# TECHNICAL PRACTICES FOR A MULTI-CDN DISTRIBUTION STRATEGY

Bram Tullemans

EBU, Switzerland

## ABSTRACT

Ideally data in a Multi-CDN (Content Delivery Network) setup is load balanced dynamically using real-time traffic information gathered throughout the delivery chain. This paper provides the rationale and architectural guidelines for an online Multi-CDN distribution backend, abstracting the content publication and playout logic from the actual delivery networks used. It describes the essential technical practices for a Multi-CDN setup covering the role of the Origin servers and essential broadcast features like geo-fencing, HTTPs secured traffic, cache purging, etc. Essential metrics like video player feedback are detailed and fall back scenarios are explained. All the elements above are deployed and tested during a pilot involving 5 European broadcasters sharing a Multi-CDN overlay to load balance their traffic over 3 different CDNs using dynamic switching algorithms. The findings presented in this paper will be updated during the IBC 2017 presentation.

## INTRODUCTION

Broadcasters are increasingly relying upon online delivery. Unfortunately, using the best effort network offered by the open internet is more expensive (4), operationally less reliable and offers insufficient traffic capacity compared to traditional broadcast distribution methods. By switching HTTP traffic between different CDNs, based on real-time performance data and business parameters, these limitations can be overcome. Stacking of CDNs improves redundancy, increases availability and capacity which should improve the audience's quality of experience while driving costs down.

From a cost perspective, it makes sense to allocate as much traffic as possible to a single supplier, but this creates an operational risk as all traffic is run through a single arrangement. With a Multi-CDN setup, the content provider can switch to a lower cost CDN if the quality is good enough. Automatically applying real-time traffic data in combination with business rules enables a dynamic optimal choice of the data flow. It leverages fluctuating live capabilities of different networks and increases operational control.

As well as optimizing for quality, the Multi-CDN load balancing solution can be used to maximize different bandwidth arrangements, i.e. CDN operators or peering relations. It will be able to fill the 'pipes' efficiently, to comply with different contractual commitments and switch networks dynamically when performance is not meeting the required standards. One does not have to bet on the service of a single provider; on the contrary, it will be possible to add promising new providers or remove low performers.

However, it is not all good news with this model, as the switching layer introduces new costs that can be substantial if it has to be operated by a single content provider. Also the upscaling from a single to Multi-CDN introduces extra complexities as roles and responsibilities get distributed over different organisational entities. The technical load balancing solution should provide the tools to manage this situation effectively. But it will also impact the choice of partners as potential competitors need to complement each other and cooperate in a single service to work on common solutions.

Recognizing these requirements for its Members, the EBU initiated a new Multi-CDN service, known as EBU Flow. It started in May 2017 as a pilot. It has two main objectives: to improve the quality of online delivery and to reduce the cost to the content providers (3). At the time of writing, the participating EBU Members are: RTÉ, VRT, RTBF, NPO and ERT.

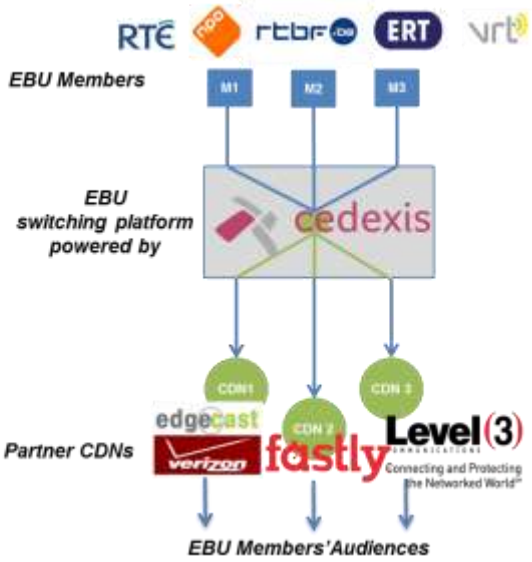


Figure 1: EBU Flow Multi-CDN pilot setup.

The service is based on a strategy of using multiple CDNs in a single service to optimize data flows from the content provider to the audience. Interviews with EBU Members and engagement with the industry have shown a clear consensus that a Multi-CDN is necessary to meet the growing demand for video delivery regionally and in terms of throughputs.

## ESSENTIALS OF A MULTI-CDN STRATEGY

Using multiple CDNs in a single delivery environment is possible due to the fact broadcasters use HTTP streaming nowadays. This commoditization allows media content to be cached, played out and monitored in a similar fashion. Content providers can combine open CDNs with other arrangements, for example, their house CDN or peering relations. All available capacities and connection speeds in the delivery chain can be measured in near real time and this information can be taken into the load balancing

equation on their own terms. For example, it should only switch traffic to an open CDN when peering capacity is fully utilized.

In the most basic scenario, CDNs provide pure HTTP or HTTPs transport service for delivering GBs of video they collect from the Origin server to end users of the content provider. An API wrapper unifies the specific calls providing a single point of integration to engage with different CDNs. This allows for example to purge all caches in the different CDNs with a single call.

The CDN overlay technology itself is a switching layer to load balance traffic between different CDNs based on dynamic switching rules. A play request from an end user generates a call in the broadcasters' backend to this switching API server for a recommendation for the optimal CDN to use for the IP-address involved. The resolved redirect URL or URI path allows the media player to start buffering the content from the intended cache inside the CDN network.

**Load Balancing rules**

Before real time performance metrics can be applied efficiently one could only use static load balancing with fixed business rules to direct traffic based on volume, access ISP or specific capabilities of a network. Dynamic switching changes the game and allows performance metrics like Quality of Service (QoS) as measured by a network probe or by the media player itself to be used in a real time automated CDN recommendation. Combine this quality metric with the cost price for a network and the optimal choice of a CDN can also be the cheapest available choice (Figure 1).

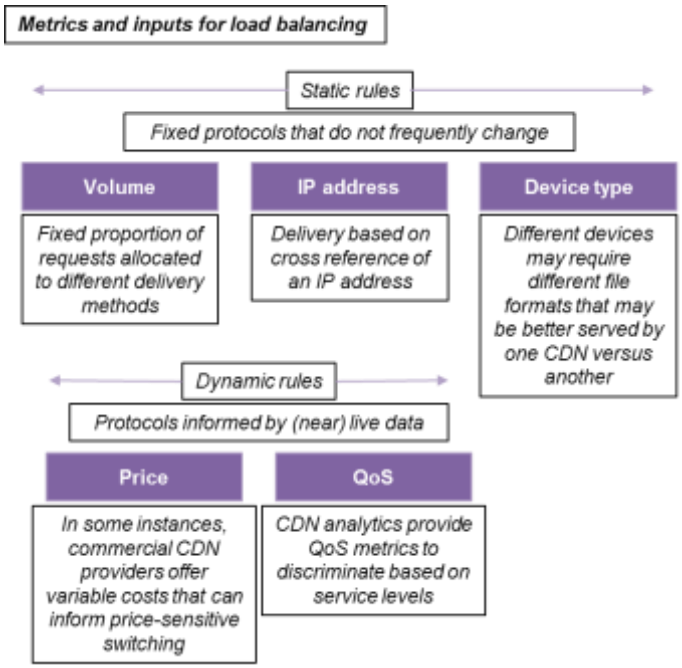


Figure 2: Business rules for load balancing traffic over different CDNs use both static and dynamic data to optimise operational efficiency

**Redundancy**

The switching platform provides a redundant setup without a single point of failure. If one of the connected CDNs is not available the traffic will be automatically switched to the other. Pilot results of from May and June 2017 prove 100% availability as at all times there was at least one CDN available.

The risk the switching platform itself fails can be mitigated by creating back-up strategies either server- or player-side to fall back to one of the CDNs. To avoid overflowing in that mitigation scenario, the provided backup CDN list should be configurable to allow the resolve to be another CDN every time. The following logic can be applied in pseudo code:

1. New asset request by client X
2. Initiate timer
3. Request API for ordered list for client X
    a. If timer exceeds 250ms
        i. Issue default API object
    b. Else
        i. Issue retrieved API object
4. Media selection process takes API and builds URI paths for player
5. Manifest issued to player with chosen URI paths

Figure 2 is an illustration of the real time monitored throughput performance of different CDNs on basis of kbps throughput on 95$^{th}$ Percentile over time. The red dotted line represents a minimal required performance for a specific video to play without interruption. At peak time (red arrow) the load balancer would switch traffic to the brown CDN to avoid service interruption. But at most times, and this is confirmed by the pilot, more than one CDN can deliver better than minimally required allowing to switch traffic for commercial reasons to a specific CDN.

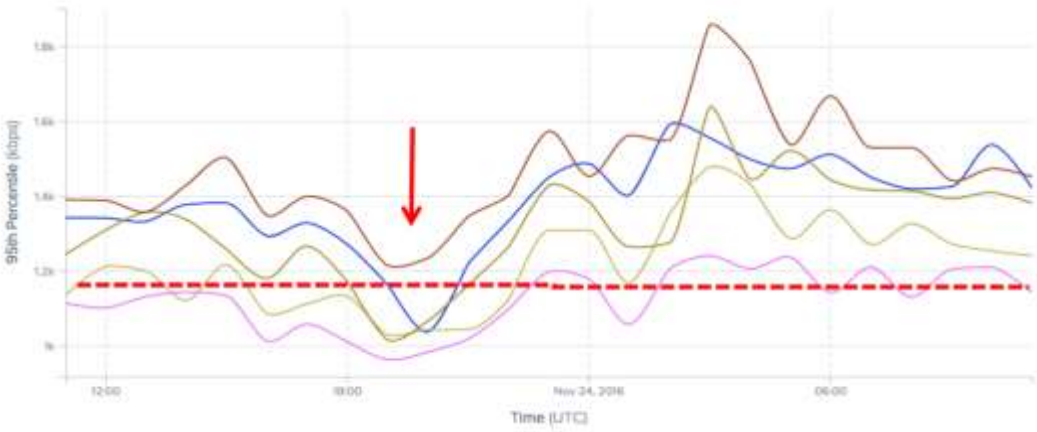


Figure 3: Switching platform selects automatically the best performing CDN of that moment for the targeted location. Illustration based on (1).

**REAL TIME METRICS**

Google Maps provides a useful analogy to explain how real time metrics are applied in recommending the optimal route for the traffic. If one has to get somewhere, Google Maps will provide potential routes and calculate the time it would take to reach the destination

based on the anonymized travel times of other users. Google collects location data from the device, using it to calculate how much traffic congestion there is on a given route. If an alternative route becomes quicker this will change the recommendation to the end user.

Ideally in a Multi-CDN service the content provider collects connectivity performance data with respect to different content sources, such as different caches in CDNs, peering relations or from the origin server. Availability (response percentage on heartbeat request) and Throughput (actual speed of the network measured by pulling through a data object) are mostly used by the backend of the connected content provider and then be informed about the best performing location. The player will use this as a preference and the other CDNs as fall-back.

**Quality of Experience**

The most promising approach is the measurement of the Quality of Experience (QoE) from the end user at a certain moment in time in a specific location. In this paper we use QoE metric proposals from Streaming Video Alliance (5) and the DASH Industry Forum 'DASH-IF (2)'. Basic data in this respect includes:

- Total Playing Time
- Video Start Failure
- Video Start Time
- Re-Buffering Ratio
- Bitrate

These basic QoE metrics can be deduced from captured data inside the media player. When events are trigged an analytics client data collector pushes reports to pre-processing servers to create data points which can be transformed into CDN recommendations. Media controller event handlers are not sufficient to use as a trigger to collect player properties but need to be complimented by a heartbeat measurement. In case the end user closes the browser the last available heartbeat registration can be used as an end situation by using the HTML5 'timeupdate' event to collect, for example every 10 seconds values representing the current user experience (6).

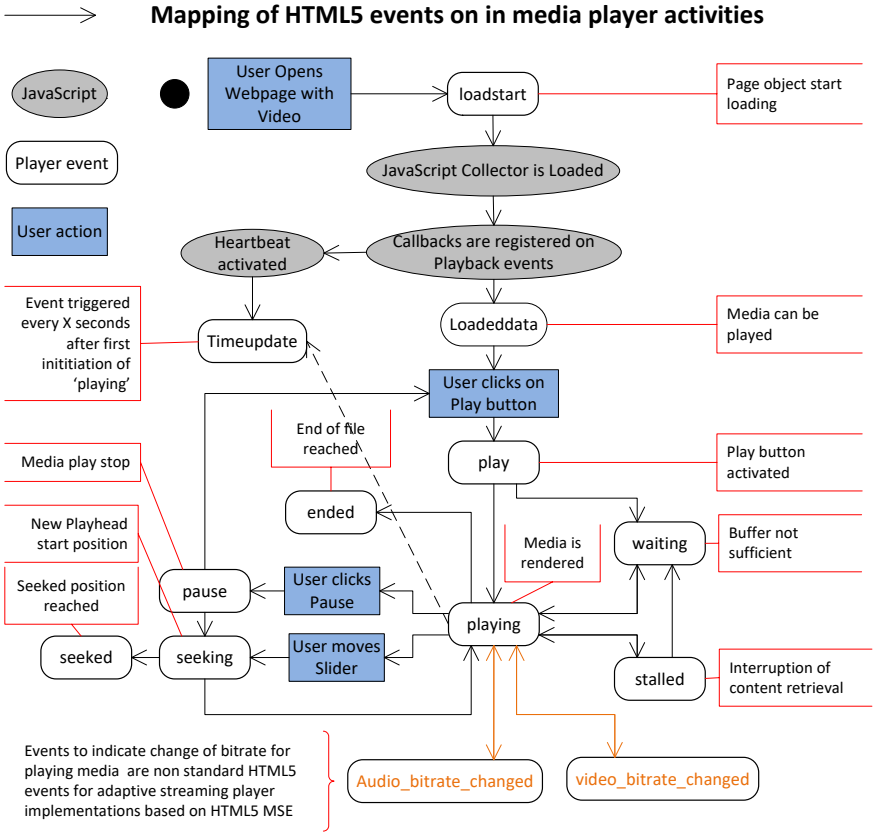


Figure 4: Example of mapping of basic HTML5 events to a playout sequence in a media. The trigger of different events during the playout session can be used to request player states or properties (6).

**Event to QoE**

With an event from a media player a process is triggered to collect properties in the client at that moment of time. In Figure 3 the relevant events are mapped to the playout sequence. Collected information is stored server side and processed into data points that can be used in the Multi-CDN setup as QoE input (Figure 4). The discussed data points can be derived from player properties in the following matter:

1) *Total Playing Time* is the SUM of
   a) Deltas of playhead wallclock timings between triggered 'playing' and 'waiting' events.
   b) Deltas of playhead wallclock timings between triggered 'playing' and one of the intended interruption events 'pause', 'seeking' or 'stop' or the last available heartbeat 'playing' event registration.
2) *Video Start Up* Time is SUM of playhead wallclock timing deltas between 'play' and 'playing' events.
3) *Re-buffering Ratio*: SUM of
   a) *Video Start Up Time.*
   b) Deltas of playhead wallclock timings between trigged 'waiting' and 'playing' events TOTAL DIVIDED by *Total Playing Time.*

4) Video Start Failure can be deduced from triggered Error events between ‘loadstart’ and ‘playing’.
5) *Bitrate*: This metadata is fixed for single file content but is a variable depending on the connection speed to the available caches when adaptive streaming formats like MPEG DASH or HLS are used. Adaptive streaming can be implemented in HTML5 environments with Media Source Extensions (7).
   a) The original bitrate and encoded resolution need to be captured the first time ‘playing’ event is triggered for example by reading this technical metadata from the MPD when MPEG DASH.
   b) Idem for all subsequent changes of playout media representations during the session including the relative positions of the playhead and corresponding wallclock timings.

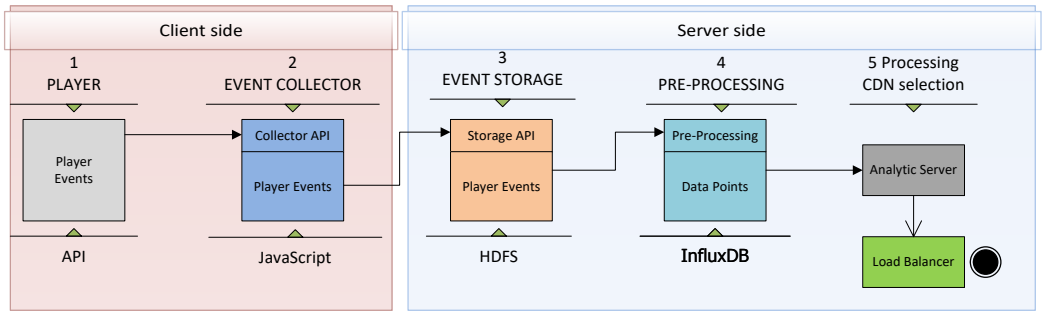


Figure 5: Example of a data flow processing architectural overview to capture reports generated in the media player via HTML5 video object to the pre-processing of this information servers side into QoE data points. In near real-time these need to be processed into decisions of the Load Balancer in the Multi CDN setup.

**Data Processing**

Combining the metrics provided per IP-address a description of the achieved quality is collated. As per content provider the webpage/application is the same any differences in quality can be attributed to the delivery chain or playout devices used. Combining more sources allows filtering out noise of home networks and playout devices statistically. With enough data at hand a prediction model of which CDN delivers the best speed to a specific location at a specific time can be made.

When not enough player requests are available, additional information can be acquired from third parties reselling real-time performance metrics acquired by pulling objects through the network or implementing functionalities in the player to test the speed towards the different available CDNs.

From a privacy perspective, the data collection and processing can be compared with the approach of website analytics tools. A QoE evaluation is accompanied with a timestamp, IP-address, content-ID and identification of the playout CDN. The content ID and CDN identification is a combination of the URL applied after retrieval algorithm that can be requested through ‘currentscrc’ attribute of the media element as described in HTML5 MSE (7).

## MULTI-CDN LOAD BALANCE ARCHITECTURE

As described above there is a central role dedicated to the switching entity or Load Balancer. It aggregates from different sources as described above performance metrics from the end user media players and other probes in the network. This data is processed into performance recommendations of the different CDNs in different regions at this specific time. From the CDN the Load Balancer receives updates on how much data is used which can fuel in combination with the performance information a business rule that for example optimises the utilisation of the bandwidth capacity procured from the CDNs.

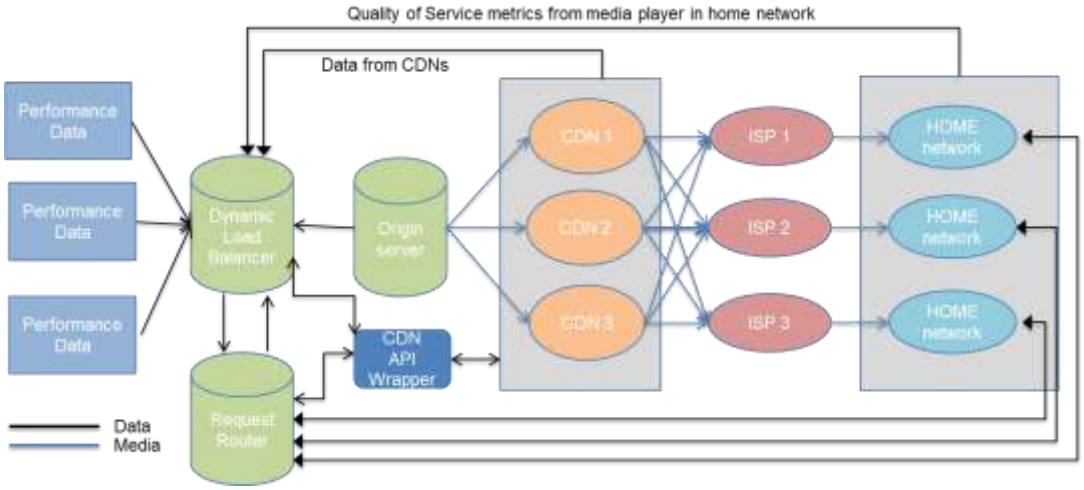


Figure 6: Example Multi-CDN architecture recognising the different essential components and the direction of the data flows.

### CDN API Wrapper

The Load Balancer should ideally perform the task of the API wrapper to centralise the communication with the CDNs. This allows not only to aggregate user reports and billing data but can also perform geo fencing and token authentication (if required). These functions are deployed differently and normally enforce the CDN specific implementations.

CDNs use different (versions) databases to check if player requests are from an area content may be played out and specific tokens exchange implementations to check if the media player is authentic. Both can be integrated in a single call from the player to the Load Balancer to check if the content can be played out or not by verifying the key and the IP-lookup. The Load Balancer is aware which CDN should be used and requests from the CDN API Wrapper to retrieve the playout URL from the CDN with the resolved token. This information is passed through to the media player. The CDN API Wrapper functionality is being monitored throughout the pilot to conclude if it is required.

Last but not least the CDN API Wrapper function should allow broadcasters to purge all caches in a single call. Whenever the rights management backend or Content Management System triggers an event that content cannot be published anymore, this should be carried out swiftly and thoroughly. Operationally it would be sub-optimal to have to delete publications from different CDN interfaces manually.


**Request Router and HTTPs**
As the final act, the Load Balancer decides which is the preferred CDN for a specific end user call. The Request router performs the handshake with the video player object to communicate what the definite URL of the media file is directing it to one of the CDNs or private edges/caches in a hybrid CDN setup.

To enable HTTPs a single wildcard can be applied over different CDNs by creating a separate private key for every CDN and registering all relevant domains using the same naming convention, for example: <sub-brand>.<brand>.<domain name>. Performance wise it is recommended to use COM or ORG domains and not to host any other activities under the same domain name.

**Role of the Origin**
In a Multi-CDN setup the Origin needs to be abstracted from the CDNs. The CDNs are white listed in the firewall protecting the Origin and can pick up data when it is not cached yet. Normally CDNs only cache content after a second end user request. Using more CDNs introduces extra traffic to the Edge servers of the live and on demand Origin setup. During the pilot 3 different mitigations of this traffic increase are tested. The first consists of simply upscaling the Edge capacity of the Origins to meet the additional traffic. The second involves changing the cache headers to enable the content to be available longer inside the CDN. Longer availability should reduce the amount of times semi-popular content needs to be picked up reducing the overall traffic to the Origin. The third option to tackle increased Origin traffic consists of creating an extra Origin inside the CDN for pre-caching the content. In any case it is recommended to have a double location redundant live and on-demand Origin setup.

**CONCLUSION**
A Multi-CDN setup abstracts the publication plane from the delivery networks. The decision logic, if content can be played and which route the media should travel over the internet to reach its audiences is a decision taken by the Load Balancer. This allows the content provider to use business rules and automatically deduce on basis of network performance, end user experience, publication rights and contractual arrangements with distribution partners what the optimal response is to a player request from a specific location at a specific time. With these proven technical practices for a Multi-CDN deployment broadcasters can gain more operational control over their online distribution strategy.

**REFERENCES**

1. Cedexis, 2017. *How to Evaluate and Implement a Multi-CDN Strategy*. Available from: http://go.cedexis.com/Implementing-Multi-CDN.html. (Accessed on 01-05-2017).
2. DASH-IF, 7-10-2016. *DASH-IF position Paper: Proposed QoE Media Metrics standardization for segmented media playback.* Available from: http://dashif.org/wp-content/uploads/2016/10/ProposedMediaMetricsforSegmentedMediaDelivery-r12.pdf (Accessed on 01-05-2017).

3. EBU, 2017. *EBU Flow*. Available from: https://tech.ebu.ch/cms/flow (Accessed on 01-05-2017).
4. OFFCOM, Redshift Strategy, 2015. *PSB distribution costs*. Available from: https://www.ofcom.org.uk/__data/assets/pdf_file/0021/62643/psb_distribution_costs.pdf (Accessed on 01-05-2017)
5. Streaming Video Alliance, 2016. Working Group Quality of Experience, 05-2016. *Key Network Delivery Metrics.* Available from: https://www.streamingvideoalliance.org/download/4482 (Accessed on 01-05-2017).
6. W3C Recommendation, 28-10-2014. *HTML5 A vocabulary and associated APIs for HTML and XHTML.* Available from: https://www.w3.org/TR/2014/REC-html5-20141028/embedded-content-0.html (Accessed on 01-05-2017)
7. W3C Recommendation 17-11-2016. *Media Source Extensions*. Available from: https://www.w3.org/TR/media-source/Streaming (Accessed on 01-05-2017).