An Optimized Adaptive Streaming Framework for Interactive Immersive Video Experiences

Maarten Wijnants - iMinds-tUL-UHasselt-EDM, Wetenschapspark 2, 3590 Diepenbeek, Belgium
Peter Quax - iMinds-tUL-UHasselt-EDM, Wetenschapspark 2, 3590 Diepenbeek, Belgium
Gustavo Rovelo Ruiz - iMinds-tUL-UHasselt-EDM, Wetenschapspark 2, 3590 Diepenbeek, Belgium
Wim Lamotte - iMinds-tUL-UHasselt-EDM, Wetenschapspark 2, 3590 Diepenbeek, Belgium
Johan Claes - Androme NV, Wetenschapspark 4, 3590 Diepenbeek, Belgium
Jean-François Macq - Alcatel Lucent Bell Labs, Copernicuslaan 50, 2018 Antwerpen, Belgium

***Abstract*—This paper describes how optimized streaming strategies, based on MPEG-DASH, can be employed to power a new generation of interactive applications based on immersive video. The latter encompasses ultra-high-resolution, omni-directional and panoramic video. The goal is to deliver experiences that are made up of multiple videos of short duration, which can be joined at run-time in an order defined through user interactions. Applications of the technology are widespread, ranging from virtual walkthroughs to interactive storytelling, the former of which will be featured in detail. The main technological challenges tackled in this paper are to deliver these experiences in a seamless fashion, at the highest quality level allowed by network conditions and on a wide range of platforms, including the Web. Besides these, the paper focuses on the two-tier software architecture of the proposed framework, as well as a short evaluation to substantiate the validity of the proposed solutions.**

I. Introduction

Streaming video on-demand is currently handled by a variety of technologies (Apple HLS, Microsoft Smooth), which are slowly being superseded by MPEG-DASH. These solutions offer major advantages to application developers and end-users alike: semi-automatic adaptation to bandwidth conditions, elimination of firewall issues and platform independence for playout components. However, most application scenarios for streaming video still focus on traditional (up to Full HD) use cases where a video sequence is considered a monolithic entity. Novel formats, referred to in this paper as immersive video, include ultra-high-resolution (4K and up), omni-directional (360°) and high-resolution panoramic video. Each of these is characterized by the fact that the resolution is much higher than the Full HD resolution typically employed by streaming applications. Additionally, in the case of omni-directional and panoramic video, only a part of the full video coverage is actually required for display at user side (i.e. the view frustum). Scalable solutions for delivering video in this way rely on the client device to request the appropriate spatial parts of the content from the server infrastructure – based on user interactions like panning and zooming. To support these application scenarios, tiled video transmission is typically used as a solution, whereby only those tiles contained within the user's viewport are requested from the server, transmitted and rendered on the end-user device. Although this technology is not the focus of this paper, it is an integral part of the proposed experience description framework.

Specifically in this paper, user interactivity is extended beyond the ability to change the virtual camera direction, by handing control of the entire flow of the immersive experience over to the end-user. In practice, this is accomplished by virtually 'pasting together' multiple video sequences, each of short duration (referred to throughout this paper as *chunks*), in such a way that choices can be presented to the user on how to proceed. A simple analogy, which will be elaborated on further in this paper, is that of a video counterpart to Google Street View. Where in the Street View case, the user needs to make decisions on how to proceed after each frame, video chunks are now used as a replacement for still imagery.

Several contributions are presented in this paper. First, the definition of a simple but flexible immersive experience graph description that can cover a number of application scenarios and technological challenges is discussed. The software architecture for the system is described, which decouples the streaming logic (back-end) from the presentation layer (front-end), the latter of which can be implemented on a range of platforms (e.g. native or Web-based). Finally, an evaluation is provided that shows the validity of the claims made on the flexibility as well as the real-world applicability of the system.

II. Related Work

Interactive video has been studied at large in literature; space limitations unfortunately compel us to include only the most recent and relevant references in this section.

In their basic form, interactive video experiences – where the user is in control over the sequencing – are sometimes referred to in literature as *Hypervideo* or *non-linear video*. A comprehensive overview of the distinction between these and other related terms is provided in [4]. The authors describe a system capable of defining the flow within such experiences through a visual editor. For the playback part, in [3], prefetching schemes are proposed that are similar to the ones that will be presented in this paper. It is important to note that the contribution and extension of the current publication should be regarded in the fact that the concepts are applied to non-traditional (immersive) video. This raises multiple additional technical challenges (e.g. increased bandwidth demands, tiled delivery, multi-depth-layer representations and increased interactivity) for which practically applicable solutions are now presented, mainly based on established standards.

Fig. 1: Example scene from the 360° walkthrough scenario

In [5], the extension of the concept of Hypervideo to 360° captured content is discussed. The authors mainly focus on how to present the content to the user and not on the underlying network aspects such as efficient delivery and caching. The ideas presented however certainly also apply to the cases described in the current paper. In particular, related to the walkthrough scenario described in section III, in a follow-up publication [6], the authors describe the extension of their ideas towards geo-referenced 360° Hypervideo. More details on tiled delivery of omni-directional video can be found in [7], [8] and [10]. A system exploiting depth information for enhanced playback purposes (see section V-C2) is Plenopticon [1], where dynamic adaptive depth of field is applied to a video sequence recorded using a plenoptic Lytro camera.

III. Example Use Case Description

To illustrate the capabilities of the proposed system as experienced by end-users, a prototype application has been developed. This technology demonstrator implements a specific usage scenario that will be elaborated on in this section. Please note that the presented scenario represents just a single, relatively straightforward example of how the proposed streaming architecture can be applied to develop a concrete immersive media experience. As said before, the technology can adapt to support a number of applications in a broad spectrum of application domains, with the main limitation being the creativity and imagination of content producers.

The prototype application is designed to familiarize foreign students and new employees with the Science Park grounds of Hasselt University by offering a virtual walkthrough of the site. The content consists of a set of chunks recorded on campus using an omni-directional setup consisting of 7 GoPro Hero3+ Black cameras mounted in a 360Heros rig (http://www.360heros.com/) and supported on a Steadicam Pilot mount (http://www.tiffen.com/pilot.html). The recorded video material includes a human guide who points out various features along the way (see Figure 1). A typical usage scenario involves users virtually moving along the roadways and choosing the desired traveling direction at various intersections in order to get a feel of the layout of the campus. While navigating the streets, users can play/pause the video sequences, change viewing direction or zoom in/out by manipulating the viewer component (either through dragging when using a mouse, or by using touch events when deployed on a tablet device). Alternatively, in the same application scenario, one may opt to display specific travel paths during different times of day (at dawn, in the afternoon, at dusk, ...) or in different climatic conditions (depending on content availability).

IV. Immersive Experience Graph

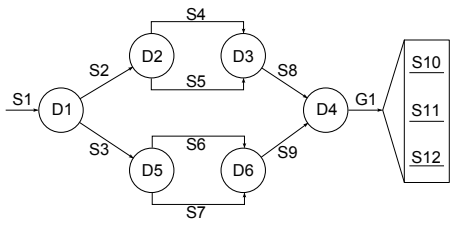
As the proposed streaming solution is targeting non-linear video consumption environments in which the end-user is given substantial freedom in determining the course of the media playback, a format is needed to capture the temporal relationships between constituting media chunks. The reader is reminded that the term *chunk* is used to refer to video sequences of limited duration (composing a single logical entity) which are concatenated to form complete *experiences*. In practice, these chunks will consist of one or more *segments* according to MPEG-DASH terminology.

An *immersive experience graph* representation is adopted to semantically describe the logical flow of individual video chunks. Based on this format, the system is able to present the relevant interaction options to the user during the experience. The graph's directed edges represent chunks (i.e., video streams), whereas its composing nodes encode decision points (two or more mutually-exclusive options) in the playback of the experience. Referring back to the use case description, a graph node would be used to implement a choice that needs to be taken on a crossroad of paths on campus. Graph edges can either refer to a single chunk or a collection of related chunks; in the latter case, no formal restrictions are imposed on the semantics of chunk clustering. In effect, such a chunk bundle could correspond with a collection of video streams that need to be visualized either simultaneously or between which may be alternated during playback. The former can be used for e.g. spatial tiling (see section V-C1) or multi-layer streaming (as discussed in section V-C2), while the latter can represent a section of street for which footage during different times of day or meteorological seasons is available (see section III).

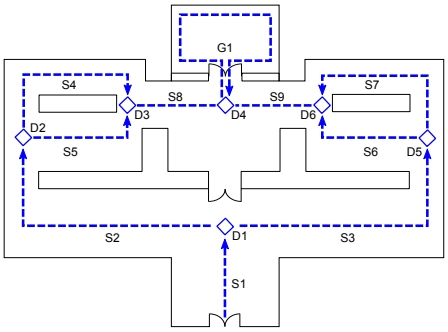
The immersive experience graph in Figure 2a describes the experience of an alternative application scenario, namely the exploration of a museum (simplified map shown in Figure 2b). Note that, for the sake of comprehensibility, only a restricted path is covered by the example graph and only a single direction of travel is considered. In reality, a more elaborate version of the same graph would likely be constructed and used. When starting the tour, the path throughout the main hall is followed (stream S1). Once decision point D1 is reached, the user can opt to proceed either left or right (stream S2 or S3). Note that the decision could be made either during playback or when the current chunk has finished playing (end of S1). Assuming that the user chooses to turn left, a similar scenario plays out when reaching point D2 and any subsequent nodes in the graph. When reaching D4, the user has the option to visit the temporary exhibition hall. Edge G1 here represents a chunk group consisting of three individual streams (S10, S11 and S12), which each correspond with a video capture of a distinct temporary exhibition as it was shown in the hall at some specific point in time. During the playback of G1, users can freely and in real-time switch between each of its composing chunks.

V. Streaming Optimizations

In this section a number of concrete optimizations to the amount of traffic being sent over the network, as well as the user experience, will be discussed.



(a) Immersive experience graph



(b) Map view of museum walkthrough

Fig. 2: Use case example scenario

A. *Content pre-caching*

Since affording viewers a seamless experience (in terms of playback smoothness) is an important objective of the proposed system, the availability of intelligent client-side content buffering schemes is highly advocated. Note that, due to the non-linear nature of the immersive experiences targeted in this paper, this task requires more sophisticated solutions than is the case in traditional systems. Also note that the proposed framework does not impose one single client-side caching scheme; instead, it is designed to allow various (possibly existing) approaches to be plugged into its implementation.

The immersive experience graph narrows down the set of navigational decisions the user is able to make in the short term. This facilitates client-side caching of relevant parts of the content, to make sure that the user does not experience any startup delay (caused by buffering) when switching between individual chunks. Consider the scenario where a choice node is encountered in the scene graph. The system is aware that the user will select one of the streams that branch from this node; however it cannot be sure which one will actually be chosen at run-time. Depending on the installed pre-caching strategy, the system will automatically start pre-fetching (some initial segments of) the future chunks associated with the outgoing edges of the choice node. Variations in the start time of the pre-caching process are supported. It is also likely in actual use cases that users will indicate their branching decision ahead of time. If so, the caching strategy can adapt by rejecting the undesired option(s) and by increasing the bandwidth resources dedicated to the chunk that is associated with the selected path.

The ultimate choice on the pre-caching strategy to use is dependent on multiple factors, including output device characteristics, network connection, type of experience targeted and possibly even user profiling. Thus far, two relatively straightforward pre-caching strategies have been proposed that define the point in time when the system has to start downloading the chunk(s) corresponding with the subsequent part(s) of the immersive experience graph. The first scheme downloads the primary (i.e. currently visualized) and pre-caching chunks at the beginning of the playback of the primary stream. This approach is clearly only suited for those usage conditions where bandwidth is abundantly available; it will enable the user to navigate freely throughout the experience, even skipping over chunks while still experiencing minimal buffering delays. When bandwidth is more scarce, an alternative strategy may be preferred, which takes a parametric approach by postponing the download of the pre-caching chunks until a certain threshold in the primary stream's playback timing has been reached (e.g. start the pre-buffering 10 seconds prior to the end of the playback of the primary chunk). Besides pure timing-based information, this strategy can also utilize the buffer fill state of the primary stream(s) to determine at which point in time it becomes viable to start downloading pre-caching content without disturbing the playback of the primary stream(s). Practical experiments based on both these strategies will be detailed in section VII.

B. *Quality and bandwidth adaptation*

The MPEG-DASH standard enables run-time media quality adaptation by allowing clients to download individual segments at specific pre-determined quality levels, for example depending on prevailing network channel conditions. However, the standard does not impose a particular scheme to be utilized and leaves this decision up to the application developer. For the technology demonstrator proposed in this paper, a total of four concrete quality adaptations logics have been integrated. The objective of these logics is to define the quality in which to download (the different MPEG-DASH segments composing) each primary and pre-caching chunk. As was the case with the pre-caching strategies, the system is extensible in terms of the quality adaptation schemes it offers. The proposed logics (see below) are hence merely examples of the type of bandwidth consumption behavior that can be enforced. Note that the installed quality adaptation strategy, combined with the utilized content pre-caching approach, are major contributors to the final Quality of Experience (QoE) of the end-user.

The example quality adaptation logics implemented are: *1)* Always use the lowest quality, which denies the system the ability to adapt by forcing each segment to be downloaded in its lowest quality level. *2)* Always use the highest quality (the opposite extreme of the first strategy). *3)* Best bandwidth fit, which instructs the adaptation layer to fetch the best possible quality, subject to the prevailing downstream bandwidth capacity (resulting in low quality for low bandwidth conditions and superior quality when sufficient bandwidth is available. Finally, option *4)* uses a 'smoothed best fit'. This is an adaptation of the algorithm proposed by [9], which is designed to provide a more stable video experience by 'filtering out' the constant quality changes that would happen under unstable network conditions, such as is the case for mobile wide-area network connections.

The fact that bandwidth needs to be divided among two categories of streams (i.e. primary versus pre-caching, see

section V-A) considerably complicates the quality adaptation task. In effect, a balance needs to be found between the priority attributed to the primary and pre-caching streams. One possible strategy could be to assign the same importance to both types of chunks, so the user can experience constant quality playback (each chunk exhibits more or less the same quality). Another strategy could be to assign more importance to the primary stream, downloading its chunks in the highest quality, while evenly distributing the remainder of the bandwidth among the pre-caching streams (which might therefore be assigned a lower quality level). The bandwidth allocation logic is an integral part of the streaming framework and can deal with all of the above-mentioned cases and is easily extended to include new strategies.

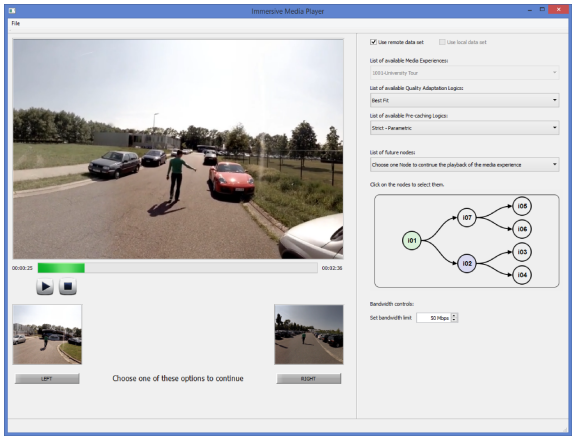


Fig. 3: Screenshot of native Qt-based front-end

C. Advanced uses of the immersive experience graph

The proposed streaming system in general, and its immersive experience graph component in particular, is sufficiently agile and expressive to tackle a number of advanced streaming use cases as well as to implement intelligent streaming optimizations, as will be demonstrated in this subsection.

1) Tiled streaming: Previously discussed in sections I and II, the huge resolution of omni-directional or panoramic video requires specific optimization to deliver these sequences to a wide range of users in optimal (visual) quality. A scalable solution is found in tiled video transmission. The immersive experience graph description and its ability to group together chunks can be employed to substitute the typical manifest that describes the spatial relationship of the different video tiles making up the entire 360° or wide-angle-recorded scene (note that this is different from the temporal relations described in e.g. MPD descriptions or HLS manifests). Clients are able to compose their personalized viewport, defined by combining chunks in a logical group (see e.g. edge G1 in Figure 2a).

2) Multi-depth-layer streaming: A specific optimization that can be identified within immersive video, which is also tightly coupled to the immersive experience graph, is that of multi-depth-layer transmission. Because (nearly) the entire scene is captured, just like it would be with a fully computer-generated representation, it becomes relevant to extract (pseudo-)depth information from the imagery. Such can be achieved using 2D segmentation algorithms or through Structure-From-Motion (SFM)-type techniques based on image data only [2]. This is an enabler for several features, including realistic augmentation of scenes with artificial content and optimized streaming.

Regardless of the technique used, it should result in the definition of a set of objects contained in the foreground, along with the classification of the rest of the image as background (with possibly multiple layers in between, depending on the accuracy of the segmentation/SFM algorithms). By stacking multiple layers on top of each other, with regions not belonging to the layer in question being rendered in a transparent fashion, this optimization is effectively visually hidden from the end-user. Priorities and/or quality levels can then be assigned and taken into account by the client software when requesting individual depth layers from the server. This maps perfectly onto the proposed pre-caching and bandwidth adaptation strategies. The layering of different chunks composing the scene can also be described using the proposed immersive experience graph, through its chunk grouping feature.

VI. The Framework

A. Platform independence

An important design goal for the streaming framework is platform independence: the proposed system must be portable to a multitude of platforms, devices and execution environments. To this end, the framework's software architecture is divided into a set of back- and front-end components. The back-end entities encapsulate the majority of the functional logic (scene handling, MPEG-DASH streaming, media quality adaptation, bandwidth management, media pre-caching, ...). Implementation-wise, the back-end is composed of a set of JavaScript modules. This decision is motivated by the near ubiquitous availability and increased performance of JavaScript engines, which renders the back-end deployable on the vast majority of modern platforms. The front-end, in contrast, fulfills the role of extended media player and is hence primarily responsible for performing media decoding, media rendering and user interfacing. The communication between back- and front-end is WebSocket-based.

Thus far, two distinct front-end implementations have been realized: a native Qt application for desktop environments (see Figure 3) as well as an HTML5-based implementation. The latter recognizes the rapidly rising popularity of the Web as content delivery and application platform. Both front-end instantiations are functionally identical. Additional front-ends could readily be developed (e.g. mobile apps for various platforms), which demonstrates the flexibility of the approach.

B. Software architecture

Figure 4 depicts the software architecture of the proposed framework. As can be seen, the bulk of the functionality is embedded in the (reusable) JavaScript-powered back-end. Marked in gray are those modules that are excluded from the focus of the described research. Solutions for these modules'

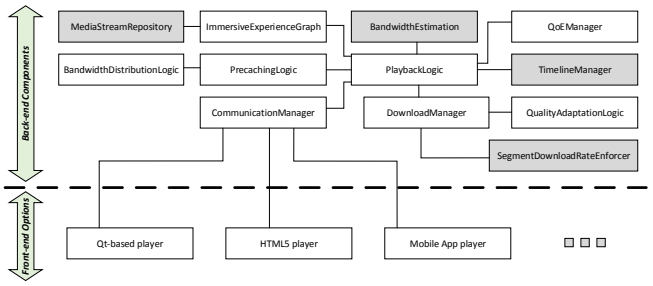


Fig. 4: Software architecture (module diagram)

responsibilities can be found in the literature (on a conceptual level) or even as off-the-shelf usable software components and are therefore not elaborated on in this paper. The non-marked modules on the other hand are integral to the system.

In general terms, the `PlaybackLogic` module acts as coordinator of the back-end process. It 1) loads all available immersive experiences (e.g. the walking tour around university grounds or the museum visit described before), 2) instructs the `DownloadManager` to fetch the appropriate MPEG-DASH segments, 3) applies the quality adaptation and pre-caching strategies as dictated by the `QoEManager`, and 4) sends the resulting content via the `CommunicationManager` to the front-end. More precisely, the `QoEManager` entity exploits heterogeneous contextual information (client device capabilities, end-user preferences, application-specific requirements, ...) to customize the client-side presentation of the immersive experience. As such, it steers the operation of other modules like the `QualityAdaptationLogic` and `PrecachingLogic`. The `BandwidthDistributionLogic` module determines how the available bandwidth, as estimated by the `BandwidthEstimation` component, must be divided among the individual chunks that are involved in the current immersive experience. Finally, the `TimelineManager` implements fine-grained media playback management.

VII. Validation

To validate the pre-fetching, quality adaptation and bandwidth distribution assets of the streaming framework, the developed prototype application has been experimentally evaluated by recording its bandwidth consumption behavior in a total of six test cases, each implementing a divergent streaming scenario. All test cases shared a single content set, which encompassed multiple representation versions of each chunk to enable MPEG-DASH-based quality adaptation, with *Rep1* being the lowest quality and *Rep7* the highest. The playback length of each individual chunk in the set varied around 50 seconds, with chunks being temporally divided into 2 second segments. All test cases applied an identical dual mode procedure for primary chunk download scheduling: in the initial *fill mode*, the objective was to fill the client-side buffer as quickly as possible with a pre-defined amount of Rep1 media segments (i.e. 10 seconds worth of media playback in our experiments), while in the ensuing *steady mode* segments were downloaded at the rate they were consumed (i.e. played back) and in the quality that best fitted the prevailing bandwidth conditions. The pre-fetching of future chunks always happened in the lowest quality level Rep1. In case primary and pre-fetching data was being streamed simultaneously in any of the experiments, these stream categories were assigned 65 and 35 percent of the downstream bandwidth budget, respectively. Finally, a stream category's bandwidth share was always broken down linearly over the active streams in that category.

The resulting network charts are shown in Figure 5. Each chart uses a stacked area representation to plot the bandwidth consumed by the different chunks involved in the respective experiment. In all diagrams, the downstream bandwidth capacity is marked using a solid red line. The horizontal and vertical axes of the charts respectively plot the time (expressed in seconds) and the bandwidth consumption (in bytes).

Figure 5a demonstrates the simplest test case, where at the beginning of the streaming of the primary chunk the options for the next chunk (in their lowest quality representation level) were downloaded concurrently with the primary stream. This resulted in the primary stream being attributed a relatively small percentage (i.e. 65 percent) of the total amount of available bandwidth throughout the pre-caching operation. During the primary stream's steady schedule mode (which commenced after 4 seconds), Rep5 segments were downloaded while the pre-caching was ongoing, and Rep6 segments afterwards (as the primary stream from that point on could claim the complete bandwidth capacity). Figure 5b shows a similar result, but under the condition that the amount of bandwidth was reduced over time (yielding a switch to a decreased visual quality for the primary stream).

In contrast to the previous experiment, in Figure 5c, the point in time at which the next chunks were pre-fetched was postponed until the very last moment in time. In this strategy, it was determined viable to implement the pre-fetching in the timespan between the reception of the last segment of the primary stream and that stream's buffer depletion (which in these examples amounted to 10 seconds). Clearly, this strategy allows for an increased quality level at the start of the primary stream. In practice, this implies that the user will not be confronted with a gradually increasing visual quality at the start of the primary stream. Similarly, Figure 5d applies the same allocation logic, but considers a scenario where the amount of available bandwidth was variable and increased over time. The chart proves that the quality adaptation logic adjusted the representation of the primary stream to optimally make use of the extra bandwidth resources, and that the pre-fetching phase consumed less time compared to Figure 5c.

Figure 5e demonstrates a longer scenario, where the same logic was applied to a sequence of multiple chunks. It can be observed that the green stream (which became the primary one after the decision point) could immediately be downloaded in the optimal representation, given that its playback buffer was already filled in the previous pre-fetching stage with the lowest quality representation. This behavior again resulted in the avoidance of buffering hickups during playback and reduced the number of visual representation upgrades.

Finally, Figure 5f represents a case where tiled streaming is applied. In these conditions, the number of pre-fetch and

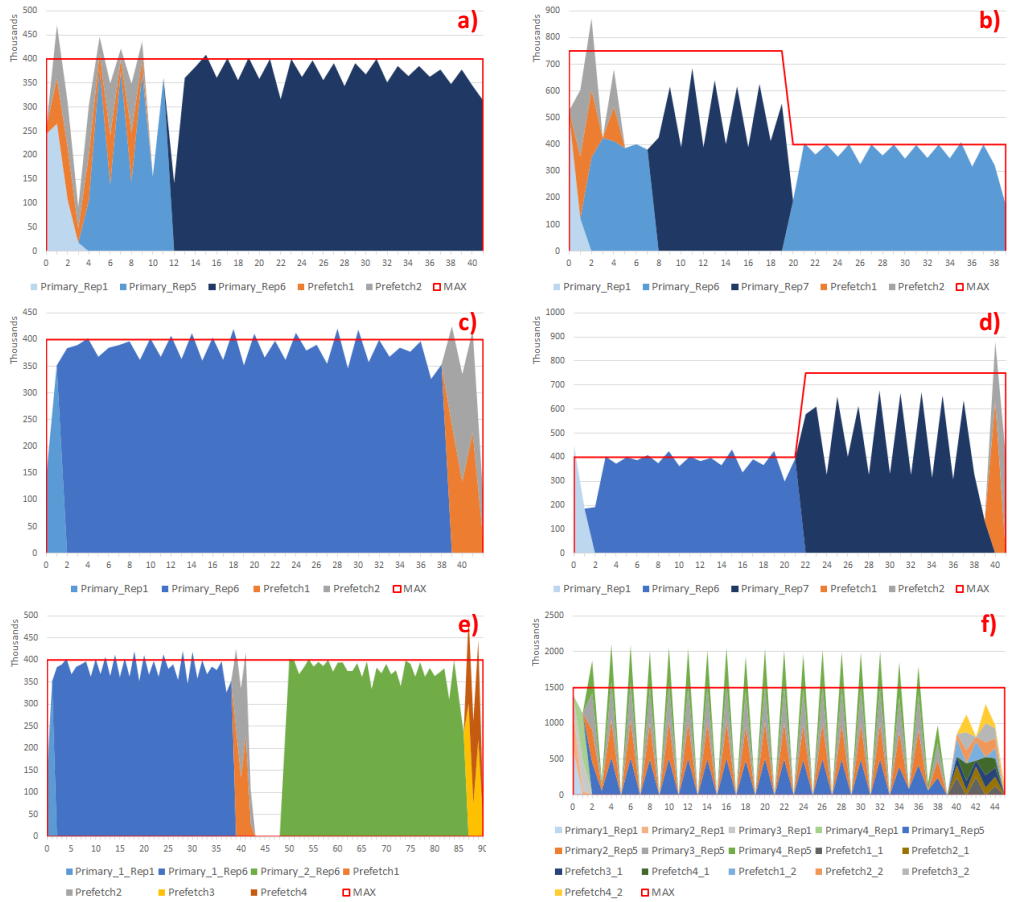


Fig. 5: Experimental evaluation results with regard to bandwidth consumption

primary streams is multiplied by the number of tiles contained in the current viewport (4 in this scenario). Still, the bandwidth distribution and quality adaptation logic were able to respect the imposed limits and optimally distributed resources over the various streams.

VIII. Conclusions

In this paper, a streaming framework has been proposed to optimize adaptive streaming of immersive media experiences. It is designed to be platform independent and extensible with respect to usage scenarios, and can integrate many advanced streaming optimizations. A technology demonstrator has been built, based on the framework, to validate the ideas.

Acknowledgements

This work is part of the iMinds ICON AIVIE project (with project partners Androme NV, Alcatel-Lucent Bell Labs and Arcadis). The research leading to these results has also received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 610370, ICoSOLE ("Immersive Coverage of Spatially Outspread Live Events", http://www.icosole.eu).

References

[1] D. P. Green, T. Smith, and G. Schofield. Plenopticon: Video playback for dynamically adaptive depth-of-field. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '14, pages 163–164, New York, NY, USA, 2014. ACM.

[2] T. Jebara, A. Azarbayejani, and A. Pentland. 3D Structure from 2D Motion. *IEEE Signal Processing Magazine*, 16(3):66–84, May 1999.

[3] B. Meixner and J. Hoffmann. Intelligent download and cache management for interactive non-linear video. *Multimedia Tools and Applications*, 70(2):905–948, 2014.

[4] B. Meixner, K. Matusik, C. Grill, and H. Kosch. Towards an easy to use authoring tool for interactive non-linear video. *Multimedia Tools and Applications*, 70(2):1251–1276, 2014.

[5] L. A. R. Neng and T. Chambel. Get around 360 degree hypervideo. In *Proc. of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '10, pages 119–122, New York, NY, USA, 2010. ACM.

[6] G. Noronha, C. Álvares, and T. Chambel. Sight surfers: 360 degree videos and maps navigation. In *Proc. of the ACM Multimedia 2012 Workshop on Geotagging and Its Applications in Multimedia*, pages 19–22, New York, NY, USA, 2012. ACM.

[7] N. Quang Minh Khiem, G. Ravindra, A. Carlier, and W. T. Ooi. Supporting zoomable video streams with dynamic region-of-interest cropping. In *Proc. of the 1st Annual ACM SIGMM Conference on Multimedia Systems*, pages 259–270, New York, NY, USA, 2010. ACM.

[8] P. Quax, P. Issaris, W. Vanmontfort, and W. Lamotte. Evaluation of distribution of panoramic video sequences in the explorative television project. In *Proc. of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video*, NOSSDAV '12, pages 45–50, New York, NY, USA, 2012. ACM.

[9] H. Riiser, H. S. Bergsaker, P. Vigmostad, P. Halvorsen, and C. Griwodz. A comparison of quality scheduling in commercial adaptive http streaming solutions on a 3g network. In *Proc. of the 4th Workshop on Mobile Video*, pages 25–30, New York, NY, USA, 2012. ACM.

[10] R. Van Brandenburg, O. Niamut, M. Prins, and H. Stokking. Spatial segmentation for immersive media delivery. In *Intelligence in Next Generation Networks (ICIN), 2011 15th International Conference on*, pages 151–156, Oct 2011.