 2018. ClusTile: Toward Minimizing Bandwidth in 360-degree Video Streaming. In *IEEE INFOCOM 2018 - IEEE International Conference on Computer Communications.*