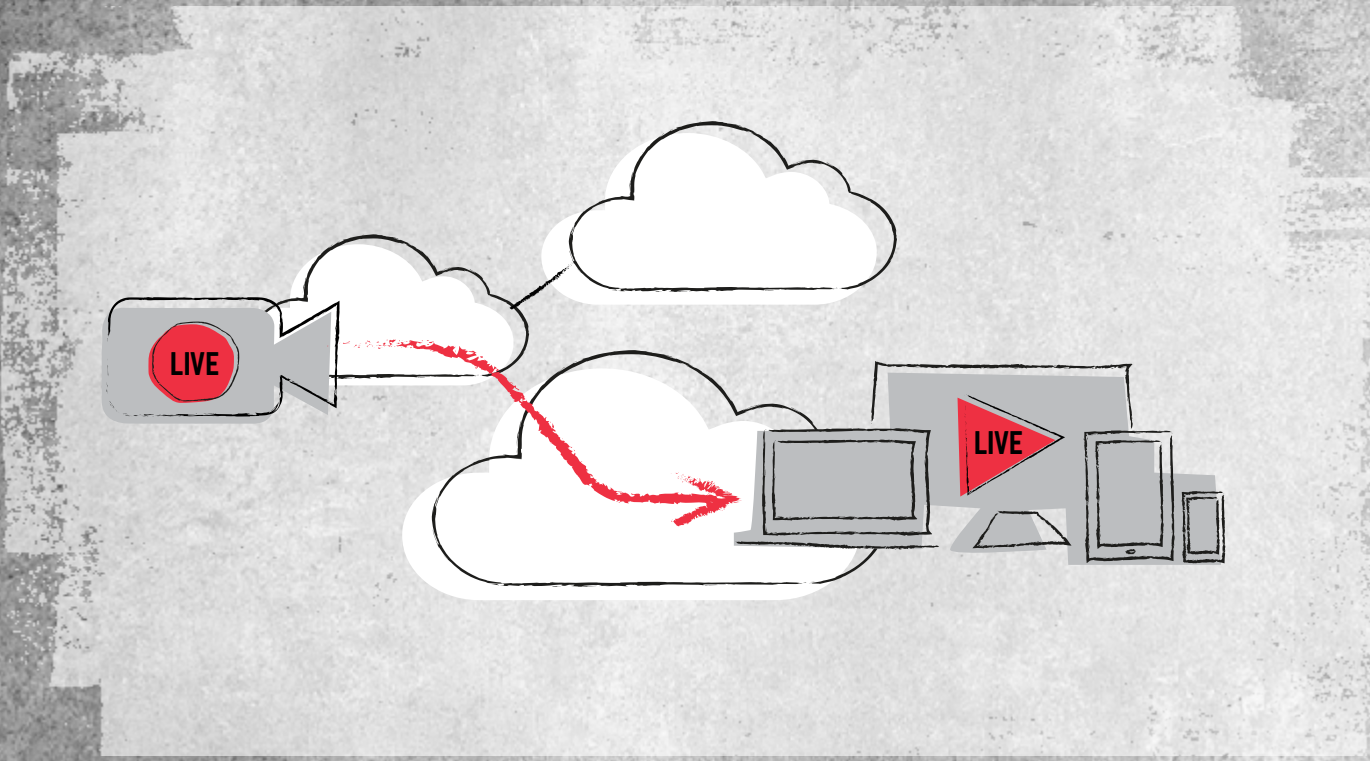
WHITEPAPER

edgeware



Technical whitepaper

OVERCOMING THE CHALLENGES OF HTTP ABR STREAMING

TABLE OF CONTENTS

Introduction – Using HTTP based adaptive bitrate (ABR) technology is quickly becoming a real alternative for delivering premium live events, such as sports and breaking news. These streams are no longer just an alternative option viewed on secondary screens, they are by many viewers becoming the main and preferred option. This introduces new challenges, both with the delivery of the actual streams, but also when comparing the viewing experience with traditional delivery services, such as satellite, terrestrial and cable.

CHALLENGES:

The challenges fall into two main categories: Bandwidth and Latency.

HTTP ABR is by nature a unicast technology, i.e. each viewer receives a unique stream. As a result, the bandwidth in the networks will increase with each additional viewer, resulting in a bandwidth explosion if not properly managed. This is in contrast with traditional broadcast methods, where the same signal is sent to all viewers and the network bandwidth is not affected by the number of viewers.

The HTTP ABR technology also introduces new sources of latency and, if not managed properly, an additional latency of 30-60 seconds can easily be introduced, compared with traditional delivery methods. Such high latencies make any kind of interactivity difficult, for example betting and voting. It is also difficult to have huge differences in latency between different delivery paths, the "hearing the neighbours cheer before you see the goal on your screen" problem. Or, the more modern version of receiving a notification from a friend through social media before the event has appeared on the screen. Reducing latency to a low value is therefore crucial for a successful HTTP ABR live service.

The following sections will digest these two issues in more detail and discuss alternatives for addressing them.

BANDWIDTH

When the first video services over IP were introduced, the natural choice for live TV was to use multicast technology. Multicast maps well to the broadcast model, even though it's not one signal sent to all users. The network replicates the signal at different connection points to reach all users, resulting in a very bandwidth efficient delivery. It is, however, non-trivial to deploy and operate a multicast network. It requires a fully managed network and is only practical within a single operator's domain. Being based on UDP means that resiliency needs to be solved by some other mechanism, such as unicast retransmission of lost packets or by using forward error correction techniques.

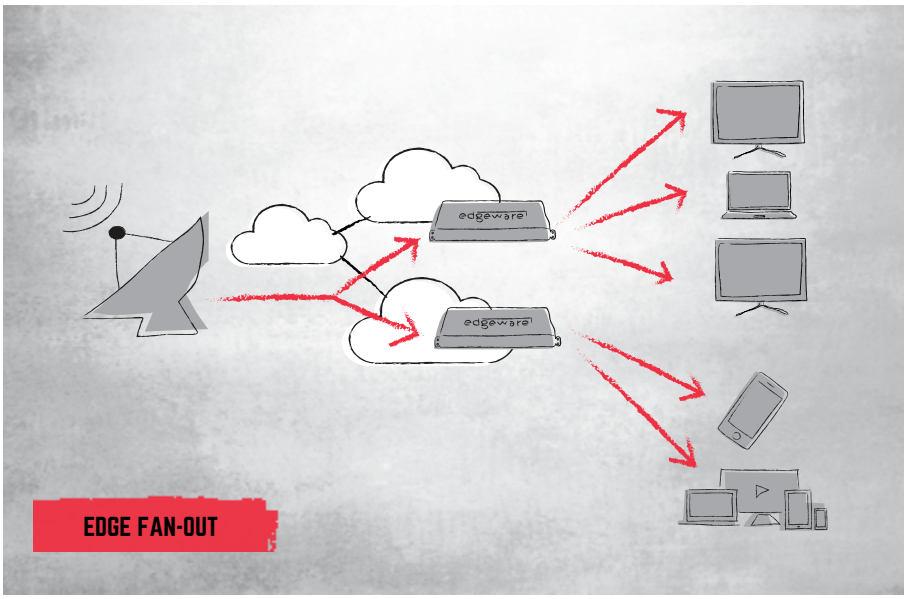
So what are the options for managing the bandwidth explosion of an HTTP ABR service?

CONTENT DELIVERY NETWORK (CDN)

The primary driver for introducing HTTP ABR technologies was to be able to use standard web technologies for the delivery. So, a CDN should be the natural choice for live streams as well? It could very well be, but there are a number of things to consider. A traditional CDN is designed to reduce the load on a central source (origin) when delivering files to a large number of clients, where clients are requesting the files over extended periods of time. This problem is solved by introducing a hierarchy of caching servers, where the most popular files are hosted in caches closer to the clients and less popular files are hosted on a smaller number of more centralized caches.

A CDN will in many cases be the best option also for delivering live streams, since it's a natural fit to the HTTP ABR technology. Live streaming is a bit special though, and special care has to be taken when designing/selecting the CDN. With live streams, many users will request the same content at the same time. Therefore, we can't benefit from any smearing over time, instead we get a peak in the traffic. Managing this peak will require the edge servers to be closer to the end users, compared with traditional file/on-demand video delivery. Most CDN service providers have their edge servers at ISP peering points, making the ISP network a potential bottleneck. Distributing edge servers into the ISP network solves this problem and the closer to the end users the servers are located the less impact on the network and the higher QoE for the end user.

A CDN can be a good choice for handling the bandwidth issue, if properly deployed. There are also latency issues with CDNs that need to be addressed, which will be discussed in more detail below.



Edge Fan-out: *Managing traffic peaks will require the edge servers to be closer to the end users.*

MULTICAST

So why not just continue using multicast for delivering these live streams? There are a several reasons why multicast is not an obvious choice for HTTP ABR delivery:

Multicast is a push technology (push the same content to all users), while HTTP ABR is based on clients pulling what they want (unicast). If multicast is used, it needs to be terminated before the content hits the client. This could be in the network (edge servers), in a residential gateway or even in a 'pre-client' running on the same device as the ABR client, even though the latter is less likely.

Multicast is designed to deliver continuous streams of data, while HTTP ABR is based on delivering sequences of files (segments). This can be managed either by converting streams to segments in the multicast termination point, or by using protocols like FLUTE, PGM or ROUTE to deliver files over multicast.

Multicast only works within an operator/ISP domain, i.e. it cannot be used for OTT services.

Multicast can be used to manage up-stream bandwidth also for HTTP ABR live streams, but what is the value added? Since multicast and HTTP are conflicting technologies (push vs pull) we need to convert from one to the other at some point. Terminating in the client is not realistic. Apart from lack of support in standard clients, multicast of streaming media does not work well over WiFi networks.

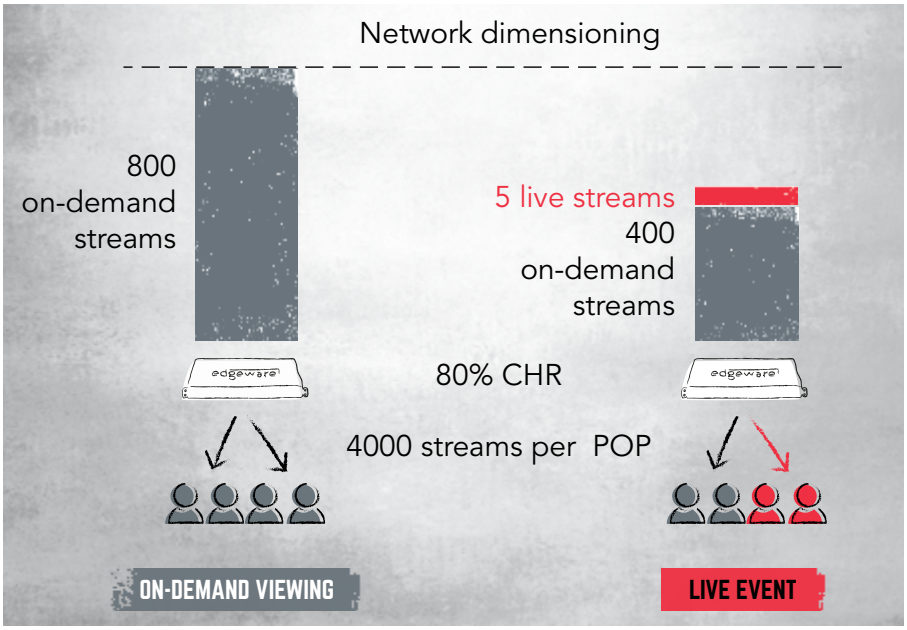
Terminating multicast in a residential gateway is beneficial if there are bottlenecks in the access network shared by many users, such as might be the case in a cable network, or if the edge servers cannot be located close enough to the end users. However, in telco networks the last mile is typically dedicated to each user, i.e. the traffic will be unicast anyway and nothing will be gained by using multicast over these links. It's rather the opposite, since all variants needs to be pushed, the bandwidth load on the last mile will be higher than if the client just pulls down the variant that is actually being used. Terminating multicast in the residential gateway also introduces a non-standard component in each home, removing the benefit of using standard web technology for the delivery.

Therefore, in most cases, the optimal termination point would be the edge servers in the CDN. Using multicast to deliver live ABR content to edge servers will reduce the up-stream bandwidth and also bypass any hierarchical caching layers, by pushing the content straight to the edge servers. So, are these benefits enough to motivate the introduction of another, rather complex, technology into the delivery chain? It depends.

A high capacity edge server only needs to receive one stream per channel and can then deliver this channel to a large number of end users. The bandwidth reduction can be several orders of magnitudes (one stream in, thousands of streams out). The question then is if the remaining up-stream bandwidth is high enough to motivate multicast? With a large number of edge servers (hundreds) it could be easy to draw the conclusion that multicast is useful, since it will reduce the bandwidth for the live streams by a large factor (hundreds). However, this bandwidth needs to be compared with other traffic in the network. Typically a CDN is designed for both on-demand and live streaming. The amount of streams in the core network from the on-demand traffic is determined by the number of users and

cache efficiency in the edge caches, while the amount of streams for the live traffic depends on the number of edge servers. When there are major live events, the on-demand traffic typically decreases. So let's look at an example, in this case a single edge server, to make the example easier to understand:

Example: *Network capacity needs for on-demand viewing vs live events*



This edge server can deliver up to 4000 streams to clients. It has a built-in cache that gives an 80% savings of bandwidth for on-demand streams, i.e. 20% of the traffic needs to come from the up-stream network, or 800 streams at peak load in this example. This is the load that the up-stream (core) network must be dimensioned for, for each edge server. Now, on Saturday night there are 5 football games at the same time. Half of the users will watch one of these live events, while the other half continues watching on-demand content. What happens with the up-stream traffic then? The number of on-demand streams will drop by 50%, to 400, and then we add the 5 live streams. Even though the client traffic is the same, the total number of streams entering the edge server is much lower than before and there is plenty of margin in the network. So even though we could have saved some bandwidth for the live streams by using multicast, there is lots of available capacity in the network, making the benefits less clear. There are two main reasons that could motivate the introduction of multicast for delivering ABR segments: 1) A very large number of edge servers and 2) If there are multiple cache layers used for the on-demand traffic the head-room in the network may not be as big as in the above example, if this hierarchy is bypassed for the live streams.

PEER-TO-PEER (P2P)

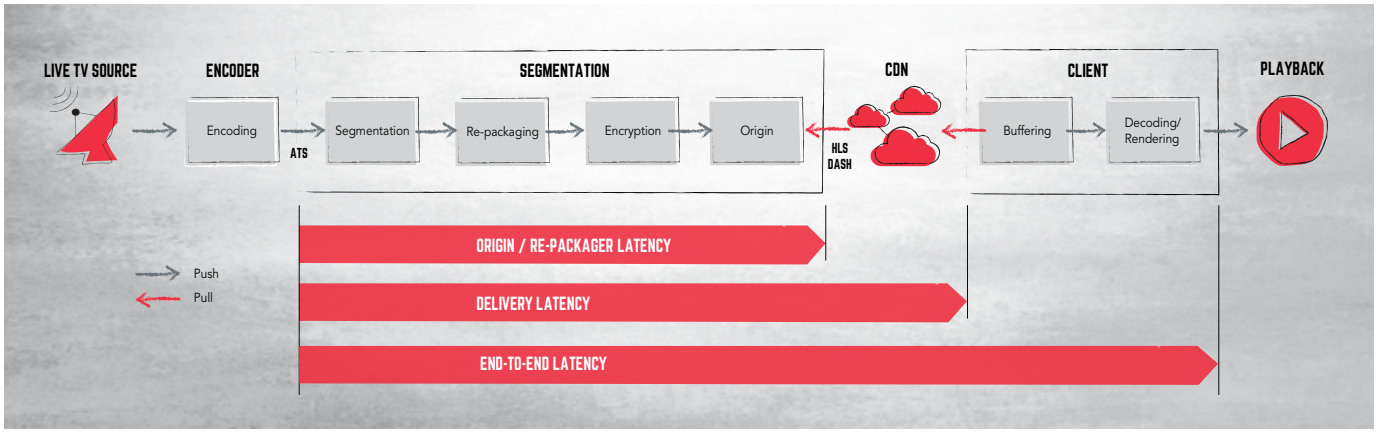
The final option is to use peer-to-peer technology. Here the end users' devices act as sources of content for other users. For a P2P network to be efficient, it requires that many users have access to the same content, something which is fulfilled with live streaming. In order to not end up with too long delivery chains, one typically strives to build trees of clients, where each client forwards the content to a small number of other clients. In this way the latency can be kept reasonably short.

The main attraction with P2P technology is that it doesn't require any network equipment and hence is 'for free' and can be run over any network without deploying anything in advance. Furthermore, today all web browsers come with support for the WebRTC protocol, which can be used to build P2P distribution solutions. Therefore, also the clients can be built with standard web tools, e.g. JavaScript.

The main drawbacks with P2P solutions are the traffic patterns generated and the vulnerability of the distribution. Lots of traffic is now going from the clients, and not to the clients, something the networks may not be designed for. Also, for a P2P network to operate efficiently there must be enough peers in the proximity and the distribution trees are very sensitive to drop outs in the root nodes. Therefore a CDN is in many cases used as a fallback.

LATENCY

In addition to the excessive bandwidth required to deliver HTTP ABR live streams, there is also an issue with the latencies that can be introduced at different points in the distribution chain. If not carefully managed the overall latency can easily be 30-60 seconds, which in many cases is intolerable. We will start by looking at the sources of latency and then discuss how to address them.



ENCODER

As an encoder will be part of any delivery chain, HTTP ABR, traditional broadcast or IPTV, we will use the encoder output as the reference point for latency, to separate the latencies introduced specifically by the HTTP ABR technology. Still, there are a couple of encoder parameters that will have an impact on the ABR latency:

- If a multiplexed format, such as MPEG-2 TS, is used as output from the encoder, the transport skew between audio and video in a single stream will have an impact on the segmentation time, since all data that belongs to one time segment must be received before the segment can be created. For example, if there is a 1 second skew between audio and video it will take an additional second to create the segment. With 4 second segments this means 25% additional segmentation time. With segmentation being one of the main contributors to latency this can be significant.

- The encoder will generate multiple streams for each channel, one per bitrate profile. It is important that these streams are synchronized, since again, the segmenter needs to receive all data from all variants before a segment can be created.

SEGMENTATION (RE-PACKAGING, ENCRYPTION, ORIGIN)

The next step is segmentation, where the continuous stream from the encoder is segmented into short time fragments of equal length, typically in the 2-8s range. A segment cannot be created until the end of the segment is known, i.e. all data that belongs to the segment must be received. Therefore, segmentation introduces at a minimum a latency equal to the segment length.

A couple of other functions are typically combined with the segmenter: re-packaging into a client format, e.g. HLS or DASH, DRM encryption and finally, storing the segments on an origin server for delivery to clients, possibly through a CDN. Even though these steps introduces latency it is typically small compared with the segmentation latency (<1s combined).

NETWORK

There are always latencies when delivering traffic over a network. In this case these latencies can however be ignored, since they are typically orders of magnitude smaller than the other latencies.

CACHING

If a CDN is used there will be caches in the network, possibly a hierarchy of caches that need to be traversed. Many generic web caches are designed to cache small objects used on web pages and will not deliver anything until an object is fully in cache. This type of behaviour is sub-optimal for live video delivery since each cache may introduce up to a segment length of delay. With a hierarchy of caches you can then have multiple segment lengths of delay.

Finally, in the client segments are buffered to cope with variations in network conditions. A typical client may buffer 3 segments, making the client buffer the biggest contributor to latency. This is very different from traditional broadcast, or IPTV, where a rate controlled signal is delivered over a managed network, requiring a much smaller buffer in the client.

In summary, segmentation and client buffering are the main contributors to latency with HTTP ABR delivery.

CLIENT BUFFERING

Before discussing how to reduce latency it is worth mentioning that some of the latencies presented above are introduced for a reason, namely to provide a robust service in a heterogeneous network with sometimes unknown and varying conditions. Therefore, minimizing latency may come at the cost of reduced resiliency.

There are several ways to minimize latencies. The following list presents some suggestions, that can be used on their own or in combination:

- Make sure the encoder settings are optimized for ABR delivery, according to above.
- Segmentation length should be kept as short as possible, since many of the latencies are expressed as a factor of the segment length. Using too short segments may however impact encoding efficiency, since a segment should preferably contain a complete GOP. Also, signalling will increase with shorter segments. Two second segments is usually not a problem, but going lower may require special care.
  The new CMAF format proposed by DASH-IF introduces 'chunks' as a way to split live segments into smaller pieces that can be delivered earlier. This will help reducing the latency due to segmentation, if delivered all the way to the clients.
- Use video optimized caches that can deliver content as soon as the first bytes are received.
- Bypass intermediate caches, live streams should go straight to the edge caches, if possible.
- Ensure there is enough bandwidth in the network to allow edge servers to download new segments at over-speed (only works with pull delivery)
- Tune the clients to use as small buffers as possible. Needs to be balanced with resilience to varying network conditions.

HOW TO MINIMIZE LATENCY?

There are many things that can be done to reduce latency, but what is 'good enough', i.e. how much do we need to reduce the latency? In some cases, it is the variation in latency between different streams that are critical, not the absolute latency. Multi-camera viewing is an example and here sub-second differences are required, even though the final synchronization of streams can be done in the client. In other cases, it is the differences versus other delivery mechanisms, e.g. broadcast vs OTT streaming, that are important. Here it is often enough if the difference in latency is within the time it takes to do some action, like sending a message to a friend. Less than 5 seconds difference seems to be acceptable by many users. Note that the delay for traditional broadcast is 5-10s, so the HTTP ABR latency doesn't have to be less than 5s but rather in the 10-15s range. Finally, for betting applications it is the delay versus the real event that matters, and every second counts (potentially as lost money!).

SUMMARY

HTTP ABR is an exciting technology that can be used to deliver premium live video content to a large number of different device types and networks. It does, however, introduce some new challenges in terms of bandwidth utilization and latency. In this paper we have looked at these challenges and how to address them. The bandwidth issue has traditionally been managed by using IP multicast, which is still an option, but it's no longer a clear choice. Latency comes from many sources and different techniques need to be used to reduce each of them.

The good news is that if your video distribution network is properly designed it can cost-effectively deliver premium live events to large audiences using the same HTTP ABR technology used for on-demand streams.

WHY EDGEWARE?

WHAT DO WE DO?

With Edgeware, you can build a TV CDN that delivers an amazing viewing experience, scales cost-effectively and gives you a unique insight into how your viewers are experiencing your content. It does this using a unique architecture that optimizes the different functions required to deliver TV in different ways.

WHY DOES A TV CDN MATTER?

TV already dominates peak-hour network traffic – and as the take-up of new TV services increases, networks are becoming even more expensive to scale, new services are hard to add and content owners have lost control over the quality of their viewers' experience.

General-purpose CDNs were developed to help distribute large amounts of frequently used content – such as PC software updates – but they were not optimized for TV. They may deliver content via lowest-cost peering points or from wherever they have spare storage capacity – and they are not built for real-time service delivery.

WHAT MAKES AN EDGEWARE TV CDN DIFFERENT?

A TV CDN is purpose-built to deliver a rich set of TV services, to scale cost-effectively and to give you a unique insight into your viewers' behavior. If TV matters to you, then a purpose-built TV CDN will deliver a far better experience for your viewers – at lower costs – than a conventional CDN. A TV CDN gives:

- **An amazing viewing experience**

With low-latency, guaranteed resources and no buffering, a TV CDN can deliver services such as Live-TV, Time-shift TV, Ad Insertion, TV Anywhere, Cloud DVR, Watermarking, Encryption and Fast Channel Change – all from one shared platform.

- **Insight and control**

A TV CDN gives you a unique insight into viewers' behavior, because it can correlate viewing analytics with network performance. So now you can find out what is really happening to your content delivery.

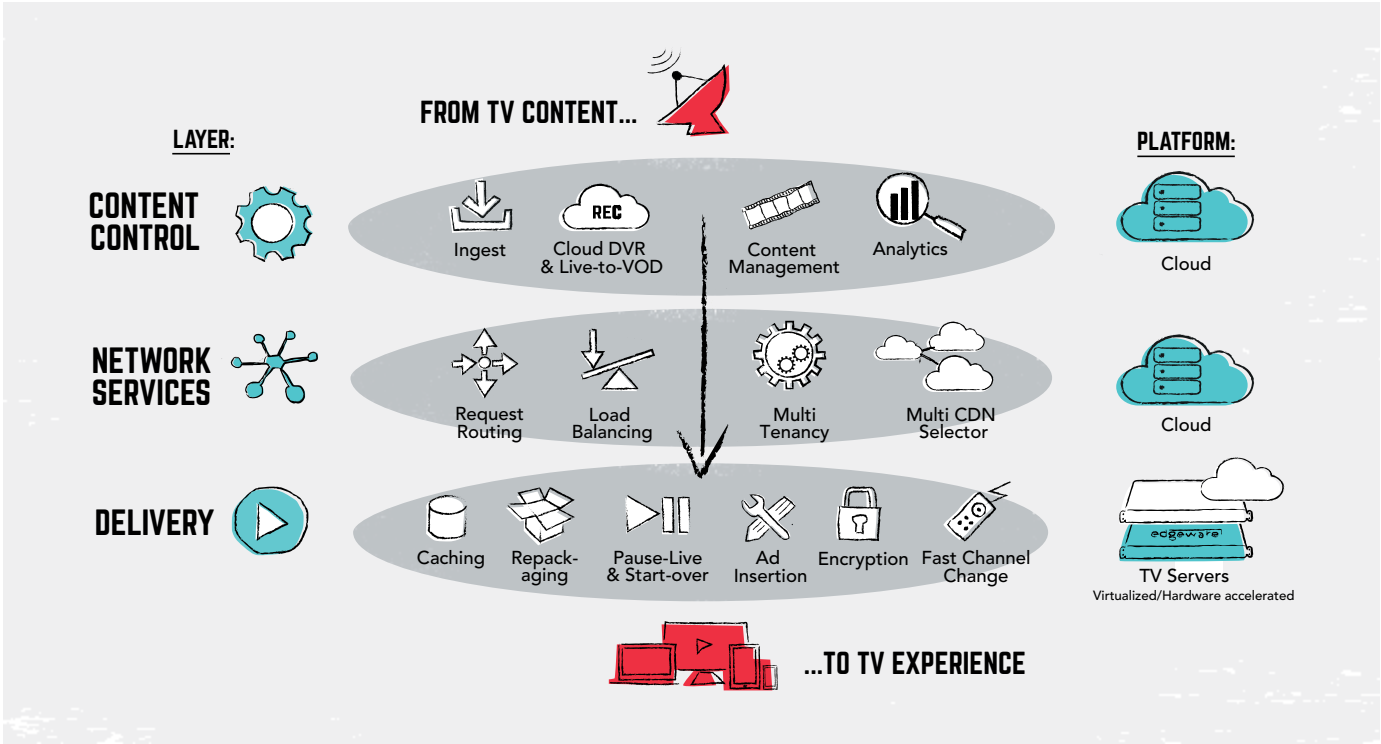
- **The cost-effective way to scale**

A TV CDN can scale to huge levels of service take-up – without huge levels of cost. That's because it uses a unique architecture to scale different functions in different ways.

EDGEWARE'S TV CDN ARCHITECTURE

Functions that are content related – such as Ingress, Content Management and creating Live-to-VoD and Catch-up TV content – need to scale to support large volumes of TV assets. Edgeware delivers these functions in software, with its Content Control Layer. They are usually centralized because they are common assets used to create the final viewing stream.

For small-scale service take-up, or initial roll-outs, this software layer can also deliver TV streams directly to viewers. But to scale to high volumes, the delivery architecture must be distributed.



Network services connect the content control layer to a distributed edge. These include request routing, load balancing and support for multi-tenancy. These functions need to scale to very large volumes of viewing requests – characterized by many small transactions – and are delivered with a software based routing layer, that also runs on standard IT or cloud infrastructure. This layer uses a TCP stack optimized for TV, that can correlate viewing analytics to network behavior.

To create a personalized TV stream to huge numbers of viewers, Edgeware's Delivery Layer uses purpose-built TV Edge Servers. The TV Edge Servers perform functions such as streaming, caching, repackaging, pause-live TV, ad insertion, encryption and watermarking.

They use hardware acceleration to outperform conventional servers, so they aren't limited by general-purpose computing architectures that suffer from CPU bottlenecks between storage and network delivery. This enables a unique deterministic experience for the viewer – so that you can deliver TV without buffering and with time-delays of a fraction of general-purpose systems.

A major incumbent European operator recently calculated that adopting Edgeware's TV CDN architecture had nearly halved the capital costs of its network as well as saving them 75% in power and 90% of their rack space – compared to delivering TV from a conventional architecture.

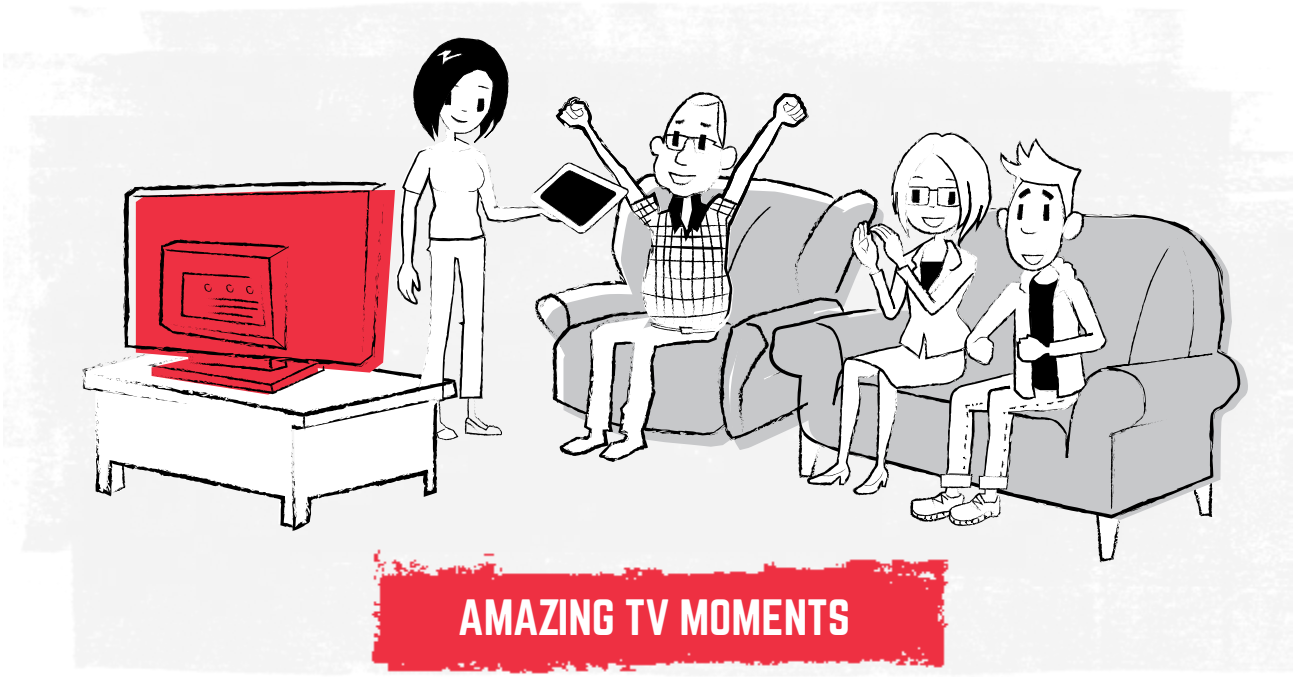
TIME FOR A TV CDN

For many years, networks were optimized for voice. Then they needed to be optimized for data. Now the time has come to optimize networks for TV. Fortunately, you don't have to start building from scratch – you can simply overlay a TV CDN. You can even build out one layer at a time as your service take-up grows.

Edgeware has already built over 100 TV delivery networks for content owners around the world – more than any other vendor – including network operators (such as KPN, Telia Company, Cincinnati Bell and Vodafone), cable companies (such as Televisa) and content owners and broadcasters (such as TVB in Hong Kong).

In fact, Edgeware TV CDNs already deliver services to more than 20 million viewers around the world, and have the capacity to deliver more than 5 million concurrent HD TV streams.

By optimizing your TV delivery, you can build a TV CDN that scales more cost-effectively than any other architecture, creates a common platform for multiple TV services and gives you unique visibility into how your viewers are experiencing your content. More importantly, your TV CDN will deliver an amazing viewing experience.



AMAZING TV MOMENTS

WWW.EDGEWARE.TV

STOCKHOLM HQ
Edgeware AB, HQ
Mäster Samuelsg. 42
12th Floor
SE-111 57 Stockholm
Sweden
+46 736 126 840
sales@edgeware.tv

USA
Edgeware, Inc.
200 E. 5th Ave., Ste. 125
Naperville, IL 60563
USA
Toll Free Phone:
+1 888 324-1970
sales_americas@edgeware.tv

MEXICO
Edgeware
Miguel de Cervantes
Saavedra 193-802
Col. Granada
Del. Miguel Hidalgo
Mexico D.F, C.P. 11520
sales_americas@edgeware.tv

HONG KONG
Edgeware
Room 2503, 25/F,
BEA Harbour View Centre
56 Gloucester Road
Wanchai, Hong Kong
Phone: +(852) 3184 0660
sales@edgeware.tv