DASH-IF position Paper: Proposed QoE Media Metrics standardization for segmented media playback

version 1.0 published October 7. 2016

ABSTRACT

Modern Quality-of-Experience (QOE) metrics are collected by a client-side analytics component which queries a media player for playback information, generates QOE data and then reports it back to a central analytics server for aggregation, analysis and display. Today, different client components from different analytics companies calculate metrics, such as “average rebuffer rate” in subtly different ways. This makes it both difficult and inaccurate to compare performance across players, devices, and analyti cs systems. This position paper proposes a client architecture as well as a set of consistent QoE metrics in order to stimulate the ecosystem to agree on such metrics. The proposals are considered as initial input to discussions in other organizations and should neither be considered final nor permanent.

Comments on this paper are welcome at https://gitreports.com/issue/Dash-Industry-Forum/Metrics

DISCLAIMER

1 INTRODUCTION

Modern Quality-of-Experience (QOE) metrics are collected by a client-side analytics component which queries a media player for playback information, generates QOE data and then reports it back to a central analytics server for aggregation, analysis and display. Today, different client components from different analytics companies calculate metrics, such as "average rebuffer rate" in subtly different ways. This makes it both difficult and inaccurate to compare performance across players, devices, and analytics systems. If every Over-The-Top (OTT) player calculated QOE metrics in a consistent manner, then the ecosystem would have a more accurate and consistent view of performance and the quality experienced by end-users would rise.

2 USE CASES AND ARCHITECTURE

2.1 Use Cases

2.1.1 Use Case 1: Monitoring of single CDN

A content owner is monitoring OTT distribution using a single CDN across SmartTVs, laptops and mobile phones. Their analytics system tells them that the rebuffer rate is higher on the TVs than the phones in a given DMA. They would like to have confidence that the problem lies with the performance of the player and is not an artifact of how the rebuffer rate was calculated.

2.1.2 Use Case 2: Monitoring of multiple CDN distribution

A content is owner is monitoring OTT distribution across multiple CDNs. They use the average bitrate delivered and the startup time of previous sessions in order to assign new sessions to the optimally performing CDN. They need the metrics to be consistently calculated in order to make the correct assignments.

2.1.3 Use Case 3: Debugging

A CDN is debugging delivery issues in a given DMA. They have rebuffer rate data from multiple different customers each of whom use a different player to play segmented content. They need to have confidence that each customer's player calculates the metrics in the same way in order to isolate the fault in the delivery system.

2.2 Architecture

A generic modern Over-The-Top (OTT) media application (player) is shown in Figure 1. Inside the physical device, the operating system supports a player execution environment, within which runs the segmented media player and a 3rd party analytics client. The job of the analytics client is to query the player for information, which it does via accessing the events and properties of the player's external public API. The client will gather the information necessary to calculate QoE metrics and then reported them back to the analytics server.

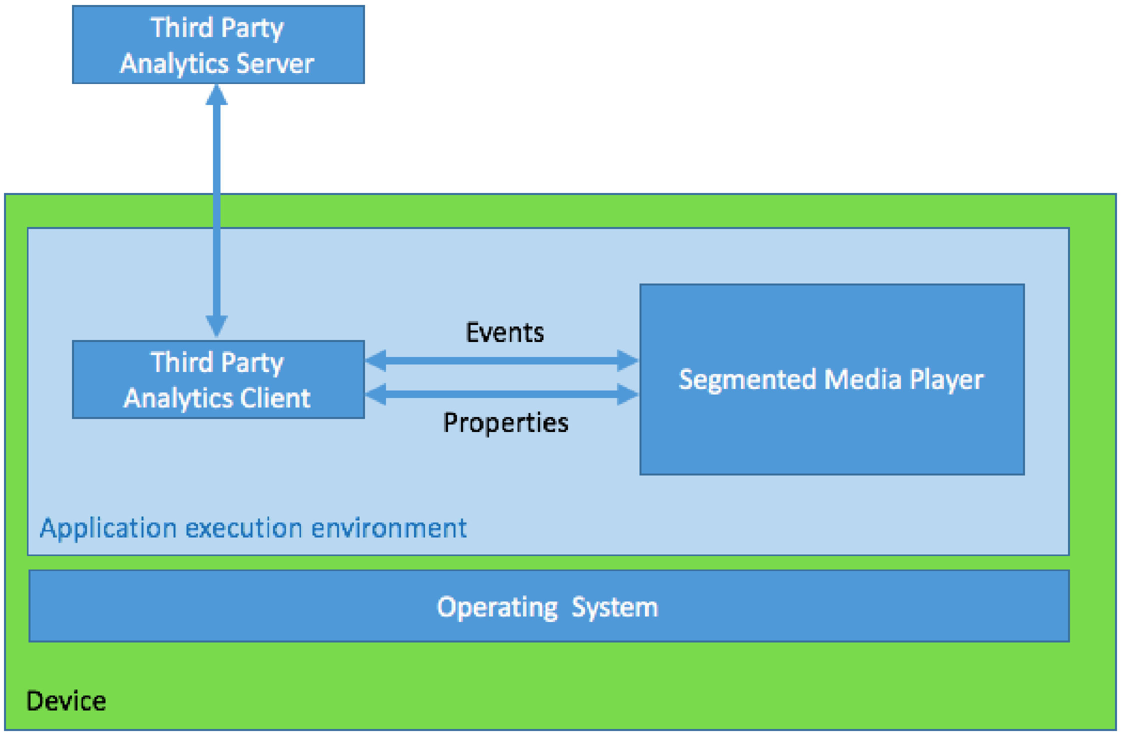
Third Party Analytics Server
Events
Third Party Analytics Client
Properties
Segmented Media Player
Application execution environment
Operating System
Device

Figure 1 Proposed Architecture

Many QoE metrics are developed from the perspective of VOD delivery, in which case the time window (the period of which the metric is calculated) can reasonably be assumed to be the duration of the video. With live linear streams, which run continuously and have no defined end-point, this can lead to problems. For example, does a rebuffer rate of 5% apply to the last minute, hour, day or week of that stream? To address this issue, this document proposes naming conventions for metrics which remove this ambiguity.

3 SCOPE OF THIS DOCUMENT

The primary scope of this document is the definition how the Third Party Analytics Client shall calculate QoE metrics.

What is out of scope is the following:

- We treat the Player and the Third party analytics client as black boxes. The player does not report the metrics on its own accord. The Analytics Client calculates the metrics by polling the player, or by listening to events, or a combination of both.
- The API between the analytics client and player may vary and is not defined by this document.
- Low-level metrics, such as HTTP and TCP session-related data, or decoding data, are out of scope.

4 STANDARDIZED PARAMETERS

4.1 Standardized QoE Time Types

When a media session is running, the time spent during the session can be accumulated into one or more timers. However, the notion of elapsed time can be expressed in several different ways, and this has an impact on how different metrics are calculated. The following types of "time" are defined in this document:

1 Media time
   a. t = 0 at beginning of media playout, i.e. first video or audio frame played
   b. runs when audio or video media is actually playing to the user, i.e. does not include bufferings, pauses etc.
   c. ends when user press stop or quits the session

2 Watched time
   a. t = 0 is when user first press play or autoplay is started
   b. runs as long as the user does not intentionally pause the playout, i.e. it also cover initial buffering and rebuffering
   c. ends when user press stop or quits the session

3 Session time
   a. t = 0 is when user first press play or autoplay is started
   b. runs whatever happen, i.e. also includes pauses etc.
   c. ends when user press stop or quits the session

4 Wallclock time
   a. t = wall clock time (local or UTC)
   b. runs whatever happens, even after user press stop or quits the session

Note that all of the above times are running at real-life speed. For instance, a 60-second content watched at half speed will accumulate a total media time of 120 seconds.

Metrics might optionally be calculated with regards to a certain time window. A time window can be expressed in any of the time types decribed above, and indicates that only the part of the session which fits inside the time window shall contribute to the metrics calculation.

For instance, a watched time window of 300 seconds means that the first metric shall cover watched time from 0 to 300 seconds, the next metric watched time 300 to 600 seconds, etc. If no time window is specified, the metric is calculated from t=0 until the end of the session.

4.2 Standardized QoE Player Properties

Properties are used by metrics agents, in conjunction with events, to synthesize QoE metrics.

1. Dropped frame count
   a. Name: droppedFrames
   b. Definition: the number of video frames dropped since start of playback session.
   c. Units: integer number
   d. Details:
2. Video bitrate
   a. Name: videoBitrate
   b. Definition: the bitrate of the video that is currently being rendered by the player
   c. Units: kbps
   d. Details
      i. This is the bitrate of the video that is visible to the end-user at the time of request, distinct from the bitrate that may be being requested at the front of the player buffer.

        ii. If the video is played at a different speed, the bitrate is adjusted accordingly. E.g. when playing at half speed, the bitrate is half of the normal-speed bitrate.

3. Audio bitrate
    a. Name: audioBitrate
    b. Definition: the bitrate of the audio that is currently being rendered by the player
    c. Units: kbps
    d. Details
        i. This is the bitrate of the audio that is visible to the end-user at the time of request, distinct from the bitrate that may be being requested at the front of the player buffer.
        ii. If the audio is played at a different speed, the bitrate is adjusted accordingly. E.g. when playing at half speed, the bitrate is half of the normal-speed bitrate.
4. Playhead time
    a. Name: playheadTime
    b. Definition: the presentation time of the media playhead at the time of request
    c. Units: seconds
    d. Details
        i. For live streams with DVR windows, playheadTime can appear static. In this case metrics agents should use playheadUTCTime, which is guaranteed to increase during live playback.
5. Playhead UTC time
    a. Name: playheadUTCTime
    b. Definition: the UTC time corresponding to the media playhead position at the time of request
    c. Units: seconds
    d. Details
        i. Use for live streams only.

4.3 Standardized QoE Player Events

Events are defined without payloads for broadest support.

1. **Initial Buffer start**
    e. Name: `initialBufferStart`
    f. Definition: event which indicates when the initial playback buffer starts to be built.
    g. Details:
        i. This event is only dispatched once per playback session (usually when requesting the manifest). It is not dispatched on every manifest request, which may occur multiple times with live streams. In the case where media is “preloaded” this event may occur before the user clicks play.
2. **Video | Audio playback start**
    a. Name: `videoPlaybackStart|audioPlaybackStart`
    b. Definition: events which indicates the video and/or audio is rendered and discernable to the end-user
    c. Details:
        i. This event must also be dispatched after a seek event.
3. **Rebuffer start**

   a. Name: rebufferStart
   b. Definition: events which indicates the time when a rebuffer starts
   c. Details:
      i. The initial buffer build at startup, or after a seek, is not considered a rebuffer. The media must be playing and then cease playback due to buffer starvation for a rebufferStart to be signaled.

4. **Video | audio bitrate changed**
   a. Name: `videoBitrateChanged`, `audioBitrateChanged`
   b. Definition: events which indicates when a rendered bitrate changes
   c. Details:
      i. This event should also be dispatched at startup when the first audio and video bitrates are selected.
5. **Pause activated**
   a. Name: `pauseActivated`
   b. Definition: indicates that the user has pressed Pause
   c. Details:
6. **Play activated**
   a. Name: `playActivated`
   b. Definition: indicates that the user has pressed Play, either initially, or after a pause.
   c. Details: If the media is auto-played, this event should also be dispatched at the same time as the `initialBufferStart` event.
7. **Playback can start**
   a. Name: `playbackCanStart`
   b. Definition: indicates when the buffer is full enough to begin playback.
   c. Details: This event is needed to track Initial Buffer Time in the case where the video is preloaded and not set to automatically play. If audio and video can start at slightly different times, the first of these are used.

4.4 Standardized QoE Metrics

Using the previously described events and properties exposed by the player, a metrics agent will be able to calculate the following QoE metrics:

1. **Initial Buffer time**
   a. Name: `initialBufferTime`
   b. Definition: time from when the initial buffer starts to be built, to when either audio or video is perceivable by the end user, or (if not autoplay) to when the buffer is full enough to immediately begin playback.
   c. Units: wallclock time seconds
   d. Example calculation: let t_1 equal the time at which the `initialBufferStart` event is received . Let t_2 equal the time at which the `videoPlaybackStart` event is received or the `playbackCanStart` event is received, whichever comes first . Initial buffer time is $t_2 – t_1$
   e. Details:
      i. Initial buffer time includes time spent retrieving the manifest, parsing the manifest, init segment retrieval if applicable, media segment retrieval, and establishment of starting buffer.

2. **Rebuffer count**
    a. Name: `rebufferCount_%%` where %% is the time window over which metric is calculated. If the time window is omitted from the name, then it implies that the metric is calculated from start of playback.
    b. Definition: the number of rebuffer start events that has occurred during a given watch time window.
    c. Units: an integer number
    d. Time window: integer seconds.
    e. Example calculation: let t_1 equal the watch time at the start of the chosen time window. Let t_2 equal the watch time at the end of the chosen time window. Let c_1 equal the total number of `rebufferStart` events at t_1 and c_2 the total number `rebufferStart` events at t_2. RebufferCount_t2-t1 is defined as (c2 – c1).
    f. Details:
        i. A rebuffer incident is a cessation in the playback of either audio or video due to insufficient forward buffer. There is no duration associated with a rebuffer incident. A rebuffer that lasts 500ms and one that lasts 5s are counted identically from a `rebufferCount` point of view.
        ii. Video and/or audio must be playing and then stop in order for it to count as a rebuffer.
        iii. Only the beginning of the rebuffering incident is counted. If a 60s video plays without rebuffering for 50s and then at the 50s mark it rebuffers and never recovers until the use terminates playback, the rebuffer count would be 0 if requested up until the 50s mark and then 1 thereafter.
3. **Rebuffer rate**
    a. Name: `rebufferRate_%%` where %% is the time window over which metric is calculated. If the time window is omitted from the name, then it implies that the metric is from start of playback.
    b. Definition: the rate of started rebufferings per a given watched time window.
    c. Time window: integer seconds.
    d. Example calculation: let t_1 equal the watched time at the start of the chosen time window. Let t_2 equal the watched time at the end of the chosen time window. Let c_1 equal the total number of `rebufferStart` events at t_1 and c_2 the total number of `rebufferStart` events at t_2. RebufferRate_t2-t1 is defined as (c2 – c1)/(t2-t1).
    e. Details:
        i. 4 rebuffering starts over the past 5 minutes would give a rebufferRate_300 = 4/300 = 0.0133 rebufferings/second
4. **Rebuffer percentage**
    a. Name: `rebufferPercentage_%%`, where %% is the time window over which metric is calculated. If the time window is omitted from the name, then it implies that the metric is from start of playback.
    b. Definition: the time spent in a rebuffer state over a given watched time interval.
    c. Time window: integer seconds.
    d. Example calculation: Let t_1 be the watched time at the start of the time window. Let t_2 be the watched time at the end of the time window. Within t_1 to t_2, there will be n rebuffering

intervals, that start at watched time t_{1n} and end at watched time t_{2n}. Define rebuffer percentage as 100 * Σ $(t_{2n} - t_{1n})/(t_2 - t_1)$ for all n.

e. Details:
   i. Rebuffer percentage is calculated with respect to watch time and not media time. For example, if the player rebuffered for 20s and played for 40s in the first minute of playback, the rebufferPercentage_60 would be 20/60 = 33.3%.
   ii. If the player is still in a rebuffering state at the end of the time window, then t_{2n} will be considered to be the value of t_2.
   iii. If the player is already in a rebuffering state at the start of the time window, then t_{11} will be considered to be the value of t_1.

5. **Average rendered video|audio|total bitrate**
   a. Name: `averageVideoBitrate_%%`, `averageAudioBitrate_%%`, `averageTotalBitrate_%%`, where %% is the time window over which metric is calculated. If the time window is omitted from the name, then it implies that the metric is from start of playback.
   b. Definition: the average media bitrate rendered by the player. The metric is specified for three different media types:
      i. video – only the video bitrate is averaged
      ii. audio – only the audio bitrate is averaged
      iii. total – the sum of video and audio is averaged.
   c. Units: kilobits per second
   d. Time window: integer seconds.
   e. Example calculation: Let t_1 be the media time at the start of the time window. Let t_2 be the media time at the end of the time window. Within t_1 to t_2, there will be n bitrate periods without switching, that start at media time t_{1n} and end at media time t_{2n}, and each use bitrate b_n Define average rendered bitrate as Σ $(b_n(t_{2n} - t_{1n}))/\Sigma$ $(t_{2n} - t_{1n})$ for all n.
   f. Details: Average rendered bitrate is measured against media time and not wall-clock time. It discounts time spent in a rebuffering state or in pause mode.

6. **Bitrate switch count for audio|video**
   a. Name: `audioSwitchCount_%%`, `videoSwitchCount_%%` where %% is the time window over which metric is calculated. If the time window is omitted from the name, then it implies that the metric is calculated from start of playback.
   b. Definition: the number of times the rendered media bitrate changed during a given window of media time. The media component can be audio or video or muxed.
   c. Units: integer number
   d. Time window: seconds
   e. Example calculation: let t_1 equal the media time at the start of the chosen time window. Let t_2 equal the media time at the end of the chosen time window. Let c_1 equal the total number of `bitrateChanged` events at t_1 and c_2 the total number `bitrateChanged` events at t_2. SwitchCount_t2-t1 is defined as (c2 – c1).
   f. Details:
      i. If a segment contains multiplexed audio and video data, then the switch count is retrieved using `videoSwitchCount`.

   ii. The first bitrate selected for start is not considered a switch. If the player selects a bitrate at start and plays it consistently through to the completion of the presentation, then the bitrate switch count is zero.
   iii. This metric tracks switches that are visible to the end user (i.e rendered) and not just queued up in the buffer.
   iv. There is no notion of total bitrate switches, i.e it is never appropriate to sum the switch counts of the audio and video segments if they are delivered de-multiplexed to the player.
   v. Bitrate switch count is measured against media time and not wall-clock time. It discounts time spent in a rebuffering state

7. **Bitrate switch rate for audio | video**
   a. Name: `bitrateSwitchRateAudio`_%%, `bitrateSwitchRateVideo`_%%, where %% is the time window over which metric is calculated. If the time window is omitted from the name, then it implies that the metric is from start of playback.
   b. Definition: the rate of bitrate switches per a given media time window.
   c. Time window: integer seconds.
   d. Example calculation: let t_1 equal the media clock time at the start of the chosen time window. Let t_2 equal the media time at the end of the chosen time window. Let c_1 equal the total number of `bitrateChanged` events at t_1 and c_2 the total number of `bitrateChanged` events at t_2. `bitrateSwitchRate`_t2-t1 is defined as (c2 – c1)/(t2-t1).
   e. Details: Bitrate switch rate is measured against media time and not wall-clock time. It discounts time spent in a rebuffering state or pause mode.

8. **Dropped frame count for video**
   a. Name: `droppedFrameCount`_%%, where %% is the time window over which metric is calculated. If the time window is omitted from the name, then it implies that the metric is from start of playback.
   b. Definition: the number of frame drops that have occurred over the indicated time window.
   c. Units: an integer number
   d. Time window: seconds
   e. Example calculation:: let t_0 equal the media time at the start of the time window, t_1 the media time at which the metric is requested and t_2 the time at which the presentation completed or playback was terminated by the end-user. Dropped frame count is the number of dropped frames which have occurred between t_0 and the lessor of t_1 or t_2.
   f. Details:
      i. A dropped frame occurs when a decoder does not have sufficient data or CPU to render a frame at the correct time and it must drop the frame in order to keep up with its clock cycle.
      ii. Example: droppedFrameCount_60=3 indicated that 3 frames have been dropped in the 60s prior to this metric being generated.