Mobile-Edge Cooperative Multi-User 360° Video Computing and Streaming

Jacob Chakareski and Nicholas Mastronarde

***Abstract*—We investigate a novel communications system that integrates scalable multi-layer 360° video tiling, viewport-adaptive rate-distortion optimal resource allocation, and VR-centric edge computing and caching, to enable future high-quality untethered VR streaming. Our system comprises a collection of 5G small cells that can pool their communication, computing, and storage resources to collectively deliver scalable 360° video content to mobile VR clients at much higher quality. Our major contributions are rigorous design of multi-layer 360° tiling and related models of statistical user navigation, and analysis and optimization of edge-based multi-user VR streaming that integrates viewport adaptation and server cooperation. We also explore the possibility of network coded data operation and its implications for the analysis, optimization, and system performance we pursue here. We demonstrate considerable gains in delivered immersion fidelity, featuring much higher 360° viewport peak signal to noise ratio (PSNR) and VR video frame rates and spatial resolutions.**

I. Introduction

Virtual reality holds tremendous potential to advance our society. It is expected to make impact on quality of life, energy conservation, and the economy [1], and reach a $120B market by 2022 [2]. As the Internet-of-Things (IoT) is becoming a reality, modern technologists envision transferring remote contextual and environmental immersion experiences as part of an online VR session. However, two main highly-intertwined challenges stand in the way of realizing this vision: VR requires **(1) ultra-low latency high data rate communications, and (2) highly data-intensive computing**. Neither of these challenges can be met by current and upcoming traditional communications systems [3], as the content to be delivered is too voluminous and the headsets' computing/storage capabilities are insufficient within an acceptable and wearable form factor. Hence, VR applications are presently limited to off-line operation, low-fidelity graphics content, tethered high-end computing equipment, and gaming/entertainment settings. 360° video is the first actual-scene content format to enable remote VR immersion. However, emerging 360° streaming practices are highly inefficient, which considerably degrades the quality of experience, as explained in detail later.

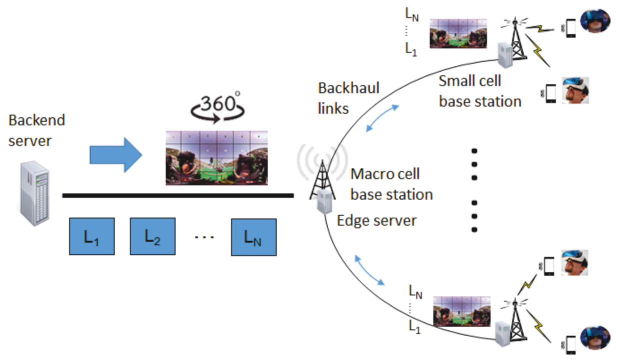

Fig. 1: System scenario under investigation.

To overcome these challenges, we investigate for the first time a novel communications system that integrates scalable multi-layer 360° video tiling, viewport-adaptive rate-distortion optimal resource allocation, and VR-centric edge computing and caching, to enable next generation high-quality untethered on-demand VR streaming. Our system is illustrated in Figure 1 and comprises a collection of 5G small cells featuring a base station and an edge server each, which pool their communication, computing, and storage resources to collectively deliver scalable 360° video content to mobile VR clients, at much higher quality. Cooperation among the small cells is enabled via backhaul links that interconnect them, and the scalable 360° content featuring multiple layers L_i of incrementally increasing quality is initially delivered from a backend server, as illustrated in Figure 1. Considerable advances in 360° video quality, frame rate, and spatial resolution are enabled.

This work was supported in part by NSF Awards ECCS-1711335, CCF-1528030, ECCS-1711592, CNS-1836909, and CNS-1821875.

II. 360° Video VR Streaming Background

360° video is an emerging video format that is captured by an omnidirectional camera that records incoming light rays from every direction (see Figure 2 top left).

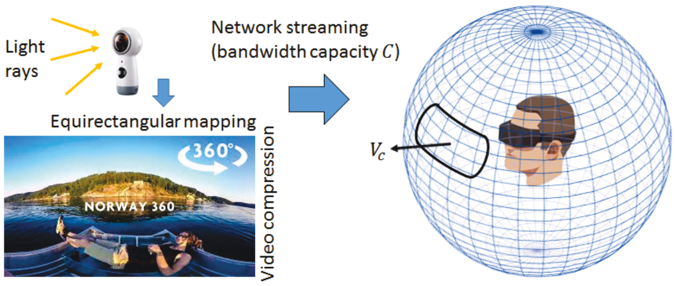

Fig. 2: 360° video capture and streaming, and user viewport V_c.

It enables a three dimensional 360° look-around of the surrounding scene for a remote user, virtually placed at the camera location, on his VR head-mounted display (HMD), as illustrated in Figure 2 right. After capture, the spherical 360° raw video frames are first mapped to a wide equirectangular panorama (illustrated in Figure 2, bottom left) and then compressed using state-of-the-art (planar) video compression such as HEVC. The former intermediate step is introduced, as compression techniques operating directly on spherical data are much less mature and performing relative to conventional video compression operating on planar (2D) video frames.

For remote service, present practices stream the entire monolithic 360° panorama to the user using MPEG-DASH [4]. However, at any point of time, the user experiences only a small portion of it denoted as V_c (current viewport). This considerably penalizes the quality of experience, due to the overwhelming volume of 360° data that needs to be delivered, which exceeds the available network bandwidth C by orders of magnitude. Similarly, the streaming also lacks the ultra-low latency interactivity required for truly immersive experiences, due to the use of traditional server-client Internet architectures.

III. Related work

Similarly to our approach, a few existing studies of single-user on-demand 360° Internet streaming [5–7] considered splitting the 360° video into spatial tiles as part of the encoding, using the tiling feature of the latest High Efficiency Video Coding (HEVC) standard [8]. However, their design choices are heuristic and lack analysis of the fundamental trade-offs between delivered immersion fidelity, user navigation patterns, coding efficiency, view switching capability, and available system resources, as we carry out. Moreover, our integration of scalability, edge-based delivery, and formal 360°

partitioning considerably enhances the VR application interactivity, by reducing its streaming latency, relative to these studies.

Existing work on wireless base station caching includes [9], which studied minimizing the total delay of content retrieval at a base station online. [10] studied reducing the delay of content delivery using caching at wireless helper nodes, differentiating available helpers based on their proximity to the served node. Moreover, [11] studied hierarchical caching in cellular back-haul networks. Joint caching and channel assignment in multi-cellular systems is studied in [12].

IV. Scalable multi-layer 360° tiling

A. 360° Panorama Tiling

We partition each panoramic frame of a 360° video into a set of $N \times M$ spatial tiles, where the first and second dimensions of the denoted tiling, (N, M), parallel the horizontal and vertical spatial dimensions of the 360° video frames. Each tile is then independently encoded and streamed to the user, according to our analysis and optimization. Denser tiling layouts increase the processing complexity and reduce the compression efficiency, but, enable more precise delineation of the user viewport and thus more efficient viewport-aware resource allocation across the 360° panorama. We empirically observed that the 6×4 and 8×6 tiling options provide good performance in terms of processing complexity and compression efficiency, for the 360° spatial resolutions available today (4K/8K).

B. 360° Navigation Data Capture

We have captured navigation data that characterizes how a mobile VR user explores a 360° video over time. Specifically, his VR head-mounted display (device) outputs the direction of the current viewpoint of the user V_c on the 360° view sphere up to 250 times per second (see Figure 2 right, for an illustration). Formally, this is the surface normal of V_c on the 360° sphere, which is uniquely characterized by the azimuth and polar angles $\varphi \in [0°, 360°]$ and $\theta \in [0°, 180°]$ that it spans on the sphere, in a spherical coordinate system with the 360° sphere center as its origin (see Figure 3, right). These two angles are equivalently denoted as yaw and pitch in the VR community, captured as rotation angles around the Z and Y axes (see Figure 3, left). We recorded the (φ_j, θ_j) pairs that coincided with the discrete temporal instances t_j of consecutive 360° video frames j from which the respective viewport V_c is selected to be displayed to the user, as he navigates the content. We leverage this data to formulate our statistical analysis of user navigation next.

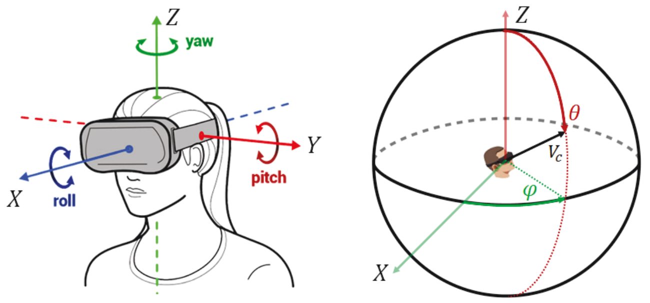

Fig. 3: 360° navigation data of current viewport V_c. Left: Rotation angles yaw, pitch, and roll around the three coordinate axis. Right: Azimuthal and polar angles (φ, θ) in spherical coordinates.

C. Statistical Characterization of User Navigation

Let the set $\{(\varphi_j, \theta_j)\}$ denote a navigation trace for a given 360° video and VR user. Let $S_j^{V_c}$ denote the set of pixels in the 360° panorama occupied by the user viewport V_c at time instance t_j (temporal video frame j). Similarly, let S_j^{nm} denote the set of pixels in the 360° panorama associated with tile (n, m), for $n = 1, \ldots, N$, and $m = 1, \ldots, M$. Now, let $S_j^{nm,V_c} = S_j^{V_c} \cap S_j^{nm}$ denote the set of pixels in tile (n, m) present in the user viewport at that time instance. That is, S_j^{nm,V_c} represents the spatial area in the 360° panorama shared by tile (n, m) and V_c at time t_j.

We illustrate later that a user viewport may occupy very different, in terms of shape and size, spatial areas of the 360° panorama, depending on its latitude (the polar angle θ on the 360° view sphere). To account for this, in developing our statistical model of user navigation, we formulate next the fractions of the spatial areas of every tile, present in the user viewport V_c at t_j, as follows: $w_j^{nm} = |S_j^{nm,V_c}| / \sum_{n,m} |S_j^{nm,V_c}|$, where $|S|$ denotes the size of a set S, in this case in number of pixels. Thus, $\{w_j^{nm}\}$ represents the normalized distribution of the spatial area of the user viewport across every tile in the 360° panorama, at time instance t_j.

Given the above analysis, we formulate the probability (likelihood) of the user navigating tile (n, m) over a time interval spanned by the time instances $[t_i, t_j]$ as: $P_{nm}^{(t_i,t_j)} = \sum_{k=i}^{j} w_k^{nm} / (j - i + 1)$. In other words, $P_{nm}^{(t_i,t_j)}$ indicates how often tile (n, m) appears (at least in part) in the user viewport during navigation of the 360° video from its temporal instance t_i to t_j, or the popularity of the 360° scene content captured by the tile for this user and time interval. For instance, if t_i and t_j correspond to the first and last video frame of the 360° video, then, $P_{nm}^{(t_i,t_j)}$ captures the navigation probability or popularity of tile (n, m) across the entire video.

D. Viewport-Adaptive Space-Time Scalability

To enable an effective allocation of system resources across a 360° video, we explore a scalable multi-layer tiling of a 360° panorama. In particular, for every tile (n, m) in the panorama, we will construct L embedded layers of progressively increasing signal fidelity. The multi-layer tiling construction is illustrated in Figure 4. It can enable carrying out effective trade-offs between delivered immersion fidelity and induced data rate, spatiotemporally over the 360° content, in response to the user navigation actions. This can be effectively accomplished by optimally selecting the number of layers l_{nm} sent for every tile (n, m) during a time interval, such that the expected user viewport quality over that interval is maximized, given the available network streaming bandwidth C.

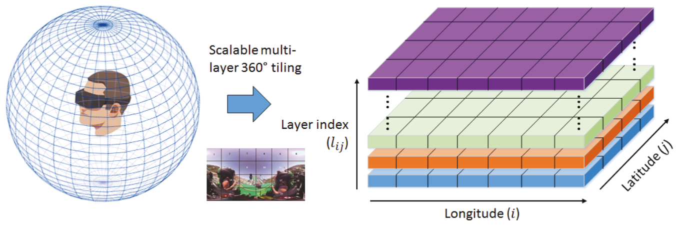

Fig. 4: Scalable multi-layer 360° tiling.

We can formally capture this optimization as:

$$\max_{l_{nm}} \sum_{nm} P_{nm} Q_{nm}(l_{nm}), \text{ subject to: } \sum_{nm} R_{nm}(l_{nm}) \leq C.$$

Here, P_{nm} denotes the likelihood of navigating tile t_{nm} during the time period under consideration, as introduced earlier. $Q_{nm}(l_{nm})$ and $R_{nm}(l_{nm})$ denote respectively the delivered immersion fidelity and induced data rate associated with tile t_{nm}, given that its first l_{nm} scalable layers are sent to a user.

Note that the proposed statistical analysis of user navigation captures as navigation likelihoods the expected overlap of a tile with the user viewport over a time interval, and the aspect that equatorial tiles are more likely navigated than polar tiles. Thus, our expected viewport quality formulation can correspond to a tile-level WS-PSNR (Weighted Spherical PSNR) [13].

V. Edge-Based Multi-User VR Streaming

A. System Modeling

There is a set of 360° videos served to mobile VR users at each small-base station. Each 360° video comprises $N \times M$ tiles. To ease the notation, we assign to every tile (n, m) of an entire 360° video a unique index j, and denote it henceforth as video j. Each base station i serves a set of VR clients that collectively induce a popularity distribution P_{ij} over the tile-videos, by their navigation actions. A base station can store a subset of the videos at its edge server, to deliver them locally to its own users. It can also serve one of its videos to a VR client at another small base station via the backhaul links through which they can cooperate, if this video is not stored locally at the edge server of that small base station. If a requested video is not available locally or from a neighboring small base station, it is delivered remotely from the back-end server.

Let $Y_{ij}^l \in \{0, 1\}$ denote the decision for small base station i to cache tile-video j comprising its first l scalable layers (see Figure 4). Let $X_{ij}^{l,k} \in \{0, 1\}$ denote the decision to deliver video j comprising its first l scalable layers from base station k to a VR user at base station i. If $k = i$, then $X_{ij}^{l,k} = 1$ will denote the event of local delivery at base station i. If k is greater than the number of small base stations in the system, $X_{ij}^{l,k}$ will capture the decision to deliver video j comprising its first l scalable layers remotely, from the back-end server, as introduced earlier. Let $Q_{j,l}$ denote the delivered immersion fidelity of tile-video j comprising its first l scalable layers.

Let $C_{ij}^{l,k}$ denote the cost of delivering tile video j comprising its first l scalable layers to a VR user at small base station i from the cache of small base station k. $C_{ij}^{l,k}$ captures the impact of the relative distance between base stations i and k, and the data volume B_j^l of tile-video j featuring l scalable layers. Similarly, let $\bar{B}_j^l$ denote the processing and rendering cost associated with tile-video j featuring l scalable layers, induced at a small base station. Let Z_i and $\bar{Z}_i$ denote respectively the storage and computing capabilities of the edge server at base station i. We model $\bar{B}_j^l$ accurately as a polynomial function of the data volume of tile-video j comprising l scalable layers.

We consider the possibility of having the data packets associated with tile-video j encoded using network coding or fountain codes [14]. Working with such packets helps use system resources more efficiently and reduce the system's transmission scheduling complexity. These advances stem from their construction which eliminates packet duplication and thus enables working with fractional network flows, instead of their discrete $\{0, 1\}$ counterparts.

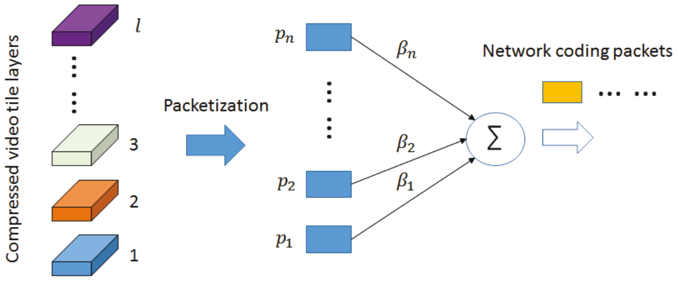

Fig. 5: Video tile layers to data packets to network coding packets.

Our method for constructing network coding packets is illustrated in Figure 5. The bitstream representing the scalable compressed video tile layers is first packetized into data packets p_i. Network coding packets are then constructed as weighted linear combinations of the data packets using $\sum_i \beta_i p_i$, where the weights β_i are selected uniformly at random from a Galois finite field. The arithmetic operation of summation is performed over the same finite field.

B. Problem Formulation

We are interested in maximizing the immersion fidelity delivered to the VR clients at the small base stations, while minimizing the induced cost. Given a tile-video j featuring l scalable layers and a small base station i, let $Q_{j,l}/C_{i,j}^{l,k}$ denote the delivered immersion fidelity per unit cost, when this content is delivered from small base station k. We recall that in this case the decision variables Y_{kj}^l and $X_{ij}^{l,k}$ would need to be set to one. Now, let $\sum_{i,j,k,l} P_{ij} X_{ij}^{l,k} Q_{j,l}/C_{i,j}^{l,k}$ denote the aggregate expected delivered immersion fidelity per unit cost. Our objective is to maximize this quantity given various system and problem formulation constraints that arise here.

We formally characterize our objective as:

$$\max_{Y_{ij}^l, X_{ij}^{l,k}} \sum_{i,j,k,l} P_{ij} X_{ij}^{l,k} Q_{j,l}/C_{i,j}^{l,k}, \tag{1}$$

$$\text{s.t.: } X_{ij}^{l,k} \leq Y_{kj}^l, \forall i,j,k,l, \quad \sum_l Y_{ij}^l \leq 1, \forall i,j, \tag{2}$$

$$\sum_k X_{ij}^{l,k} \leq 1, \forall i,j,l, \quad \sum_{j,l,k} X_{ij}^{l,k} B_j^l \leq Z_{bh}, \forall i, \tag{3}$$

$$\sum_{j,l} Y_{ij}^l B_j^l \leq Z_i, \forall i, \quad \sum_{j,l} Y_{ij}^l \bar{B}_j^l \leq \bar{Z}_i, \forall i, \tag{4}$$

where the first constraint in (2) captures the notion that tile-video j cannot be delivered from small base station k, unless it is cached there. The second constraint in (2) captures the condition that only one replica of tile-video j comprising l scalable layers is stored at base station i. The first constraint in (3) captures the notion that tile-video j comprising l scalable layers is streamed to VR clients at small base station i from the edge server of only one small base station k. The second constraint in (3) ensures that the tile-video data streamed to any small base station i does not exceed the transmission capacity of the carrier backhaul links, denoted here as Z_{bh}. Finally, the two constraints in (4) capture the limited caching and computing capabilities of the edge servers at each base station.

C. Analysis and Approximation

The problem (1) - (4) is discrete and has a combinatorial nature. Thus, it is difficult to solve. Showing that (1) - (4) is NP-complete requires showing that any given solution can be verified quickly and that a known NP-complete problem can be reduced to (1) - (4) [15]. We can verify a given solution by checking its feasibility against the constraints (2) - (4) in polynomial time. This meets the first requirement. We meet the second requirement by mapping the known NP-complete multi-knapsack problem [16] to (1) - (4).

The multiple knapsack problem comprises N items, characterized with profit and weight factors α_n and γ_n, and K knapsacks, characterized with holding capacity factors c_k. The objective is to select K disjoint subsets of items such that their aggregate profit is maximized and each can be assigned to a knapsack k such that its aggregate weight does not exceed c_k. We extend this definition to include two weight factors per item n, namely γ_n^1 γ_n^2, and respectively two capacity factors c_k^1 and c_k^2 per knapsack k. The NP-complete nature of the problem remains under the extension. We map this problem then to (1) - (4) as follows. First, we map each knapsack to the edge server associated with one small base station, such that $Z_k = c_k^1$ and $\bar{Z}_k = c_k^2$. Next, we map item n to tile video j comprising l scalable layer such that $B_j^l = \gamma_n^1$ and $\bar{B}_j^l = \gamma_n^2$, and $\sum_i P_{ij} = \alpha_n$. Finally, we set $Q_{j,l}/C_{i,j}^{l,k} = 1, \forall i, j, l, k$.

Given the above, the knapsack problem and the mapped instance of our problem share a feasible solution with a common objective function value. Moreover, we can carry out the problem mapping reduction above in polynomial time. This completes the verification that (1) - (4) is NP-complete.

To formulate an approximation solution to solve (1) - (4), we note that given our system setting in Figure 1, the cost factors $C_{i,j}^{l,k}$ will all be equal when streaming tile video j featuring l scalable layers from any neighboring small base station $k \neq i$, and smaller than the cost factor of delivering this content from the remote back-end server. Thus, we can rewrite (1) - (4) as:

$$\max_{Y_{in}, X_{in}^k} \sum_{i,n} P_{in} X_{in}^i Q_n / C_{i,n}^i + \sum_{i,n,k\neq i} P_{in} X_{in}^k Q_n / \bar{C}_n, \quad (5)$$
$$\text{subject to: } (2) - (4).$$

To simplify the notation, we have mapped each original index pair (j, l) to a unique single variable n in the reformulation above. $\bar{C}_n$ denotes the common cost factor of delivering content item n from any neighboring small base station $k \neq i$.

Let $\alpha_{in} = P_{in} Q_n / C_{i,n}^i + \sum_{k\neq i} P_{kn} Q_n / \bar{C}_n$ denote the maximum prospective benefit of caching item n at small base station i. We define $\mathcal{V} = \{1, \ldots, K\} \times \{1, \ldots, N\}$ to be the set representing the vector product of the sets of small base stations and content items, where K and N denote their respective sizes. Let $v = (i, n) \in \mathcal{V}$ denote a member of this set. Facilitating $\mathcal{V}$, we can solve (5) as a multiple knapsack problem with multiple constraints associated with each small base station k, as follows.

Using dynamic programming [17], we formulate the optimal value function $f_v(\cdot)$ associated with (5) as

$$f_v(s_1, \bar{s}_1, \ldots, s_K, \bar{s}_K) =$$
$$\begin{cases} \max_{x_v \in \{0,1\}} \{\alpha_{in} x_v + f_{v-1}(\ldots, s_i - B_j^l x_v, \bar{s}_i - \bar{B}_j^l x_v, \ldots)\}, \\ \quad \text{if } \nexists v' < v : x_{v'} = 1 \wedge n' = n \wedge s_i \geq B_j^l, \bar{s}_i \geq \bar{B}_j^l, \\ \max_{x_v \in \{0,1\}} \{(\alpha_{in} - P_{in} Q_n / \bar{C}_n) x_v + \\ \qquad f_{v-1}(\ldots, s_i - B_j^l x_v, \bar{s}_i - \bar{B}_j^l x_v, \ldots)\}, \\ \quad \text{if } \exists v' < v : x_{v'} = 1 \wedge n' = n, i' \neq i \wedge s_i \geq B_j^l, \bar{s}_i \geq \bar{B}_j^l, \\ f_{v-1}(s_1, \bar{s}_1, \ldots, s_K, \bar{s}_K), \\ \quad \text{if } \exists v' < v : x_{v'} = 1 \wedge n' = n, i' = i \vee s_i < B_j^l \vee \bar{s}_i < \bar{B}_j^l, \end{cases}$$

for $v = |\mathcal{V}|, \ldots, 1$, where $f_0(\cdot) = 0$. The state variables $s_i \in \{0, \ldots, Z_i\}$ and $\bar{s}_i \in \{0, \ldots, \bar{Z}_i\}$ capture the slack caching and computing capacity, respectively, at small base station i.

We develop the optimal value function $f_v(\cdot)$ using the above Bellman optimality condition recursion, in stages, iteratively, starting from stage $v = 1, \ldots, |\mathcal{V}|$. Our objective is to determine $f_{|\mathcal{V}|}(Z_1, \bar{Z}_1, \ldots, Z_K, \bar{Z}_K)$, which corresponds to the objective in (5) at the optimal solution $\{x_v^*\}$. The latter can then be obtained by backtracking from $f_{|\mathcal{V}|}(Z_1, \bar{Z}_1, \ldots, Z_K, \bar{Z}_K)$.

Completing $f_v(\cdot)$ requires a total running time of $O(|\mathcal{V}|KM)$, where $M = \max_i\{Z_i, \bar{Z}_i\}$. Thus, this approach represents a pseudo-polynomial time algorithm for solving (5).

We proceed one step further to formulate a fully-polynomial time approximation scheme [15] for solving (5) that will leverage the above development. In particular, we first scale the benefit factors α_{in} associated with caching item n at small base station i, such that they are all small numbers, polynomially bounded in $|\mathcal{V}|$. Applying dynamic programming via the optimal value function, as formulated above, to the scaled instance of (5) would then result in a polynomial running time (in $|\mathcal{V}|$) solution strategy, with an induced approximation factor ϵ. Moreover, we integrate a desired ϵ into the scaling of the factors α_{in}, so that this strategy is fully-polynomial time, i.e., with respect to $1/\epsilon$, as well.

Let $\alpha^{\max} = \max_{i,n} \alpha_{in}$ denote a scaling factor we will use. Let $p = \lfloor \log(\epsilon \alpha^{\max} / |\mathcal{V}|) \rfloor$ capture the precision at which we will approximate/quantize the benefit factors α_{in}. Then, we define $\alpha_{in}^s = \lfloor \frac{P_{in} Q_n / C_{i,n}^i}{10^p} \rfloor + \sum_{k\neq i} \lfloor \frac{P_{kn} Q_n / \bar{C}_n}{10^p} \rfloor$.

Algorithm 1 outlines our efficient branch-and-prune algorithmic implementation of the above strategy, as noted earlier.

Algorithm 1 Branch-And-Prune Approximation Scheme

```
1: Initialize $\mathbf{T} = \emptyset, F_0 = 0, \mathbf{Q}_0 = \{(\mathbf{T}, F_0)\}$
2: for $\forall v \in \mathcal{V}$ do
2:   Expansion Phase
3:   for $\forall(\mathbf{T}, P_{v-1}) \in \mathbf{Q}_{v-1}$ do
4:     Branch $\mathbf{Q}_v = \mathbf{Q}_{v-1} \cup \{(\mathbf{T} \cup \{v\}, F_{v-1} + \alpha_v)\}$
       Subject to:
4:       (i) $\sum_{v' \in \mathbf{T}} B_{n'} + B_n \leq Z_i \wedge \sum_{v' \in \mathbf{T}} \bar{B}_{n'} + \bar{B}_n \leq \bar{Z}_i$
         (ii) If $\nexists v' \in \mathbf{T} : n' = n$,
              Set $\alpha_v = \alpha_{in}^s$
         (iii) If $\exists v' \in \mathbf{T} : n' = n \wedge i' \neq i$,
              Set $\alpha_v = \lfloor \frac{P_{in} Q_n / C_{i,n}^i}{10^p} \rfloor - \lfloor \frac{P_{kn} Q_n / \bar{C}_n}{10^p} \rfloor$
5:   end for
5:   Prunning Phase
5:   if $\exists(\mathbf{T}', F_v'), (\mathbf{T}'', F_v'') \in \mathbf{Q}_v$
       Subject to:
       (i) $F_v' = F_v''$, and
       (ii) $\sum_{v \in \mathbf{T}'} B_n > \sum_{v \in \mathbf{T}''} B_n \wedge \sum_{v \in \mathbf{T}'} \bar{B}_n \geq \sum_{v \in \mathbf{T}''} \bar{B}_n$,
          or
       (iii) $\sum_{v \in \mathbf{T}'} B_n \geq \sum_{v \in \mathbf{T}''} B_n \wedge \sum_{v \in \mathbf{T}'} \bar{B}_n > \sum_{v \in \mathbf{T}''} \bar{B}_n$,
6:   then prune $\mathbf{Q}_v = \mathbf{Q}_v \setminus \{(\mathbf{T}', F_v')\}$
7:   end if
8: end for
9: Select $(\mathbf{T}^*, F_{|\mathcal{V}|}^*) : F_{|\mathcal{V}|}^* = \arg\max_{F_v} \{(\mathbf{T}, F_{|\mathcal{V}|}) \in \mathbf{Q}_{|\mathcal{V}|}\}$
```

In particular, Algorithm 1 leverages an efficient tree data structure $\mathbf{Q}_v$ that is dynamically updated during execution. Member elements $(\mathbf{T}, F_v) \in \mathbf{Q}_v$ comprise subsets $\mathbf{T}$ of size $\leq v$, of the first v elements in $\mathcal{V}$, such that they induce the maximum achieved benefit (F_v), given the constraints (2) - (4). For every subsequent v, Algorithm 1 comprises *an expansion phase*, where the optimal paths (subsets of cached data) maintained in $\mathbf{Q}_{v-1}$ are branched out by considering the next decision variable v (to cache item n at base station i), while observing constraints (2) - (4), and *a pruning phase*, where only the optimal paths after the expansion are retained.

At completion of stage $v = |\mathcal{V}|$, Algorithm 1 terminates by selecting the caching configuration $\mathbf{T}^*$ in $\mathbf{Q}_{|\mathcal{V}|}$ that exhibits the maximum achieved benefit $F_{|\mathcal{V}|}^*$. Then, the corresponding optimal streaming variables can be selected as follows. First, $\forall\, v \in \mathbf{T}^*$, we set $X_{in}^i = 1$. Next, if $\exists\, i, k \neq i : X_{in}^i = 1 \wedge X_{kn}^m = 0, \forall\, m$, we set $X_{kn}^i = 1$. Last, if $\exists\, i : X_{in}^k = 0, \forall\, k$, we set $X_{in}^{K+1} = 1$.

Finally, we verify the approximation guarantees of Algorithm 1. Let OPT denote the optimal objective in (5) and let $\{X_{in}^k\}^*$ denote the respective solution. We want to verify that the solution $\{X_{in}^k\}'$

computed by Algorithm 1 satisfies $O(\{X_{in}^k\}') \geq (1-\epsilon)\cdot\text{OPT}$, where $O(\cdot)$ denotes the objective function in (5). In particular, Algorithm 1 operates on scaled benefit factors, where $\Delta = \epsilon\alpha^{\max}/|\mathcal{V}|$ denotes the scaling aspect. Thus, the benefit α_{in}^s achieved by selecting item v in the solution $\{X_{in}^k\}'$ will satisfy $\Delta\alpha_{in}^s \leq \alpha_{in}$. This implies that the achieved benefit induced by $\{X_{in}^k\}'$ can drop at most Δ, for every item v cached according to $\{X_{in}^k\}^*$. Hence, we can bound the overall achieved benefit drop as $O(\{X_{in}^k\}^*) - \Delta \cdot O'(\{X_{in}^k\}^*) \leq |\mathcal{V}|\Delta$. Here, O' denotes the objective function in (5) evaluated on the scaled benefit factors.

On the other hand, the solution $\{X_{in}^k\}'$ computed by Algorithm 1 represents the optimal solution for the scaled instance of the problem (5), (2) - (4). Thus, $O'(\{X_{in}^k\}') \geq O'(\{X_{in}^k\}^*)$. Leveraging these two inequality relationships, we can write

$$\begin{aligned} O(\{X_{in}^k\}') \geq \Delta \cdot O'(\{X_{in}^k\}') &\geq \Delta \cdot O'(\{X_{in}^k\}^*) \\ &\geq O(\{X_{in}^k\}^*) - |\mathcal{V}|\Delta \qquad (6) \\ &= \text{OPT} - \epsilon\alpha^{\max} \\ &\geq (1-\epsilon)\cdot\text{OPT} \qquad (7) \end{aligned}$$

where (6) follows from the first inequality relationship established earlier and (7) holds as $\text{OPT} \geq \alpha^{\max}$. This verifies the desired approximation guarantees of Algorithm 1.

The running time of Algorithm 1 is polynomial in $|\mathcal{V}|$, as it corresponds to completing a table of at most $|\mathcal{V}|^2\lfloor\alpha^{\max}/\Delta\rfloor$ entries. The scaling enables the running time of Algorithm 1 also to be polynomial in $1/\epsilon$, as $|\mathcal{V}|\alpha^{\max}/\Delta = 1/\epsilon$.

D. Streaming network coding packets

Using network coding packets reduces the complexity of (1) - (4), as the decision variables Y_{kj}^l and $X_{ij}^{l,k}$ can be continuous in that case. Thus, (1) - (4) becomes linear programming, which can be solved exactly in polynomial time [18]. In particular, relaxing $Y_{kj}^l, X_{ij}^{l,k} \in [0,1]$ to be fractional will capture that now portions of the network coding packets representing tile-video j featuring l scalable layers can be cached at small base station k, and streamed from this base station to users at small base station i, respectively. Hence, the objective (1) will become a linear weighted-sum function of continuous variables indicating the proportions of network coding packets associated with a specific tile-video streamed to users at a given small base station, from each base station. Moreover, the constraints (2) - (4) will become linear functions as well and will still hold after relaxing the original discrete decision variables. Specifically, all constraints from (2) - (4) will apply in a straightforward manner, with for instance $Y_{kj}^l B_j^l$ and $X_{ij}^{l,k} B_j^l$ indicating in this case the data volumes associated with the proportions of network coding packets representing tile-video j comprising l scalable layers cached at small base station k, and streamed from this small base station to users at base station i, respectively.

VI. Experiments

For performance evaluation, we implemented our setting from Figure 1 as follows. There are three small base stations and five users assigned to each base station. There are eight 360° videos the users can request for streaming. These are Coaster, Wingsuit, Paris, Basketball, Runner, Angel Falls, Kite Flite, and Dolphins, featuring spatial resolutions of 4K or 8K, and frame rates of 30 or 60 fps. We constructed our scalable 360° tiling to feature $L = 3$ scalable layers, Group of Pictures (GOP) size of 30 frames, and 6×4 spatial tiles. The reference methods used in our evaluation use instead the above 360° videos compressed using HEVC into GOPs of 30 frames. We related the cost factors $C_{i,j}^{l,k}$ from Section V to the energy consumption of streaming the 360° content, proportional to the number of network hops the data needs to traverse to the destination. We considered that the energy per bit consumed by wireless transmission is 3.5 μJ and that for wireline transmission is 0.5 μJ per hop [19, 20]. To examine the pure streaming performance of our 360° tiling and the associated viewport-adaptive allocation of resources, we implemented a state-of-the-art method based on MPEG-DASH [21]. We also implemented two caching methods, *Blasco2014* [9] and *LRU* (Least-Recently-Used), to compare against our cooperative edge-based multi-user VR streaming and the associated Algorithm 1 for optimal allocation of storage and computing resources.

Advantages of scalable tiling and user-viewport adaptation. We carried out this experiment. A sender transmits 360° content to a VR client over a network link of a given data rate C (see Fig. 2). We implemented our optimization from Section IV-D to allocate C across the scalable tile layers of the 360° content, in response to the navigation actions of the user, and measured the resulting expected immersion fidelity experienced by the user. We formally measure the latter as the expected Y-PSNR of the user's viewport. We measured the respective performance of the reference method noted earlier. We used the content Wingsuit and Angel Falls in this experiment.

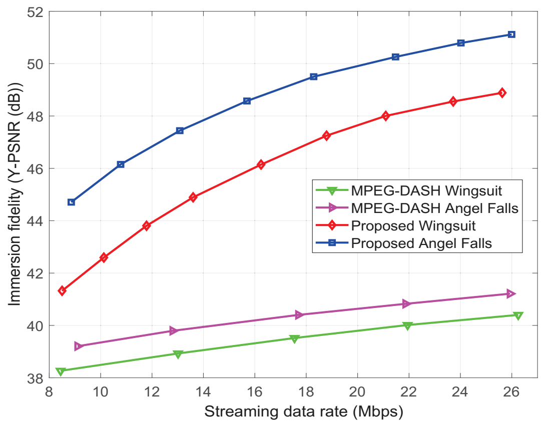

Fig. 6: Transmission efficiency vs. MPEG-DASH.

We can see from Figure 6 that our approach enables considerable benefits over MPEG-DASH by integrating the user navigation actions and the rate-distortion trade-offs across the 360° panorama, in deciding how transmission resources should be allocated. Approximately 6-7 dB of immersion fidelity improvement have been enabled across both 360° sequences and all streaming data rates considered in Figure 6. These advances can in turn enable much higher operational efficiency for a streaming system for 360° video delivery that integrates our methodology from Section IV, as our subsequent results demonstrate.

End-to-end system performance. In Figure 7, we study the energy consumption of the several methods under comparison, as a function of the available storage capacity of the base stations' edge servers. We set the computing power $\bar{Z}_i$ of each server to 65W [22]. We can see that our approach considerably outperforms the two reference methods, demonstrating three to six times lower energy consumption across the entire range of x-axis values examined in Figure 7. These gains are enabled by our optimization framework that can optimally pool and allocate the transmission, storage, and computing resources available at the small base stations. Moreover, our approach enables much higher energy consumption versus storage capacity efficiency, for low values of the latter, as observed in Figure 7.

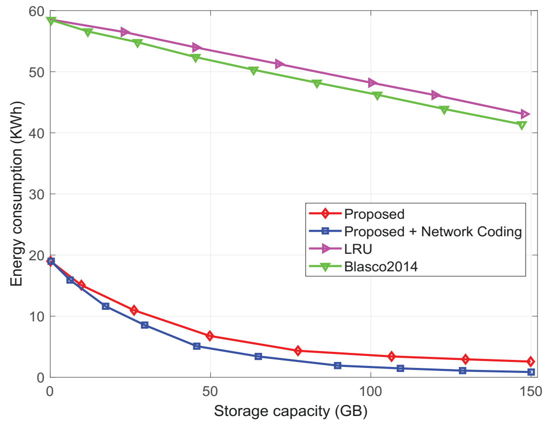

Fig. 7: Energy consumption vs. storage capacity at base stations.

As expected, the reference method *Blasco2014* outperforms the other reference method *LRU* due to the optimization it implements, though its gains appear to be marginal. Finally, we also evaluated the performance of our framework when operating on network coding packets. We can see from Figure 7 that this introduces further energy consumption savings, as in this case the optimization (1) - (4) can be solved exactly, as explained earlier. This then leads to even higher utilization of the available system resources, as expected.

We also studied the energy consumption of the three competing methods, as a function of the available computing capacity of the edge servers collocated with the small base stations. These results follow the same trends and relative performances as those observed in Figure 7 and cannot be included here due to limited space.

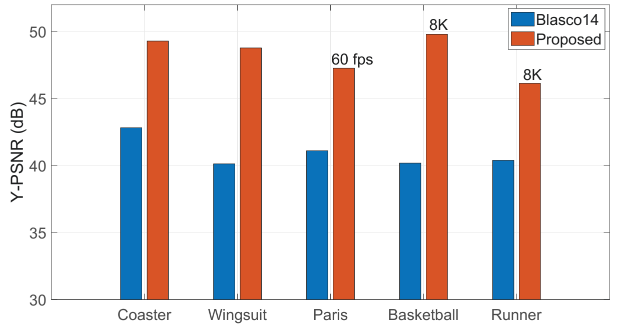

Fig. 8: Delivered immersion fidelity across different 360° videos.

Finally, in Figure 8, we assessed the delivered immersion fidelity across the different 360° videos streamed in our system. We can see that across all five 360° videos considered in Figure 8 our approach enables consistently significant gains in immersion fidelity delivered to the mobile users, ranging between six to ten decibels of the viewport Y-PSNR for a mobile VR user. Moreover, our approach enabled delivering some 360° videos also at higher spatial resolutions or temporal frame rates, thereby augmenting further the user's immersion quality of experience. In particular, the video Paris was delivered at temporal frame rate of 60 fps by our approach, as indicated in Figure 8, while the reference method *Blasco2014* could only stream it at 30 fps. Similarly, our approach enabled delivering the videos Basketball and Runner at higher 8K spatial resolution for the 360° panorama, as indicated in Figure 8, while *Blasco2014* could only stream them at 4K spatial resolution. The results for the second reference method *LRU* are not included, to avoid cluttering Figure 8, as they are lower than those for *Blasco2014*.

VII. Conclusion

We explored a novel communications system that integrates for the first time scalable multi-layer 360° video tiling, viewport-adaptive rate-distortion optimal resource allocation, and VR-centric edge computing and caching, to enable next generation high-quality untethered VR streaming. Our system comprises a collection of 5G small cells that can pool their communication, computing, and storage resources to collectively deliver scalable 360° video content to mobile VR clients at much higher quality. The major contributions of the paper are the rigorous design of multi-layer 360° tiling and related models of statistical user navigation, and analysis and optimization of edge-based multi-user VR streaming that integrates viewport adaptation and server cooperation. We also investigated the possibility of network coded data operation and its implications for the analysis, optimization, and system performance we pursue in this setting. The advances introduced by our framework over the state-of-the-art comprise considerable gains in delivered immersion fidelity, featuring much higher 360° viewport peak signal to noise ratio (PSNR) and VR video frame rates and spatial resolutions.

References

[1] J. G. Apostolopoulos, P. A. Chou, B. Culbertson, T. Kalker, M. D. Trott, and S. Wee, "The road to immersive communication," *Proceedings of the IEEE*, vol. 100, no. 4, pp. 974–990, Apr. 2012.
[2] T. Merel, "Ubiquitous $90 billion AR to dominate focused $15 billion VR by 2022," Digi-Capital Research, Jan. 2018.
[3] B. Begole, "Why the Internet pipes will burst when virtual reality takes off," Forbes Magazine, Feb. 2016.
[4] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, Apr. 2011.
[5] C. Ozcinar, A. De Abreu, and A. Smolic, "Viewport-aware adaptive 360° video streaming using tiles for virtual reality," in *Proc. IEEE Int'l. Conf. on Image Processing*, Beijing, China, Sep. 2017.
[6] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck, "Improving virtual reality streaming using HTTP/2," in *Proc. ACM Multimedia Systems Conference*, Jun. 2017, pp. 225–228.
[7] M. Hosseini and V. Swaminathan, "Adaptive 360 VR video streaming: Divide and conquer," in *Proc. IEEE Int'l Symp. Multimedia*, Dec. 2016.
[8] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 22, no. 12, Dec. 2012.
[9] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE Int'l Conf. Comm.*, Jun. 2014.
[10] K. Shanmugam, N. Golrezaei, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," *IEEE Trans. Information Theory*, Dec. 2013.
[11] H. Ahlehagh and S. Dey, "Video caching in radio access network: impact on delay and capacity," in Proc. *IEEE WCNC*, 2012.
[12] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE J. Selected Areas in Communications*, Aug. 2016.
[13] Y. Sun, A. Lu, and L. Yu, "Weighted-to-spherically-uniform quality evaluation for omnidirectional video," *IEEE Signal Processing Letters*, vol. 24, no. 9, pp. 1408–1412, Sep. 2017.
[14] N. Thomos, J. Chakareski, and P. Frossard, "Prioritized distributed video delivery with randomized network coding," *IEEE Trans. Multimedia*, vol. 13, no. 4, pp. 776–787, Aug. 2011.
[15] V. Vazirani, *Approximation Algorithms*, 2nd ed. Springer-Verlag, 2003.
[16] S. Martello and P. Toth, *Knapsack problems*. Wiley New York, 1990.
[17] R. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*, 1962.
[18] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Programming*, 3rd ed. Athena Scientific, Feb. 1997.
[19] J. Huang, F. Qian, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in Proc. *ACM MobiSys*, 2012.
[20] V. Sivaraman, A. Vishwanath, Z. Zhao, and C. Russell, "Profiling per-packet and per-byte energy consumption in the netfpga gigabit router," in Proc. *IEEE INFOCOM Workshops*, 2011, pp. 331–336.
[21] M. Graf, C. Timmerer, and C. Mueller, "Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP: Design, implementation, and evaluation," in *Proc. ACM Multimedia Syst.*, Jun. 2017.
[22] "CPU power dissipation," Wikipedia Online. [Online]. Available: https://en.wikipedia.org/wiki/CPU_power_dissipation/