INCREASING AD PERSONALIZATION WITH SERVER-SIDE AD INSERTION

Lionel Bringuier

Elemental Technologies, USA

ABSTRACT

Advertising is an integral part of premium broadcasters' and video content providers' monetization strategies. To optimize revenue opportunities, this advertising needs to be personalized accurately for the user and delivered so that it does not impair the quality of a viewer's experience.

The industry is moving towards server-side advertising insertion (SSAI). With SSAI, a single uninterrupted stream containing both program and commercial content is delivered at a consistent quality, and commercials are personalized for each individual stream at the moment of delivery.

This paper examines the architectures required to achieve SSAI at scale so that thousands or even millions of concurrent, individually-tailored advertising manifests can be delivered in a timely fashion, even for live streamed events. To achieve this scale, cloud and cloud-assisted software solutions are required. This paper assesses the effectiveness of this approach and the long-term practicalities of delivering targeted and secure dynamic advertising with high quality of experience.

THE CHALLENGE

There is no question that consumers have come to expect content anywhere, any time and on any device, whether fixed or mobile, on demand.

That expectation does not waiver when it comes to live content – particularly sports. If the consumer cannot watch live events on a large television at home, then the same content should be available wherever the consumer happens to be, preferably with the same functionality including pause and rewind.

To put this in perspective, an estimated 1.6 billion people worldwide watch online video on connected devices (1). 61% of these global consumers watch television and video on smartphones (2). That number rose to 71% between 2012 and 2015, and is still rising. Video traffic to mobile devices is set to grow at 55% a year through 2020 (3).

Today's content delivery industry includes relative OTT newcomers such as Amazon, Hulu, and Netflix and traditional broadcasters such as BBC, NHK and CBS, and specialist content companies such as UEFA, which provides comprehensive Europe-wide football coverage. These organizations all share two key issues that drive operational and technical planning.

First, these companies want to ensure a consistently high quality of experience for their subscribers. Although video delivery may be IP over broadband, the expectation is that this will be akin to broadcast television which more or less "just works," provides a consistent level of image quality and has very few disturbances today. This is in contrast to the early

days of the internet which relied heavily on text and a handful of still images. Today's expectations are video-centric with increasingly intense demands for bandwidth to support this.

Second, content providers need to earn revenue from the internet, an idea which runs counter to consumer expectations that the internet is "free." In the earliest days of the internet, content was provided by hobbyists and people looking to build brands. Now, content has to generate a commercial return. This is further complicated by the growing complexity and respective bandwidth requirements of the video era.

Today, the costs of delivery are significant: US$0.01 a gigabyte just for the content delivery network (4), not including processing and storage costs. Pay TV operators and content owners are accustomed to traditional linear TV monetization models. But the internet is an unmanaged network with unclear monetization models. It is also not possible to apply multicasting approaches; over the internet and with the current IP architectures, it is necessary to use "unicast" (one-to-one) delivery and not multicast (one-to-many). With each piece of internet-delivered video a unique point-to-point connection, delivering video at scale means significant costs. Commercial operations must be able to cover these costs and make a margin.

The most common source of revenues for OTT services is spot advertising. In traditional linear TV models, advertisers leverage broad demographics and geographies to target commercials for the right country or region. With the ability to more precisely identify the Internet subscriber to a video service through a mix of information gathering and cross-site tracking techniques, there are many ways to achieve a much more personalized advertisement strategy. With traditional linear TV advertising strategies falling short, the optimal approach to increasing OTT delivery monetization is to get ads tailored at the individual level, based on previous viewing histories and other "big data."

It is important to note that traditional methods of personalizing advertising fail the quality test because they depend on content coming from one source and advertising from another. In addition, combining the two when viewed on a consumer's device empowers that user to install clever ad-blocker software that can identify advertising content and skip it entirely.

Finally, video streaming and OTT business model depends upon "impressions" – specific consumer views reported from the client to the player. If there is no way to acknowledge that an advertisement has been played on a specific device, then no money is due from the advertiser to the service provider.

CLIENT-SIDE ADVERTISING INSERTION

Earliest attempts at online video advertising had the commercial spot "burnt in" to the video asset. This so-called static advertising did not take demographics into consideration and was always attached to specific content long after an on-air campaign had ended. Advertisers saw no additional benefit and were reluctant to pay fees over and above the on-air slot charges.

As a result, online services moved to a client-side model which inserted specific, relevant advertising into the video stream at the point of playback. This ensured that the commercials were appropriately targeted and timely, because the commercial could be changed for each viewing.

This approach offers two main benefits. The first is that the CPM (cost per thousand impressions) is higher than with static advertisements because the demographics are better identified. But that is not all: end users find a personalized advertisement less intrusive because it offers something more in line with their interests. A luxury car fan will be much

more interested in an ad from BMW or Mercedes-Benz than a generic soda or grocery store spot.

To implement this, service providers moved to a manifest-driven means of delivering content. When a consumer makes a request to view a particular video, the response from the service provider is to send a manifest – an XML or plain text file that lists the elements which make up the requested video – along with the IP addresses from which each can be obtained.

As the video plays, the player makes the necessary calls to obtain the next chunk of content to be shown from the manifest. In a well-designed player, where the segments come from the same source, there should be a seamless display and a reasonable quality of experience.

If advertising is to be included, then the manifest includes tags to mark calls to advertising servers. The player then stitches ads from external ad servers into the relevant portion of the manifest for the individual viewer.

The established standard for this is VAST (video ad serving template), published by the Interactive Advertising Bureau (IAB) (5). The fundamental architecture of client-side advertising provision is shown in figure 1.

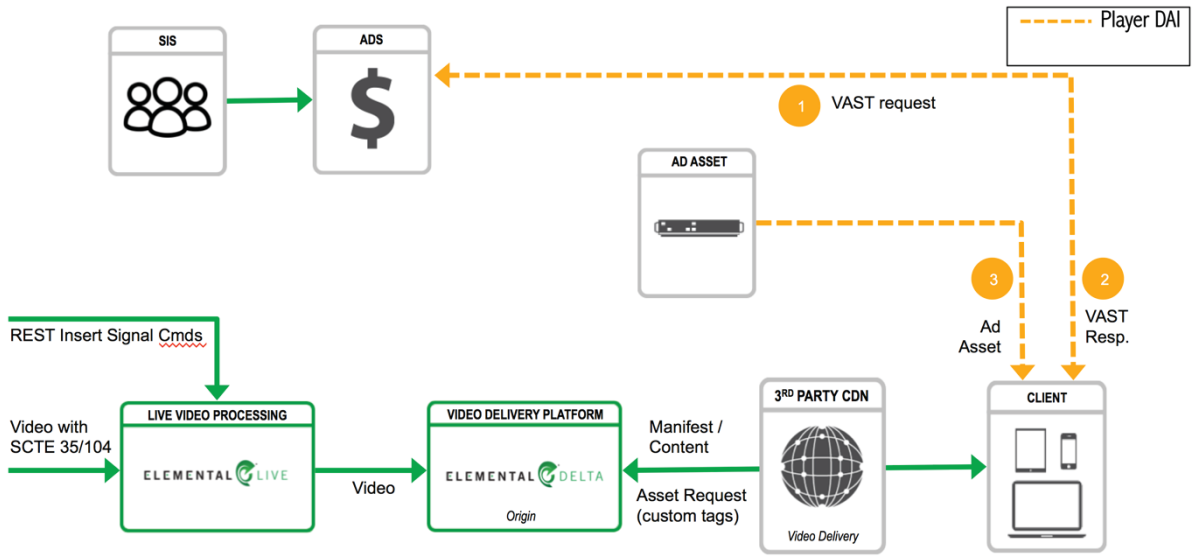

Figure 1 – client-side advertising infrastructure

The advertising servers may not be using the same compression codecs – or even the same aspect ratio – as the content servers, which can result in a visible shift in quality between program and commercial. There may be differences in encryption or no encryption of commercial messages. A change of codec and/or encryption may delay content display (both when switching from program to commercial and back again), presenting the consumer with a freeze or a spinning wheel.

That is a frustrating drop in quality of experience for video-on-demand consumers, and can be catastrophic when watching live streaming such as sports. Consumers will inevitably feel they have missed key parts of the action while their computers or phones buffer on a return from a commercial break.

Explicitly telling the receiving device which parts of the manifest are advertising – through the IP addresses for those calls – also makes it easy to skip the advertising, manually or

through ad-blocking software. Content owners attempt to circumvent this for mobile device delivery by offering custom players in the form of native apps that block ad-blockers. But, the trend is for this content to be viewed via the mobile web rather than through dedicated apps. In the US, mobile web audiences are 2.5 times larger than app-viewing audiences, and mobile web audiences are growing twice as fast (6).

On the other hand, client-side advertising insertion – adding commercials in the consumer's device – has one significant advantage. It provides very accurate metrics for the advertiser, because the individual consumer in effect calls for the specific advertising. It is very simple to see exactly who has watched each spot, and exactly how many people have viewed it, making revenue attribution clear and precise. Having said that, there is always a risk of clever software spoofing the advertising server by claiming the commercial has been viewed when it has actually been skipped.

What is required, then, is a method of inserting targeted advertising into individual delivery paths that provides clear metrics, protects against advertising blocking or skipping, and maintains a consistent quality of experience for the consumer.

The solution lies in upstream insertion of the advertising – before encoding or "server-side" – so that a continuous stream arrives at the consumer device eliminating any possibility of discrimination between content and commercials, and avoiding freezes, black screens and spinning wheels.

SERVER-SIDE ADVERTISING INSERTION

Server-side advertising insertion (SSAI) is, in a way, a return to the origins of online advertising, because the material – both content and commercials – arrives as a single stream from a single source. The difference is that the advertising is no longer static, and is dynamically inserted before encoding.

SSAI requires an architecture that supports just-in-time packaging of content. Rather than content being pre-packaged for each delivery format, just-in-time packaging takes the native content from both the program store and the advertising servers, and adapts it on the fly at the point of request.

In this architecture (see figure 2), the manifest remains with the service provider. When a consumer requests a particular video stream, the program is requested from the server. It arrives complete with markers for commercial insertion, usually in the form of SCTE 35 (for compressed video) or SCTE 104 (for SDI video) standard cues.

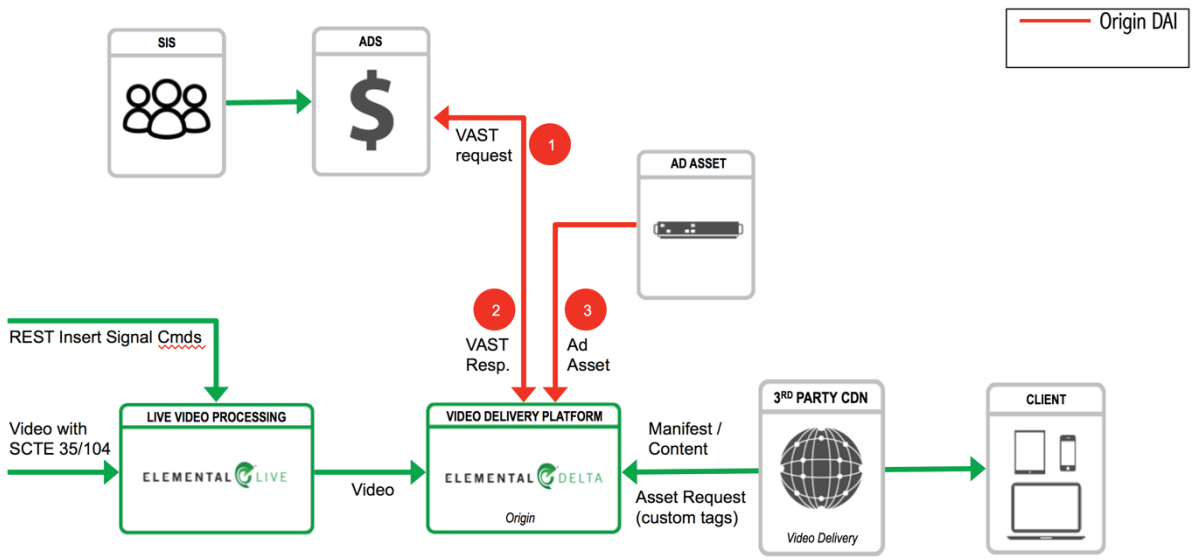

Figure 2 – server-side advertising infrastructure

In contrast to CSAI in which the receiving device makes calls to the advertising servers and provides the response for the impression count, in the SSAI model, the video delivery platform makes VAST calls with end-user metrics to the advertising servers and receives the appropriate commercials. These are cut into the video upstream of encoding or transcoding.

This architecture allows for the dynamic insertion of advertising into live streams, and even into sports broadcasts when it is not known in advance how long a program break will be. By using the same SCTE 104 or SCTE 35 cues that broadcast playout systems use, a server-side advertising insertion platform can perfectly fill the break on the fly.

The entire video sequence – program and commercials – is now packaged as a single stream and delivered to the consumer in a consistent codec and quality, and with no markers to delimit content and advertising. The resulting stream makes it hard to detect or eliminate the advertising, and provides a uniform quality of experience.

This stream also simplifies the requirements of the player at the target device, making for lighter-weight apps or reducing demands on web players. The growing assumption is that web clients will use HTML5 to call and display video, as devices converge to OTT and internet-based delivery formats. That means that a service can be deployed faster, because there is no need to develop a player for every platform that will be targeted. It also is a more future-proof approach, as it mitigates the need to develop new players whenever there is a new OTT device launch or a significant update in existing devices' operating systems.

In this approach, it becomes the responsibility of the service provider to offer the reporting and metrics on delivered advertising. But the architecture does support highly-targeted advertising, even down to the individual viewer if the advertising campaign software has that level of granular data. Targeting to the individual is seen as the holy grail of advertising, and is likely to command a significant premium.

STANDARDS

One of the most significant challenges in delivering online video and ads is the large number of combinations of screen resolution, codec, operating system in target device and streaming type available. Apple iOS is relatively well managed but it is far from universal that users upgrade to the latest release. Android, because of its complex ecosystem of hardware, software and release management, results in many more options: there are at least four different Android operating systems that currently have more than 10% penetration.

The complexities of mobile devices cannot be ignored because, according to McKinsey, mobile will become the principal digital platform. Mobile broadband penetration is set to overtake fixed broadband, reaching 58.3% of the total penetration by 2019 (7).

Beyond these mobile issues, there are four major television streaming devices: Amazon Fire, Apple TV, Google Chromecast and Roku, each with significant market share. There are also connected TVs, and each manufacturer has different ways of handling OTT content.

An attempt to provide dedicated services for each platform would quickly collapse under sheer complexity. While many solutions have been proposed, MPEG-DASH is increasingly seen as the best.

Through applications such as Google's Shaka Player or the DASH-IF JS player, MPEG-DASH allows a single format to be displayed on any device. It provides support from mobile

phones to hybrid broadcast/broadband television (HbbTV). It is a native ABR (adaptive bitrate) system, ensuring that the best quality is always achieved for a given bandwidth.

MPEG-DASH is codec-agnostic. HEVC is fully compatible with MPEG-DASH, so high performance compression can be used to maximize quality, or to use a common platform to deliver 4K Ultra HD. In 2014, IBC honored the Vienna State Opera with a special award for its 4K transmissions to the home using HEVC encoding and MPEG-DASH delivery.

It also hosts common encryption (CENC), an emerging standard which uses the encrypted media extensions (EME) defined in HTML5 to provide the DRM decryption in the player. This in turn allows a common encrypted file to be delivered by MPEG-DASH, whatever the DRM system used in the player.

However, MPEG-DASH is not the only solution currently available. For Apple hardware, and a number of other very popular devices, the rule is to use Apple's HLS, or HTTP Live Streaming protocol for ABR streaming.

HLS is a very mature technology and implemented in a wide range of manufacturers' devices beyond Apple, even though that in and of itself is a major market. There are third-party players available, such as THEOplayer from OpenTelly or Flowplayer, which do not require any plug-in for HLS playback, making them widely applicable.

On the other hand, HLS is a proprietary technology, developed and owned by Apple. It has not changed much since its launch ten years ago and is very stable or getting outdated, depending on your point of view. As a proprietary standard, it carries risk of obsolescence.

Contrast that with MPEG-DASH, which is developed by the very long-standing and independent MPEG forum. The MPEG forum continues to develop the standard and will undoubtedly develop new MPEG formats in future. The ubiquity of MPEG formats will almost certainly see DASH support in all browsers, most mobile devices, set-top boxes and smart TVs.

MPEG-DASH can deliver substantially lower end-to-end latency compared to HLS. It also uses a templated manifest that can be cached at the edge, while an HLS manifest is updated routinely and needs to be propagated several times a minute.

The one significant philosophical difference between the two is that MPEG-DASH is open to a range of encryption solutions, whereas HLS only supports its own DRM. To provide heavy duty content protection in HLS means wrapping the streams in another technology that can be interpreted by the receiving device. CENC common encryption technology is the usual solution to this issue.

HLS and MPEG-DASH have a lot in common, though, in the philosophy of how content is presented through manifests or playlists, and how video content is sliced into small chunks. The architecture and the underlying protocols are what allow server-side advertising insertion without recourse to complex proprietary extensions, and therefore allow users to build open infrastructures while retaining the benefits of advanced capabilities.

Finally, to be fully exhaustive, there are other ABR protocols in the field, like Smooth Streaming from Microsoft and HDS (HTTP Dynamic Streaming) from Adobe. These two streaming formats have less flexibility in the way manifests are generated, and while they can also be used for server-side ad insertion with static ad replacement, it is much more complex to deal with personalized server-side ad insertion.

HLS EXAMPLE

For this paper, HLS examples are used because the format is currently more widely deployed and better understood than MPEG-DASH.

Take a typical piece of content, such as a live HLS channel workflow that is streamed with the following format in the manifest:

```
#EXTINF:10,
http://customer.cdn.com/segment-0013.ts
#EXT-X-CUE-OUT:30
#EXTINF:10,
http://customer.cdn.com/segment-0014.ts
#EXTINF:10,
http://customer.cdn.com/segment-0015.ts
#EXTINF:10,
http://customer.cdn.com/segment-0016.ts
#EXT-X-CUE-IN
#EXTINF:10,
http://customer.cdn.com/segment-0017.ts
```

The "#EXT-X-CUE-OUT" manifest declaration is the signal to the player that this is a commercial break. When using CSAI, the player will have some sort of logic to initiate a VAST call to get the advertisements. The player will probably download ads to play within the time between the CUE markers. The player also reports back to the ad service that one or more impressions has occurred, with typically a report when 25%, 50%, 75% and 100% of the ads have been watched (quartile beaconing).

In turn, the commercial manifest could look like:

```
#EXTINF:10,
http://customer.cdn.com/segment-0013.ts
#EXT-X-CUE-OUT:30
#EXTINF:10,
http://adserviceprovider.cdn.com/ad-45/seg-01.ts
#EXTINF:10,
http://adserviceprovider.cdn.com/ad-45/seg-02.ts
#EXTINF:10,
http://adserviceprovider.cdn.com/ad-45/seg-03.ts
#EXT-X-CUE-IN
#EXTINF:10,
http://customer.cdn.com/segment-0017.ts
```

It is clear from this part of the manifest that some calls are not to the content provider but to advertising servers.

By moving the advertising insertion to the server side and within the packaging process, the http://customer.cnd.com and http://adserviceprovider.cdn.com differences are eliminated and made to look exactly the same from a manifest perspective. There is no need to put the #EXT-X-CUE tags anymore as the ad insertion or replacement is already done.

Finally, the differences between the “segment-XXXX.ts” and “ad-45/seg-YY.ts” segments can also be eliminated and the regular content and the ad video chunks all made to look the same, thus obfuscating the names. The resulting manifest would then be something like:

#EXTINF:10,

http://customer.cdn.com/out/i/695_1_13.ts

#EXTINF:10,

http://customer.cdn.com/out/i/695_1_14.ts

#EXTINF:10,

http://customer.cdn.com/out/i/695_1_15.ts

#EXTINF:10,

http://customer.cdn.com/out/i/695_1_16.ts

#EXTINF:10,

http://customer.cdn.com/out/i/695_1_17.ts

There is no way anyone can tell that 695_1_13.ts is a chunk of regular content while 695_1_14 is an obfuscated ad. As well as eliminating the prospect of skipping the advertising, a consistent stream of content is also presented in the same resolution, codec and encryption, which maintains the quality of experience.

The final question is: If there is no way to tell what is content and what is an advertisement, how can the players acknowledge that the advertisements have been played back for accounting purposes? That is where vendors can offer creative solutions to demonstrate that, because the spots were embedded in a continuous stream, they have been delivered to the viewer. Elemental has a solution for this, but it is beyond the scope of this paper.

SCALABILITY

The advantage of client-side advertising is that program content can be encoded and packaged, in all the different formats, in advance. When a request comes from a consumer, all that happens is that the appropriate file is played out from a server.

Server-side advertising depends fundamentally upon just-in-time packaging; the content is stored in some suitable high-quality delivery format, and packaged for delivery at the time it is requested. This has other benefits for the operator, including reduction of storage and bandwidth costs by delivering content best-suited to network conditions, and the target device, at the moment of delivery.

To cope with fluctuations in demand for just-in-time server-side ad insertion, a highly scalable architecture is required. Demand will inherently vary throughout the day, but for broadcasters there will inevitably be sharp peaks: a breaking news story, or the return of a highly popular television series.

Sporting events can create enormous demand. During the 2014 FIFA World Cup in Brazil, as many as three million consumers a day worldwide used online video services provided by the host broadcaster. Twenty four million unique users watched tens of millions of hours of content online during the tournament. The peak streaming rate – during the Argentina Netherlands game – was 6.9 terabytes per second.

The solution lies in software encoding and packaging that can be virtualized for rapid deployment, and put in a public cloud infrastructure for immediate auto-scaling. Dedicated single path hardware encoders and packagers lack flexibility. The only practical solution is to spin up instances of software-defined video processing as they are needed.

Further, for dynamic server-side advertising insertion, the best place to prepare the content stream is at the edge, as close as possible to the consumer. This implies a cloud service. In the 2014 World Cup example quoted above, the encoding and packaging used Elemental software hosted by Amazon Web Services. The cloud is required to create millions of

individually tailored manifests of content and advertising, even for live streamed events.

That in turn demands a flexible architecture to manage requests and determine where best to create the output stream. The manifest must be created and manipulated according to the rules and demands of the advertising management provider. Instructions must be delivered to the just-in-time encoding and packaging software, wherever it is. Finally a single contiguous stream is sent to the client.

CONCLUSION

Online and mobile consumption of video content continues to rise inexorably, with the inevitable conclusion that advertising spend is moving towards new platforms.

Broadcasters and content owners want to deliver a premium experience to online consumers. They also need to protect their monetization.

Client-side advertising can and does deliver monetization, but at the cost of quality of experience and at the risk of skipping commercials.

Server-side advertising insertion achieves both goals. By delivering a single stream that contains both programs and commercials in the same resolution, aspect ratio, codec and encryption, it avoids buffering, freezes, blocking, spinning wheels and other interruptions. It is especially vital when streaming live content to deliver a consistent quality of experience so audiences do not feel that they have missed something vital.

Through the use of individual ID and carefully targeted campaigns, it is now possible to get the right commercials to the right individuals. Because commercials are dynamically inserted into each stream at the point of delivery, server-side advertising insertion opens up the possibilities of real-time bidding for spots, an important benefit of the rapidly-growing practice of programmatic advertising.

Implementation of server-side advertising in a practical and cost-effective manner, across a broad range of platforms and delivery fabrics, demands a software architecture in which manifest manipulation and content encoding and streaming can be performed in real time. This processing needs to scale instantly to meet the demand of concurrent users, which could rise into the millions.

The resultant service is likely to use the cloud to provide the additional, flexible capacity and to reach the edge to minimize content delivery costs.

REFERENCES

1. The OTT Playbook, Parks Associates, June 2015
2. Consumer Lab TV & Media Report, Ericsson, September 2015
3. Mobility Report 2015, Ericsson, November 2015
4. Josh Stinehour, Devoncroft Partners, NAB 2016 conference session
5. www.iab.com
6. The 2015 US Mobile App Report, comScore, September 2015
7. Global Media Report, McKinsey, October 2015