Optimizing Adaptive Tile-Based Virtual Reality Video Streaming

Jeroen van der Hooft[†], Maria Torres Vega[†], Stefano Petrangeli[§], Tim Wauters[†] and Filip De Turck[†]
† IDLab, Department of Information Technology, Ghent University - imec
§ Adobe Research
E-mail: jeroen.vanderhooft@ugent.be

***Abstract*—The increasing popularity of head-mounted displays and 360° video cameras has encouraged content providers to provide virtual reality video streaming over the Internet, using HTTP adaptive streaming to deliver a two-dimensional representation of the immersive content. However, since only a limited part of the video (i.e., the viewport) is watched by the user, the available bandwidth is not optimally used. Recently, adaptive tile-based video streaming has been proposed; rather than sending the whole 360°video at once, the video is cut into temporal segments and spatial tiles. Each tile can be requested at a different quality level, giving priority to content within the viewport. This results in higher video quality and an increased bandwidth utilization. In this paper, we address three open research questions, concerning viewport prediction, tile-based rate adaptation, and application layer optimizations. First, we present a content-agnostic viewport prediction scheme based on spherical walks. Second, we introduce a new rate adaptation heuristic for tile-based video, which prioritizes tiles according to the great-circle distance between the viewport's and the tile's center. Third, we investigate the advantages of using HTTP/2 server push. We show that the proposed optimizations result in significant improvements in terms of viewport prediction error and video quality, compared to state-of-the-art solutions.**

I. Introduction

The popularity of virtual reality (VR) has increased significantly over the last years. Well-known content providers such as YouTube[1] and Facebook[2] now allow streaming VR content up to 4K resolution. To provide an acceptable user experience, the principles of HTTP Adaptive Streaming (HAS) are often applied to deliver the immersive video over the best-effort Internet. In HAS, the video is encoded at different quality levels and temporally divided into multiple segments with a typical length of 2 to 10 seconds [1]. Using a rate adaptation heuristic, the client requests these segments at the most appropriate video quality, based on e.g., the available bandwidth and the amount of buffered content. The client stores the segments in a buffer, before decoding the sequence in linear order and playing the video out on the user's device.

Content providers typically use HAS to deliver 360° video in a similar manner as traditional video. First, the content is mapped on a two-dimensional representation, using e.g. equirectangular projection. The resulting video is then encoded at multiple resolutions, temporally divided and made available on an HTTP server. Since the quality is fixed for the whole view, the available bandwidth is suboptimally used: the user only has a limited view when wearing a head-mounted display (referred to as the viewport), so that a significant part of the bandwidth is wasted on content which is never consumed. To solve this issue, the equirectangular content can be split into $m \times n$ tiles of the same resolution. The client can then request each tile at a different quality level, prioritizing tiles within the viewport by assigning a higher video quality.

Tile-based HAS for VR introduces a number of challenges. First, tile-based video streaming is prone to user movements: even when the considered buffer contains a mere two seconds of video, the user could temporally perceive a lower quality when moving around within the video scene. Therefore, accurate predictions of the future viewport location are required. Second, the client's rate adaptation heuristic needs to take into account the new spatial dimension in the decision-taking process. This increases the heuristic's complexity, which attempts to provide a high video quality and avoid buffer starvation at all times. Third, since requests over HTTP are issued one by one, tile-based solutions are more likely to be affected by network latency. This is especially true for mobile networks, where round-trip times are in the order of 30-70 and 50 to 200 ms for 4G/LTE and 3G/HSPA(+) respectively [2].

In this work, we aim to address the above-mentioned challenges of tile-based HAS for VR video. The main contributions of this paper are thus threefold. First, a content-agnostic viewport prediction scheme is proposed based on unidirectional spherical trajectories. Second, a new rate adaptation heuristic is presented for tile-based quality assignment. This heuristic prioritizes tiles according to their great arc distance to the viewport center, resulting in a higher overall quality. Third, we revisit the application of HTTP/2 server push, in an effort to counter the impact of latency on tile-based HAS solutions. The proposed optimizations are evaluated in detail, showing significant improvement in terms of prediction error and video quality, compared to state-of-the-art solutions.

The remainder of this paper is structured as follows. In Section II, the general HAS architecture for VR streaming is presented and state-of-the-art solutions are discussed. The suggested optimizations are detailed in Section III, focusing primarily on viewport prediction and a novel rate adaptation heuristic. The experimental setup and results are presented in Section IV, before coming to final conclusions in Section V.

[1]https://www.youtube.com
[2]https://www.facebook.com

978-3-903176-15-7 © 2019 IFIP

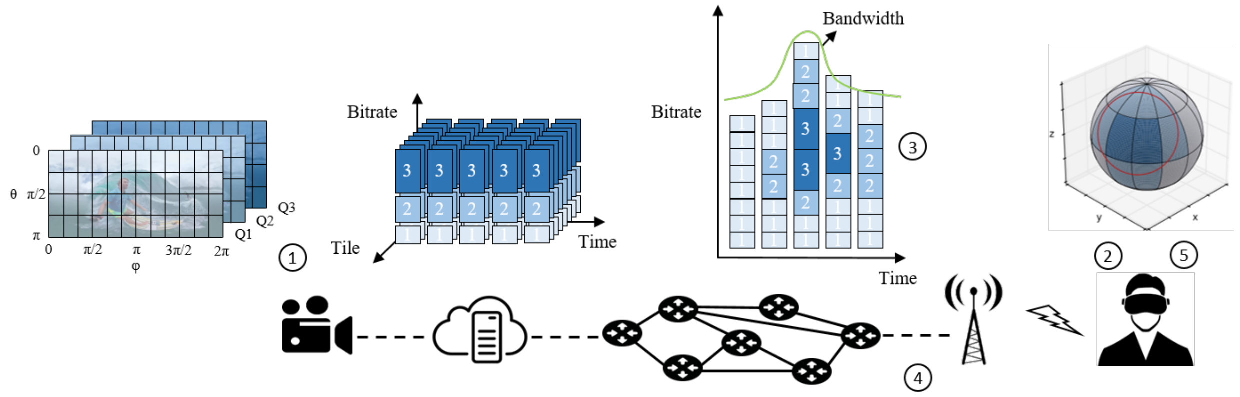

Figure 1. The HAS-based principle applied to VR video streaming.

II. State-of-the-Art and Challenges

As illustrated in Figure 1, the HAS principle can be adopted to provide tile-based VR video streaming. Content is captured by a 360°camera, encoded at different qualities, temporally and spatially segmented, and made available on the server. At client-side, a head-mounted display is used to consume the immersive content, registering the user's movement and regions of interest within the video. Based on the available bandwidth and current or future viewport coordinates, a rate adaptation heuristic decides at which video quality to download each of the tiles of the next video segment. The content is requested from the server, buffered in a client-side buffer and finally played out on the user's device. Below, we elaborate on the five most relevant components in this setup.

A. Video Capture & Encoding

Tile-based encoding is often achieved through the HEVC standard, implemented by encoders such as HEVC Test Model (HM[3]). Many schemes for tiling exist, including cubic, pyramid and dodecahedron mappings [3]. Still, the equirectangular mapping is most commonly used. This projection maps the sphere into a rectangle, resulting in stretching at the poles of the sphere. To overcome this, Budagavi et al. propose to gradually smoothen the quality of the polar parts of the video, reducing the bitrate for VR video by 20% [4]. Using a similar starting point, Hosseini et al. propose to use a different tiling structure altogether, using four central and two polar tiles [5]. Although there are benefits to these approaches, we will use traditional equirectangular projection in this work.

B. Viewport Prediction

Viewport prediction is an important means to tile-based HAS: if the future position is accurately predicted, the client can compensate for the user's movements and proactively download relevant parts of the video at higher quality. In related work, a distinction is made between content-agnostic and content-aware viewport prediction. Content-agnostic approaches take into account the location and/or angles of the viewport only, in order to predict how the user will move next. Petrangeli et al. propose to use linear extrapolation of the user's recent trajectory on the equirectangular projection of the video [6]. Given the user's most recent position P_c and a position P_p ΔT_p seconds ago, the user's position ΔT seconds from now is estimated by $P_c + \frac{\Delta T}{\Delta T_p}(P_c - P_p)$. Qian et al. propose (weighted) linear regression on the yaw, pitch and roll angles [7]. The authors show that the prediction accuracy decreases for a higher ΔT. Similarly, Xu et al. use linear regression to show that the variation of the prediction error increases for a higher ΔT [8]. Content-aware solutions also take into account information on the content itself, defining regions of interest, moving objects and key frames within the video. Focusing on tile-based VR streaming, Fan et al. use neural networks that concurrently leverage sensor-related features (i.e., the viewport position) and content-related features (i.e., image saliency and object motion maps) to predict the future viewport position [9]. Sitzmann et al. observe a fixation bias in immersive video, which is used to apply existing saliency predictors for two-dimensional video to VR [10]. Although there are benefits to content-aware solutions, a significant amount of user data and processing efforts are required for new video content. In this work, we will focus on content-agnostic approaches only.

C. Tile-Based Rate Adaptation

Adding an additional spatial dimension increases the complexity of the rate adaptation heuristic. Recently, a number of solutions have been proposed to address the extension to the spatial domain. Le Feuvre et al. propose a heuristic based on fixed tile priorities [11]. A number of priority schemes are presented, including uniform and center-based priorities. The authors do not consider viewport changes, however, and evaluate results for a non-interactive environment (segment duration 10 s). Hosseini et al. propose a rate adaptation heuristic which is based on three regions of priority: the tile at the center of the viewport, the tiles surrounding this tile (for a maximum of eight tiles), and all others [5]. Zone per zone, the quality of each tile is increased from the lowest to the highest level, as long as the available bandwidth is not exceeded. Then, the quality of the last considered tile is increased to the highest quality supported by the remaining bandwidth budget.

[3]https://hevc.hhi.fraunhofer.de/HM-doc/

Petrangeli et al. propose a scheme based on center and polar zones [6]. Starting from a 4×4 tiling sheme, the top and bottom row tiles are concatenated, and the tiles on the second and third row are concatenated column-wise. The resulting six tiles are divided on a per-zone level, depending on the current and predicted viewport position. Then, qualities to tiles are assigned zone by zone. Using this approach allows to reduce the number of GET requests by the concatenation of tiles. Although this is important in high-latency environments, the granularity of the video i strongly reduced.

D. Application Layer Optimization

Temporally and spatially splitting the video in multiple parts results in an increased number of GET requests issued by the client. When the requested files are small and the network latency is high, this results in idle round-trip time cycles and unused bandwidth resources.To mitigate these effects, HTTP/2 server push has been proposed to actively push resources from server to client. Pushing shorter video segments back to back, for instance, can reduce the video's startup time and camera-to-display delay in HAS [12]. Recently, Petrangeli et al. showed that server push can also be applied in the case of tile-based VR streaming, where the client can request the server to push all tiles belonging to a single segment simultaneously [6]. Results show that this approach results in a higher perceived throughput, and thus in a higher video quality with fewer playout freezes. The application of HTTP/2 server push opens other opportunities as well. For instance, if a scenario is considered in which playout continues even when not all tiles for the current segment have arrived (in which case a black box is shown for the corresponding tiles), prioritization of certain tiles at the application layer can be defined by the distance between each tile and the predicted viewport center. In this way, relevant tiles can be delivered sooner, and playout of the viewport can continue/resume unimpeded.

E. Video Quality Evaluation

Compared to regular HAS, the application of tiles introduces new factors which affect the user's Quality of Experience (QoE). For instance, frequent quality switching on the temporal and on the spatial level can result in an unpleasant and even nauseating experience. Furthermore, the speed with which the quality is adjusted when moving around plays a crucial role as well: if the user has to wait several seconds before the quality is adjusted, the QoE will be strongly affected. With regard to the video quality perceived within the viewport, a number of evaluation metrics have been used in related work. Some works report the average peak signal-to-noise ratio (PSNR) values for the obtained video [13], [8], or show results in terms of video bitrate [8], [13], [14]. Other works consider the quality of the tile in the center of the viewport only, averaging this value over all segments, or to measure the time spent on each layer [6]. Although the latter has shown to be an excellent evaluation metric for regular HAS, it is uncertain whether it is directly applicable to VR: the (potentially lower) video quality of surrounding tiles can have a significant impact on the overall perceived quality. Rai et al. recently showed that the users eyes are rarely fixed to the center of the viewport; rather, a peak is observed for angles between 12 and 20 degrees [15]. In previous work, we therefore proposed to weigh the quality of each of the tiles within the viewport, taking into account the distribution of the user's gaze [16]. Sampling points from the viewport according to this distribution, the overall video quality is expressed as the average video quality over all these points. The quality in a point can be defined as the quality level (ranging from 1 (lowest) to q max (highest)), the average bitrate, the PSNR or the structural similarity index (SSIM) of the corresponding tile.

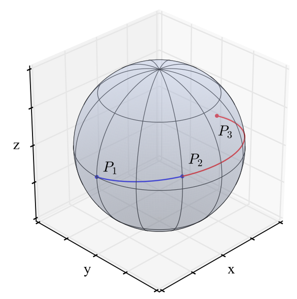

Figure 2. Viewport prediction using spherical walks. When the user moves from point P_1 to P_2 in a certain amount of time (e.g., $100\,\mathrm{ms}$), point P_3 is predicted by extending the current trajectory unidirectionally.

III. Approach

In the previous section, the state-of-the-art for VR-based HAS was discussed, along with its challenges and possible ways to tackle these. Below, we present the different approaches and optimizations adopted in this paper. These encompass a content-aware viewport prediction scheme, a new rate adaptation heuristic for tile-based quality selection and on application layer optimizations through HTTP/2.

A. Viewport Prediction

We propose a content-agnostic approach for viewport prediction, extending the solution adopted by Petrangeli et al.[6]. This extension is twofold. First, we consider user movement to be a trajectory on the surface of the unit sphere, rather than a two-dimensional one over the equirectangular projection. When a user has covered a certain distance in a given time interval, the current path is extended unidirectionally to predict the user's future trajectory (Figure 2). This approach reflects the user's movement within the video scene more naturally, and should therefore result in a higher prediction accuracy. Second, we recognize that the user's movement is largely volatile, with changes occurring in the order of hundreds of milliseconds. Even when the user has been showing a unidirectional trajectory in the recent past, chances are that this trajectory changes significantly in the near future. Therefore, we propose to predict the next viewport coordinates not by completing the full trajectory, but rather a fraction thereof. This reduces the impact of changes in the user's movement, and should result in a lower prediction error.

Algorithm 1 Proposed rate adaptation heuristic.

```
Input: $s$, the segment number (starting at 1)
       $m, n$, the number of rows and columns
       $BW$, the available bandwidth $[b/s]$
       $dur$, the segment duration $[s]$
       $buffer$, the buffer size $[s]$
       $n_{qual}$, the number of qualities (1 to $n_{qual}$)
       $vp$, the width of the viewport $[deg]$
       $\phi, \theta$, the spherical coordinates of the viewport
Output: $qualities$, a list specifying the quality for each tile
 1: $qualities \leftarrow ones(m \cdot n)$
 2: if $s \leqslant buffer/dur$ then
 3:     return $qualities$
 4: $budget \leftarrow BW \cdot dur$
 5: $tiles \leftarrow [1; m \cdot n]$
 6: if $\sum_{t \in tiles} Size(s, t, 1) \geqslant budget$ then
 7:     return $qualities$
 8: else if $\sum_{t \in tiles} Size(s, t, n_{qual}) \leqslant budget$ then
 9:     return $qualities \cdot n_{qual}$
10: $distances \leftarrow [Distance(t, \phi, \theta), \forall t \in tiles]$
11: $tiles_{in} \leftarrow [t, \forall t \in tiles : distances[t] \leqslant vp/2]$
12: $tiles_{out} \leftarrow tiles \setminus tiles_{in}$
13: $n_{bits} \leftarrow \sum_t Size(s, t, 1)$
14: for $tiles \in [tiles_{in}, tiles_{out}]$ do
15:     for $q \in [2; n_{qual}]$ do
16:         for $t \in Sort(tiles, distances)$ do
17:             $cost \leftarrow Size(s, t, q) - Size(s, t, q-1)$
18:             if $n_{bits} + cost > budget$ then
19:                 return $qualities$
20:             $n_{bits} \leftarrow n_{bits} + cost$
21:             $qualities[t] \leftarrow q$
```

B. Tile-Based Rate Adaptation

The proposed rate adaptation heuristic is presented in Algorithm 1. Given the segment index s and the perceived bandwidth BW, this heuristic determines the most appropriate quality level for each of the video tiles. If the client has just started buffering, all tiles are downloaded at the lowest quality (lines 1-3). When enough segments have been retrieved, the total budget is calculated, based on the available bandwidth and the segment duration dur (line 4). If the total file size of all tiles at the lowest quality exceeds this budget, the lowest quality is selected for each tile (lines 5-6). Similarly, if the total file size of all tiles at the highest quality does not exceed this budget, the highest quality is selected for each tile (lines 7-8). If none of these conditions is met, the heuristic will proceed to allocate the tile qualities. First, the great arc distance between the center of the viewport, indicated by ϕ, θ, and the center of each of the tiles is calculated (lines 10-11). Then, tiles are divided according to whether or not the center is within the viewport (i.e., the distance is lower than half the viewport size vp_{deg}), and the total number of bits for the current configuration (at the lowest quality) is calculated (lines 11-13). Next, the heuristic attempts to increase the quality of each tile within the viewport until either all tiles have the same quality, or the budget has been exceeded (lines 14-22). This process is repeated until either the budget is exceeded or all the tiles within the viewport are assigned the highest video quality. If there is still budget left, a similar process starts for each of the remaining tiles. The choice for a more homogeneous quality within the viewport is a deliberate one: the human eye is sensitive to changes in the perceived quality, especially when changes occur within the spatial domain (contrary to regular HAS, where changes in video quality only occur in the temporal domain). Rather than increasing the video quality of the closest tiles to the highest quality, the available bandwidth is spread across the whole viewport.

CRF	Sandwich	Spotlight	Surf
15	21.9 ± 6.6	20.8 ± 13.9	26.4 ± 12.7
20	10.3 ± 3.2	10.6 ± 8.9	16.7 ± 8.7
25	4.5 ± 1.4	5.2 ± 5.1	9.6 ± 5.4
30	2.2 ± 0.7	2.6 ± 2.7	4.8 ± 2.8
35	1.2 ± 0.3	1.4 ± 1.3	2.4 ± 1.4

Table I
OBTAINED BITRATES [MB/S] FOR THE THREE VIDEOS USING A 4×4 TILING SCHEME, PRESENTING THE AVERAGE VALUE AND THE STANDARD DEVIATION AMONG ALL SEGMENTS.

C. Application Layer Optimization

Similar to the approach adopted by Petrangeli et al., we will use HTTP/2 server push to deliver all tiles of each segment back-to-back [6]. To enable this, a custom request handler will be used at server-side, allowing the client to define a list of quality levels for each of the tiles to retrieve. The decisions of the applied rate adaptation heuristic can thus be plugged in to retrieve all tiles at the desired quality level. As discussed in Section II, adopting this approach will eliminate idle RTT cycles, resulting in higher throughput and video quality when the latency is high.

IV. EVALUATION AND DISCUSSION

To evaluate the proposed optimizations, a dataset provided by Wu et al. was chosen [17]. This dataset contains traces for 48 unique users and 9 different VR videos, specifying the coordinates of the viewport center throughout all video sessions with an average sampling rate of 47 Hz. From the list of videos, we selected three which show significantly different features: Conan (an indoor performance), Spotlight (an action movie) and Surf (an ensemble of surf clips made using a GoPro camera). Below, we first discuss the experimental setup, and then present results for each of the proposed optimizations.

A. Experimental Setup

Each of the considered videos is encoded using the HM encoder, applying a 4×4 tiling scheme at 4K resolution and 30 FPS. The Group of Pictures (GOP) length is set to 32, resulting in a segment duration of around 1.067 s. The CRF factor for five different quality representations is set to 15, 20, 25, 30 and 35, resulting in the average bitrates specified in Table I. To evaluate the impact of the available bandwidth and network latency on the resulting video streaming sessions, a network setup is emulated using MiniNet[4], where a client is connected to an HTTP/1.1- and HTTP/2-enabled Jetty server[5] (Figure 3). The open-source code of the Java-based server is

[4]http://mininet.org
[5]https://www.eclipse.org/jetty/

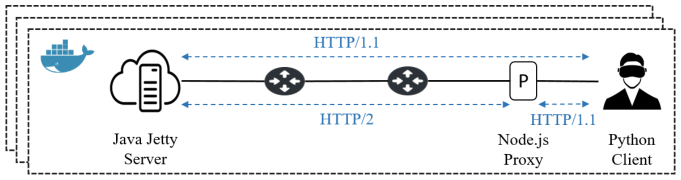

Figure 3. Experimental setup. MiniNet is used to host a virtual network within a Docker container. A Python-based VR player is used to play video streaming sessions from the HTTP/2-enabled Jetty server.

slightly modified, in order to provide a custom request handler for the pushing of tile-based video segments. The client is a headless Python-based implementation, which has been made available online[6]. This client provides support for different viewport prediction schemes, rate adaptation heuristics and quality metrics. A buffer size of two segments, or 2.133 s, is used in our evaluations. The viewport size is set to 110°, similar to the VIVE head-mounted display used by Wu et al. during data collection [17]. To allow seamless connection over HTTP/2, a Node.js proxy is provided. This proxy serves as a client to the HTTP/2-enabled server, forwarding the required push request and handling incoming files as an intermediate for the VR client. The complete setup is wrapped in a docker container, increasing portability and allowing parallel execution of video streaming sessions. Experiments are carried out on imec's Virtual Wall[7], with at most four docker containers running simultaneously on a hexacore Intel(R) Xeon(R) CPU E5645 @ 2.40GHz with 24 GB of RAM.

B. Evaluation Metrics

Using the provided traces and video content, the performance of the proposed approaches is evaluated. The viewport prediction accuracy, on the one hand, is evaluated by considering the great-circle distance between the predicted coordinates and the coordinates actually visited by the user. The video quality, on the other hand, is evaluated using two different metrics. First, our weighed quality metric ($n_1 = n_2 = 50$), taking into account the distribution of the user's gaze within the viewport [16]. Second, the average time spent on each quality layer for the tile corresponding to the viewport center.

C. Viewport Prediction

Given a buffer size of 2 s, we first tuned the settings of the proposed viewport prediction scheme. Based on a preliminary parameter sweep, we found that the best performance is obtained by observing the movement made during the last 100 ms, and continuing this movement on the great-distance circle for another 400 ms. Using these settings, we compared our proposed viewport prediction scheme with its equirectangular counterpart, and with an approach in which the most recent viewport coordinates are predicted. Figure 4 shows the average prediction error as a function of the average speed within the video session, for three configurations: i)

[6]https://github.com/jvdrhoof/VRClient

[7]https://doc.ilabt.imec.be/ilabt-documentation/

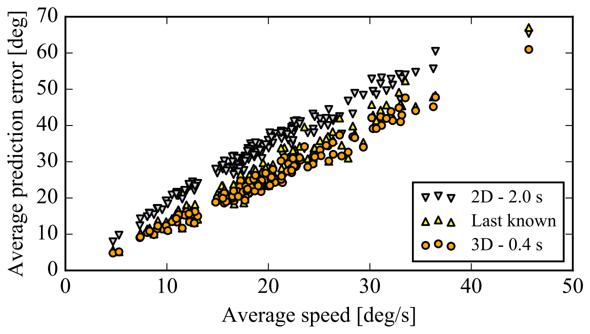

Figure 4. Average prediction error for the viewport center 2 s in the future, as a function of the average speed within the video session.

"2D - 2.0 s", the original prediction approach proposed by Petrangeli et al.; ii) "Last known", predicting the last known viewport location; iii) "3D - 0.4 s", the proposed approach based on spherical trajectories. Obtained results are shown for the three videos and all 48 users, or a total of 144 samples per configuration. From these results, we conclude that the proposed approach significantly reduces the average prediction error: at an average speed of $19.7\,\mathrm{deg/s} \pm 7.2\,\mathrm{deg/s}$, the average prediction error for the viewport center 2 s in the future is reduced from $33.7\,\mathrm{deg} \pm 10.7\,\mathrm{deg}$ and $26.5\,\mathrm{deg} \pm 10.5\,\mathrm{deg}$ to $25.0\,\mathrm{deg} \pm 9.7\,\mathrm{deg}$, a reduction of 25.8% and 5.7% respectively ($p < 0.001$). Furthermore, results indicate that the prediction accuracy is linearly dependent with the speed at which the user moves within the video. In the evaluations below, we will use the proposed approach to predict the future viewport location.

D. Rate Adaptation Heuristic

Next, we evaluate the performance of the proposed rate adaptation heuristic, and compare it with the polar heuristic proposed by Petrangeli et al [6]. Two values for the viewport size (the vp parameter in Algorithm 1) are used: 110° and 360°. While the former will result in an increased video quality within the viewport, the latter will increase the video quality for the whole video. To evaluate the heuristics under different network conditions, traffic control (tc) is used to fix the available bandwidth between client and server to values uniformly spread between 2 and $24\,\mathrm{Mb/s}$. The former will result in the client streaming the lowest quality only, while the latter should result in a high video quality within the viewport. Note that, in this paper, we do not evaluate the impact of variable bandwidth on the client's performance, and thus do not consider playout freezes; this will be left as future work.

Figure 5 shows the weighed video quality as a function of the available bandwidth, for thee videos and rate adaptation heuristics. From these results, we conclude that the proposed rate adaptation heuristic outperforms the other two approaches. This is because i) the heuristic proposed by Petrangeli et al. is more conservative, since all tiles belonging to the same zone are requested at the same quality; and ii) the viewport-aware approach ($vp = 110°$) prioritizes tiles within the viewport, rather than the whole video. Figure 6 shows results for the

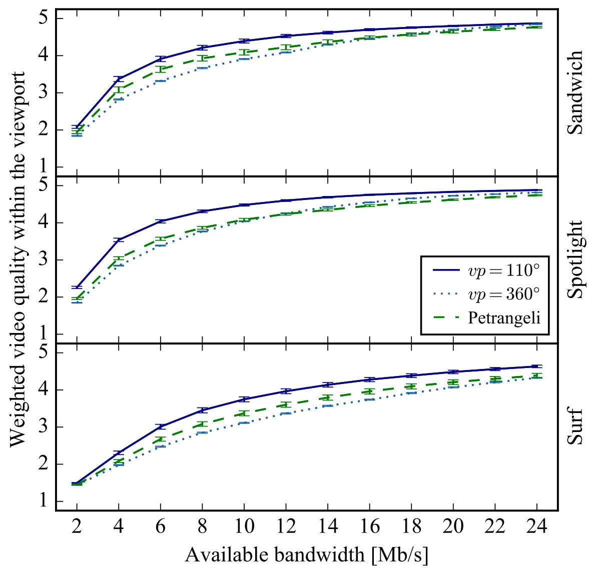

Figure 5. Observed video quality as a function of the available bandwidth, for a negligible network latency. Average values over all 48 users are shown, along with the standard deviation.

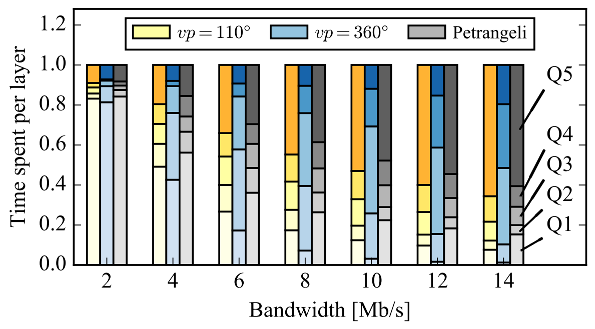

Figure 6. Time spent on each layer as a function of the available bandwidth, for the Surf video. Average values over all 48 users are shown.

Surf video in terms of the time spent on each quality layer (the quality of the tile corresponding to the viewport center). As an example, for a bandwidth of $8\,\mathrm{Mb/s}$, the relative time spent on the highest quality layer is 44.7% for $vp = 110°$, while this is 10.4% and 38.6% for $vp = 360°$ and Petrangeli, respectively. From these results, we conclude that our rate adaptation heuristic outperforms the other two.

E. HTTP/2 Server Push

In a final evaluation, both the bandwidth and the latency are changed using tc. Figure 7 shows results for the observed video quality as a function of the latency, for a bandwidth of 6 and $12\,\mathrm{Mb/s}$ and the proposed rate adaptation heuristic, with $vp = 110°$ and our viewport prediction approach. When HTTP/1.1 is used, the quality quickly decreases: this is an immediate consequence of the idle RTT cycles lost to retrieve the tiles for each segment. Indeed, $4 \times 4 + 1 = 17$ requests are issued for each segment, resulting in a total idle time of more than $500\,\mathrm{ms}$ when the network latency is $30\,\mathrm{ms}$. In contrast, when HTTP/2 is used, server push allows to deliver segments back-to-back, resulting in a similar throughput and thus allows

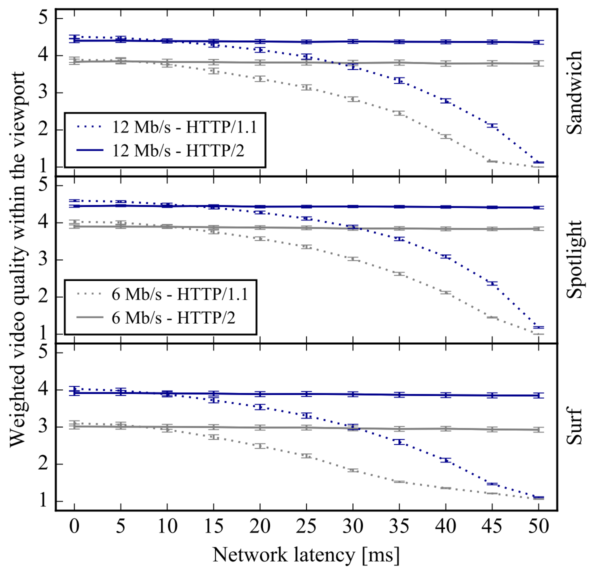

Figure 7. Observed video quality as a function of the network latency, for the proposed rate adaptation heuristic. Average values over all 48 users are shown, along with the standard deviation.

to maintain the video quality for higher latency values. It is worth noting that an improvement is observed in terms of video quality, even when the network latency is negligible. This is because the number of GET requests to the server and the overall download time for all tiles is reduced, resulting in a higher estimation of the available bandwidth. This, in turn, results in a slightly higher video quality for the next segment, with larger file sizes and therefore, more accurate and higher bandwidth estimations. Overall, we conclude that the use of HTTP/2 has a beneficial impact on the perceived video quality, especially in high-latency network scenarios.

V. Conclusions and Future Work

In this paper, we have presented solutions to three of the most important challenges of adaptive tile-based 360° video streaming. Using an existing dataset of viewport traces, where the movement of 48 users is tracked throughout immersive video sessions, we have shown that our proposed content-agnostic viewport prediction approach is able to reduce the prediction error significantly. Through emulation, we have proven that our novel tile-based rate adaptation heuristic outperforms other heuristics in terms of video quality and time spent per quality layer. Finally, we have shown that the application of HTTP/2 server push allows to counter negative effects introduced by network latency, especially given the many GET requests required by tile-based solutions.

In future work, we will perform a more extensive evaluation of the proposed optimizations under highly variable network conditions. We will also extend the proposed rate adaptation heuristic to take into account probabilistic information on the viewport position, rather than a single predicted value. Finally, we plan to investigate the benefits of HTTP/2's stream prioritization and termination on tile-based video streaming.

Acknowledgment

Jeroen van der Hooft is funded by grant of the Agency for Innovation by Science and Technology in Flanders (VLAIO). Maria Torres Vega is funded by grant of the Research Foundation - Flanders (FWO). This research was performed partially within the imec PRO-FLOW (150223) project and the FWO-funded "Optimized source coding for multiple terminals in self-organising networks" (G025615N) project.

References

[1] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[2] OpenSignal. (2018) State of Mobile Networks: USA (January 2018). [Online]. Available: https://opensignal.com/reports/2018/01/usa/state-of-the-mobile-network/

[3] M. Yu, H. Lakshman, and B. Girod, "A Framework to Evaluate Omnidirectional Video Coding Schemes," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, 2015, pp. 31–36.

[4] M. Budagavi, J. Furton, G. Jin, A. Saxena, J. Wilkinson, and A. Dickerson, "360 Degrees Video Coding Using Region Adaptive Smoothing," in *Proceedings of the IEEE International Conference on Image Processing*, 2015, pp. 750–754.

[5] M. Hosseini and V. Swaminathan, "Adaptive 360 VR Video Streaming: Divide and Conquer," in *Proceedings of the IEEE International Symposium on Multimedia*, 2016, pp. 107–110.

[6] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck, "An HTTP/2-Based Adaptive Streaming Framework for 360-Degree Virtual Reality Videos," in *Proceedings of the ACM Multimedia Conference*, 2017, pp. 306–314.

[7] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 Video Delivery over Cellular Networks," in *Proceedings of the Workshop on All Things Cellular: Operations, Applications and Challenges*, 2016, pp. 1–6.

[8] Z. Xu, X. Zhang, K. Zhang, and Z. Guo, "Probabilistic Viewport Adaptive Streaming for 360-Degree Videos," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2018, pp. 1–5.

[9] C. Fan, J. Lee, W. Lo, C. Huang, K. Chen, and C. Hsu, "Fixation Prediction for 360-Degree; Video Streaming in Head-Mounted Virtual Reality," in *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2017, pp. 67–72.

[10] V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein, "Saliency in VR: How Do People Explore Virtual Environments?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 4, pp. 1633–1642, 2018.

[11] J. Le Feuvre and C. Concolato, "Tiled-Based Adaptive Streaming Using MPEG-DASH," in *Proceedings of the International Conference on Multimedia Systems*, 2016, pp. 41:1–41:3.

[12] S. Wei and V. Swaminathan, "Low Latency Live Video Streaming over HTTP 2.0," in *Proceedings of the Network and Operating System Support on Digital Audio and Video Workshop*, 2014, pp. 37:37–37:42.

[13] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-Based HTTP Adaptive Streaming," in *Proceedings of the ACM Multimedia Systems Conference*, 2017, pp. 315–323.

[14] H. Ahmadi, O. Eltobgy, and M. Hefeeda, "Adaptive Multicast Streaming of Virtual Reality Content to Mobile Users," in *Proceedings of the Thematic Workshops of ACM Multimedia*, 2017, pp. 170–178.

[15] Y. Rai, P. Le Callet, and P. Guillotel, "Which Saliency Weighting for Omni Directional Image Quality Assessment?" in *Proceedings of the International Conference on Quality of Multimedia Experience*, 2017, pp. 1–6.

[16] J. van der Hooft, M. Torres Vega, S. Petrangeli, T. Wauters, and F. De Turck, "Quality Assessment for Adaptive Virtual Reality Video Streaming: A Probabilistic Approach on the User's Gaze," in *Proceedings of the International Workshop on Quality of Experience Management*, 2019, accepted for publication.

[17] C. Wu, Z. Tan, Z. Wang, and S. Yang, "A Dataset for Exploring User Behaviors in VR Spherical Video Streaming," in *Proceedings of the ACM Multimedia Systems Conference*, 2017, pp. 193–198.