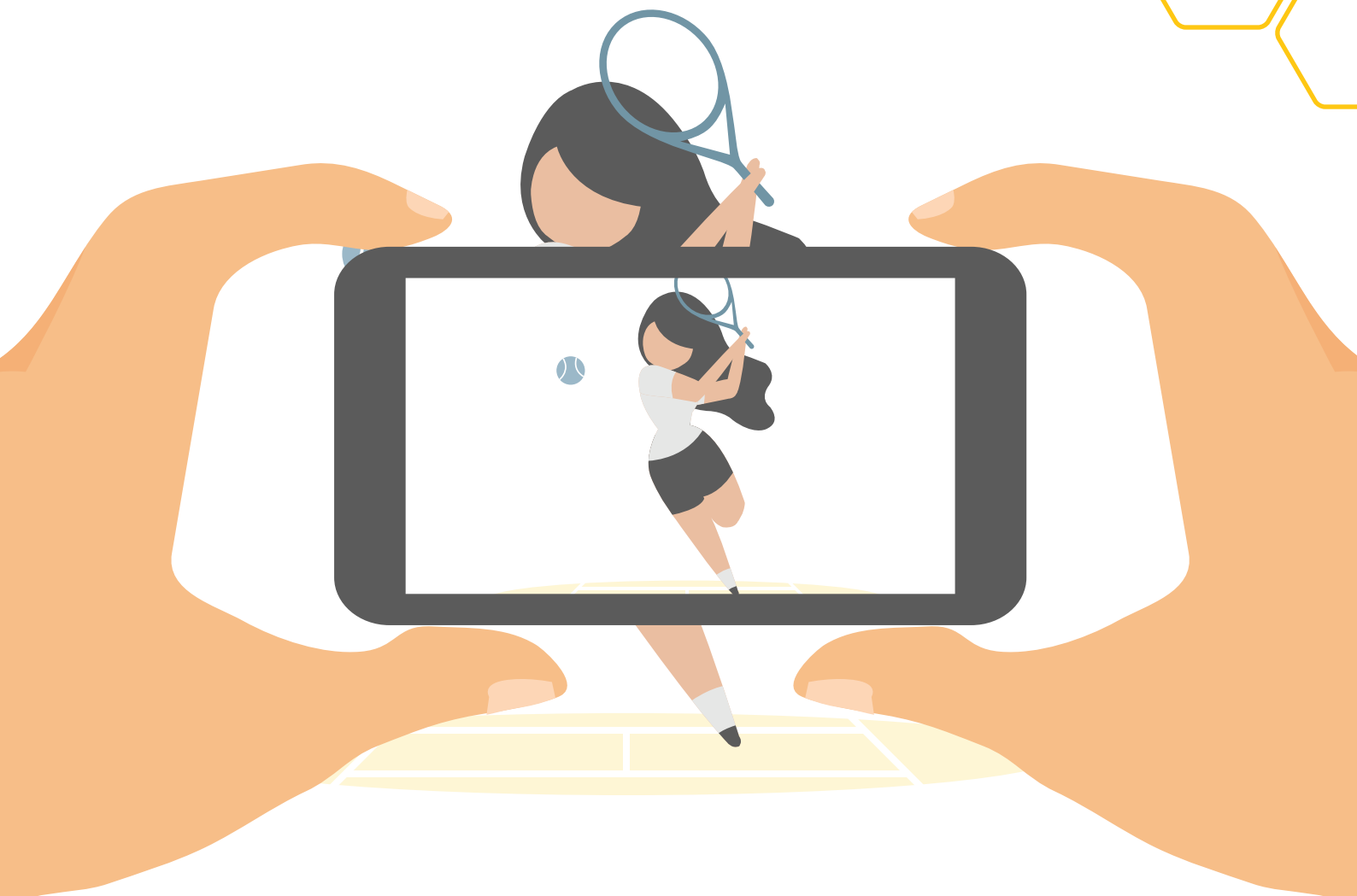
THEO

HIGH EFFICIENCY STREAMING PROTOCOL

Ultra-low latency, ultra-fast zapping, low bandwidth at scale.



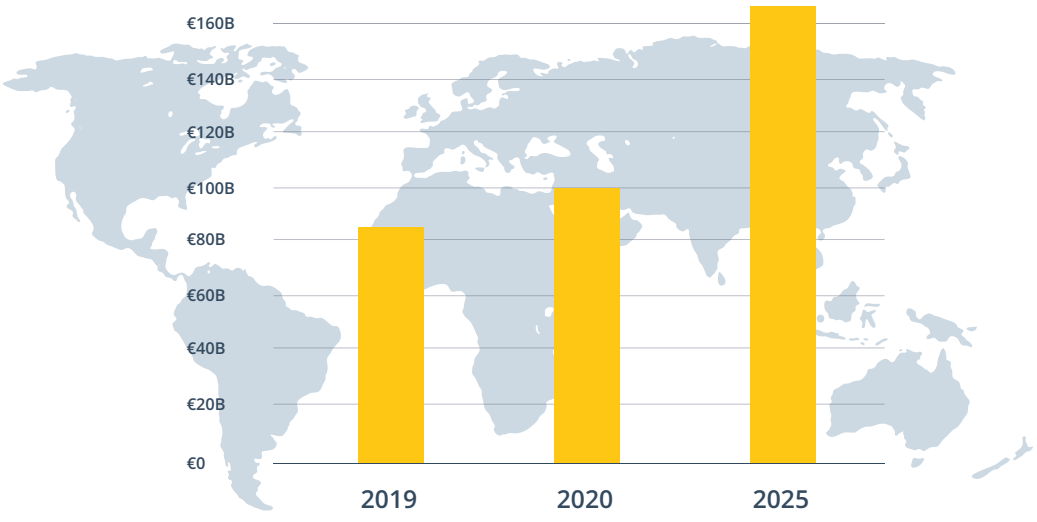
High Efficiency Streaming Protocol: ultra-low latency, ultra-fast zapping, low bandwidth at scale

The video delivery protocol to reduce latency and bandwidth at scale

The media industry is constantly pushing to innovate and improve viewer experience to attract and tie customers to their service. However, this is hampered by technical hurdles such as high latencies, long video start-up times and bandwidth constraints. With the High Efficiency Streaming Protocol (HESP), a massive leap forward is made possible. The new protocol enables streaming services to be delivered at scale with sub-second latency, while reducing bandwidth. This already impressive list of improvements is further complemented by almost instant channel change and start-up times, pushing the viewer experience and engagement to the next level.

Online video is booming, calling for scalability

Year after year the industry sees an increase in online video consumption, both in terms of hours viewed and in terms of online video revenues. According to Digital TV Research, global online video revenues will reach $167 billion in 2025, which is double the $83 billion recorded in 2019. Each year the number and variety of target platforms and devices grows tremendously. This calls for online streaming solutions that are both scalable and deployable on virtually every connected device with a screen. HTTP Adaptive Streaming (HAS) is used for just that reason. HAS ensures scalability over the Internet and adaptability to variations in the available bandwidth. The use of HAS protocols leads to universal access on every device at every location.



Online viewers are more demanding than ever

Not only is the volume of online video streaming skyrocketing, but viewers are also more demanding than ever. Thumbnail size, blurry videos are no longer exciting people, and viewers are no longer prepared to go for a cup of coffee before a video starts.

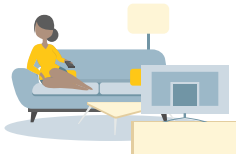
Today's viewers are no longer satisfied with delays between live events and online viewing. A latency of 30 to 40 seconds, or sometimes even a minute, is no longer accepted. About 60% more people would watch live events online if the stream is not delayed from the broadcast[1]. More than a third of the respondents indicated sub-second latency requirements in a recent survey.[2] Sub-second latency will also enable applications such as interactive conferences and online teaching. People expect low start-up and zapping times, and they poorly rate[3] and abandon the service[4] if start-up and zapping take too long. All of these expectations continue to increase. Today's viewers want the same quality of experience they know from mainstream TV on any device and any type of connectivity.

60%

Almost 60% of people would be more likely to watch live sports online if the stream was not delayed from broadcast.

100ms

Instantaneous zapping substantially improves quality of experience and viewer retention.



Lean back experience becomes the minimal requirement.

HESP Characteristics

HESP offers a true broadcast-like video streaming experience combined with advanced interactivity.

Sub Second Latency

- No spoiled user experiences
- In-sync delivery across devices
- Interactive user experiences

Up to 20% reduction in bandwidth and costs

- Cost savings for content distributors
- Re-invest bandwidth saving in delivery of more and higher quality videos

Scales over existing infrastructure

- Cost efficient delivery over existing HTTP infrastructure
- Compatible with standard encoders
- Deliver over standard CDN's
- Cross-platform playback support

Enhanced viewer experience

- Instant zapping and seeking times
- Full adaptive bitrate (ABR) to respond to all different kind of network conditions

[1] https://www.limelight.com/resources/white-paper/state-of-online-video-2019/
[2] https://www.wowza.com/wp-content/uploads/Streaming-Video-Latency-Report-Interactive-2020.pdf
[3] Kooij, R., Ahmed, K. and Brunnstrom K. (2006), Perceived quality of channel zapping, fifth IASTED International Conference Spain, pp. 155-158.
[4] http://www.iariajournals.org/systems_and_measurements/sysmea_v2_n23_2009_paged.pdf

Online video is booming, calling for scalability

As shown in the table below, HESP outperforms other streaming protocols.

	HESP	CMAF-CTE	LL-HLS	HLS/DASH	WebRTC	RTMP
Latency	Ultra-Low	Low	Low	High	Ultra-Low	Ultra-Low
Bandwidth	Low	High	High	Low	High	High
Zapping Time	Ultra-Low	Trade-Off	Trade-Off	Trade-Off	Low	Low
Scalability	Low Cost	Low Cost	Average	Low Cost	High Cost	High Cost
Cross-Platform	Yes	Almost	Yes	Yes	Almost	No
ABR	Yes	Yes	Yes	Yes	No	Yes

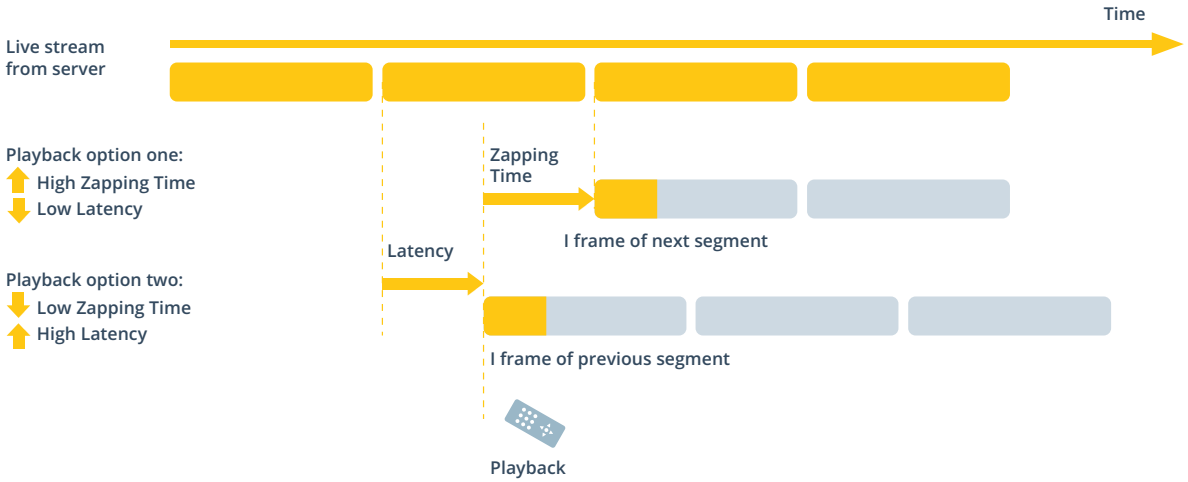
HESP brings the same ultra-low latency as RTMP and WebRTC, while bandwidth usage and the cost of scaling remain low.

The reason for this is the fundamental difference in interaction between the player and the backend (server/CDN). In WebRTC clients connect directly to an active streaming server. That enables sub-500 millisecond latency, but it can only be scalable for live streaming at high infrastructure costs as it requires an active streaming server for each client. For HESP, as well as for other HAS protocols, there is no need for an active streaming server dedicated to each client. An HTTP/file server, with less resources, is enough to serve the small segment files: a player will request the segments; the server very efficiently delivers them to the player. This approach implies more cost-efficient scaling.

HESP is more bandwidth efficient than other HAS protocols, especially when used for ultra-low latency. By using longer segments and less I-frames, the bandwidth usage drops. Better encoding quality in HESP further improves bandwidth efficiency. Since HESP is ultra-low latency, it is possible to trade off some of the latency gain to give more time to the encoder. With more time available, the encoder will be able to compress more for the same visual quality, which leads to less bandwidth usage.

Furthermore, fewer manifest updates are needed with HESP, as there is no need for a player to fetch an updated manifest file for every single segment. This is contrary to DASH, for example, where a manifest update takes place for every segment (e.g. every 6 seconds). For LL-HLS a manifest update even takes place for every single part, so maybe every 200 milliseconds (ms), or 5 times per second, which causes a lot of traffic.

HESP has the lowest zapping time among HAS and non-HAS protocols as a new HESP stream can be started at any moment with no waiting time. This is in contrast with LL-HLS and LL-DASH where start-up and zapping time are dependent on the first available key-frame in either the current or the next segment. In the first scenario, the I-frame of the current segment is already available on the server, which results in a low zapping time, but increases latency as start playback from a position in time that can be several seconds in the past. In the second scenario, while waiting for the next segment to arrive, the viewer could have a higher zapping time, but a lower latency.



Protocols such as webRTC offer low channel change times. However, the player cannot start displaying at any given moment. The player needs to wait for the ongoing GOP to be completed (similar to the segment for LL-HLS and LL-DASH), and a key frame to be available.

HESP offers advanced ABR functionalities and provides cross-platform support. HAS protocols have already established a decent footprint on most mobile devices and streaming devices. For IPTV and delivery to STB, however, existing HTTP Adaptive Streaming approaches have significant downsides. HESP aims to solve this problem by optimizing delivery towards these platforms as well, targeting smart TVs, connected devices, RDK and Android TV-based STBs. HESP is well suited for lean back TV.

HESP fundamentals: simplicity, reduced overhead and better performance

HESP offers a true broadcast-like video streaming experience combined with advanced interactivity.

HESP video streaming

HESP is delivered over HTTP/1.1[5]. The protocol uses Chunked Transfer Encoding (CTE) at a very small granularity, i.e. each chunk contains a frame, to allow for very low latency. HESP additionally uses byte range requests, as these allow to start at a given position in the video, which is very beneficial to reduce the start-up latency.

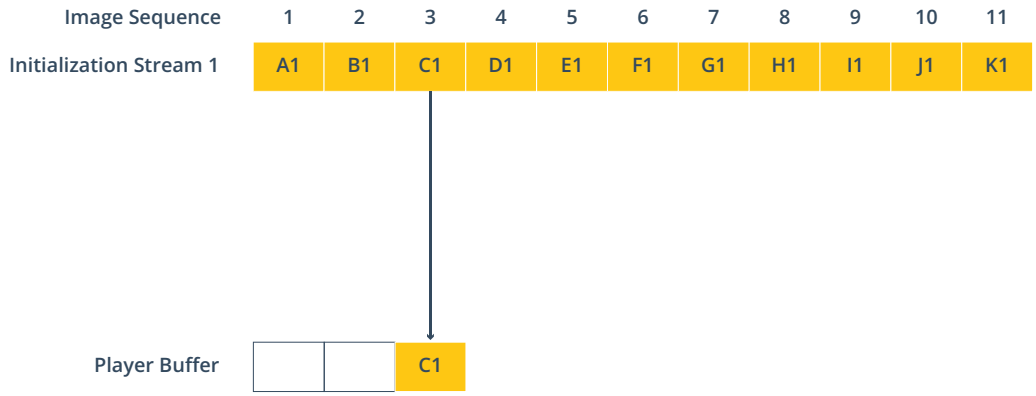
HESP relies on two complementary streams to achieve its astonishing results.

The first stream, the initialization stream, contains only key frames. This stream is solely used when a new stream starts. At that moment, the most recent image that is available in the initialization stream is requested (or another image if we want to start at a specific location). As the initialization stream's images are key frames, playback can start immediately.

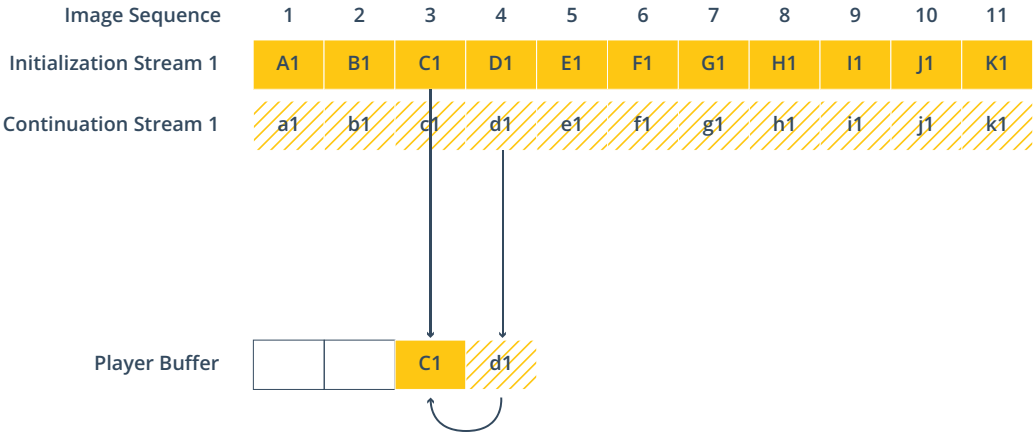
Key frames are expensive in terms of bandwidth, so it is important not to continue playing out the following images from the initialization stream. This is where the continuation stream kicks in. After the key frame of the initialization stream, images are requested from the continuation stream. The continuation stream is a regularly encoded stream for low latency purposes. The images from the continuation stream are requested, starting at exactly the right location (the image following the image fetched from the initialization stream), by using a byte range request. The start position of the byte range request is sent alongside the initialization stream's image.

This mechanism is explained in the figures below: Assume Initialization Stream 1 is available with frames A1, B1, and so on. At some point in time, a user wants to start watching the video. The player will then request the most recent initialization frame, e.g. C1.

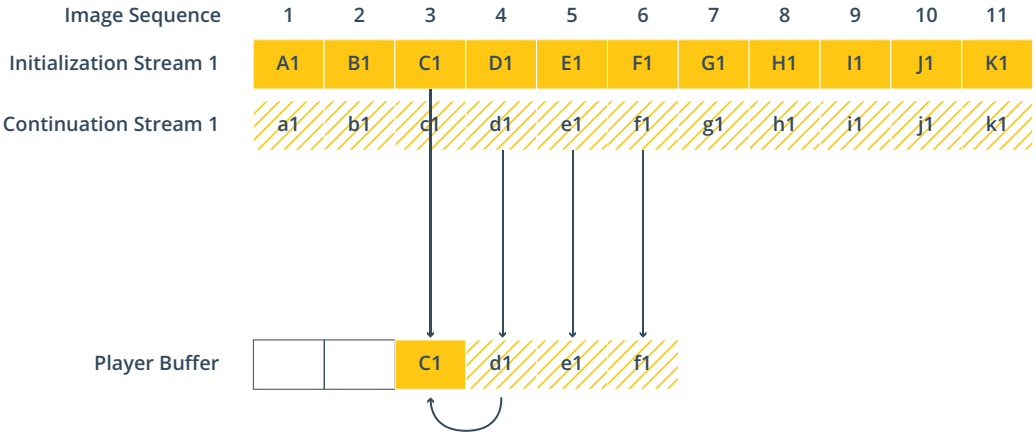
[1] Alternatively, HTTP/2 frame-based streaming could be used.



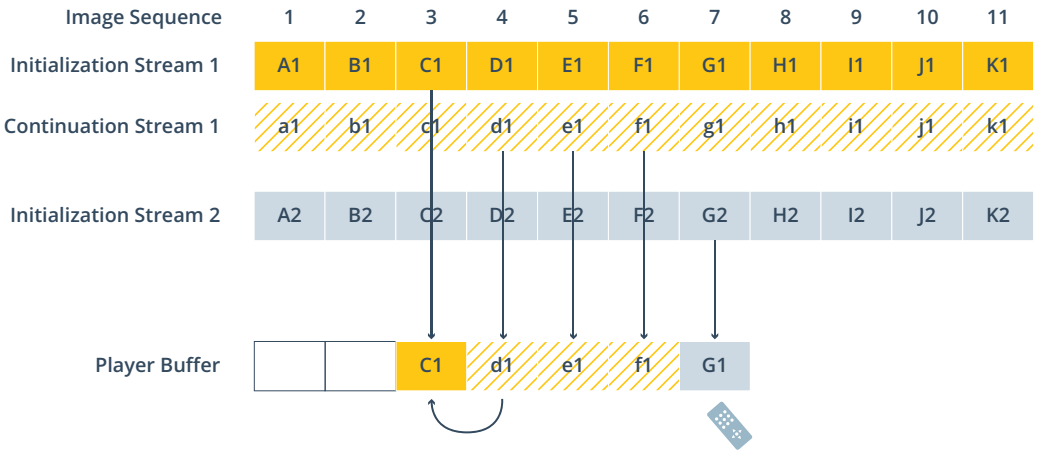
Next, the player will automatically request the following images from the corresponding continuation stream 1, at exactly the right location: d1.



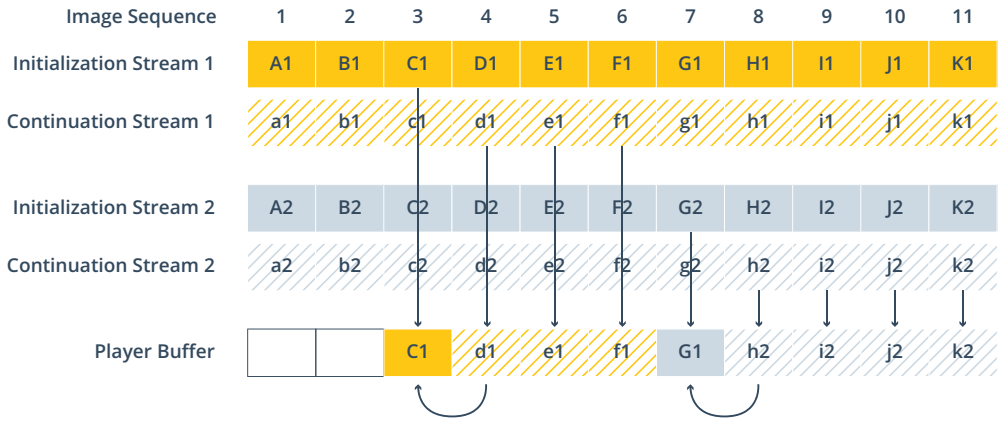
And then followed by e1, f1, and so on. The byte range request ensures that all the following frames are automatically fetched as well in a single request. The protocol details of HESP define how to find d1, e1, etc.



When the viewer wants to change channels and watch Stream 2, the player then requests the most recent initialization frame of this second video feed, e.g. G2.



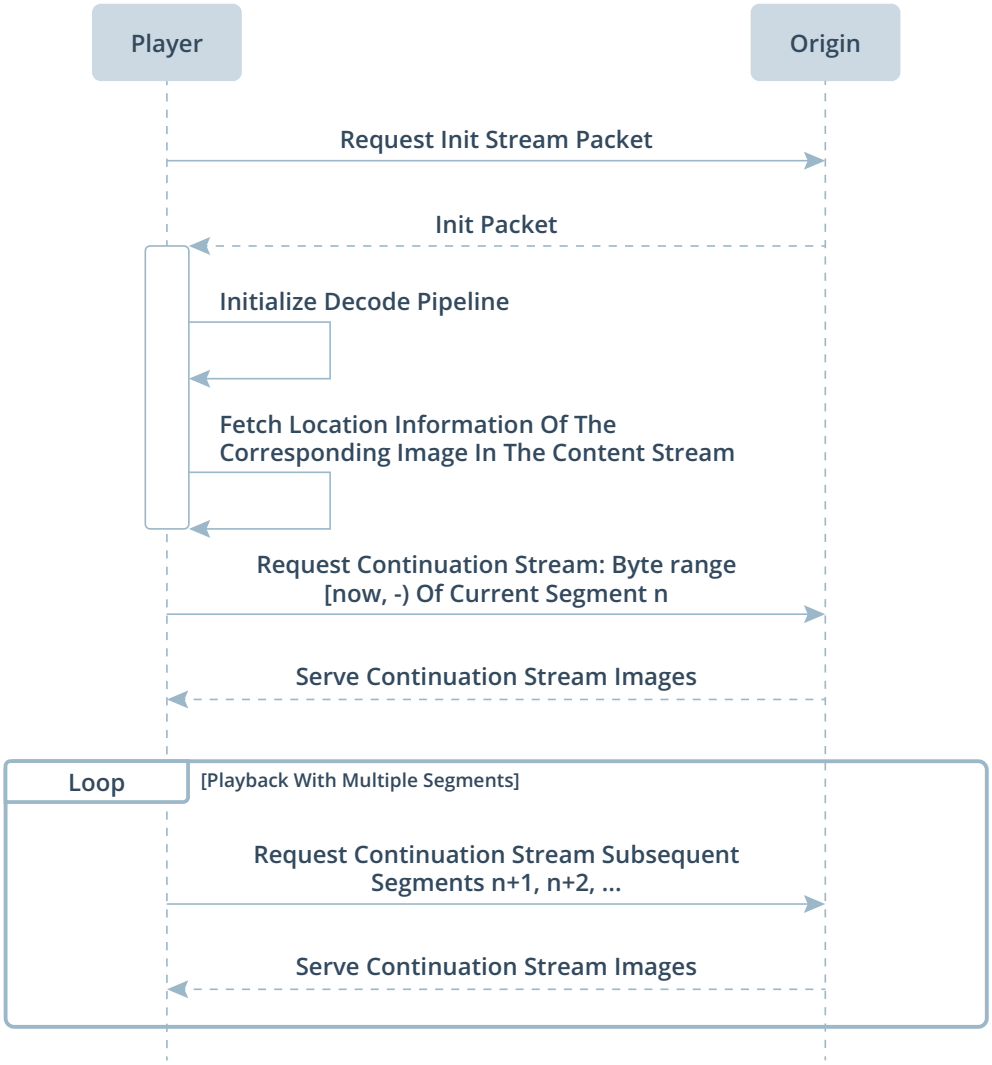
After that, the continuation stream takes over again.



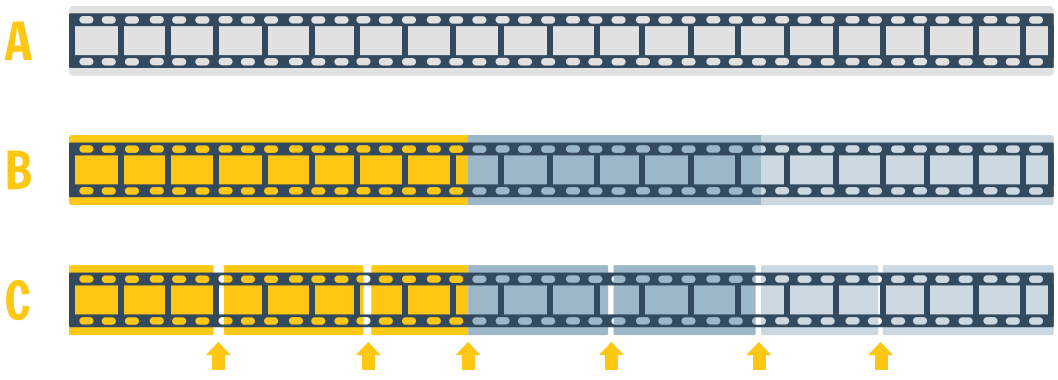
HESP manifest and video control flow

HESP uses manifests in the same way as DASH and HLS, but with a slight difference: an HESP manifest only contains minimal information of the video stream, such as the available qualities. This information does not change often, unless a new piece of video arrives, such as advertisement, if a new quality is added, or when a new audio track is exposed.

Contrary to HLS and DASH the player must not regularly fetch the manifest to ensure the playback of the different video segments in the continuation stream. Instead, a player will automatically request all segments. The segment addressing is calculated automatically for an efficient and continuous delivery of the continuation stream.



Typically, a video feed consists of several sections (called presentations in the HESP terminology). A presentation is a piece of video that logically belongs together, such as the first, second, third and fourth quarter of a sports game, an advertisement insert, or a set of news items. In that case the manifest file changes.



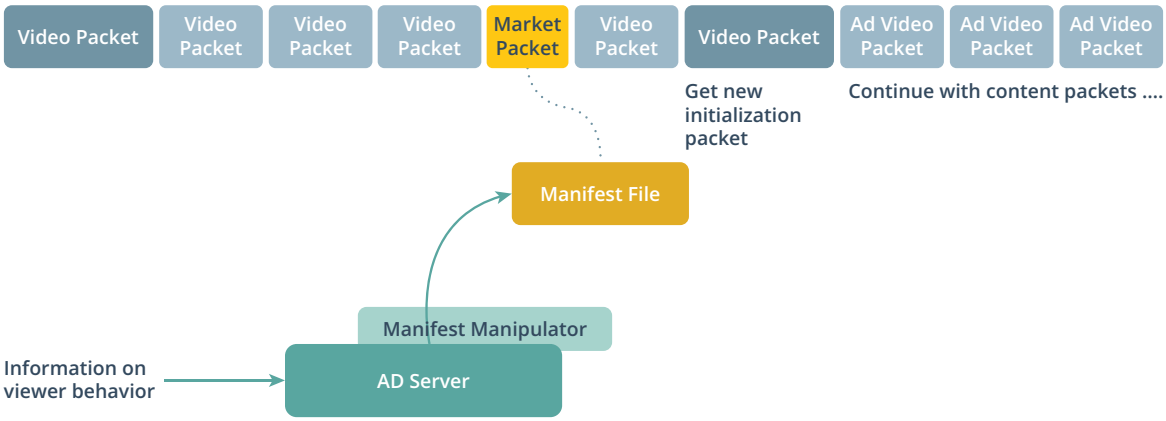
A A video is a complete feed to the viewers.

B A video is a set of presentations (in this example we have a yellow, a dark grey and a light blue presentation). A presentation is the lowest granularity inside a manifest. A manifest only is updated when presentations are updated.

C A presentation is a set of segments (in this example, segment boundaries are indicated with arrows). Segment addressing happens automatically within a presentation for an efficient and continuous delivery of the continuation stream.

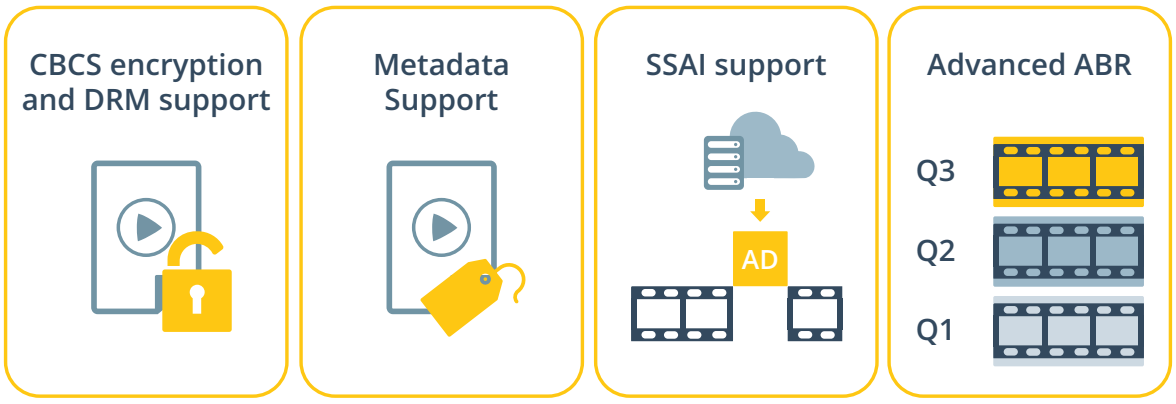
The player is informed that the manifest has been updated by inserting a marker in the continuation stream. The marker itself does not contain manifest information. It simply triggers the player to download (a new version of) the manifest file again. This manifest file contains information on the new presentation (when it will start, its URL, its qualities, ...). At the right moment, the player will request a new initialization packet, corresponding to the new presentation, and will subsequently request the frames from the continuation stream.

An example use case for these markers is insertion of advertisements. The ad server will then be the driver to modify the manifest and to insert a marker.



HESP Features

HESP is an HTTP Adaptive Streaming (HAS) approach, fully leveraging the existing HAS infrastructure. HESP adopts CMAF-CTE as a packaging format, which means it can tap into a rich ecosystem of functionalities.



CBCS encryption and DRM support

The HESP streaming format is based on CMAF-CTE. Therefore it inherits everything available for CMAF-CTE, such as encryption or DRM) support. HESP supports CENC and CBCS encryption methods, as well as DRM systems such as Fairplay on Apple devices and Widevine on non-Apple devices.

Metadata support

Since HESP is based on the CMAF-CTE format, existing metadata formats, such as subtitles or non-linear ads are also supported. However, having the possibility to add metadata is not enough. Special attention needs to be paid to the ultra-low latency nature of HESP. As an example, subtitles typically come with a start time and an end time for display. However, in case of ultra-low latency, the end time is not yet known e.g. a subtitle starts at frame 1252 and ends at frame 1401, but frame 1401 is not yet available. HESP includes measures to ensure a smooth and flicker-free display of subtitles.

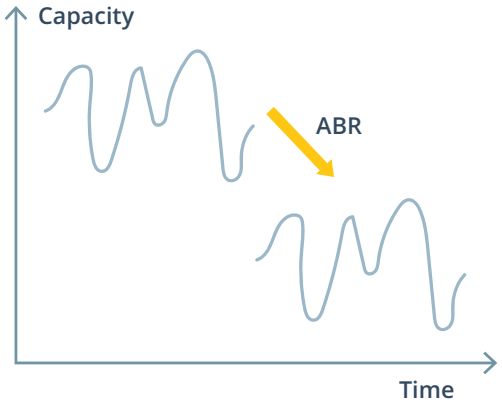
SSAI support

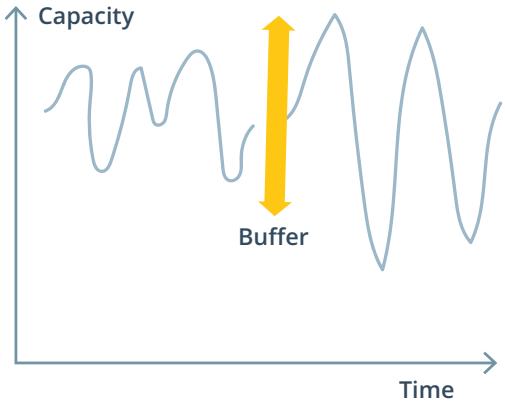
HESP supports multi-content streams, such as an advertisement insert, with guaranteed and efficient continuity. A new advertisement insert will lead to an update of the manifest file to include the advertisement content. This new manifest file is signaled through the mechanism of using markers. Markers are used to trigger the download of a new manifest file and are inserted using Event Message (emsg) boxes.

ABR (Adaptive Bitrate streaming) and buffer management for HESP

To effectively cope with network variations HESP includes solutions for (i) measuring the available bandwidth capacity and selecting the best suited quality (and corresponding bitrate) of the video for the available network bandwidth – ABR – and (ii) managing the player buffer size efficiently, both continuously and in real-time.

ABR and buffer management are the two key components for handling network variations. In an ideal network condition, the images arrive at a fixed pace. In practice, however, images can be delayed because of packet loss, jitter, different image sizes, or simply because there is not enough bandwidth. ABR is then needed to adjust to the available capacity and the buffer will cope with high frequency network variation.

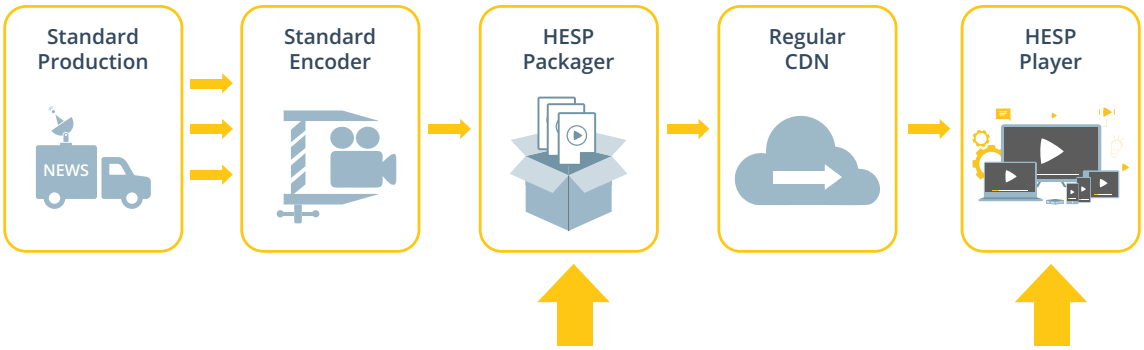




In order to have a smooth and continuous playback of images, it is important that the quality of the stream optimally matches the bandwidth capacity, and also the buffer is neither empty, to avoid stalls or freezes, nor too large, to minimize latency.

Impact on the workflow

To implement HESP, two components of the video workflow need to be modified: the packager and the player. HESP works with regular encoders and with regular CDNs, if these support CTE (as for LL-DASH) and byte ranges (as for LL-HLS).



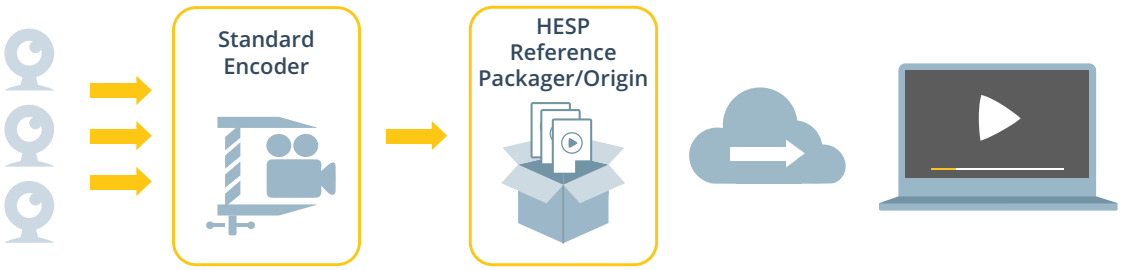
HESP Profiles

The HESP streams can be configured to meet different purposes by changing the size of the chunks and segments, by using B frames or not, ... This leads to different profiles, two of which are the Maximal Gain profile and the Maximal Compatibility profile.

Maximal Gain
In the Maximal Gain profile, the target is to provide the lowest latency, the lowest bandwidth and the lowest zapping and start-up times. The continuation stream is made of I & P frames only, referring to only one previous frame. The CMAF-CTE chunks are ultra-short - 1 frame - while the segments are long i.e. several minutes or more.

Maximal Compatibility
In the Maximal Compatibility profile, the goal is to re-use LL-DASH and LL-HLS streams. The continuation stream is a CMAF-CTE stream with regular sized segments (~6 seconds) and chunk sizes around ~200ms (6 frames at 30fps). The continuation stream is made of I, P and B frames (B B B P subGOPs in a chunk). There is no bandwidth gain in this profile, although latency, zapping time and start-up time decrease remarkably.

PoC setup

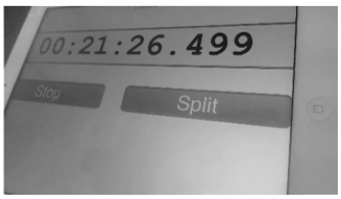


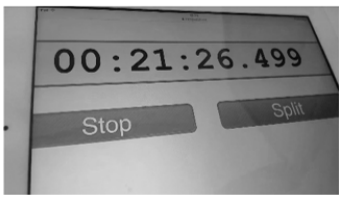
The purpose of this PoC is to validate HESP's low glass to glass latency, HESP low zapping times and the synchronized multi-viewing enabled by HESP.

The test setup consists of webcams capturing the environment. The webcam streams are encoded using standard encoders and packaged with an HESP packager. Next, the video feeds are streamed to the HESP player over regular HTTP Internet connections.

HESP synchronized multi viewing

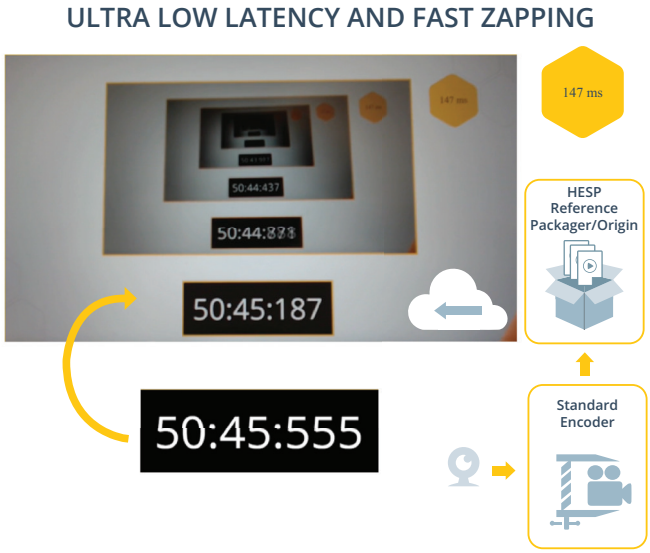
In a first setup, three webcams are pointing towards the same clock. The camera feeds are shown in a browser with multiple (3) player windows. Each player window is showing one camera feed. The synchronization between the feeds can be measured by comparing the displayed clock times.
The screenshots below show that the HESP feeds are nicely in sync, with at most one or two frames difference. This HESP feature is important to allow viewers to watch different camera feeds coming from the same event, such as the different camera angles of a sports game.





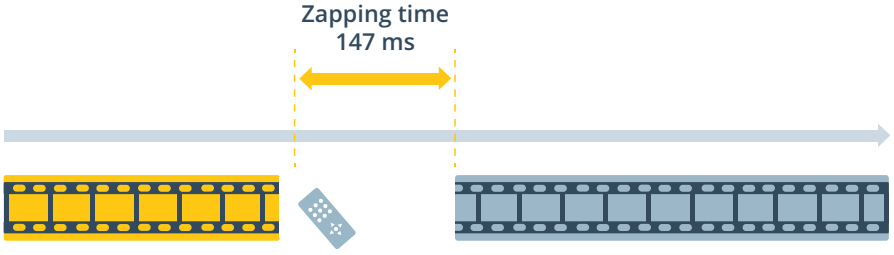
HESP low glass to glass latency

In a second setup, the end to end latency is measured by displaying a reference clock, filming this clock, and displaying the filmed clock. The glass to glass latency is then simply derived by calculating the time difference between the reference clock and its playback in the video player.
The screenshot below shows the HESP glass to glass latency, which is between 300 and 400ms in this example.



HESP fast zapping

The zapping time is measured by changing from one camera feed to another. Zapping time is the time from the request to change to a different video channel until the moment that the first image of this new video feed is sent to the display. Mostly, values of a few 100ms are achieved. In the shown test result, the zapping time was 147 ms, which is lightning fast to the human eye.



Summary

As outlined and shown by the test results above, HESP has a number of interesting properties:

1. Sub second latency
2. Bandwidth reduction
3. Instantaneous channel change and ultra-fast zapping time as low as 100ms
4. Synchronized viewing
5. Scalability across standard HTTP CDNs
6. Instant ABR switching capabilities.
7. DRM, SSAI and Metadata support
8. Easy integration in existing streaming architectures.

These properties enable streaming service providers to improve their service offering in multiple ways:

1. Setting up interactive streaming experiences where viewers can interact in near real-time with the streamed content.
2. Increase user satisfaction by providing instant video startup and channel zapping.
3. Reduce bandwidth cost.
4. Allow to scale a low latency streaming solution over HTTP to reach all popular platforms and devices.
5. Deliver a low latency stream which can dynamically adapt to viewer environments, switching between qualities immediately (ABR).
6. Reduce latency for time sensitive content, avoiding spoiled experiences for viewers.

CONTACT US

Discover what THEO can do for you.

www.theoplayer.com
contact@theoplayer.com