ETSI TR 126 929 V16.1.0 (2020-11)

5G;
QoE parameters and metrics relevant to the Virtual Reality (VR) user experience
(3GPP TR 26.929 version 16.1.0 Release 16)

Reference
DTR/TSGS-0426929vg10

Keywords
5G

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Legal Notice

This Technical Report (TR) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

User experience based network management is important for operators so that they can provide best quality services. In case of a VR service, which is much more complex than traditional video streaming in terms of content creation, network transmission and device requirements, the provision of a satisfying immersive experience is more challenging. The first step of this effort would be to find out what factors have an impact on user experience, and define reference metrics and parameters that would help operators make assessment of user experience, do trouble shooting and design target solutions.

The analysis of impact on user experience involves the whole E2E VR service chain:

- Creation of content. VR content creation would involve multiple steps such as capture, stitching, projection, and encoding, each step would have an impact on the content itself. It is necessary to look into each step and find out what factors are relevant to the VR experience;
- Network transmission. The amount of video data of VR content entails a high streaming bitrate, and may lead to a risk of network and/or access link congestion and re-buffering, and thus an impediment to limiting latency. Latency is one of the key elements to create the feeling of immersiveness, and thus has considerable impact on user experience.
- Device requirement. Compared with a traditional device, be it a mobile phone or a tablet, a VR device exhibits many more attributes, designed to help create immersive experience. The degree of freedom it provides to users, the sensitivity of sensors that enable quick catching of head movement, and many other attributes, all need to be studied and evaluated about their relevance to user experience.

The present document investigates the QoE metrics relevant with VR experience from the aforementioned three aspects, and also the way of reporting these QoE metrics to the network for further analysis.

1 Scope

The present document provides a study on the QoE metrics relevant to VR service. The study focuses on:

- Defining a device reference model for VR QoE measurement points.
- Studying key performance indicators that may impact the experience of VR service.
- Identifying the existing QoE parameters and metrics defined in SA4 standards such as TS 26.247, TS 26.114 which are relevant to Virtual Reality user experience;
- Identifying and defining new QoE parameters and metrics relevant to Virtual Reality user experience, taking into consideration the use cases listed in TR 26.918, and any sources that show the relevance of new metrics, e.g. scientific literature, specifications/solutions from other standard organizations.
- Analysing potential improvements to the existing QoE reporting so as to better accommodate VR services.
- Providing recommendations to future standards work in SA4 on the QoE parameters and metrics and, as necessary, coordinate with other 3GPP groups and external SDOs, e.g. MPEG, ITU-T.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TR 26.918: "Virtual Reality (VR) media services over 3GPP".

[3] 3GPP TS 26.247: "Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH)".

[4] ISO/IEC 23009-1: 2014/Amd. 1:2015/Cor.1:2015: "Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats".

[5] Recommendation ITU-T P.1203 (10/2017): "Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport".

[6] A. Singla, S. Fremerey, W. Robitza, A. Raake, "Measuring and comparing QoE and simulator sickness of omnidirectional videos in different head mounted displays", in 9th International Conference on Quality of Multimedia Experience (QoMEX), May 2017.

[7] A. Singla, A. Raake, W. Robitza, P. List, B. Feiten, "AhG8: Subjective Quality Evaluation for Omnidirectional (360°) Videos", Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-G0152, 7th Meeting, July 2017.

[8] Recommendation ITU-T P.910 : "Subjective video quality assessment methods for multimedia applications", International Telecommunication Union, April 2008.

[9] Recommendation ITU-R BT.500-13: "Methodology for the subjective assessment of the quality of television pictures", International Telecommunication Union, Jan. 2012.

[10] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness," The International Journal of Aviation Psychology, 1993.

[11] E. Asbun, Y. He, Y. He, "AHG8: InterDigital Test Sequences for Virtual Reality Video Coding", Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-D0039, 4th Meeting, Oct. 2016.

[12] S. Schwarz et. al., "Tampere pole vaulting sequence for virtual reality video coding", Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-D0143, 4th Meeting, Oct. 2016.

[13] W. Sun, R. Guo: "Test Sequences for Virtual Reality Video Coding from LetinVR", Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-D0179, 4th Meeting, Oct. 2016.

[14] 3GPP TS 26.118: "3GPP Virtual reality profiles for streaming applications".

[15] ISO/IEC 23090-6: "Immersive media metrics".

[16] 3GPP TS 26.114: " IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction".

[17] 3GPP TR 26.909: "Study on improved streaming Quality of Experience (QoE) reporting in 3GPP services and networks".

3 Definitions of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

AV	Audio/video
AVC	Advanced Video Coding
3DOF	3 Degrees of freedom
6DOF	6 Degrees of freedom
DASH	Dynamic Adaptive Streaming over HTTP
FHD	Full High Definition
FOV	Field of view
HMD	Head Mounted Display
MCC	Metrics collection and computation
MPD	Media Presentation Description
QoE	Quality of Experience
VR	Virtual Reality
UL	Up-link

4 VR QoE overview

4.1 Introduction

The VR service is much more complex than traditional 2D video streaming, and thus defining relevant metrics is also more challenging. The following clauses address some of these challenges.

4.2 VR QoE metrics evolution

The goal with this study is to suggest improvements to the existing QoE reporting, so that suitable QoE metrics are available to better understand the VR service quality as experienced by the VR users. A complicating factor for the study is the lack of a thorough scientific understanding on exact how different VR conditions and impairments relate to the final user quality.

As VR services becomes more mature, and also more standardized, this understanding will become more clear over time. It is not unlikely that in a long-term perspective there might be standardized objective quality models, similar to the ITU-T P.1203 [5], which translate measurable QoE metrics into the final user experience.

However, there is a significant delay from the time a standard (3GPP, MPEG, ITU-T etc.) is ready, until the corresponding standard features are actually implemented large-scale by the device industry. There is also a further delay until the penetration of such standard-compliant devices gets high enough to become useful for wider analysis.

Thus, waiting with defining VR-related QoE metrics until a full and complete understanding has been gained will cause a significant gap in time, where VR service-quality monitoring cannot be adequately performed, at least not by standardized means.

The question is how to move forward during this transition period, where a detailed understanding of the QoE relations is not always available? And how to do it in a way where stepwise QoE metrics refinement can be done over several releases?

The key for how to succeed with this can actually be found in the existing QoE metrics in TS 26.247. When these metrics were standardized in Rel-10, several of the defined "metrics" are actually not really stand-alone QoE metrics. Rather they are just lists of events initiated by the client or by the user.

A typical example is the PlayList, which contains user and client actions, together with timestamps and a minimal set of related metadata. As discussed in earlier SA4 meetings, these "event lists" are not QoE metrics by themselves, and there has also been suggestions to enhance TS 26.247 with additional "real QoE metrics".

However, a big advantage with the event lists is that they contain the basic information necessary to calculate other derived QoE metrics. For instance, ITU-T P.1203 needs as one input a metric containing the number of rebufferings. As P.1203 was not fully standardized until 2017, there was no knowledge during the Rel-10 QoE work that such a metric would later be needed. However, due to the event-based PlayList it is now actually possible to derive this metric, without changing the TS 26.247 standard. The same is true for several other metrics needed by P.1203.

A drawback with event lists, especially for the VR case, is that they can potentially contain a lot of events. For normal 2D streaming the interaction from the user is much more limited, basically play, stop, jump, etc. For a VR service all of those are also there, but the main interaction is continuous head (or even body) movements, which then might also cause the player to interact towards the network in different ways (fetching different tiles etc.).

However, there are several ways to handle the report size problem, for instance using a constant (but configurable) sample time for the "movement list", and/or a (configurable) movement threshold before a movement is logged. Although it might be difficult to define exactly what sample time or movement threshold that would be the best compromise between accuracy and report size, actually there is no need to know that right now. It is a decision which can be made (years) later by the operator or the service provider, depending on the QoE use case at that point in time.

When VR services become more mature, it is expected that more optimized versions of the QoE metrics can possibly be standardized, resulting in more tailored and smaller metrics. But having a basic QoE reporting available as early as possible is a big advantage, as it is typically during the early phases of a new service adoption that the need for quality-related feedback and analysis is most important.

Thus a stepwise evolution of VR QoE metric could be to use event lists whenever possible for the initially defined metrics, as this seems to be the most future-proof representation. The lists could allow some basic configurability to enable a flexibility between accuracy vs. report size. This would allow later derivation of more specific QoE metrics, as well as other quality-related aspects, some of which might not even be known today.

4.3 Viewport-related QoE aspects

4.3.1 Introduction

From the user point of view, one of the main differences between normal 2D video streaming and VR video streaming is the notion of a viewport. Instead of always seeing the complete video, the user only sees a cropped part of the video, the viewport, depending on the direction of the device.

The resulting impairments and quality experience will vary depending on content authoring strategy, the client rendering strategy, and any network impact. The following clauses show some (non-exhaustive and simplified) examples of possible delivery scenarios and resulting impairment types. Note that while the viewport also affects audio, no audio aspects are included in the examples.

The examples should not be seen as any endorsement of specific authoring or delivery technology, they only illustrate some possible impairments as seen from the user point of view. Also, for the sake of simplicity, all examples are drawn with square regions, but in practice regions can have different shapes and also do not need to have clear quality boundaries. Any spatial distortion due to the mapping into spherical rendering is also not considered.

4.3.2 Single stream region-independent coding

In this scenario the content is encoded with the same resolution and quality settings over the complete 360 content, and delivered as a single stream. The client decodes the complete stream but shows only the cropped part corresponding to the viewport (the red rectangle).

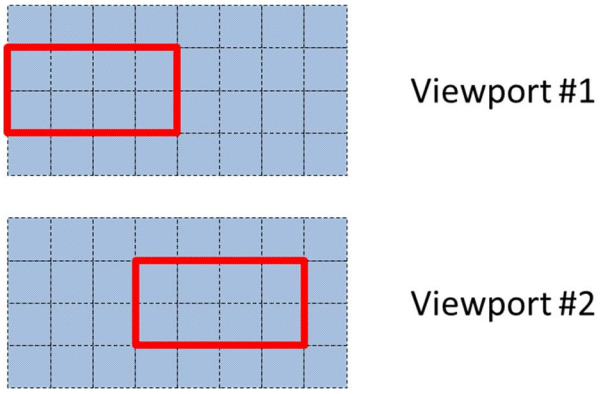

Figure 4.3.2: Single stream region-independent coding

In the example above the user moves the viewport from left to right, but as the encoding is the same for any viewport, this scenario is very similar to the 2D case. The main additional impairments would likely be related to projection artefacts and any device-internal rendering delay.

4.3.3 Single stream region-dependent encoding

The content is encoded with emphasis on one or more regions, where the content producer believes that most users will direct their viewport. Thus the resolution and/or coding quality is higher for selected parts of the spatial area, corresponding to these regions. The video is still delivered as one single 360 stream, and the client decodes and shows a cropped part corresponding to the viewport.

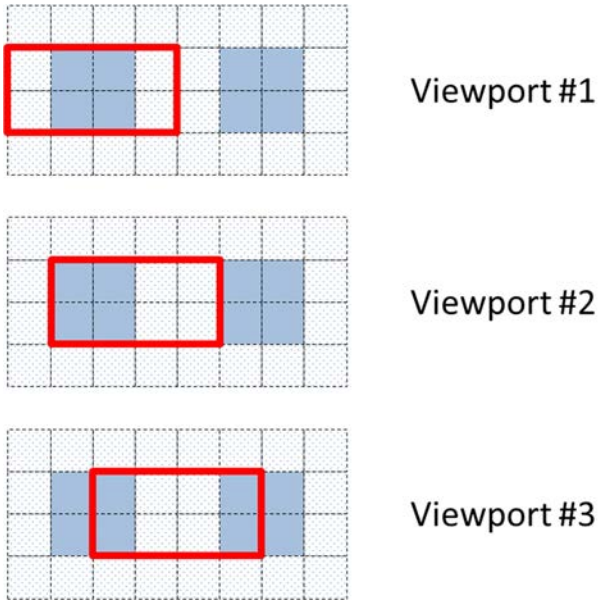

Figure 4.3.3: Single stream region-dependent encoding

In the example above, two emphasized regions are defined (dark grey), and the user moves the viewport from left to right. For all three viewports the average viewport quality is the same (i.e. 50 % high quality and 50 % low quality), but it is likely that the quality in the central part of the viewport has largest importance for the user. Thus you would expect viewport #1 to be experienced as best, and viewport #3 as worst.

4.3.4 Multiple stream region-dependent encoding

Multiple streams can be used, each emphasizing a given region. The receiver selects to download and render the stream which emphasized region best corresponds to the actual viewport.

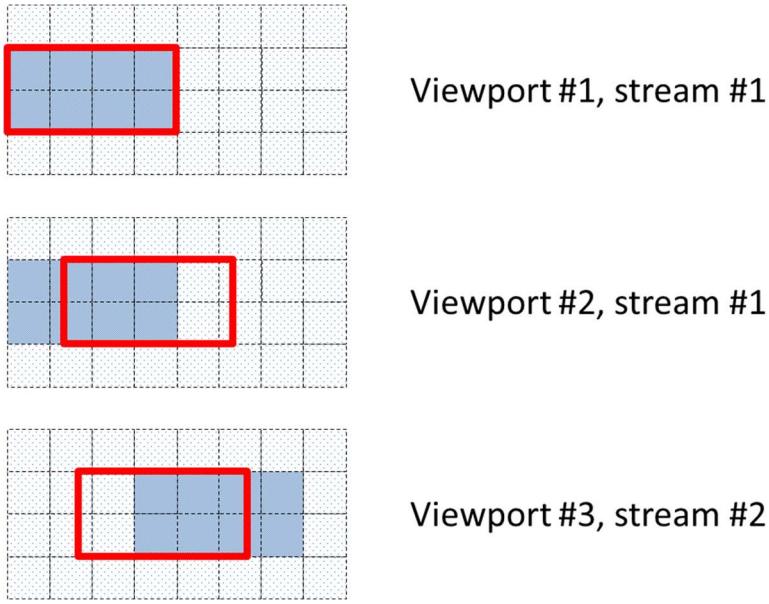

Figure 4.3.4: Multiple stream region-dependent encoding

In the example above, stream #1 and stream #2 have 25 % overlapping region-optimized encoding, and when reaching viewport #3 the receiver decides to switch to stream #2. Thus while turning the head between viewport #2 and #3, the user will see an instant change of quality for the left part (going high to low) and right part (going low to high) of the viewport.

Note that the average viewport quality is the same for viewport #2 and #3 (i.e. ~67 % high quality and ~33 % low quality) but the dynamic effect of switching between them is probably clearly visible. If the user moves his head back

and forth between viewport #2 and #3, and the receiver selects to switch streams, such quality changes can likely be rather annoying.

4.3.5 Region-based encoding, simple head movement

With region-based encoding the client typically fetches viewport regions with high quality, while using low-quality regions for the background around the viewport (or even for the complete 360). The figures below (left and right) illustrate two variants of a simple head movement.

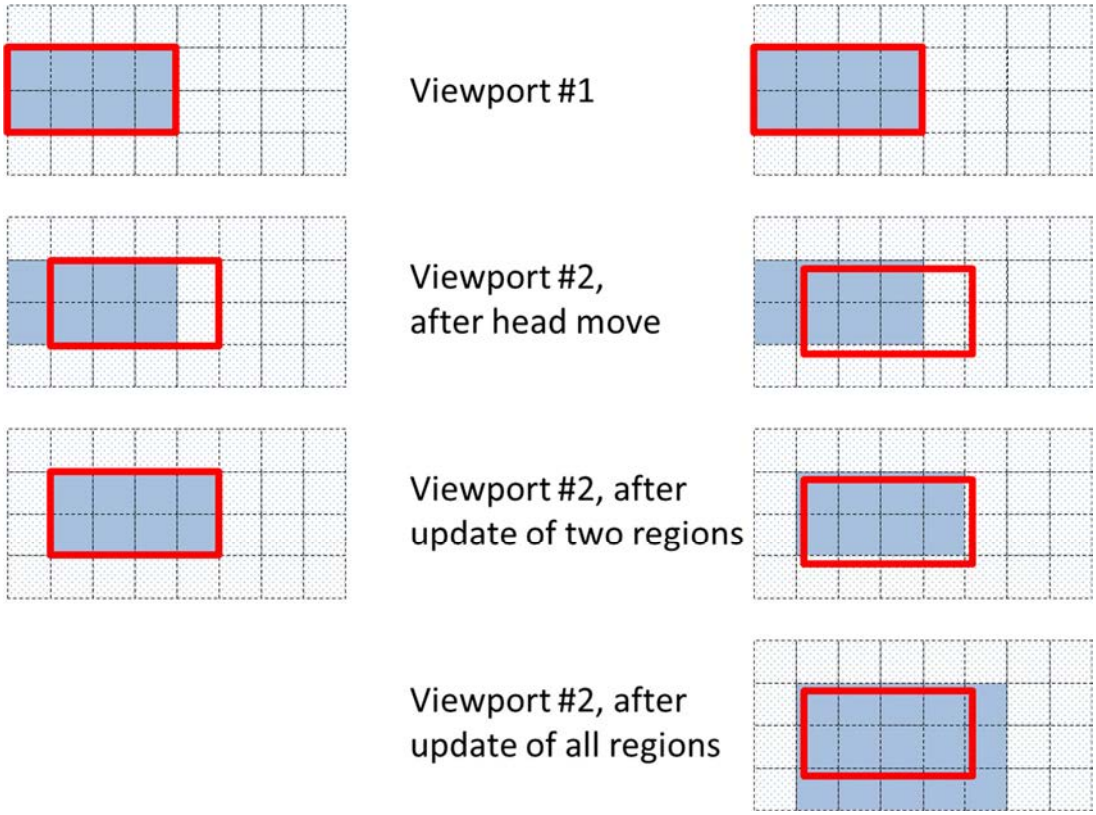

Figure 4.3.5-1: Region-based encoding, simple head movement

On the left side, the head movement is aligned with the regions, while on the right side it is moved a small bit into the next region column, and also a bit downward. In both cases the main impairment is the visible delay before high-quality versions of the new viewport regions have been fetched and rendered.

In the example it likely takes a bit longer to update the right scenario (nine regions instead of two), but due to the minimal viewport coverage of the seven outer regions, the user is unlikely to note any quality difference between the left and right scenario after the update of the first two regions. Thus although the final update delay probably is different, the experienced quality might be the same.

Note that the client could in principle instead decide to skip updating the seven outer regions due to their minimal viewport coverage. Alternatively, the client could decide to update them, but use an intermediate quality level instead of the highest quality, resulting in the example below.

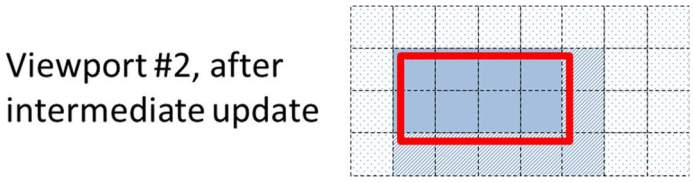

Figure 4.3.5-2: Region-based encoding, simple head movement, use of an intermediate quality level

4.3.6 Region-based encoding, complex head movements

In practice, a user might move his head in more complex patterns, with continuous movements of varying speed. During these movements the region update times will likely vary depending on network conditions etc. The example below shows a possible scenario, where the update most of the time lags somewhat compared to the current viewport at any given time.

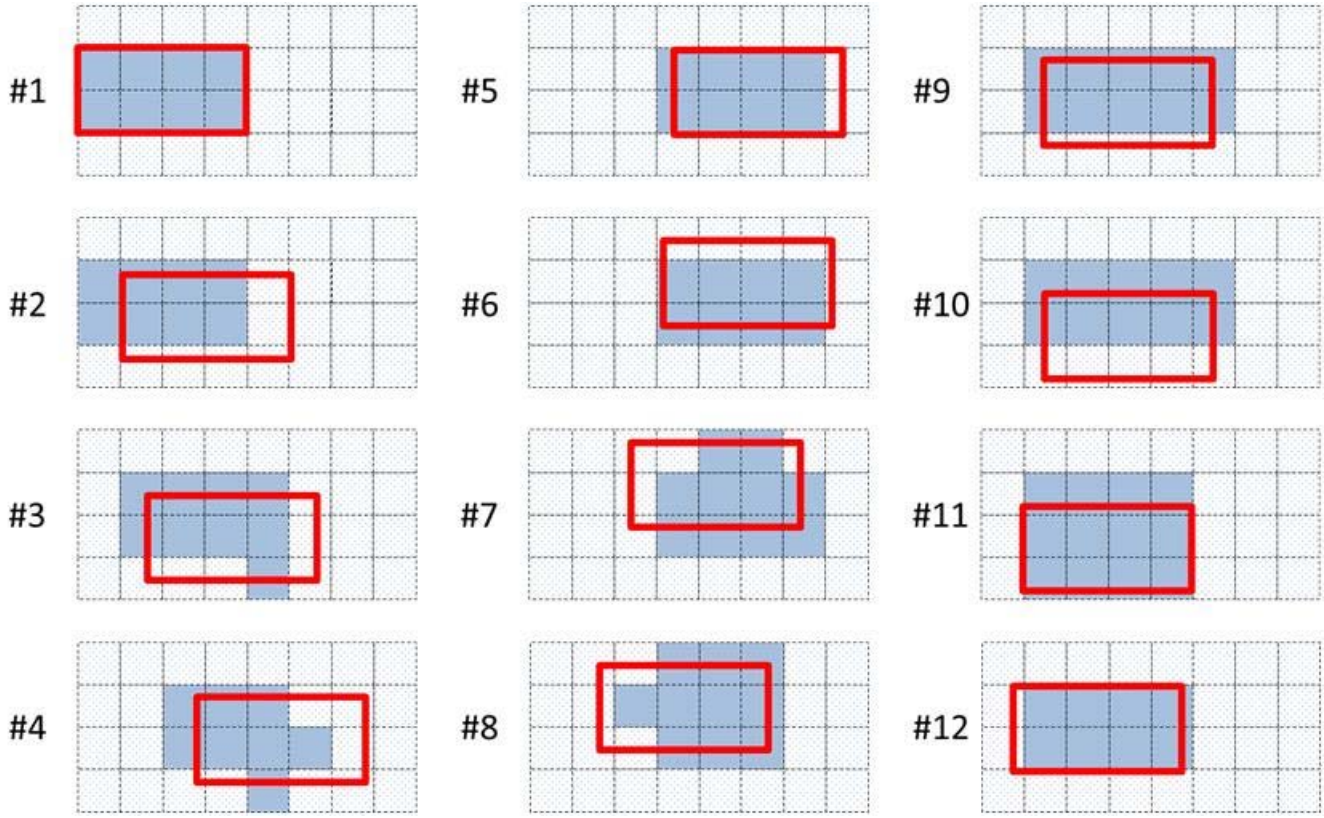

Figure 4.3.6: Region-based encoding, complex head movements

4.3.7 Summary

The shown examples form only a small subset of many possible scenarios, but they still illustrate that the viewport-related interaction between the user, the client, and the server is complex, and the final impact on the end-user quality will depend on many factors.

This interaction and its principal effect on the perceived quality needs to be understood. The interaction also needs to be mapped into a reference architecture, identifying what information (for instance regarding the viewport) that might be available at different sub-parts of the system.

Note that QoE metrics as such need not map perfectly to the perceived end-user quality, as this is a much wider task usually handled by ITU-T with extensive subjective tests, and advanced quality models. However, QoE metrics should be designed in such a way that they will be able to characterize typical impairments in a consistent, discriminative and meaningful way.

4.4 Viewport-related information flow

To better understand what metrics are possible to measure, the flow of information inside the VR streaming application is of importance. A reference model for VR QoE measurements is described later in clause 6, defining a number of observation points. However, the current clause will not specifically focus on observation points, but more on the flow of information, and will instead use the following picture to illustrate a possible information flow for a simple use-case:

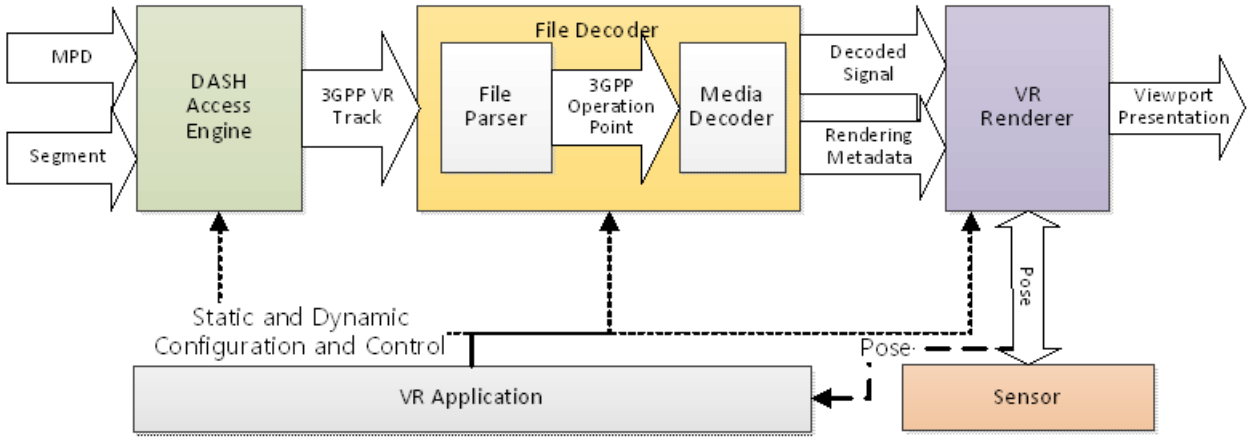

Figure 4.4.1: Client Reference Architecture for VR DASH Streaming Applications (from [14])

The DASH data model in Figure 4.4.2 below is also used as a basis for the use-case:

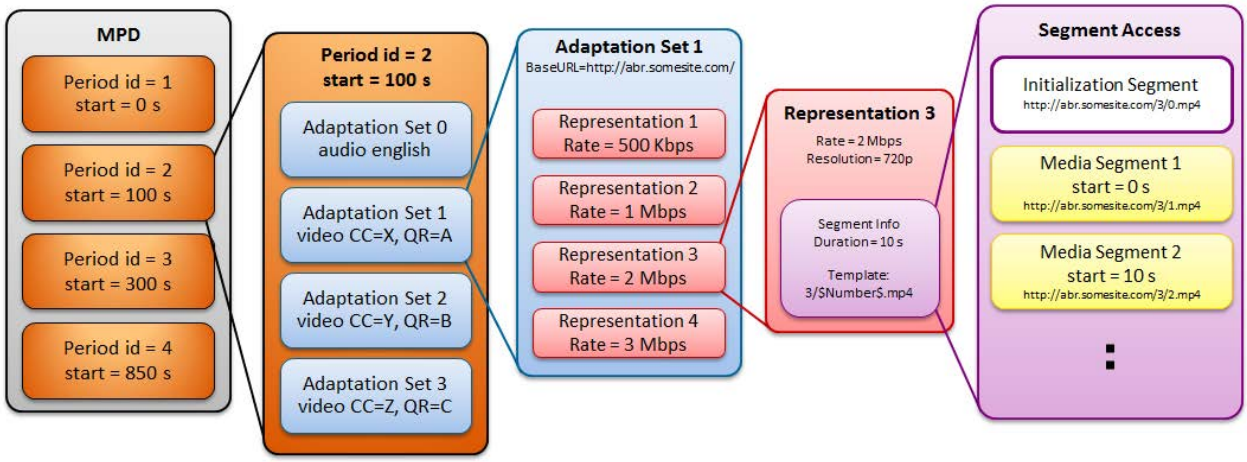

Figure 4.4.2: DASH Data Model

As seen in Figure 4.4.2, the adaptation sets are the central part for any viewport-related handling of the DASH media. Each adaptation set covers a certain spatial area of the sphere (signalled in CC), and it may also contain additional information on the relative quality ranking (QR) between the sets. Thus adaptation sets are selected depending on the user pose, and within each adaptation set there might be several representations with different encoding bitrates.

Note that depending on the scenario (e.g. single stream region-independent, single stream region-dependent, multiple stream region-dependent) multiple adaptation sets may be used at the same time, for instance a low-quality low-bitrate full-coverage set used for the full 360 view, and one or more high-quality sets which mainly covers the intended viewport. In tiled distributions each adaptation set typically contain only one tile, at a certain resolution.

Figure 4.4.3 below shows a simplified signalling diagram for a use-case with different changes of user pose:

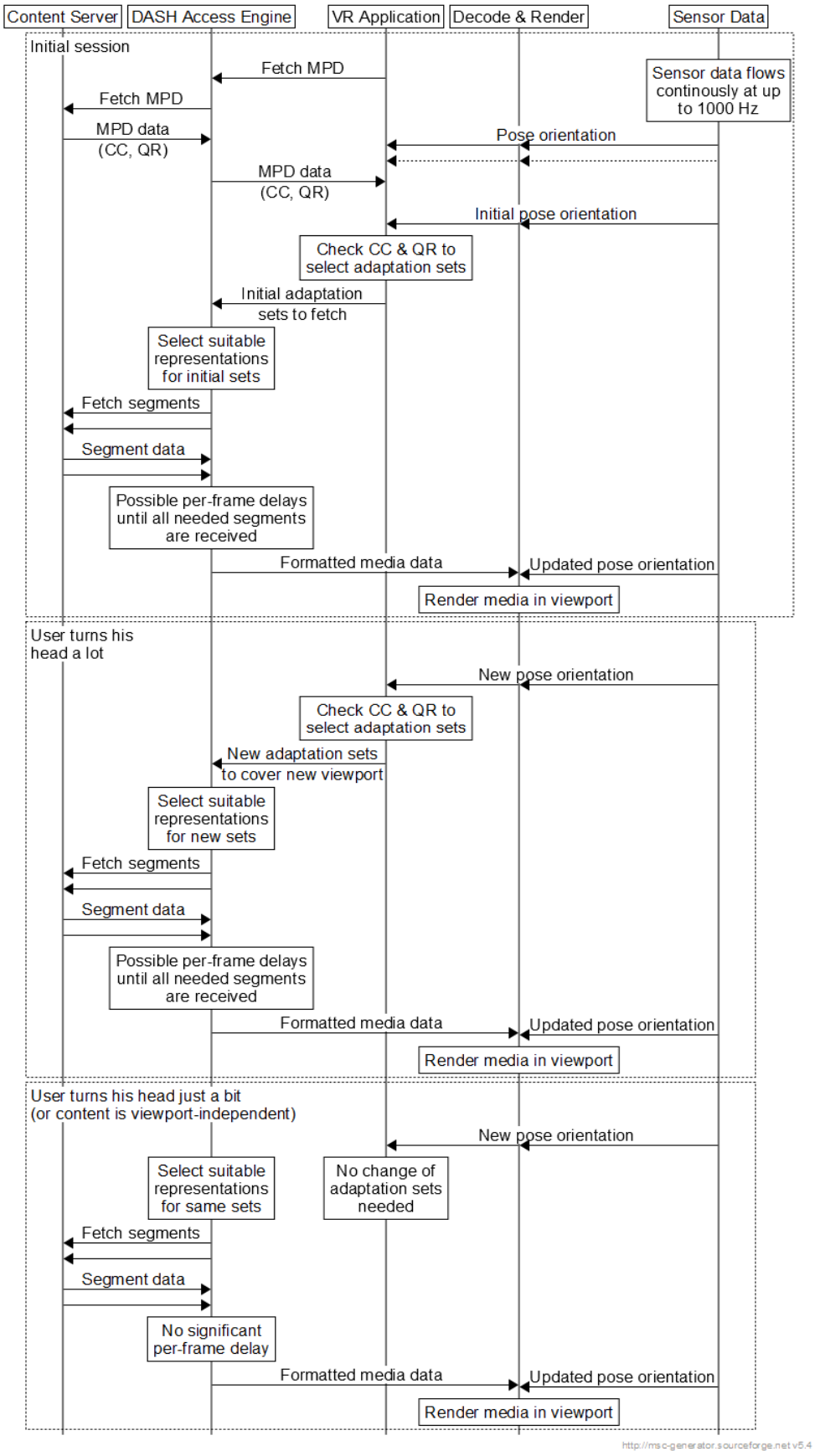

Figure 4.4.3: Simplified signalling use-case

Although very simplified, the use-case illustrates the possible division of responsibility between the VR application and the DASH access engine. The VR application uses the sensor information and the MPD coverage and quality metadata to continuously decide which adaptation sets that will be used. The task of the DASH access engine is to continuously fetch the media segments for the selected adaptation sets, while possibly adapting between different representations depending on the bitrate conditions in the network

In this use-case the DASH access engine does not have any knowledge about the pose or viewport orientation, or other viewport aspects such as field-of-view. It only tries to fetch suitable segments for the adaptation sets specified by the VR application, and deliver these for decoding and rendering.

Thus any metrics related to pose or viewport handling need either use specific non-viewport-related events known at DASH level (such as change of requested adaptation sets and the later delivery of the related media frames), or use combined trigger events derived from the VR application and the Decoder/Renderer (which might be difficult).

5 Use case for VR QoE

5.1 Introduction

There are 12 use cases defined in TR 26.918 [2], classified as UE consumption of managed/3rd party VR content and VR services including UE-generated content.

5.2 3DoF VR Streaming

Although all the 12 use cases are possible scenarios of VR service, the current standard work majorly focuses on 3DoF VR streaming, e.g. in MPEG.

According to [2], the use case of VR streaming is defined as:

"*A user watches a VR on-demand video content with a HMD. The content is streamed over the 3GPP network using unicast. The user can navigate within the 360 degree video by moving his head and watch different fields of view within the video*".

For this use case:

1) Content part: study needs to be conducted on which metadata would help analyse user experience.

2) Delivery part: changing network conditions may lead to problems in user experience, especially the impact of transmission latency on user experience, e.g. the initial loading latency, the transmission latency contributing to the latency between head/eye movement and presentation of high-resolution content to the user.

3) Device part: device capabilities will also have impact on user experience, e.g. the decoder capability, the sensor detect latency in case of head movement.

QoE metrics relevant with the above aspects need to be studied under this study item, and based on the result of this study, user experience of 3DoF VR streaming could be evaluated, and relevant provisioning and streaming techniques toward improving user experience could be designed.

6 VR QoE reference model under consideration

6.1 General description

A reference model for VR QoE measurement is illustrated in Figure 6.1.1, defining a number of observation points where specific metric-related information can be made available to the Metrics Collection and Computation (MCC) function. The MCC can use and combine information from the different observation points to calculate more complex metrics.

Note that these observation points are only defined conceptually, and might not always directly interface to the MCC. For instance, an implementation might relay information from the actual observation points to the MCC via the VR application. It is also possible that the MCC is not separately implemented, but simply included as an integral part of the VR application.

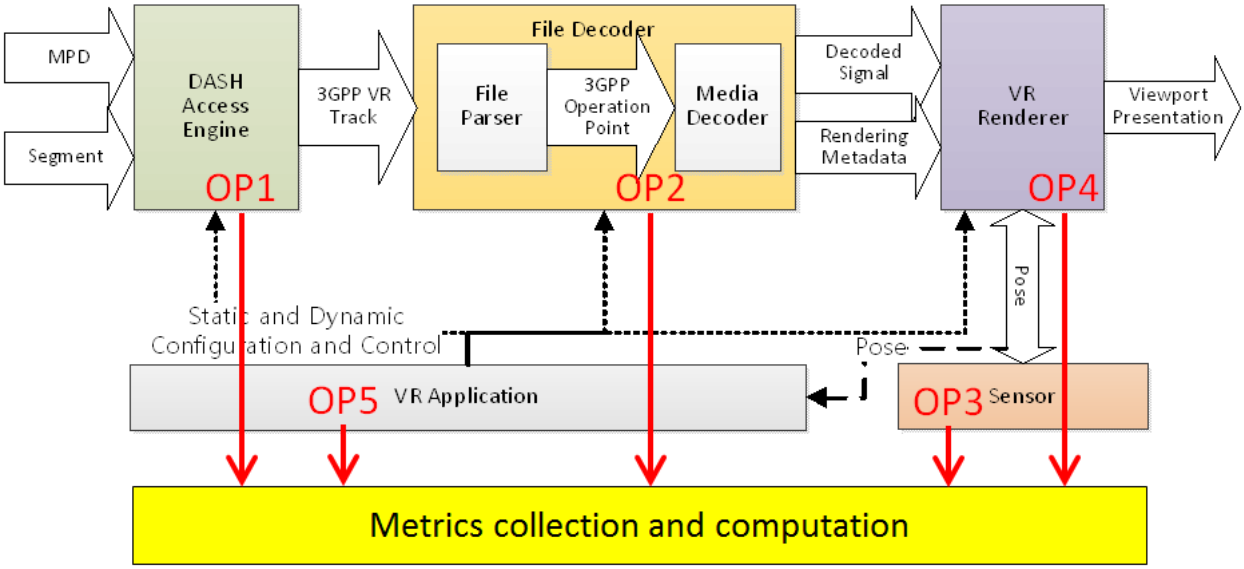

Figure 6.1.1: Client reference model for VR QoE measurement

6.2 Observation point 1

The access engine fetches the MPD, constructs and issues segment requests for relevant adaptation sets or preselections as ordered by the VR application, and receives segments or parts of segments. It may also adapt between different representations due to changes in available bitrate. The access engine provides a conforming 3GPP VR track to the file decoder.

The interface from the access engine towards MCC is referred to as observation point 1 (OP1) and is defined to monitor:

- A sequence of transmitted network requests, each defined by its transmission time, contents, and the TCP connection on which it is sent
- For each network response, the reception time and contents of the response header and the reception time of each byte of the response body
- The projection/orientation metadata carried in network manifest file if applicable
- The reception time and intended playout time for each received segment

6.3 Observation point 2

The file decoder processes the 3GPP VR Track and typically includes a file parser and a media decoder. The file parser processes the file or segments, extracts elementary streams, and parses the metadata, if present. The processing may be supported by dynamic information provided by the VR application, for example which tracks to choose based on static and dynamic configurations. The media decoder decodes media streams of the selected tracks into the decoded signals. The file decoder outputs the decoded signals and metadata which is used for rendering.

The interface from the file decoder towards MCC is referred to as observation point 2 (OP2) and is defined to monitor:

- Media resolution
- Media codec

- Media frame rate
- Media projection, such as region wise packing, region wise quality ranking, content coverage
- Mono vs. stereo 360 video
- Media decoding time

6.4 Observation point 3

The sensor extracts the current pose according to the user's head and/or eye movement and provides it to the renderer for viewport generation. The current pose may also be used by the VR application to control the access engine on which adaptation sets or preselections to fetch.

The interface from the sensor towards MCC is referred to as observation point 3 (OP3) and is defined to monitor:

- Head pose
- Gaze direction
- Pose timestamp
- Depth

6.5 Observation point 4

The VR Renderer uses the decoded signals and rendering metadata, together with the pose and the knowledge of the horizontal/vertical field of view, to determine a viewport and render the appropriate part of the video and audio signals.

The interface from the media presentation towards MCC is referred to as observation point 4 (OP4) and is defined to monitor:

- The media type
- The media sample presentation timestamp
- Wall clock counter
- Actual presentation viewport
- Actual presentation time
- Actual playout frame rate
- Audio-to-video synchronization
- Video-to-motion latency
- Audio-to-motion latency

6.6 Observation point 5

The VR application manages the complete device, and controls the access engine, the file decoder and the rendering based on media control information, the dynamic user pose, and the display and device capabilities.

The interface from the VR application towards MCC is referred to as observation point 5 (OP5) and is defined to monitor:

- Display resolution
- Max display refresh rate
- Field of view, horizontal and vertical

- Eye to screen distance
- Lens separation distance
- OS support, e.g. OS type, OS version

7 VR interaction metrics

7.1 Introduction

Besides the conventional QoE aspects such as video playout delay, stalling, and interruption, AV synchronization, and visual quality as specified in TR 26.247 and TR 26.909, additional VR specific QoE metrics and practical measurements are critical to analyse the immersion quality of VR content creation, network delivery condition, and VR device capability. Some VR metrics may be obtained from a single observation point of the VR QoE client reference model, while more complex VR metrics may be derived from multiple observation points.

One of the enabling features of VR is interactivity. A user may consume immersive VR content via HMD or conventional displays with a wide range of interaction capabilities, such as manipulation of an object within VR scene, or simply changing the user's observation point (continuous or discrete), turning the viewing orientation, or using gestures to experience different aspects of the VR content. The majority of use cases described in TR 26.918 support these interactions.

For example, the audience may select the viewing position in the "infinite seating content" use case; the experience/sound of the VR scene may adapt continuously as the viewer moves in the "event multicast to VR receivers" use case; the user may navigate within the 360-degree video by moving his head and watching different fields-of-view in the "VR stream" and "360 AV content library distribution" use cases; multiple users may connect socially to watch an event and communicate to each other in the "social VR" use case; "remote class participation" allows the student to remotely view and listen other students physically present in virtual class and interact with colleagues.

Even for "HMD-based legacy content consumption" use cases, the user may receive multiple streams in thumbnail version displayed around the user and be able to monitor a channel by turning the orientation towards a particular direction. And a channel may turn into the main channel with higher resolution and quality in front of user when being selected.

In the VR QoE client reference model, the interaction may be recognized by the sensor (OP3) and converted into pose data. The pose data can be head pose, gaze direction, skeleton or hand gesture from wearable devices, external sensors or VR applications. The VR application (OP5) collects these pose data and forwards them to the corresponding modules. For example, when the user's viewing orientation is switching from one viewport toward another viewport, the VR client response may include requesting new segment (OP1), decoding a new viewport (OP2), rendering a new viewport (OP4) at a different quality, and the latency between the interaction and system response may vary depending on VR technologies implemented.

7.2 VR interaction latency

7.2.1 Introduction

VR interaction latency is a key metric to evaluate the VR experience and it varies based on the VR content encoding, packing and delivery technologies, as well as VR device capabilities. The following clause describes the factors attributing to the VR interaction latency.

For the single-stream region-independent streaming approach, VR content is encoded with the same resolution and quality settings over the complete 360 degrees and delivered as a single stream. The pose data may only be used by the VR renderer to present the corresponding new viewport, and the latency between the user interaction and the viewport fully reflected on the HMD display is merely interaction detection latency by the sensor and motion-to-photon latency by the renderer. However, for multiple stream region-dependent encoding and streaming approach, the pose data may be used by the VR application to determine the new stream with a high-quality region mapping to the new viewport, and then drive the DASH access engine to request the new segments and refresh the receiver buffer if applicable.

The latency between user interaction and the corresponding response may involve interaction detection latency, DASH request latency, network delivery latency, receiver re-buffering latency and motion-to-photon latency. For the tiled streaming approach, additional latency may be introduced for the file decoder to handle different resolution tile streams, and the user may experience transition from low quality, low resolution viewport to high resolution, high quality viewport presentation due to the fall-back schemes where hybrid tiling approaches combining low and high resolution or quality tiles may be applied.

7.2.2 VR Interaction detection latency

The interaction detection latency is the time interval between the start of user interaction (OP3 or OP5) and the detection of such interaction by the VR application (OP5). In case such interaction is generated by HMD or external inputs such as gesture control or mouse tracking, the latency is determined by the sensitivity of HMD sensor, gesture or mouse tracking to perceive subtle motions and the sampling rate of VR application to receive the pose data. In case the interaction is simulated by the VR application, such latency may be negligible. Such latency applies to all VR streaming approaches and may be derived from APIs of OP3 and OP5.

7.2.3 VR content access latency

Once VR application detects an interaction, it may identify the corresponding VR content to be presented in response to the interaction. For 360 videos, the interaction such as changing viewing orientation or viewpoints may result in a different viewport presentation. VR content access latency is the time interval between VR application (OP5) requests VR content segment (OP1) and the corresponding segment being accessed by file decoder (OP2).

For single stream region-independent approach, VR client may continue stream the same VR representation and experience consistent viewport quality since VR content is encoded with the same resolution and quality settings over the complete 360-degree video content and delivered as a single stream.

For single stream region-dependent approach, VR client may continue stream of the same VR representation but experience different viewport quality since the VR content is encoded with emphasis on one or more regions, where the content producer believes that most users will direct their viewport.

For multiple stream or tile stream approach, different stream may emphasize a given viewport and VR application (OP5) may determine the appropriate stream to be requested at each time instance (OP1) based on the viewing orientation and bandwidth available.

VR content access latency may be affected by the segment length and receiver buffer size. Small receiver buffer size may be vulnerable to buffer underflow but able to reduce initial playout delay and VR content access latency. Figure 7.2.3.1 illustrates an example of interaction detection latency and content access latency. The user switches orientation from front view to right view at time t1, VR application detects such interaction and decides to request new segment associated with right viewport at time t2, the requested segment R5 is decoded at time t3 after all queued segments (e.g. F1, F2, F3 and F4) are being decoded or flushed. The time interval between t1 and t2 is interaction detection latency and the time interval between t2 and t3 is content access latency. VR content access latency can be derived from APIs of OP1 and OP2.

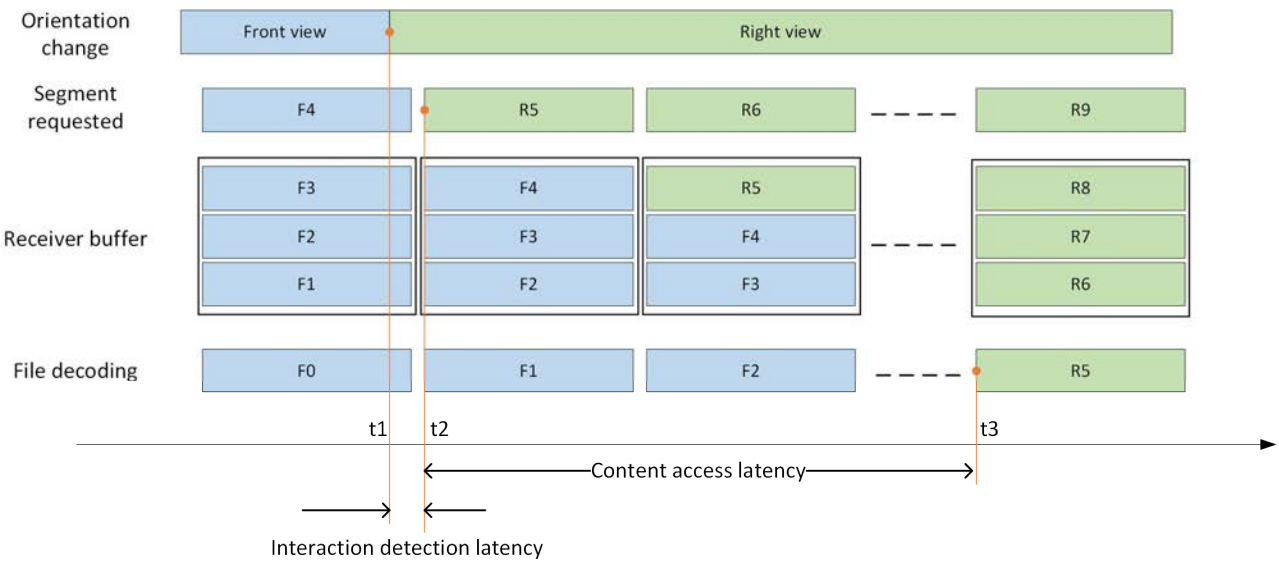

Figure 7.2.3.1: Example of interaction detection latency and content access latency

7.2.4 VR decoding latency

VR decoding latency is the time interval between the time a file decoder (OP2) receives a VR segment or frame and the time a decoded video segment or frame is passed to renderer (OP4). The decoding latency may be measured by the average decoding time consumed for each frame or segment depending on the file decoder design and performance. High decoding latency may cause presentation delay when the decoded video frame is available later than the associated presentation time stamp. The decoding latency may be reduced when a file decoder only decodes partial of the video frame (e.g. viewport) or region-wise packed frame with multiple resolution regions.

7.2.5 VR rendering latency

VR rendering latency is the time interval between the presentation time associated with the video frame or viewport and the actual playout time of the video frame or viewport being reflected on the display (OP4). Depending on the VR device design, the renderer may parse the rendering metadata to determine a region to display, and project VR content from 2D projection format to 3D sphere for HMD display or to 2D viewport plane for conventional display presentation. The display refresh rate may attribute to the rendering latency. VR renderer may also compose or overlay a variety content from different sources together to present a mixed reality scene for the use cases such as social TV or remote class participation. VR rendering latency may be derived from APIs of OP2 and OP4.

8 VR content impact on QoE

8.1 Introduction

While transmission-related factors (such as delay) certainly play an important role for the VR experience, the VR content itself will also have some impact. The following clause specifically investigates the content aspect.

8.2 Impact of Content Complexity on QoE

8.2.1 Introduction

QoE provided by the immersive technologies such 360-degree videos play an important role how much users are going to interact with the technology. Therefore, there is a need to assess the QoE of the new emerging technology, as QoE is one of the contributing factors in making the technology successful.

In this work, the influence of resolutions, camera motion, motion in the content, and simulator sickness on QoE is investigated. Some of the users are prone to simulator sickness, therefore, it is of interest to investigate how the simulator sickness interacts with the QoE and vice-versa.

8.2.2 Preparation of datasets

Six contents were chosen showing significant differences with respect to motion in the scene. Two resolutions, 4K and FHD were chosen which was motivated by the resolution limitation of the HMDs. The resolution of both devices is 2160×1200. The dataset was downloaded from the Internet, because the duration of these video sequences could be chosen much longer compared to the standard dataset [11–13]. Off-the-shelf contents was used as provided by the services without re-encoding. . Table 8.2.3.1 provides an overview of the content. The H.264/AVC encoded video sequences with highest provided bitrates were downloaded. Visible inspection by experts assured that the encoding quality for all contents was high. Then the duration was cut to a length of 60-65 seconds [6] and [7].

8.2.3 Technical setup and equipment

Two HMDs were used from two different companies – named HMD1and HMD2 here. The resolution and field of view (FOV) for both devices are 2160×1200 and 110° respectively. Whirligig player (version 3.89) was used in order to display the 360° videos in both HMDs. The HMDs were connected to a desktop PC equipped with an NVIDIA GTX980 graphics card and an Intel Core i7 processor. The names of the HMDs were hidden to the subjects to decrease contextual effects [6] and [7].

Table 8.2.3.1: Description of the Dataset [6,7]

No.	Name	Resolution	FPS	Bit-rate (in mbps)	Timestamp	Content characteristic
1	Roller Coaster	3840×2048	30	17.7	00.10 – 01:10	- Dynamic Shooting - Roller Coaster moving at high speed
		1920×1080		3.7		
2	League of Legends	3840×2160	30	17.4	01:36 – 02:40	- Dynamic Shooting - A lot of characters in game moving in random direction at a very high speed
		1920×1080		3.4		
3	Cockpit	3840×1920	25	9.8	07:35 – 08:40	- Static Shooting - View from cockpit is shown - Medium motion in the content
		1920×1080		3.6		
4	Sky Diving	3840×1920	29	16.4	03:22 – 04:25	- Dynamic Shooting - Random motion of the camera - medium motion in video
		1920×1080		4.5		
5	Aeroplane	3840×2048	29	8.5	03:29 – 04:31	- Dynamic Shooting - Slow motion of camera
		1920×1080		2.4		
6	Nature documentary	3840×2048	30	13.0	01:19 – 02:24	- Static Shooting - Almost no motion in the video
		1920×1080		3.5		

8.2.4 Test Method

For assessing the QoE of 360° videos, the Absolute Category Ratings (ACR) [8] method was used. Subjects were asked to rate the "Integral quality of VR representation", and a paper-based scale was provided after each sequence. Instructions were given to the subjects not to consider stitching and ghosting artefacts while rating the videos. A total of 28 subjects participated in the subjective test. Out of 28, 15 were female and 13 were male, with an average age of 26.25 and a median of 25. Prior to the experiment, subjects were screened for correct visual acuity using Snellen charts (20/20) and for colour vision using Ishihara charts. A short training was performed at the beginning of each test session to familiarize the subjects with the test procedure, and to help adjusting the HMD(s) according to their head size and inter-ocular distance [6] and [7] .

8.2.5 Results

8.2.5.1 Audio-visual quality of the entire VR session

Outlier detection was performed on the raw scores of the subjects based on ITU-R Recommendation BT.500-13 [9]. In this experiment, no outliers were found. Mean Opinion Score (MOS) along with the associated 95 % Confidence Interval (CI) were computed for each test stimulus.

From Figure 8.2.5.1.1, it is clear that subjects were able to find a difference between the two resolutions irrespective of the device and content. Figure 8.2.5.1.2 shows the difference in the MOS for HMD2 and HMD1 for 4K and FHD resolution. For 4K resolution, HMD1 was slightly preferred over HMD2 for all the contents except content #5. For the FHD resolution, in tendency, the HMD2 was preferred for the contents #1, #3 and #5, but the difference was not significant. For contents #2 and #6, the difference in the MOS is considerable and for content #2 the difference is statistically significant [6] and [7] .

It is interesting to note that video sequence "Project 360" (Content #2) which has the highest amount of motion due to the motion in the content and camera motion provides the least QoE irrespective of the device and resolution. Whereas, the video sequence "Surrounded by Wild Elephants" (Content #6) which has the least motion in the content provides the highest QoE irrespective of device at 4K resolution.

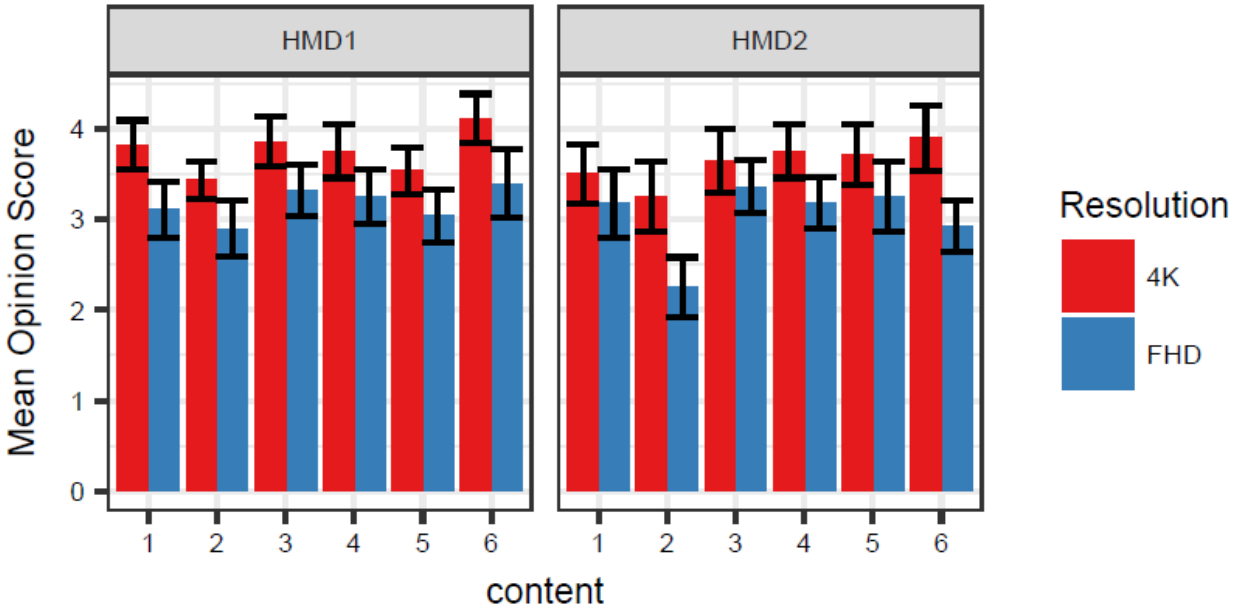

Figure 8.2.5.1.1: MOS with CIs obtained for all contents for different devices and resolutions [6,7].

A one-way ANOVA was carried out on the individual ratings to analyse the impact of the resolution, device, and content on the users' judgements. Results show that content and resolution have a significant impact on the users' ratings with all p - values < 0.01, whereas the device has a slight but non-significant effect (p = 0.07) [6] and [7].

8.2.5.2 Assessment of simulator sickness

While watching the 360° videos in HMD, users may experience symptoms of simulator sickness. Therefore, assessing the simulator sickness is a relevant additional step. Simulator sickness is a sub-category of motion sickness and symptoms comprise fatigue, sweating, vertigo, nausea, etc. [6, 10].

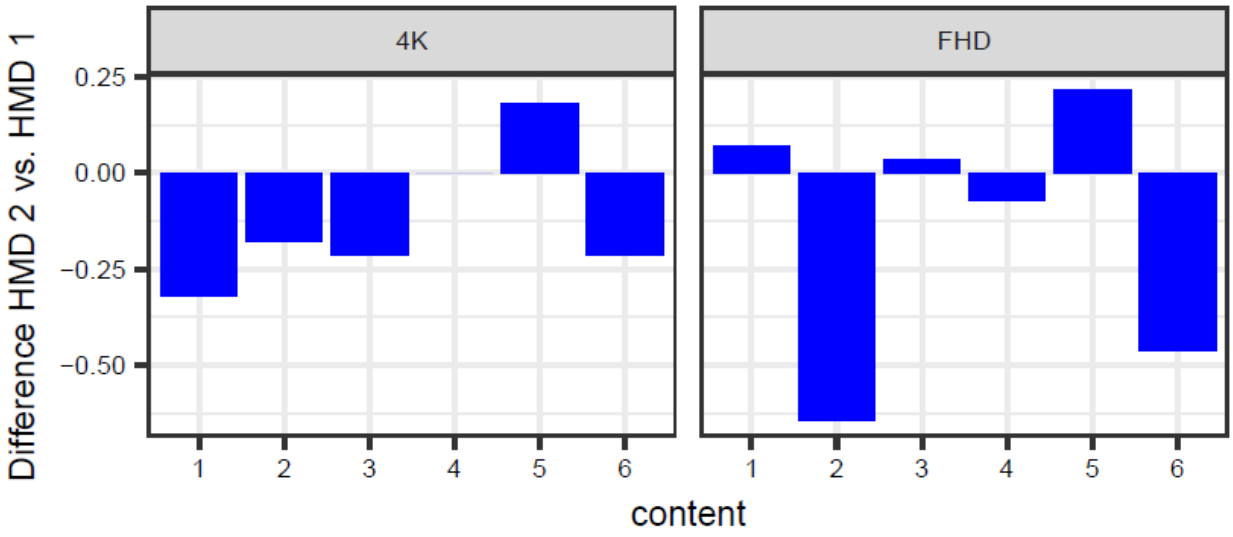

Figure 8.2.5.2.2: Difference in the MOS for all contents between devices for different resolutions [6] and [7]

The most popular Questionnaire for assessing the simulator sickness is Simulator Sickness Questionnaire (SSQ) published in 1993 [10], which was used for this experiment.

SSQ was derived from the motion sickness questionnaire (MSQ) [10], selecting 16 out of the original 28 symptoms for analysis. These symptoms are further classified into three sub-categories: nausea (N), oculomotor (O), and disorientation (D). Not all 16 symptoms are used for calculating N, O, D, and unit weights are assigned in each category. For obtaining the scores, a 4-point scale (0, 1, 2, 3) is used and weighted values are added to get the scores for each category. N, O, D, and Total Score (TS) are then calculated using the method shown in [10].

From Figure 8.2.5.2.3 it is clear that for HMD1 for 4K resolution, content 2 leads to the highest simulator sickness scores. It is worth to note that content 2 lead to the lowest quality scores. Inversely, content 6 has the lowest simulator sickness scores and the highest QoE. For HMD2 for FHD resolution, content 2 lead to the highest simulator sickness scores among all devices and resolutions as well, and was judged to have the lowest quality among all devices and resolutions. These observations indicate that simulator sickness interacts with quality when 360° videos are watched in HMDs, or may be the cause for lower quality scores.

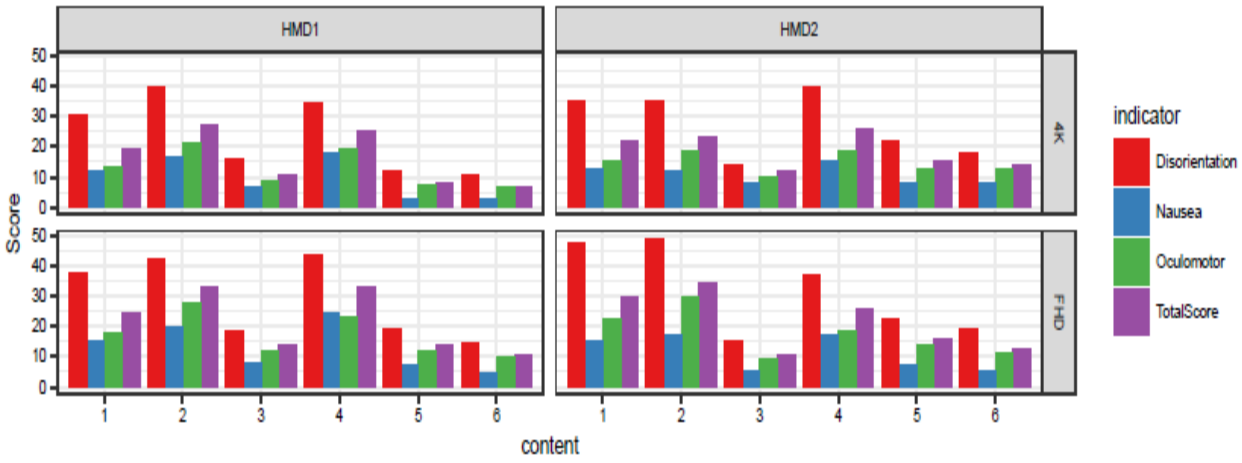

Figure 8.2.5.2.3: Simulator Sickness Scores for all contents for different devices and resolutions [6] and [7]

In order to find the influence of resolution on the simulator sickness scores, the difference between the simulator sickness scores of 4K and FHD have been computed for HMD1 and HMD2, as shown in Figure 8.2.5.2.4. It can be seen from the graph that – in case of HMD1– users are generally more prone to simulator sickness in FHD resolution as compared to 4K resolution. For HMD2, contents 3 and 6 show slightly lower simulator sickness scores for FHD [6] and [7].

An ANOVA was carried out on the Total Score of simulator sickness with content, resolution and gender as factors. Results indicate that content, resolution and gender as random factors. Results indicate that content, resolution and gender have a significant impact on the simulator sickness scores with all p-values < 0.01.

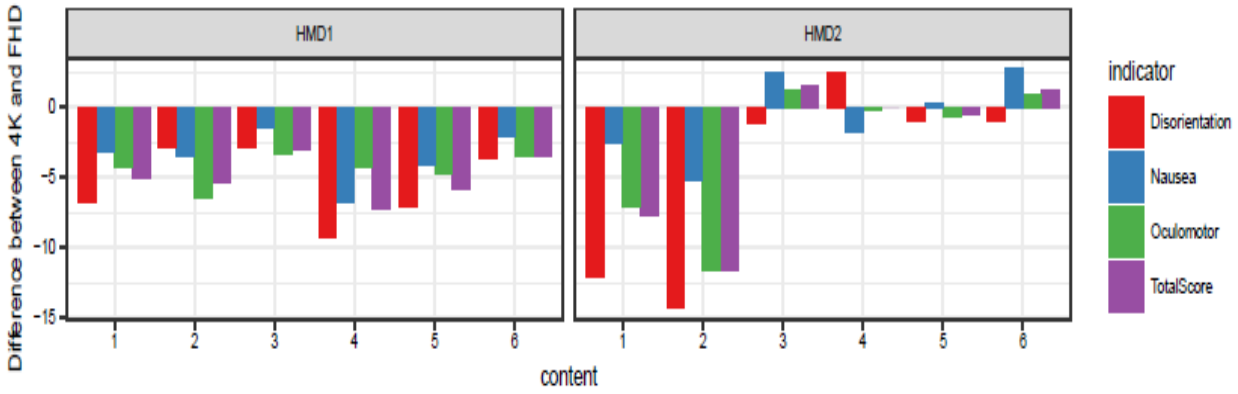

Figure 8.2.5.2.4: Difference in Simulator Sickness Scores for all contents comparing 4K and FHD [10]

8.2.6 Summary

In this clause the influence of resolutions, camera motion, motion in the content, and simulator sickness on QoE have been investigated. Results show that 4K/UHD provides better QoE as compared to FHD resolution. In addition, is has been shown that camera motion and motion in the content interacts with the QoE. Simulator sickness has also an impact on the QoE and vice-versa. Content that has the lowest simulator sickness scores provides the maximum QoE to the users and vice-versa.

Hence it is essential to include detection of features for camera motion and motion in the content for deriving metrics that help to estimate simulator sickness and consequently the influence of simulator sickness on the VR QoE.

9 Transmission impact on VR QoE

9.1 Introduction

There are many similarities between VR and traditional streaming in terms of transmission impact on user experience. And some of the QoE metrics defined in TS 26.247 could be directly applied for VR service. And of course new QoE metrics are also possible considering specific properties of VR service. All the transmission impact information could be collected by OP1 and OP2 of the reference model described in clause 6.1.

9.2 QoE metrics relevant with network transmission

9.2.1 Average Throughput

Section 10.2.4 in [3] defines the metric for average throughput information. This information could be observed by OP1 of the reference model.

9.2.2 Buffer Level

Annex D.4.5 in ISO/IEC 23009-1 [4] defines the metrics for buffer level status events. This information could be observed by OP1 of the reference model.

9.2.3 Play List

Section 10.2.7 in [3] defines the metrics for event that may happen due to user action, the end of the content or a permanent failure. This information could be observed by OP1, OP2, OP3 and OP4 of the reference model.

9.2.4 Presentation Delay

9.2.4.1 Introduction

A possible presentation delay metric is measuring the delay between the wanted presentation time for a DASH segment, and the actual (maybe delayed) presentation time. Note that this is not exactly the same as "motion to photon delay" or "motion to high-quality content", as segments might arrive late not only due to user movement and interaction, but also due to other reasons.

On the other hand, from a user perspective, both are equally bad and give rise to the same type of artefacts and quality degradations, so it makes sense to just measure the presentation delay, and disregard the actual reason (e.g. head movement) which caused the delay. However, for most practical VR cases the main reasons for late segment arrival is likely to be fast head movement.

During playout the playhead position continuously moves forward unless the playout stops due to rebuffering. Assuming that any such rebuffering can be handled by the rebuffering metric (i.e. derived from the existing PlayList metric), the presentation delay metric only needs to deal with the possibly late arrival of video data during ongoing playout.

Thus the presentation delay for any DASH segment received is in principle the difference between the intended segment start time and the current playhead position. If the segment start time is later then the playhead position, the data is available on time (at least if any further device-internal processing delay is disregarded). If the segment start time is earlier than the playhead position, the segment is late.

The question is how this delay can be practically measured, and how it relates to what the user really experiences. There are several technical possibilities, with varying degree of accuracy and implementation complexity, and the following (non-exhaustive) clauses outline some possible variants.

9.2.4.2 Ignore viewport and encoding quality

A relatively low-complex implementation is to ignore the actual viewport seen by the user, and assume that the VR client does only request segments which are relevant. For instance, for multiple-stream region-dependent encodings, it is assumed that the client changes to a more appropriate track when the user moves his head, and that all data requested and later received is actually used and relevant. In the same way, for region-based encodings, the regions (e.g. the tiles) requested and received are assumed to be relevant.

Thus for every DASH segment received, a delay is calculated based on the current playhead position and the defined start time for the received segment. If the segment request was done after the defined start time, the time of the segment request is used instead of the defined start time (to handle the case when only a part of a segment is needed). If the delay is negative (i.e. the segment arrived on time), the segment delay is set to zero.

Note that all segments are treated equally even if some segments are likely much more important, so this implementation is only a crude estimation of the impact on the user experience.

Input needed: Segment start time, playhead position.

9.2.4.3 Consider viewport but not encoding quality

A more complex implementation is to only consider delays which are probably visible to the user. For instance, when a DASH segment is received, the segment delay is calculated as in the previous clause, but the content coverage of the segment is also derived (e.g. via the CC metadata from the MPD).

The current viewport coverage is also derived (e.g. from the sensor and device data), and the overlap between the segment and the viewport is calculated. The segment delay is only considered if there is any overlap (alternatively, the delay is weighted by the percentage of the viewport covered by the segment).

Note that as the encoding quality of the content is not considered, a low-resolution background segment (which likely has large coverage) might be weighted higher than a high-resolution segment. Also, as the overlap is calculated when the segment is received, the user might have moved the viewport when the data in the segment is later rendered.

Input needed: Segment start time, playhead position, content coverage, viewport coverage, MPD.

9.2.4.4 Consider viewport and encoding quality

An even more complex implementation is to also consider the encoding quality. Encoding quality can be based on the relative QR quality ranking from the MPD, or it can be approximated as BPSPA (bits per second per area, i.e. segment size divided by segment length divided by the angular area coverage).

For instance, the current "steady state" quality level can be estimated by using the maximum quality value (QR or BPSPA) seen within a certain measurement window. In addition to the content and viewport overlap calculated as in the previous clause, the segment delay is only considered if the segment quality is the same (or close to) the steady state quality level. Thus low-quality background segments will not contribute to the delay measurements.

Note that also in this implementation the evaluation is done when a segment is received, so any head movements between evaluation and actual rendering is not accounted for.

If QR quality ranking is used, the QR value for a viewport may be derived as an average of the QR values of those regions used to render the viewport (e.g., when region-based encoding is used). This can be calculated as the weighted average of the QR values of each region, where the weights may correspond to the size of the area within the viewport being covered by the region. Alternatively, the lowest or highest quality ranking value among the different regions may be chosen to represent the quality of the viewport. It should be noted that the size of a region's overlapping area with the viewport is calculated in the spherical domain. The observation points involved in the calculation of the viewport QR value are OP1, OP3, and OP4 in the VR QoE client reference model described in clause 6.1.

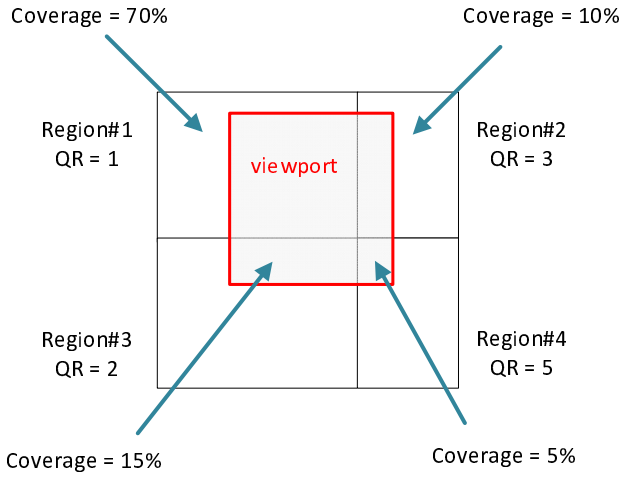

Figure 9.2.4.4.1: Viewport QR based on weighted average of quality ranking of overlapping regions

Figure 9.2.4.4.1 is an example showing a viewport that is covered by four regions, each having its own QR value. The QR value of regions #1, #2, #3, and #4, are 1, 3, 2, and 5, respectively. And the percentages of viewport area covered by the regions are 70 %, 10 %, 15 %, and 5 %, respectively. The quality of the viewport can be calculated as follows:

$$\mathrm{QR}(\mathit{viewport}) = \sum_{k=0}^{n-1} \mathrm{QR}(\mathit{region}(k)) \times \mathrm{coverage}(\mathit{region}(k))$$

Therefore, in the example shown in Figure 9.2.4.4.1, the quality of the viewport can be derived as:

QR(viewport) = QR(region#1) * coverage(region#1) + QR(region#2) * coverage(region#2) + QR(region#3) * coverage(region#3) + QR(region#4) * coverage(region#4) = 1*0.7 + 3*0.1 + 2*0.15 + 5*0.05 = 1.55

Input needed: Segment start time, playhead position, content coverage, viewport coverage, quality ranking, MPD.

9.2.4.5 More advanced delay measurements

If more accurate delay measurements are needed the evaluation likely needs to be done close to rendering or viewport generation. However, assuming that the delay measurements will mostly be used to approximate the end user experience, it is not obvious that such advanced implementations are worthwhile. More detailed understanding is needed before proposing more advanced implementations.

9.2.4.6 Aggregation

The previous clauses only describe how individual delay measurements could be done, but to be practically useful these need to be aggregated into meaningful metrics (and reporting all individual segment delays is also bandwidth-demanding, especially for region-based (e.g. tiled) encodings).

Aggregation can in principle be done over a complete QoE reporting period, but it is probably better to enable aggregation over shorter intervals to increase the time resolution of the metrics.

Assuming that most VR services will have relatively decent quality (as otherwise they would not be used), many of the individual segment delay measurements should be zero or at least pretty small. Thus using simple linear averaging is probably not very useful, as this might hide any delay peaks.

Histogram binning could be used, as this will clearly show the distribution of the delays. To keep the amount of bins low bin sizes can be varied, e.g. by using logarithmic bins. Another alternative is to use percentiles to catch the best and the worst part of the delay distribution.

Yet another possibility would be to report on detailed segment level, but to only reports segments which:

- arrive later than a certain delay threshold (e.g. >0 ms, report only late segments);
- cover more than a certain percentage of the viewport (e.g. >0 %, report only visible segments);
- have a BPSPA value higher than a certain percentage of the steady-state BPSPA value (e.g. 75 %).

Assuming that most segments do arrive on time, the resulting report will normally be reasonably small. This approach is exemplified in the next subclause.

9.2.4.7 Metric definition

The table below describes the metric for presentation delay. It is assumed that as part of the QoE configuration the three thresholds for delay, coverage and quality have been set. If not set, all segments will be reported.

Only the timestamps, the playheadPosition and the presentationDelay fields are mandatory, while the viewportCoverage and the relativeQuality fields are only reported if supported by the client.

Table 9.2.4.7.1: Presentation delay metric

Key	Type	Description
`SegmentList`	`List`	A list of received segments fulfilling the delay, coverage and quality thresholds
`Entry`	`Object`	An object containing information for one segment
`timestamp`	`Real-Time`	Reception time for the segment
`playheadPosition`	`Media-Time`	Playhead position when receiving the segment
`presentationDelay`	`Integer`	Delay compared to intended segment presentation time, in milliseconds
`viewportCoverage`	`Integer`	Percentage of viewport covered by the segment (optional)
`relativeQuality`	`Integer`	Segment BPSPA percentage relative to the steady-state BPSPA value (optional)

9.2.4.8 Metric examples

The following figures illustrate a few possible scenarios for calculation of presentation delay for a given segment. The assumption in the examples are that the content has different regions, so that the client will fetch different content depending on where the user is watching.

The examples show when segments are requested, when they are received, decoded and rendered. For practical illustration purposes the buffer (i.e. the time difference between segment reception and the corresponding decoding) is probably unrealistically small, but the metric calculations would be the same even if a longer buffer would be used.

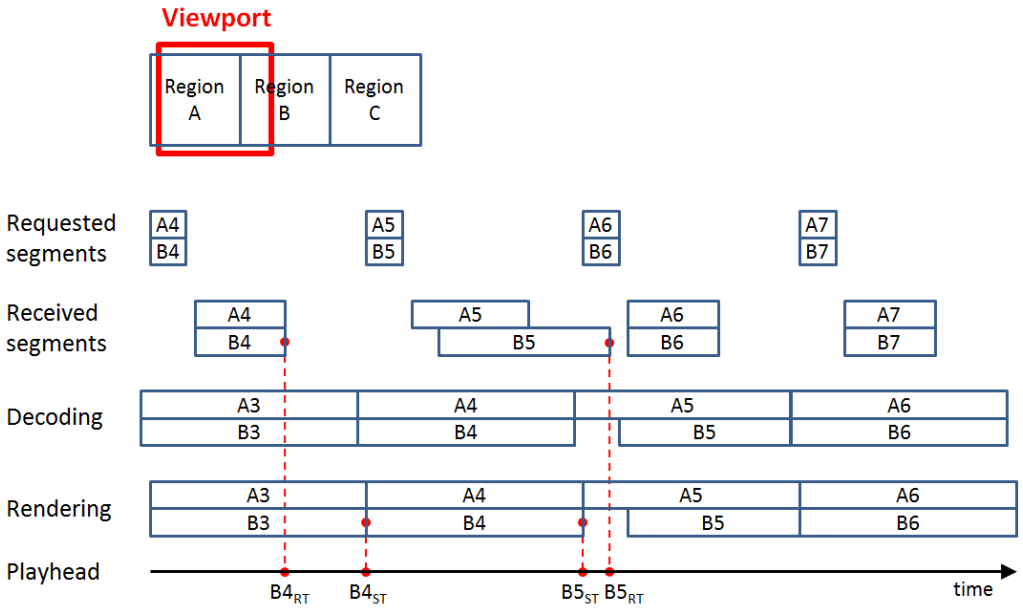

Figure 9.2.4.8.1: Example 1

The first example above illustrates two cases, one where a segment (B4) arrives on time, and one where another segment (B5) is arriving late. Note that there are no head movements included in this example.

In the first case, the receive time (RT) of B4 is earlier than the nominal start time (ST), and thus the segment is on time, and will be assigned a presentation delay of 0 (zero).

In the second case, the segment B5 arrives late, but the client selects not to stop and rebuffer as the main portion (region A) of the viewport is still available. Thus the B region will not be visible, or only visible as a low-resolution background for some time. This time (i.e. the presentation delay) is calculated as the difference between $B5_{RT}$ and $B5_{ST}$. Note that the calculation does not account for the additional decoding and rendering delay, so the real delay will be slightly longer.

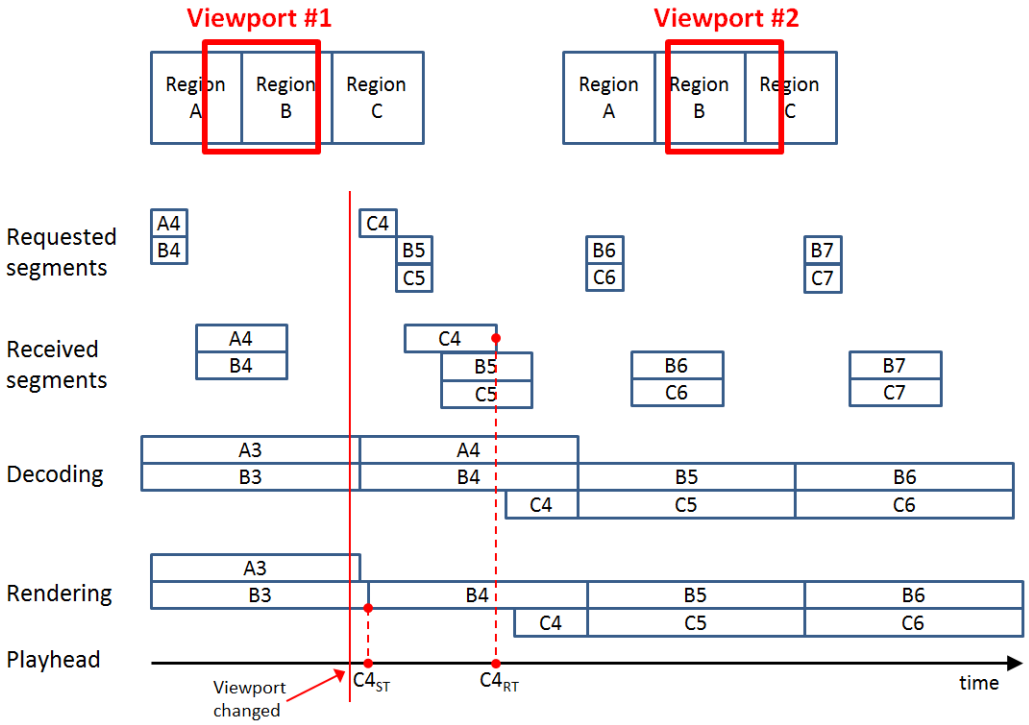

Figure 9.2.4.8.2: Example 2

Example 2 above illustrates a scenario where the user moves his head and changes pose. First the user is seeing region A and B, and after the pose change he instead sees region B and C. The viewport is changed the first red line, and after some small delay the client now requests segment C4, to fill in the missing part of the viewport.

The segment arrives at $C4_{RT}$, and the presentation delay is calculated as the difference between $C4_{RT}$ and $C4_{ST}$. The segments B5 and C5 have also been requested by the client, and manage to arrive in time, so these are not causing any additional degradation.

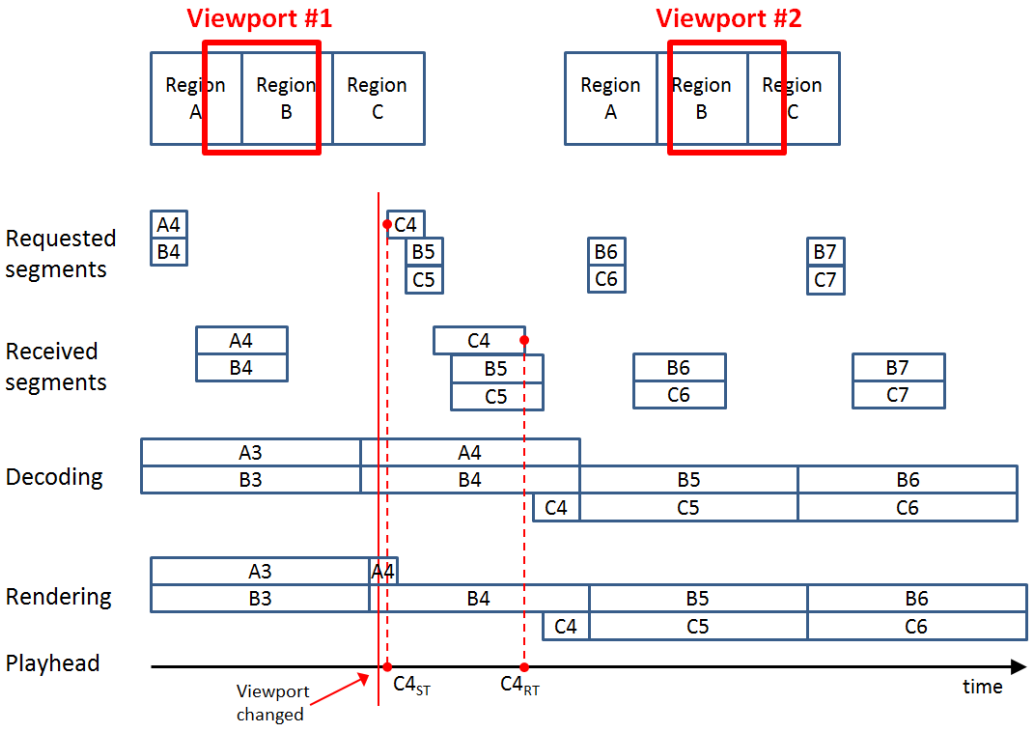

Figure 9.2.4.8.3: Example 3

Example 3 above is very similar to example 2, except that here the head movement and the subsequent request for segment C4 happens after the nominal start time of C4. In this case the start time is adjusted to be the request time, so the presentation delay covers the time between the two dashed line, i.e. $C4_{RT}$ and $C4_{ST}$.

As shown in all examples, the calculation of the presentation delay is straightforward, and easy to implement. While it does ignore the initial delay between head movement and the time of the segment request, as well as the decoding and rendering delay, these should normally be small as compared to the transport delay for the segments.

10 VR device impact on QoE

10.1 Introduction

Compared with traditional streaming video, the key feature of VR service is to create immersive experience and enable smooth interactivity between user and the environment, in which VR device would play an important role. This contribution proposes device information relevant to user experience of VR service. All the device property information could be collected by OP5 of the reference model described in clause 6.1.

10.2 QoE metrics relevant with VR device

10.2.1 Field of View

One of the factors that contribute to the uniqueness of 360 video experience is the level of immersion induced by the wider FoV of HMD, which represents the extent of observable environment at any given time. A wider FoV could help provide a more authentic feeling of immersion. Thus, FoV of the HMD is an important parameter that helps evaluate to what extent a VR device could help create immersive experience.

The user may adjust the FoV for a programmable HMD device to achieve a better experience. The adjustment may depend on the VR content or the user's preference. A record of the rendered FoV may provide information about the user's most preferred FoV during the VR playback.

10.2.2 Resolution

Resolution here is defined as for per eye. An appropriate screen resolution would provide the best and comfortable experience.

10.2.3 Refresh Rate

Refresh rate is the number of times per second the display grabs a new image from the graphic processing unit. Lower refresh rate would contribute to processing latency and lead to VR sickness, i.e. viewing glitches on the screen. While higher refresh rate adds to the sense of presence in virtual worlds.

10.2.4 Decoder capability

The support of codec profile and level is an important property that decides the content types it could decode.

10.2.5 Detailed QoE metrics

The QoE metrics relevant with VR device as listed in Table 10.2.5.1 is necessary for assessment of device impact on user experience, and suggested to be collected by VR client. This metric is logged at the start of each QoE reporting period, and whenever the characteristics change during the session (for instance if the user adjusts the FOV).

Table 10.2.5.1: QoE metrics relevant with VR device

Key	Type	Description
DeviceInformationList	List	A list of device information objects.
Entry	Object	A single object containing new device information.
resolution	Object	Display resolution for each eye.
videowidth	Integer	Number of pixels in display width.
videoheight	Integer	Number of pixels in display height.
refreshRate	Integer	The number of times in a second that a display hardware updates its buffer.
decoderCapability	Set	Codec profile and level the device supports.
fieldofview	Object	Information of end device FoV capability.
horizontalFoV	Integer	Horizontal FoV of the device in degrees.
verticalFoV	Integer	Vertical FoV of the device in degrees.
RenderedFOV	Object	Information of the rendered FOV.
hRenderedFOV	Integer	Horizontal rendered FOV in degrees.
vRenderedFoV	Integer	Vertical rendered FOV in degrees.

11 VR QoE configuration and reporting

11.1 Introduction

For the existing (non VR) QoE functionality, configuration is done on DASH level according to TS 26.247. There are three different methods for configuration:

- QoE configuration is placed in the MPD
- QoE configuration is done via OMA-DM
- QoE configuration is done via RRC signalling

All three cases re-use similar configuration objects, basically specifying which metrics to report, and how the reporting is done (at regular intervals, at end of streaming etc.). Reporting is done with an XML-defined schema which is identical for all three cases.

11.2 VR metrics

As stated earlier in this document, several of the existing metrics in TS 26.247 are also valid for a VR case, while there is a need to add new VR-specific metrics. However, from a configuration and reporting perspective, these can just be added into the existing configuration and reporting structure. Thus only minimal changes are expected due to these additions.

This is true as long as all metrics can be specified in TS 26.247, i.e. they can be calculated based on information available to the DASH client. If a new metric would need input from a functional component which is not covered by TS 26.247 there might be a need of further changes in configuration and or reporting methods.

12 Conclusions

It is evident that VR streaming services add a lot of complexity as compared to traditional streaming services. While the latter are more statically consumed, for VR services the user actions and movements becomes an important part of the total user experience. Also the actual content delivery will often depend on what the user is doing, which presents new challenges, mostly in terms of throughput and delay.

While the exact relationship between potentially measurable metrics and the complete user experience will probably not be fully researched and understood until VR services are becoming more widely used, at least basic issues should be addressed now, further improvements added and more elaborate metrics later.

Some existing QoE metrics, such as throughput, buffer level and playlist are useful also for a VR use case, and can probably be kept more or less as-is, as described in clauses 9.2.1 to 9.2.3.

A very basic addition would be to define new device metrics, as outlined in clause 9. These are necessary to understand the context of the VR service, as well as limitations and other characteristics of the device.

A more challenging task is to define metrics which describe the interaction between user actions and movements, and the responsiveness and quality of the content. Due to the complexity of the VR service it might be useful to define scalable interaction metrics, e.g. as outlined in clause 9.2.4, so that a lower-accuracy metric can be measured with a relatively simple client implementation, while higher-accuracy variants could be available if the client supports more advanced metric features.

Very advanced metrics, such as relations to content complexity as outlined in clause 8.2, most likely need further understanding before any implementable metrics can be defined.

To get as wide support as possible for new VR metrics, any metrics defined in a future normative work should ideally be coordinated with other standardization fora (for instance MPEG as the current traditional QoE metrics have a more or less a full MPEG alignment, see Annex A for MPEG metrics).

Configuration and reporting should as far as possible re-use and extend existing mechanisms, as described in clause 11, preferably in a backward-compatible way.

To facilitate a timely development of VR metrics, it is recommended that a normative work item is started. The work item should at least implement:

- Defined VR metrics observation points, aligned with the VR Client Reference Model in TS 26.118.
- Metrics describing the characteristics of the VR device. This enables a service provider to understand the device capabilities of the users, so that VR content production can be optimized.
- Metrics describing the interaction delay. This makes it possible for a service provider to monitor the actual performance of the VR content in real-life usage, and thus optimizing content authoring and delivery methods to achieve the best possible user experience.
- Extensions to the DASH Metrics configuration and reporting schemas to support the new metrics.

Annex A:
MPEG-I part-6 immersive media metrics

A.1 Client Reference Model

The MPEG-I group has adopted the generic client reference model shown in Figure A.1. The reference model consists of five key functional modules, including: network access, media processing, sensors, media playback, and client controller. In addition, a metrics computing and reporting (MCR) module, possibly residing outside the VR client, is responsible for querying the measurable data from the functional modules and calculating the different metrics. Metrics calculated by the MCR module are reported to an analytics server or other interested entities. The client reference model defined in MPEG-I Part 6 [15] and the five observation points are aligned with and directly map to those in FS_QoE_VR TR.

Note that the information in this Annex represents the MPEG status as of April 2019, and might change before final approval in MPEG-I.

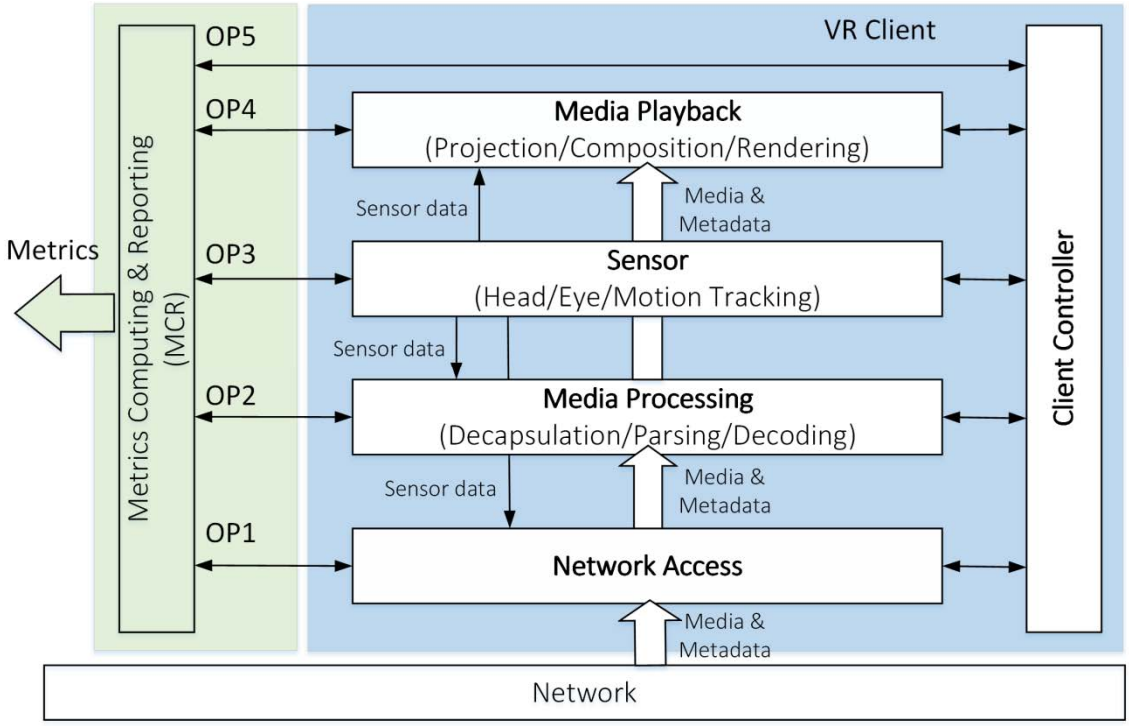

Figure A.1: Immersive media metrics client reference model

A.2 MPEG Immersive media metrics

A.2.1 Display information set

This metric reports the characteristics of the display used by the client for rendering the VR content, including display resolution, pixel density, and refresh rate. The display is the direct presentation interface to the end user and its rendering capability and performance impacts the QoE severely.

Table A.1: Display information set metric

Key	Type	Description
DisplayInfoSet	Set	Set of display information
Entry	Object	
displayResolution	String	Display resolution, in units of pixels
displayPixelDensity	Integer	Display pixel density, in units of PPI
displayRefreshRate	Integer	Display refresh rate, in units of Hz

A.2.2 Rendered field-of-view set

This metric reports a set of FOVs rendered by VR client devices. Field of view is display capability and the user may adjust FoV for 2D display or programmable HMD devices, the adjustment may depend on the VR content or user's preference, a record of rendered FoV may provide information of device capabilities and user's most comfortable FoV during the VR playback.

Table A.2: Rendered FOV set metric

Key	Type	Description
RenderedFovSet	Set	Set of rendered FOVs
Entry	Object	
renderedFovH	Integer	The horizontal element of the rendered FOV, in units of degrees
renderedFovV	Integer	The vertical element of the rendered FOV, in units of degrees

A.2.3 Rendered viewports

During playback of VR content, a sequence of viewports may be rendered on the display. While some VR experiences are created to encourage the users to explore all 360 degrees, other VR experiences are meant to direct viewers to specific areas at a certain time. Therefore, viewports may be rendered based on user interest or director's cut signalling. The rendered viewports metric reports which viewport has been rendered at what times. The VR headset tracking sensors can provide such measurable data. Aggregated viewport views metrics from multiple users can be used to identify the popularity of each region within the VR content and assist the content distributor or CDN for better caching.

The rendered viewport metric is defined in Table A.3.

Table A.3: Rendered viewports metric

Key	Type	Description
RenderedViewports	List	List of rendered viewports
Entry	Object	
startTime	Media-Time	Specifies the media presentation time of the first played out media sample when the viewport indicated in the current entry is rendered starting from this media sample.
duration	Integer	The time duration, in units of milliseconds, of the continuously presented media samples when the viewport indicated in the current entry is rendered starting from the media sample indicated by `startTime`. "Continuously presented" means that the media clock continued to advance at the playout speed throughout the interval.
viewport	ViewportDataType	Indicates the region of the omnidirectional media corresponding to the viewport that is rendered starting from the media sample indicated by `startTime`.

Where ViewportDataType identifies a viewport with six integer keys as shown in Table A.4.

Table A.4: ViewportDataType

Key	Type	Description
ViewportDataType	Object	
viewpoint_id	Integer	Specifies the identifier of the viewpoint to which the viewport belongs.
centre_azimuth	Integer	Specifies the azimuth of the centre of the viewport in units of 2^{-16} degrees. The value will be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive.
centre_elevation	Integer	Specifies the elevation of the centre of the viewport in units of 2^{-16} degrees. The value will be in the range of $-90 * 2^{16}$ to $90 * 2^{16}$, inclusive.
centre_tilt	Integer	Specifies the tilt angle of the viewport in units of 2^{-16} degrees. The value will be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive.
azimuth_range	Integer	Specifies the azimuth range of the viewport through the centre point of the viewport, in units of 2^{-16} degrees.
elevation_range	Integer	Specifies the elevation range of the viewport through the centre point of the viewport, in units of 2^{-16} degrees.

A.2.4 Comparable viewport switching latency

One factor impacting the viewing experience is the consistency of viewport quality. The comparable quality (CQ) viewport switching latency metric reports the latency experienced by the user when switching from one viewport to another viewport until the presentation quality of the new viewport reaches a comparable presentation quality as the first viewport. This metric can be used to assess the QoE of overall immersive media streaming approach. The CQ viewport switching latency can be affected by various factors, including bandwidth, segment duration, decoder buffer size, and the segment scheduling algorithm.

The comparable quality viewport switching latency metric is specified in Table A.5.

Table A.5: Comparable quality viewport switching latency metric

Key	Type	Description
CQViewportSwitchingLatency	List	List of comparable quality viewport switching latencies
Entry	Object	
firstViewport	ViewportDataType	Specifies the spherical region corresponding to the first viewport (i.e., before the switching).
secondViewport	ViewportDataType	Specifies the spherical region corresponding to the second viewport (i.e., after the switching).
firstViewportQuality	Integer	Specifies the quality value of the first viewport
secondViewportQuality	Integer	Specifies the quality value of the second viewport
t	Real-Time	Specifies the measurement time of the viewport switching latency in wall-clock time.
latency	Integer	Specifies the delay in milliseconds between the time a user movement from first viewport to second viewport and the time when the presentation quality of the second viewport reaches a comparable presentation quality as the first viewport.
reason	List	Specifies a list of possible causes for the latency.
Entry	Object	
code	Enum	A possible cause for the latency. The value is equal to one of the following: 0: Segment duration 1: Buffer fullness 2: Availability of comparable quality segment

A.2.5 Viewpoint switching latency

A viewpoint is the point from which the user views the scene, it usually corresponds to a camera position. The viewpoint can be statically or dynamically positioned along the timeline. Multiple viewpoints scenarios are supported in OMAF version 2. An event such as sport match or music concert may have multiple viewpoints on the field or stadium to offer different viewing perspective to the users. The user may request one viewpoint at a time and switch among multiple viewpoints on-the-fly. The amount of time it takes to switch from one viewpoint to another may impact user experience.

Viewpoint switching latency may be caused by a variety of factors, such as the device's response time, the player's download and decoder buffer sizes, the random access period at which video was encoded, and network conditions. Service providers and device manufactures may use such metric to assess and improve the user experience.

Table A.6 defines a viewpoint switching latency metric.

Table A.6: Viewpoint switching latency

Key	Type	Description
`ViewpointSwitchingLatency`	`List`	List of viewpoint switching latencies
`Entry`	`Object`	
`targetViewport`	`ViewportDataType`	Specifies the spherical region corresponding to a viewport of target viewpoint (i.e., after the switching).
`t`	`Real-Time`	Specifies the measurement time of the viewpoint switching latency in wall-clock time.
`latency`	`Integer`	Specifies the delay in milliseconds between the time when switching from a source viewpoint to the target viewpoint is initiated, and the time when content corresponding to the target viewpoint is reflected on the display.

Annex B:
Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
06-2019	SA#84	SP-190334				Presented to TSG SA#84 (for information)	1.0.0
06-2019	SA#84					Approved at TSG SA#84	16.0.0
09-2019	SA#85	SP-190653	0001	1	F	VR QoE Minor Corrections	16.1.0
09-2019	SA#85	SP-190653	0002	2	B	VR QOE metrics updates	16.1.0
09-2019	SA#85	SP-190653	0003	2	F	VR QOE metrics correction	16.1.0

History

Document history		
V16.1.0	November 2020	Publication