Synamedia

WHITE PAPER
Video Network

# How to Deliver a Live OTT Video Service that Matches Broadcast

# How to Deliver a Live OTT Video Service that Matches Broadcast

In the video distribution industry today, there is a growing trend among service providers to replace traditional live broadcast video delivery with Over-The-Top (OTT) delivery. There are a couple of reasons for this move:

Reducing workflows: Currently, most service providers need to maintain two completely different workflows: one workflow, using broadcast delivery, serves the legacy set-top boxes. The second workflow, delivered via Adaptive Bitrate (ABR) streaming, serves all other client devices, such as PCs, smartphones, tablets. The differences in these workflows are quite significant. They employ different encoders, ad insertion workflows, and content recording workflows. They also run on different access networks. Because of the high-cost involved in maintaining these two distinct workflows, it would make financial sense to converge them into one single workflow.

Introducing new value-add services: OTT delivery allows service providers to add new value-add services that would not be possible with broadcast video. For example, they can deliver more granular advertising that is targeted to individual subscribers with tailored content.

If a service provider decides to reap the benefits of OTT and converge to a single video delivery workflow, it is essentially the OTT workflow that they are choosing. When replacing live broadcast video with OTT delivery, the expectation would obviously be that the user experience should improve rather than degrade. This is the point where some weaknesses of traditional OTT delivery can be exposed, posing problems that require solving before any convergence can take place. The main issues with traditional ABR streaming for live video delivery are:

- **Latency**
- **Video Quality and Bandwidth Efficiency**
- **Scalability**

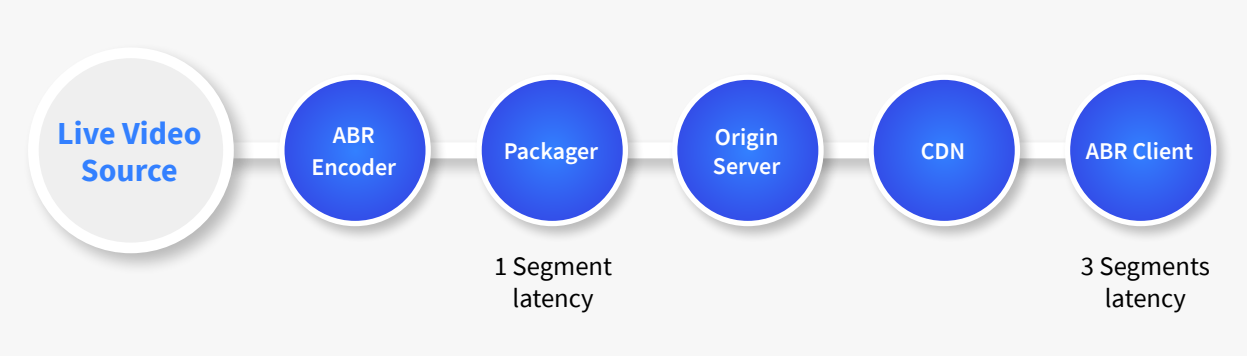
This White Paper explains these challenges in more detail and proposes solutions for overcoming each of them.

## Latency

Traditional OTT delivery of live content has been notorious for having a high latency. When talking about latency in the context of this paper, we mean live-latency. In other words, the latency between a live event—such as a goal being scored in a soccer game—and the moment you see that live event on your client device. Live-latencies of 45-60s with traditional OTT delivery are not unusual.

The reason for these high latencies is that traditional ABR streaming protocols (such as Apple HLS) were originally designed with reliability in mind. They were never intended to compete on latency with live broadcast. When using ABR technology to stream live content over the "unmanaged internet", all kinds of issues can occur, such as, network congestion, packet loss etc. These make it difficult for the ABR client to maintain a high-quality service. The popular way to overcome this challenge was to buffer a large amount of content in the ABR client. This gave the client enough time to adapt to the changing networking conditions and maintain a continuous video delivery to the user while avoiding the dreaded buffering icon.

When looking at a traditional ABR pipeline, the main contributors to latency are the ABR encoder, the packager, and the ABR client. The origin server and the Content Distribution Network (CDN) typically contribute very little towards latency.




The latency of the ABR encoder is mainly determined by Video Quality requirements (the higher the latency of the ABR encoder, the easier it is to increase the Video Quality) and is highly independent of the ABR delivery. As such, modifying the ABR streaming protocols will not help to reduce the latency of the encoder.

The latency of both the Packager and the ABR client, however, are highly dependent on the segment duration used by the ABR streaming protocol. As an example, the Apple HLS recommendation is to use 6s-segments. A Packager requires at least a single segment duration before it can make its content available to the client. The client itself needs to download and buffer at least one segment before it can start displaying the content. However, for reliability reasons as explained above, a typical client buffers at least three segments, resulting in a live-latency of at least three segment durations in the client.

Since the live-latency has a one-to-one correlation with the segment duration in use, an obvious and easy solution would be to lower the segment duration. To some extent this is correct but there are constraints on how short you can make ABR segments.

The first reason is that each ABR segment must start with an Instantaneous Decoder Refresh (IDR) video frame. This effectively means that you need to start a new group of pictures (GOP) in the encoder. Short

GOPs have a negative impact on video quality. In order to achieve broadcast-quality video, GOPs shorter than about one second are not recommended[1].

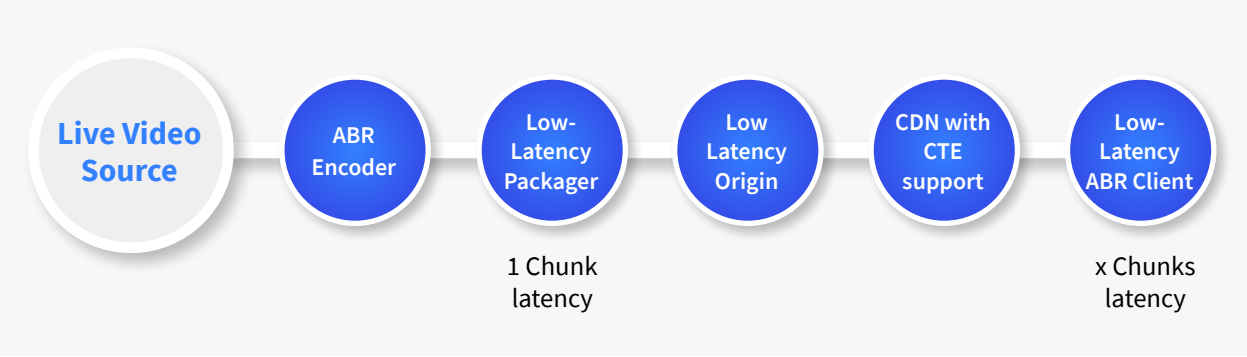
The second reason is that the shorter the ABR segments, the more segments the client will request from the CDN in a certain timeframe. This means that very short segments will significantly increase the load on the CDN, again impacting the delivery of the content at scale. For these reasons, the practical lower limit to segment duration is about 2 seconds (for some specific applications 1 second segments may also work).

## Work on ABR Streaming Protocols

Over the past couple of years, much work has been done to modify existing ABR streaming protocols in order to lower the live-latency and bring it on par with broadcast delivery. Examples are low-latency DASH and low-latency HLS.

Low-Latency DASH provides a way to decouple the live-latency of the OTT pipeline from the segment duration. It does this by processing and delivering ABR segments in smaller pieces, so-called chunks. Chunks can have a much shorter duration than segments. For example 200ms chunks versus 6s segments. Chunks also don't need to start with an IDR frame (except for the first chunk of a segment). A low latency packager can make content available to the client as soon as the first chunk of a segment is available. This means the latency of the low-latency packager is no longer related to the segment duration but rather to the chunk duration. As soon as the packager makes the first chunk of a segment available, the client can start downloading the first chunk of that segment. This means that the latency of the client is also no longer related to the segment duration but rather to the chunk duration. The protocol to transport chunks in the packager and the client in low-latency DASH is called HTTP/1.1 Chunked Transfer Encoding. This technology is not new and is commonly used for webpages. Its use for ABR streaming, however, is quite recent.

The following diagram depicts a low-latency DASH pipeline.




Note that in order to support low-latency, all components in the pipeline from the packager onwards must support HTTP/1.1 Chunked Transfer Encoding, otherwise latency will be sub-optimal. There are vendors today that can offer an end-to-end low-latency DASH pipeline.

Next to low-latency DASH, there is also a community-created low-latency version of Apple's HLS protocol which has been successfully deployed at scale in the field. This implementation makes use of the same HTTP 1.1 Chunked Transfer Encoding protocol as low-latency DASH and as such has many similarities in the way it is deployed and used.

[1] You could of course compensate the impact of a short GOP by increasing the video bitrate, but then you will run into other issues when having to deliver the content at scale.

Recently, Apple announced its own proposal for low latency HLS. It takes a fundamentally different approach to low-latency DASH and the community low-latency HLS implementation. Apple makes use of small partial segments which are delivered to the client via an HTTP/2 server push. As explained above, using small (partial) segment files creates a large amount of transactions and load in the CDN. In order to limit that, Apple came up with the idea of an active origin server that can deliver very specific playlists to a client while at the same time automatically pushing a segment along with the playlist. Apple's approach has a significant impact on the packager, origin server, and CDN.

## Video Quality and Bandwidth Efficiency

Delivering the same quality of service with OTT as with broadcast also means that video quality must be on par.

For over 20 years, the industry has been optimizing broadcast delivery to ensure premium video quality to its subscribers. In the early days of digital video broadcast over satellite or cable, video channels were encoded at a constant bitrate. This means that independently of the actual video content, each channel was encoded 24/7 at the same bitrate.

Since video complexity can change every second, encoding at a constant bitrate generates fluctuating video quality (VQ). For easy-to-encode video sequences, VQ will be high, while for difficult-to-encode sequences, VQ will be low. A popular way to improve video quality in broadcast delivery is to use statistical multiplexing. This method exploits the video complexity statistics of multiple video channels that are aggregated in a shared constant-bandwidth channel (for example, 10 video channels combined in a QAM channel). Video sequences within a channel will fluctuate over time between easy-to-encode and difficult-to-encode. But when combining multiple channels, chances are low that they all peak in complexity at the same time. In reality, there will be a statistical distribution of the video complexity over the different channels that will make some channels peak in complexity, while at the same time other channels will have low complexity. At a certain moment in time, the channels that peak in complexity will get a relatively large portion of the available shared channel bandwidth while the low complexity channels will get a relatively small bandwidth allocation. This statistical distribution over the different channels will change continuously and the bitrate distribution will be automatically adjusted on the encoders.

### Constant Bitrate Encoding: Throwing Money Down the Pipe?

Traditional ABR delivery uses constant bitrate encoding of the video profiles. Using constant bitrate encoding for ABR delivery has a number of drawbacks. First of all, encoding in constant bitrate means that you deliver inconsistent video quality to the subscribers, as already described above. Secondly, for easy-to-encode video content, you use more bandwidth than you really need. In ABR delivery using more bandwidth for a channel than you really need means your infrastructure costs will increase since either a public CDN is used where you pay per egress byte, or a private CDN is used that must be scaled to support this higher bandwidth.

Since ABR delivery means unicast delivery, there is no fixed shared bandwidth pipe and you cannot use the statistical multiplexing techniques that are used in broadcast.

You can, however, use a related technique called Constant Quality encoding (as opposed to constant bitrate encoding). With Constant Quality encoding, the encoder creates a bitstream that provides a constant video quality experience to the user. Since the complexity of the video changes over time,

encoding at constant quality means that the bitrate that is generated fluctuates over time. Constant quality encoding is configured by providing a video quality target on the encoder. However, next to a video quality target, a maximum bitrate ("cap rate") must also be configured in order to avoid highly-complex video content resulting in an excessive bitrate.

By using constant quality encoding, a consistent video quality is delivered to the subscribers.

Since you only use the bandwidth that is really needed, none is wasted. This not only saves on CDN delivery cost, it also allows you to set the cap-rate of the channels higher than what would normally be used as constant bitrate in CBR encoding. In doing so, the video quality of the channels will increase for highly-complex channels and for highly-complex sequences within a channel. As such, just like with statistical multiplexing in broadcast delivery, a similar statistical effect can be seen in ABR delivery but with potentially many more channels participating (i.e. all available channels versus only the channels in a statmux pool).

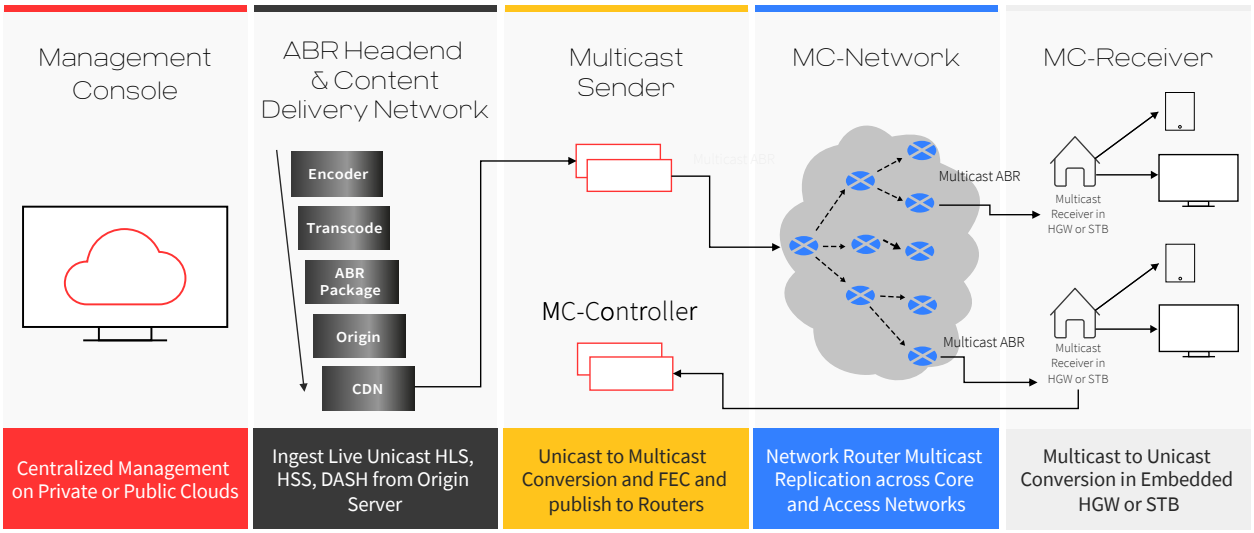
## Scalability

With broadcast delivery of video services, a video channel is only distributed once and can be consumed by many clients. As such, it is very efficient from a distribution network point of view.

Traditional OTT delivery however, requires unicast delivery over a CDN and over the access network to the subscribers. In cases where millions of people simultaneously watch a popular live event—such as the soccer World Cup final—this means that the very same content needs to be delivered millions of times to the different ABR clients in the network. That is exactly the reason why CDNs are used for ABR delivery. CDNs allow you to cache and distribute identical content to these millions of subscribers. Naturally, the CDN will need to be designed and scaled to handle those high loads of traffic.

The remaining problem, however, is the access network (for example cable network) that sits between the CDN and the subscriber at home. The access network is shared between potentially hundreds of subscribers and so doesn't have the capacity to provide unicast access for live video delivery to each of these individual subscribers.

One way to solve this problem is by using Multicast-ABR.




*Multicast-ABR solution*

With Multicast-ABR, video channels that are being watched by multiple subscribers on the shared access network are only sent once over the network. This is done by encapsulating the ABR segments into a multicast protocol and multicasting those segments over the access network to the home gateway or set-top box. The latter contains a Multicast-ABR receiver that extracts the ABR segments again from the multicast packets. The Multicast-ABR receiver then caches and proxies this content to ABR clients in the home via unicast. The ABR clients are not even aware that the content they request is delivered to the home via multicast. The decision on which content is delivered over the access network via unicast or via multicast is taken by the Multicast Controller and is based on viewing activity that is continuously monitored.

Combining low-latency ABR with Multicast-ABR poses some additional challenges on the Multicast-ABR system in order to keep the latency of the end-to-end system as low as possible.

## Conclusion

Replacing live broadcast video delivery with live OTT delivery, while maintaining the same quality of experience, clearly poses multiple challenges. This white paper has attempted to address the most important issues and propose relevant solutions.

## About Synamedia

Synamedia has an end-to-end solution for Multicast ABR delivery, including low-latency ABR support adding minimal latency overhead. They can provide an end-to-end solution to deliver live content over OTT while maintaining the same quality of experience as broadcast delivery and at the same time adding new revenue-generating features.

For more information about Synamedia's ABR streaming solution, contact us or check out our Video Network solutions.

**Author:** Samie Beheydt, Principal Engineer, Synamedia

Synamedia

Global Headquarters
Synamedia
One London Road
Staines, United Kingdom TW18 4EX

Visit us online at synamedia.com