THEO

A COMPREHENSIVE GUIDE TO

LOW LATENCY

THEO TECHNOLOGIES

Latency, low latency, ultra-low latency are becoming increasingly important.

New developments like LL-HLS and CMAF-CTE both confirm and support this statement, in addition to other streaming protocols such as webRTC and RTMP.

With all the technology and the definition of latency, it can be difficult to see the forest for the trees. So let us start with a few, all equally valid, definitions of latency.

WHAT IS LATENCY?

Glass-to-Glass or End-to-End Latency

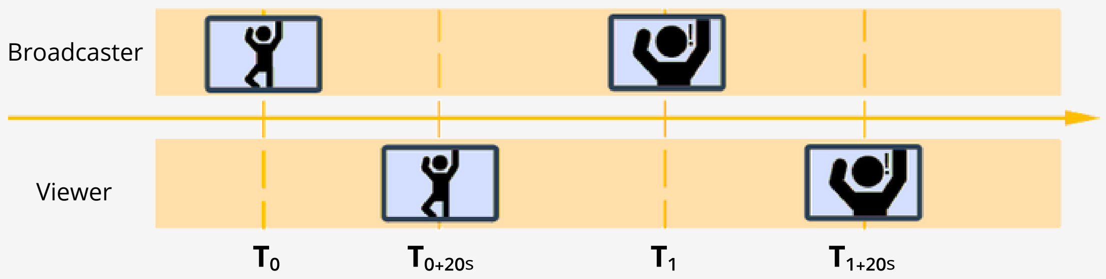


Figure 1 - Latency between broadcaster and viewer

The most trivial definition of latency is the so-called glass-to-glass latency or the end-to-end latency. That is **the time it takes between the moment that action happens (and is in front of the first glass, the camera) and the moment that a viewer sees this action on his screen (the other glass)**. This definition of latency is especially useful for the streaming of live and interactive events.

Protocol, Startup and Channel Change (Switch Latency)

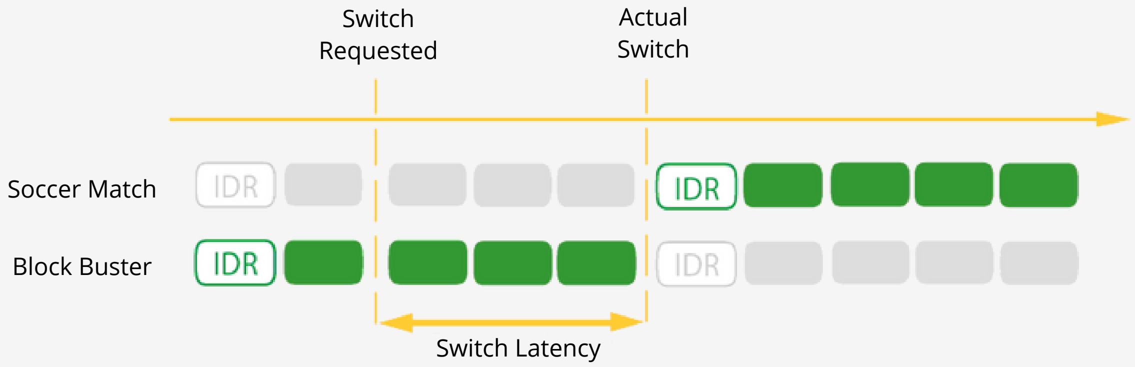


Figure 2 - Switch Latency

A second definition is the so-called protocol latency. **That is the latency between the output of the encoder and the actual playback**. This latency is interesting for low latency applications where we do not want to compromise the quality of the encoder. We also have the startup and channel change latency. That is the time it takes to start a video or to change channel, once the command has been given. This is an important parameter for streaming video applications that want to provide a leanback TV experience where viewers are used to instantaneously change channels with a simple push on a button.

Not every latency is equally important for every use case.

In the table below we indicate which latency really matters for five typical use cases:

Use Case	Startup Time	Channel Change Time	Protocol Latency	Glass-to-Glass Latency	Description
Broadcast	+++	++	+++		Protocol latency is important to ensure simultaneous arrival on main screen and on OTT devices. Startup and channel change times are crucial to ensure a leanback TV experience and to ensure that people stick to the service.
VoD	+++				Playback need to start rapidly. VOD user interface are designed not to need fast channel changes. Protocol and glass-to-glass latency is not important.
Live Events	++	(++)	Implicit	+++	Glass-to-glass latency is crucial. Startup latency is important as for every video service. Channel change is important for large events with multiple stages or multiple cameras
Video Calls	++	(++)	Implicit	+++	Glass-to-glass latency is the major criterion for video calls (and even more so for the audio)
Interactive Events	++	(++)	Implicit	+++	Glass-to-glass latency is crucial for interactuive events (though mostly slightly lower than for video calls). Channel change is important for setups with multiple concurrent interactive events.

Table 1 - The Importance of Latency in Different Use Cases

WHERE IS LATENCY INTRODUCED?

Latency is introduced at many different steps in the video distribution chain.

1. Firstly, **the encoding/transcoding takes time with a direct impact on the glass-to-glass latency**. A use-case dependent trade-off will be needed between latency, quality and bitrate. Typically, quality and (smaller) bitrate will be preferred unless for applications where the glass-to-glass latency is crucial.

2. Secondly the **distribution networks between source and playback device adds to the latency**, as well glass-to-glass, protocol, startup and channel change latency. CDNs allow them to benefit from dedicated networks and to reduce the overall load on the distribution network by caching as much as possible.

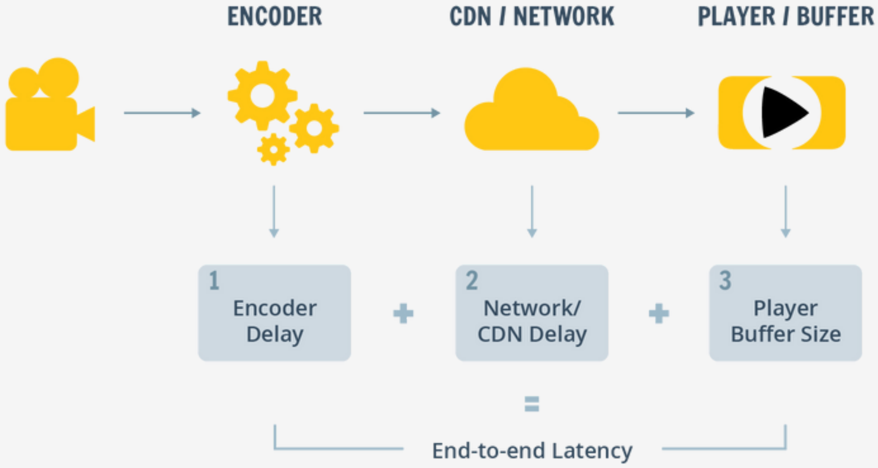


Figure 3 - End-to-end Latency

3. Thirdly, **the player buffer adds to the latency**. Players use buffers to cope with network variations and to avoid stalls. A trade-off is necessarily dependent on the importance of the latency and the quality of the network. This is also true for startup latencies and channel change times. One needs to define the minimal amount of buffered video before the playback actually starts.

4 Overall **the streaming protocol has a large impact on the different types of latency**, because it defines how the video is divided into packets that are transferred and it directly impacts the buffer depth. Tuned or dedicated protocols are needed to achieve ultra low latencies and startup times.

The different streaming protocols all have a different glass-to-glass / protocol latency. The table below gives an impression of the capabilities of the different protocols:

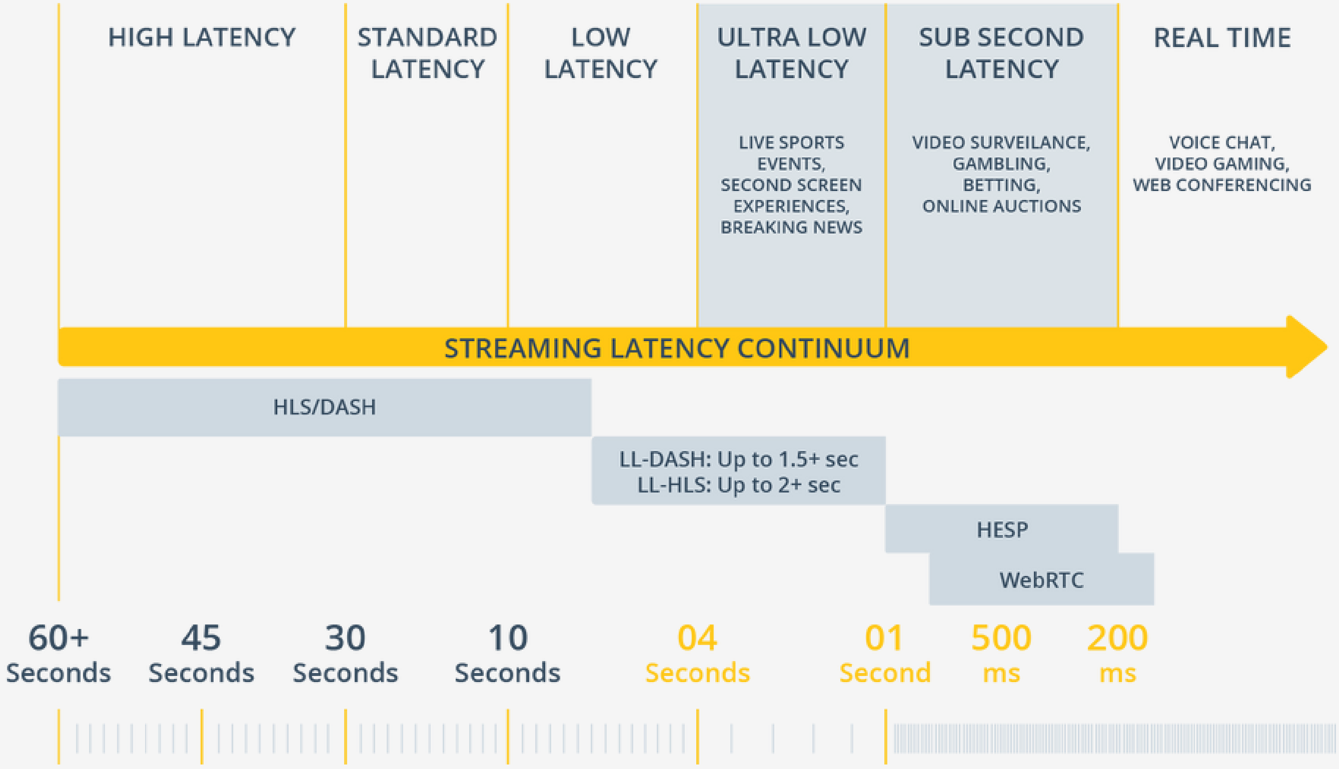


Figure 4 - Protocol Latency

We see that **the use of the traditional DASH and HLS protocols leads to large latencies**. These latencies can be reduced by shortening the segments. But the latency remains high because a segment is handled as an atomic piece of information. Segments are created, stored and distributed as a whole. **LL-DASH and LL-HLS overcome this problem by allowing a segment to be transferred piece-wise**. A segment does not need to be completely available before the first chunks or parts of the segment can be transferred to the client for playback. This significantly improves the latency.

For ultra-low latency, approaches are needed that allow for a continuous flow of images that are transferred as soon as they are available (rather than grouping them in chunks or segments). This can be done using webRTC and HESP (HESP Webinar & Whitepaper), THEO Technologies' next generation streaming protocol. HESP using

Chunked Transfer Encoding over HTTP whereby the images are made available to the player on a per image basis. **That ensures that images, extremely rapidly after they are generated, are available at the client for playback.**

Of course, this only gives one aspect of these protocols.

The zapping time is also important. For DASH and HLS this is a trade-off with the latency since playback can only start at segment boundaries. That implies that a player needs to choose between waiting for the most recent segment to start or starting playback of an already available segment. In case latency is not critical this allows for a very fast startup. If latency is critical, there is a penalty for the startup time.

> *WebRTC allows for a shorter zapping time,*
> *but still is bound to GOP size boundaries.*

HESP allows for ultra-low start and channel change times, without compromising on latency. HESP does not rely on segments as the basic unit to start playback. HESP can start playback at any image position. As explained in (reference), HESP uses range requests to tap into the stream of images that is made available for distribution as soon as they are created.

Low latency and fast zapping is fine, but **scalability is equally important, especially for video services reaching out to tens or hundreds of thousands of concurrent viewers**. HTTP based approaches (DASH,LL-DASH, HLS, LL-HLS, HRSP) have an edge over webRTC, since HTTP based approaches ensure the highest possible network reach, can tap into a wide range of efficient CDN solutions and achieve scalability by file servers. **WebRTC on the other hand relies on active video streaming between server and client and supports much less viewers on an edge server compared to a regular CDN edge cache**.

.

LOW LATENCY USE CASES

Video-on-Demand

VoD traditionally focuses on the highest possible quality for the lowest number of bits. Fast startup is the only latency metric that really impacts the user experience. Besides adopting the right streaming video approach, user interfaces are adopting latency hiding techniques to give the viewers the impression of instantaneous startup times. This includes prefetching the video or starting at lower qualities so that the video is transferred faster and starts earlier.

Broadcast

Broadcast traditionally focuses on quality of experience for a large audience. This calls for longer encoding times to ensure the best possible visual quality of the video for a given bandwidth budget, but also for fast start-up and channel change times and for scalability. Latency is becoming increasingly important as well, driven by the desire to have the playback on online devices occur at the same time as the existing broadcast distribution.

Therefore, this industry gradually moves to shorter segment sizes, and to LL-DASH and LL-HLS. This still does not give a really good solution in combination with fast channel change though, because **a trade-off will have to be made between start-up and channel change times on one hand than the latency on the other hand.**

Live Event Streaming

Live event streaming critically depends on low glass-to-glass latency. Traditional HLS and DASH protocols are not satisfactory. Therefore, live event organizers are using WebRTC. This works fine for small audiences. The cost of scaling for WebRTC is high though. Consequently, live event organizers targeting mass audiences are looking for LL-DASH and LL-HLS when they can afford the increased latency. HESP brings an answer here, with slightly higher latencies than webRTC but the same scaling characteristics as any HTTP based approach, largely outperforming webRTC.

Bi-directional video conferences are using webRTC.

REAL-LIFE EXAMPLES

To make this more concrete we will zoom in on a few examples with a focus on latency:

THEO reaches between 1 and 3 seconds latency with LL-DASH and LL-HLS depending on the player and stream configuration. In the past few months THEO engaged with several customers worldwide, both for PoCs and for real deployments, reaching latencies of around 2 sec in real life conditions..

Synamedia, Fastly and THEO set up an end-to-end demonstrator with HESP, **reaching out to a virtually unlimited number of viewers with sub-second protocol latency and zapping times well below 500msec [MHV 2020].**

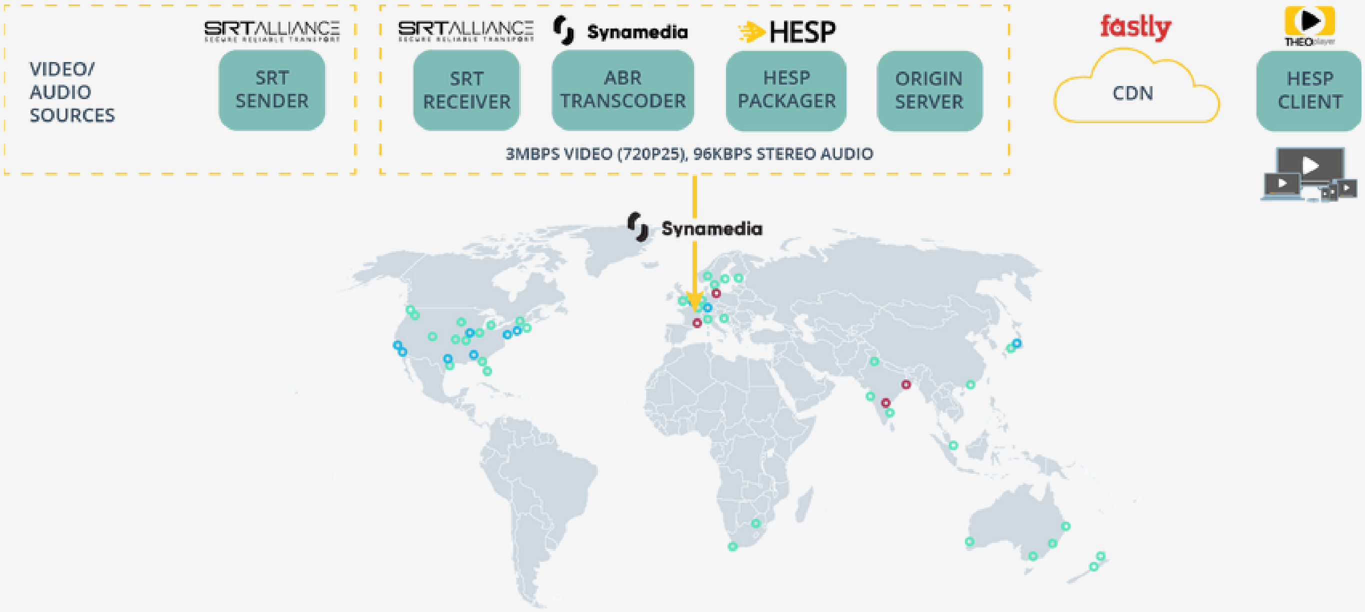


Figure 5 - Synamedia, Fastly and THEO HESP Demonstration

WebRTC is being used for video conference tools such as Google meet.

Youtube Live brings live content **with a delay of several seconds.**

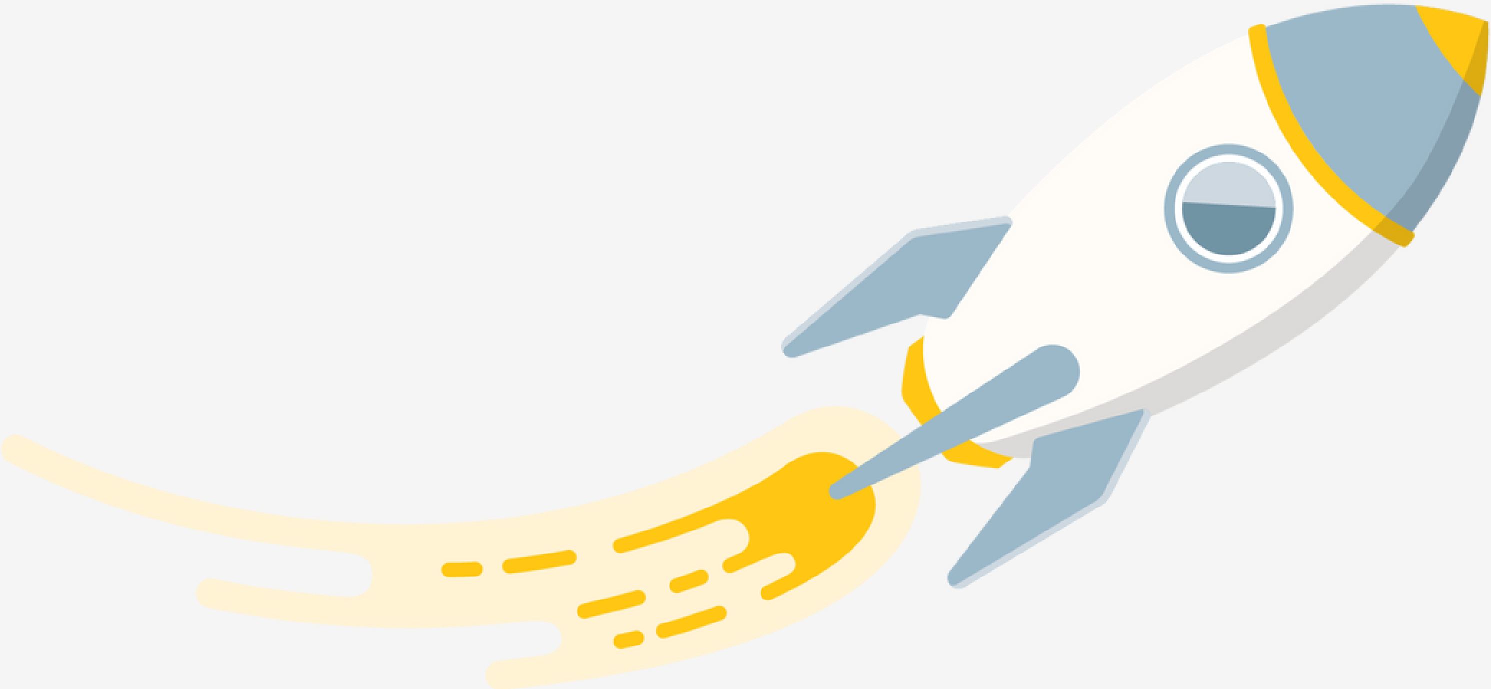
Wowza has a hybrid system for live events, using WebRTC for a limited number of ultra-low latency critical participants and LL-HLS for the rest. [LL-HLS Webinar with THEO and Wowza]

CONCLUSION

Different applications come with different low latency expectations.

Existing technologies are typically designed to cover a range of latency needs. For stringent latency requirements (down to a few seconds) we need LL-HLS and LL-DASH. Sub-second latency at scale is made possible by HESP, the High Efficiency Streaming Protocol. **WebRTC is capable of achieving even lower latencies, but often at the expense of quality of experience, and falls short when it comes to scalability to a large number of viewers.**

Any questions left? Don’t hesitate to reach out to our team.



THEO

INTERESTED IN REDUCING YOUR LATENCY?

Get in contact with one of our THEO experts.

PIETER-JAN SPEELMANS
CTO & Founder

BART VAN OOSTERHOUT
VP Product Management

WILLEM DE SAEGHER
VP Global Sales

Ask an expert