Designing the ideal video streaming QoE analysis tool

Contents

1. Introduction

Eurofins Digital Testing has built a purpose designed framework for conducting video streaming Quality of Experience (QoE) analysis. In this white-paper we describe the problem analysis, requirements definition and design process that resulted in the resulting tool that we have since used on multiple projects.

Our basic goal was to play back a range of video streams on a range of devices and measure the resulting QoE. However, this is not as simple as it sounds. In this article we start by examining all the factors (or inputs) that can ultimately affect the end user's QoE when watching streamed video, and how these inputs can be repeatedly controlled.

We then consider what QoE metrics (or outputs) should be measured and put this altogether to deduce a set of requirements that should be met by an ideal QoE analysis framework.

Finally, we describe how we engineered a framework to meet these requirements, resulting in a tool that can measure QoE for different content, network conditions and devices/apps in a way that is automated and easily visualised while providing deep offline analysis capabilities.

2. Factors that impact QoE

The factors that affect QoE can be summarised in the following picture:

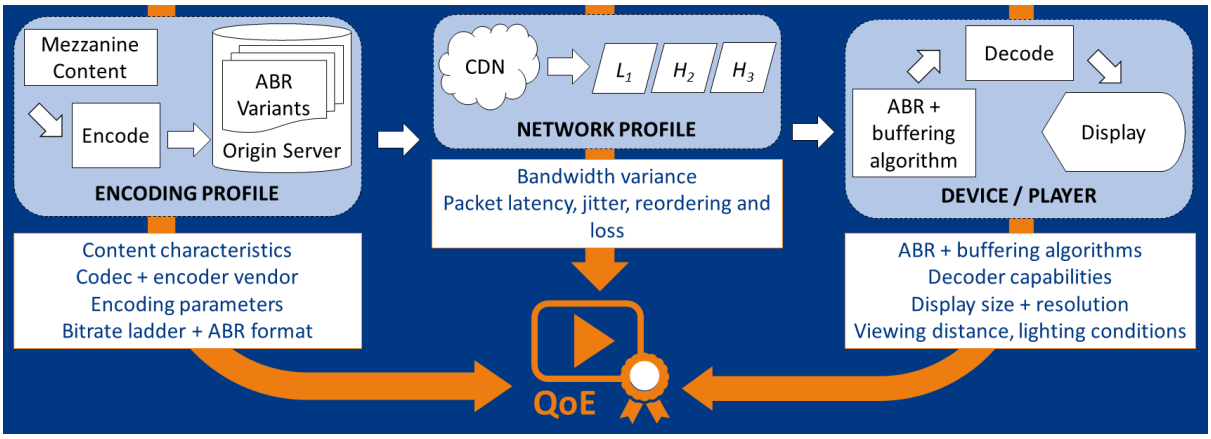


Figure 1: Overview of factors that impact QoE

Broadly speaking, these factors can be put into three categories: encoding profile, end-to-end network conditions, and the device/application the content is played back on. Let's consider each of these in turn.

Encoding Profile

For our purposes an "encoding profile" refers to the entire set of audio video data and parameters describing a single content asset (containing audio and video, in multiple adaptation sets in the case of ABR content). This is typically stored as a single fragmented MP4 file but doesn't have to be. An encoding profile can vary in many ways:

a) Content characteristics – e.g. sport/movie/news/cartoon
b) Codec – e.g. H.264/H.265/VP9/AV1
c) Encoder vendor choice – identical codecs can have very different performance according to the encoder used
d) ABR format:
   - HLS vs MPEG-DASH
   - Segment length
e) Number and bitrate of representations or variants – i.e. the design of the bitrate ladder
f) Video resolution
g) Video frame-rate
h) Other elementary stream encoding parameters, including:
   - Profile/Level
   - GOP structure
   - CBR/VBR

Typically, higher bitrates are used for higher picture resolutions and frame rates and are implicitly associated with higher QoE. However, it's also widely understood that there is a non-linear relationship between bitrate and video quality for a given video resolution and that, above a certain threshold, simply increasing the bitrate available to encode the video results in insignificant increases in picture quality. This means there are situations where, if there is more bandwidth available, simply allocating more

bitrate based upon the current profile/resolution/frame-rate used in the encoding profile may not result in noticeably improved QoE and, instead, perhaps a higher resolution picture should be offered.Of course, most of the above video factors have an audio equivalent. For most AV content, the video bit rate is an order of magnitude greater than the audio bit rate so has greater impact on QoE, which is why we focussed on varying and controlling video characteristic

Network Conditions

In an ABR streaming scenario, the selection of video stream to consume from a set of video streams of varying bitrates is based on the prevailing network conditions at the time of viewing. Fluctuations in the network conditions can cause the quality of the viewing experience to degrade or improve. There is no simple model of the effect different network impairments have on any given viewing event since the algorithm used by different players will respond differently, and the impact on quality is often highly non-linear.

The significant factors that impact network conditions for one client playing one stream include the following:

a) Effective bandwidth / throughput (Mb/s)
b) Packet latency, jitter, reordering and loss
c) HTTP connection error rate

All stages in the end-to-end network connection are relevant: the performance of the origin server, the CDN, the ISP's last mile connection, and the consumer's home network all combine to determine the overall network conditions. Our goal was to be able to model and control the end-to-end network characteristics, not any individual stage.

The factors listed above all vary over time and the nature of this variation is a critical factor in modelling real-world behaviour. We are able to simulate a wide range of behaviour through access to a very large and rich dataset of real-world end-to-end network conditions.

Since the majority of ABR video is today carried via HTTP over TCP, we assumed that packet level behaviour (i.e. the second bullet point above) is hidden by the TCP layer and we constrained just the effective available bandwidth, rather than individually controlled factors such as latency and jitter. In the case of many CDNs the HTTP connection error rate (i.e. the number of times a client receives an HTTP 404 response and has to re-request a segment) is so small that it does not have a material impact on QoE.

The time varying bitrate throughput available to the client we call the "network profile"

Devices and Players

The device and viewing environment used to watch the content has a clear impact on the QoE due to the following factors.

a) Decoder capabilities – format support
b) Device resource load (CPU/memory)
c) Display size
d) Display resolution
e) Other display properties:
    - Pixel refresh speed
    - Display filtering and "pixel improvement" algorithms
    - Brightness
    - Viewing angle

f) ABR implementation and buffering algorithm
g) Network stack characteristics
h) Viewing distance & lighting conditions

Factors a) and b) are characteristics of the devices processing hardware, while c), d) and e) are characteristics of the device's display hardware.

Sometime the content might not be supported by the device at all because the media codec level of the content is not supported by the hardware components within the device (especially when considering mobile devices such as phones and tablets). This is a particular problem on Android devices where the mandated level of codec support is quite low relative to other devices, but it is known that many devices have higher levels of codec support.

While the end-user device hardware and display itself is obviously associated with a viewer's QoE, the impact of the video player client software used to watch the content is often not considered explicitly. Factors f) and g) above are both determined by the device's software. The player software, whether at the application layer or part of the built-in media player, has a material impact on the QoE because in ABR video consumption the player is responsible for choosing which variant bitrate to stream, when to switch, and what buffering strategy to adopt. ABR algorithms can vary very considerably between player applications running on the same device.

Understanding how the software impacts the QoE and affects the choice of encoding profiles is complicated by the fact that there are different types of player agent software and the ability of an OTT service provider to affect the behaviour of the player depends on the type of player in use. Broadly speaking there are two primary categories of player:

Embedded. This is where the choice of bitrate to stream and present is determined by an underlying capability of the platform the player application is running on. In the case of Apple iOS, HLS playback and bit rate selection are controlled by iOS itself and the app author can't affect the adaptation behaviour directly (ignoring one or two exceptional cases, such as Netflix). In the case of Android, there is native HLS playback available via the MediaPlayer API, but it is limited based on the specific version of Android. Another example would be for HbbTV devices which support MPEG-DASH via providing the manifest (MPD) directly to an HTML5 video element and the underlying middleware controls the adaptation algorithm; in other words, although it's an HTML application, HTML Media Streaming Extensions (MSE) are not used.

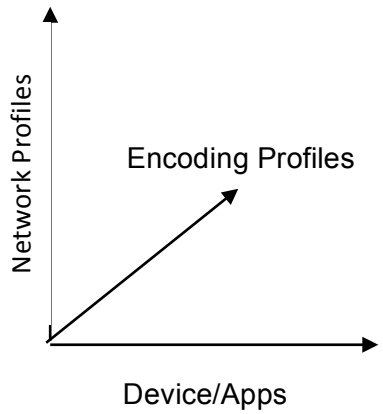
Application controlled. This is where the service provider's application itself controls the selection of the bitrate and has an algorithm that determines when to switch profile. Examples include: HTML based applications using MSE, e.g. the open source dash.js and HLS.js players: native media player libraries that can be embedded in the application, e.g. the open-source ExoPlayer application for Android; and commercially available third-party player applications that the service provider can customise the UI for.

In this proposal the combination of device and player software is considered as a single variable dimension when considering impact on QoE.

Combinations of Factors

All of the factors described within these three categories come together to affect the QoE. They can be visualised as dimensions of search space where each point in that space corresponds to different input conditions that will result in different output QoE.

Input conditions			Resulting output QoE metrics		
Device/App Combinations	Encoding Profiles	Network Profiles	QoE metric 1	QoE metric 2	…
DA_1	EP_1	NP_1			
		…			



		NP_o			
	…	NP_1			
		…			
		NP_o			
	EP_n	NP_1			
		…			
		NP_n			
…	EP_1	NP_1			
		…			
		NP_o			
	…	NP_1			
		…			
		NP_o			
	EP_n	NP_1			
		…			
		NP_n			
DA_m	EP_1	NP_1			
		…			
		NP_o			
	…	NP_1			
		…			
		NP_o			

Figure 2: A search space representation of the factors that impact QoE. The table rows indicate the test run results for 'm' device/app combinations, 'n' encoding profiles and 'o' network profiles.

A brief consideration of all the factors listed above quickly suggests that testing all possible combinations leads to a vast search space due to the combinatorial explosion. For example, the following experiment would result in 25,000 test runs!

- 50 encoding profiles, from, say, 5 content types, 2 codec types, 5 bitrate ladder designs.
- 10 network profiles
- 50 device/player combinations, from, say, 10 devices with 5 different web-player applications

Even a relatively narrow experiment – e.g. examining which software player on device X performs best for a specific type of content and codec – can require hundreds of test runs to enable

statistically robust evaluation. For this reason, enabling efficient and repeatable running of a large number of test runs was a fundamental requirement of our tool.

3. Measuring QoE – what are the output metrics?

The following measures are all relevant to assessing the QoE of streaming video.

- Video start-up time: time from initiation of playback to first video frames being displayed.
- Buffering events: the number, duration and location of events during playback where the video is paused or not showing while further segments are acquired.
- Switching events: the number and location of events where the player adapts from one bitrate to another.
- Frame-by-frame objective video quality. To calculate this, we used the SSIMPLUS algorithm – a full-reference, objective video quality metric. We made this choice after evaluating a number of alternatives, including VMAF, and the reason for the choice of SSIMPLUS has been explained in a separate article.
- Audio quality, including audio-video sync.

Measuring ABR QoE is an active and relatively immature research topic and there is no agreed way of combining these measurements to a simple overall QoE metric. Worse, for some of the metrics research experiments have contradictory conclusion as how even a single measure impacts QoE. For example, consider the frequency and visibility of switching events. Some researchers have concluded that switching events should be avoided as far as possible leading to “widely spaced” encoding ladders with fewer representations. Other researchers have shown that barely perceptible switching events have little impact on QoE, leading to “tightly packed” encoding ladders with a higher number of representations, where each adjacent representation has a quality difference that is only just distinguishable. In practice it seems that encoding and storage costs are the more important factor in determining this choice: OTT providers with high value, relatively static catalogues (e.g. Netflix) will prefer “tightly packed” ladders, whereas those with fast turnover or live content (e.g. broadcaster catch-up services) are likely to prefer “widely spaced” ladders.

The impact on QoE of the number and duration of buffering events is equally poorly understood. While less buffering is obviously desirable, there is no conclusive research on whether, say, 5 x 6s buffering events is better or worse than a single 60s event.

4. What does the ideal tool need to do?

So, having evaluated the impact factors that need to be varied (network conditions, encoding profiles, device/players), and the output metrics that need to be measured, we arrived at the following set of requirements for our analysis framework.

- **Automated**. This is the only way to cost efficiently run hundreds of

experiments: it is simply not viable to use manual execution. Additionally, measuring QoE accurately and in a repeatable fashion makes test automation the best choice.

- **Device agnostic**. The framework must be usable on any type of device (set-top-box, HDMI stick, mobile phone, tablet, computer, smart TV) and a wide variety of screen types without relying on frame capture and analysis techniques that work on some devices but not on others.
- **Content and encoder agnostic**. We wanted the solution to be agnostic of the codec, encoding and packaging platform without relying on particular tricks or features of a particular encoder pipeline. This was to make sure we could evaluate different codecs, encoders and ABR formats without having to re-architect the framework.
- **Network reproducibility**. To precisely and repeatedly control network conditions means the framework has to run within a dedicated local network, where a variety of typical network conditions can be accurately reproduced.
- **Powerful results visualisation**. All the measurements must be automatically captured into a spreadsheet to allow detailed and comprehensive analysis. However, we also wanted to provide easy, visual and intuitive browsing of individual test runs, allowing the user to see how metrics vary over time while also being able to see the corresponding video that was being displayed on the client device. It was also important to enable deep analysis of the results without having to go back and re-run a particular test – meaning that all relevant test date, such as network traffic logs, should be stored for later use.
- **Accurate**. The framework must offer frame accurate analysis of the results when deducing metrics like amount of buffering time, switching events and start-up times. Gathering these metrics with such high precision is not trivial.

5. Framework design

The above requirements resulted in the following framework design, shown in the diagram on the next page.

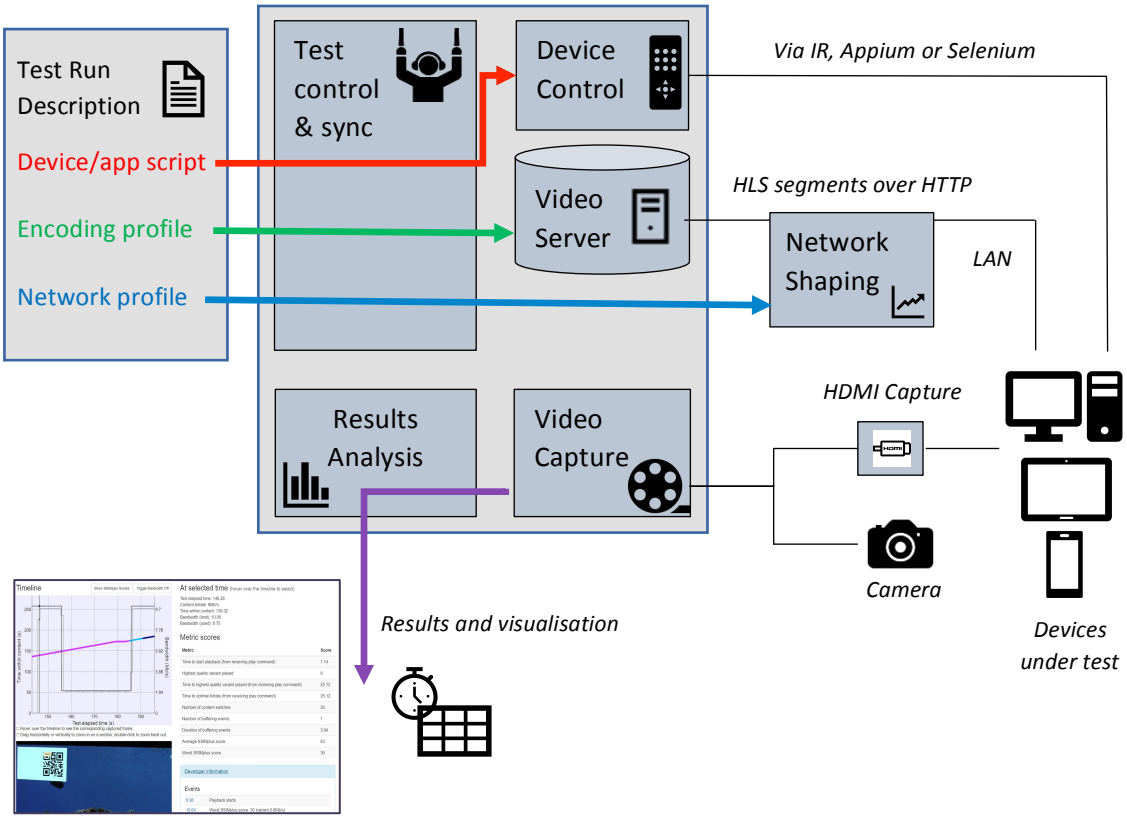


Figure 3: Schematic representation of test framework

Each test run consists of playing back a single piece of video content on a single test device, while modifying properties of the network link between the video server and the device-under-test. A test run can be broken into the following steps:

- Using Appium, Selenium or IR, the device is driven to run the app and play a video file. The precise sequence of control steps depends upon the device in question and the app being used. Careful time synchronisation is maintained between the device control script, the network emulator, and the video capture unit.
- The network profile shaping script is executed which varies the bandwidth for the duration of the test as required. The network shaping unit is also responsible for logging all network traffic during the tests.
- The camera or HDMI capture records the display screen of the device-under-test for twice the known duration of the content (to allow for any buffering events). The recorded video is stored for every test run for subsequent analysis and for use within the results visualisation.
- The device starts to play back the local video content and continues until playback is complete.
- The recorded video is stored and analysed offline, giving a list of all the frame/representations played at every point of time during the content playback.
- This data is used to calculate a set of quality metrics (listed above) and store them in a spreadsheet. At the same time an HTML/JavaScript visualisation of all the results is created and hosted in the cloud

to enable visualisation and analysis of the results from any location.

These steps are then repeated for each test run, with the URL of the video content and the network profile adjusted to what is required for that test run. Between each test run the device is rebooted and the application is restarted to ensure that all caches are cleared.

6. Video capture and analysis

Each source video type has two small QR code embedded in the source content that varies for every frame and for every representation/variant. Using a camera recording or HDMI Capture from the device/app playing the streamed video, we robustly and automatically detect every frame that was played out on the device/app under test and determine which representation within the ABR format the frame was taken from. This allows for detection of dropped frames, blank frames or buffering, repeated frames, and detection of switching events.

The captured video and contained QR codes are only used to calculate the exact frame that is being displayed at a given moment in time. The quality of the video displayed by the device is not determined using the recorded video. Instead, each encoding of the source content is pre-processed to determine the SSIMPLUS score of every frame within every representation/variant. When analysing the test run this SSIMPLUS score is then looked up for every captured frame, meaning that the quality of the recorded video has no impact on the SSIMPLUS scores.

The result set for each test run is very rich, containing information about how the displayed frame, SSIMPLUS score, available bandwidth and utilised bandwidth vary over time. To facilitate interpretation of this information, displayed alongside the video and the inferred QoE metrics the results can be presented in an interactive manner in any web browser. By hovering the mouse over the timeline, the user can see the value of various properties at that moment in time, as well as the precise video frame that was captured at that time.

Figure 4 on the next page shows a typical results visualisation for a single test run.

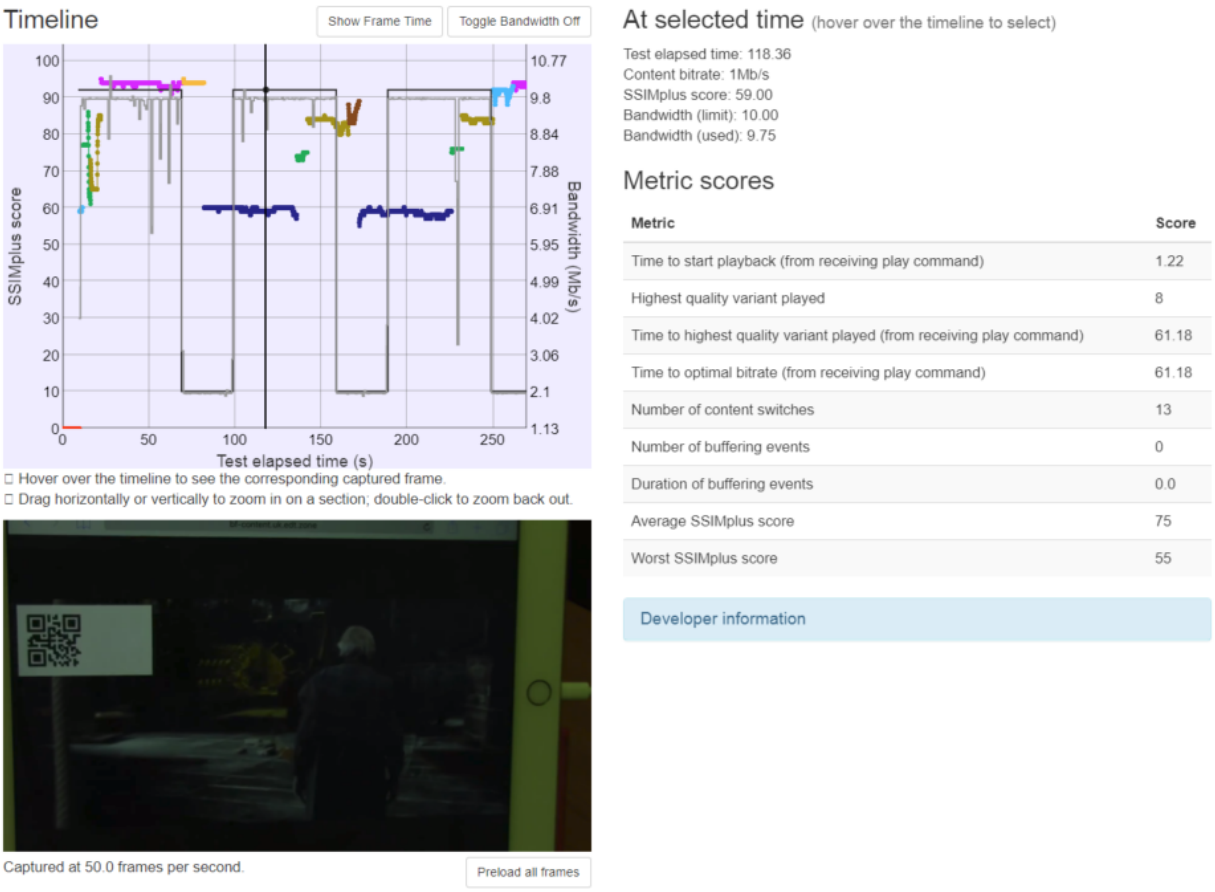


Figure 4: A typical results visualisation for a single test run

7. Summary

Our QoE analysis tool has succeeded in fulfilling our design requirements and has been successfully used to evaluate video streaming QoE performance across multiple projects. This has involved more than 20 device/app combinations, 10 encoding profiles and over 10 network profiles; leading to hundreds of individual test runs. Already this has led our customers to have a greater deeper insight into how their system choices impact QoE – in many cases reaching conclusions that are non-obvious and counter-intuitive.

If you would like to see a demonstration of the system, please contact us at digitaltesting@eurofins.com. Alternatively, you can explore a subset of the results by visiting https://ott-qoe.eurofins-digitaltesting.com.