ENHANCING MPEG DASH PERFORMANCE VIA SERVER AND NETWORK ASSISTANCE

Emmanuel Thomas[*], M.O. van Deventer[*], Thomas Stockhammer[†],
Ali C. Begen[‡], Jeroen Famaey[§]
[*]TNO, the Netherlands, [†]Qualcomm, Germany,
[‡]Cisco, Canada, [§]University of Antwerp – iMinds, Belgium

ABSTRACT

MPEG DASH provides formats that are suitable to stream media content over HTTP. Typically, the DASH client adaptively requests small chunks of media based on the available bandwidth and other resources. This client-pull technology has proven to be more flexible, firewall-friendly and CDN-scalable than server-push technologies. However, service providers have less control given the decentralized and client-driven nature of DASH, which introduces new challenges for them to offer a consistent and possibly higher quality of service for premium users. This issue is addressed by MPEG in a new work referred to as SAND: Server and Network-assisted DASH. The key features of SAND are asynchronous network-to-client and network-to-network communication, and the exchange of quality-related assisting information in such a way that it does not delay or interfere with the delivery of the streaming media content. MPEG is expected to publish and complete the work on SAND as a new part of the MPEG DASH standard by early 2016.

INTRODUCTION

Over the last few years, HTTP-based adaptive streaming has become the technology of choice for streaming media content over the Internet. In 2012, MPEG published a standard on Dynamic Adaptive Streaming over HTTP (DASH) [1], which has been adopted and profiled by other standards and industry bodies, including DVB, 3GPP, HbbTV and DASH-IF. The DASH formats are primarily designed to be used in client-pull based deployments with HTTP the protocol of choice for media delivery. A client first retrieves a manifest in a Media Presentation Description (MPD), and then it selects, retrieves and renders content segments based on that metadata, as seen in Figure 1.

DASH when deployed over HTTP offers some fundamental benefits over other streaming technologies. DASH requests and responses pass firewalls without any problem, like any other HTTP messages. As the content is typically hosted on plain vanilla HTTP servers and no specific media servers are necessary, DASH is highly scalable: DASH segments can be cached in HTTP caches and delivered via Content Delivery Networks (CDN), like any other HTTP content. Most importantly, a DASH client constantly measures the available bandwidth, monitors various resources and dynamically selects the next segment based on that information. If there is a reduction in bandwidth, the DASH clients selects

segments of lower quality and size, such that a buffer underrun is prevented and the end user retains a continuous media consumption experience. From many studies, it is well known that start-up delays and buffer underruns are among the most severe quality issues in Internet video and DASH constitutes a solution to overcome and minimize such problems.

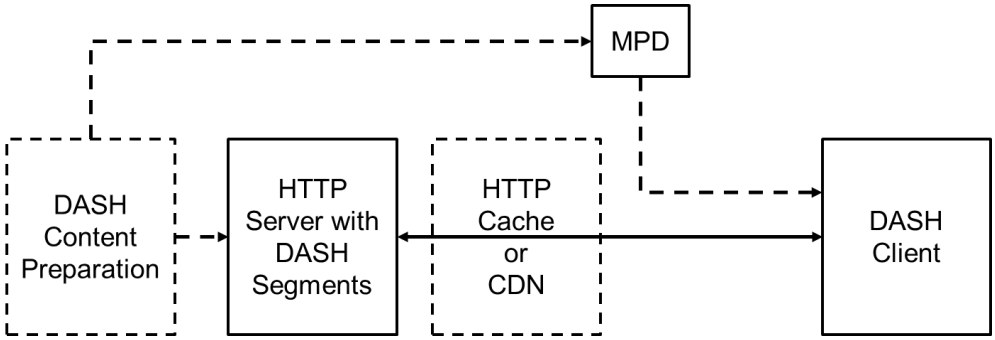


Figure 1 – Conceptual architecture of MPEG DASH.

However, the fundamental decentralised and client-driven nature of DASH also has some drawbacks. Service providers may not necessarily have control over the client behaviour. Consequently, it may be difficult to offer a consistent or a premium quality of service. Examples include that the resources announced in the MPD may become outdated after a network failure or reconfiguration, resulting in misdirected an unsuccessful DASH segment requests by the client. A DASH client can mistakenly switch to lower quality segments, when a mobile hand-over or a cache miss is interpreted as a bandwidth reduction. Massive live DASH streaming may lead to cascades of cache misses in CDNs. A DASH client may unnecessarily start a stream with lower quality segments, and only ramp up after it has obtained bandwidth information based on a number of initial segments. Multiple DASH clients may compete for the same bandwidth, leading to unwanted mutual interactions and possibly oscillations [6]. As a consequence, service providers may not be able to guarantee a premium quality of service with DASH, even in managed networks where regular DASH clients may not fully take advantage of the offered quality of service features.

In 2013, MPEG and the IETF organised a joint workshop [2] to discuss the issues and potential solution directions, as input to the 2nd edition of the DASH standards. Soon after, MPEG started the Core Experiment on Server and Network-assisted DASH (CE-SAND), in which use cases are defined and solutions are explored. Based on the results of CE-SAND, MPEG is developing an architecture, data models and a protocol solution, expected to published as part 5 of the MPEG DASH standard in 2016. The use cases and status of the work as of mid of 2015 are summarized in the remainder of this paper.

SAND USE CASES AND EXPERIMENTS

The CE-SAND addresses the following topics:

- Unidirectional, bidirectional, point-to-point and multipoint communication, with and without session management between servers/CDNs and DASH clients
- Providing content-awareness and service-awareness towards the underlying protocol stack including server and network assistance

- Various impacts on the existing Internet infrastructure such as servers, proxies, caches and CDNs
- Quality of service (QoS) and quality of experience (QoE) support for DASH-based services
- Scalability in general and specifically for logging interfaces
- Analytics and monitoring of DASH-based services

From these topics, MPEG experts derived a set of use cases to illustrate the scope of this core experiment including:

- Network mobility, e.g., when the user physically moves, which makes the device switching from one network to another
- Server failover scenario, e.g., when the segment delivery node crashes, which potentially leaves the DASH client without segments to retrieve
- Server-assisted DASH adaptation logic, e.g., when a server assists DASH clients for selecting representations
- Operational support of live service by real-time user reporting, e.g., when DASH clients report useful information in order to improve the overall quality of service
- Bi-directional hinting between servers, clients and the network, e.g., where a DASH client lets the delivery node know beforehand what it will request in the near future
- Inter-device media synchronization, e.g., when one or more DASH clients playback content in a synchronised manner

Over the past two years, MPEG experts have collected evidences from experiments and relevant feedback from the industry showing the benefits of such assistance for DASH clients. For example, one of these experiments on start-up time and quality convergence is reported in more detail below. This falls into the category of "server-assisted DASH adaptation logic".

Experiment: Faster Quality Convergence Through QoS Signalling

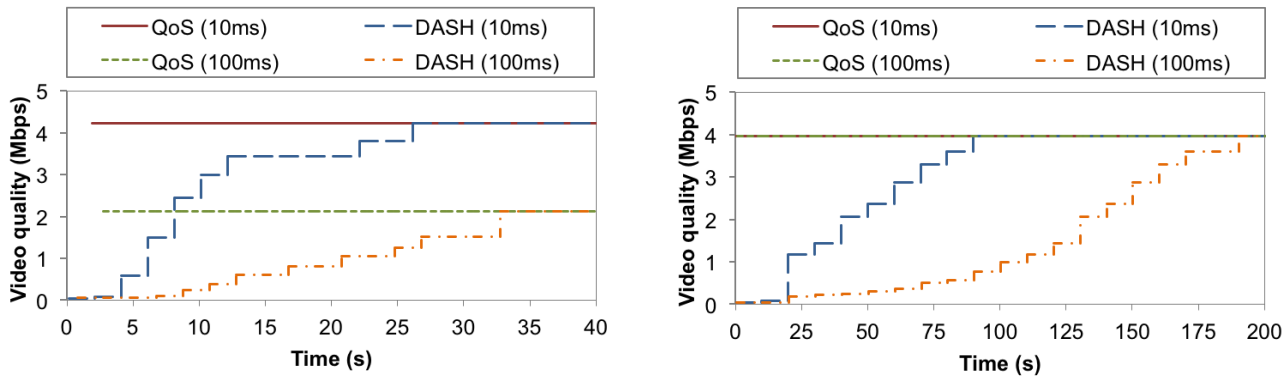
A major source of QoE degradation in DASH is the slow convergence towards the optimal quality during the start-up period of a stream. The client will not only wait to start playback until its buffer has reached a certain size, but will also start at the lowest video quality and slowly ramp it up to find the optimal point. This will occur both at the start of a streaming session, as well as when the user switches to a different stream (e.g., channel change) or to a different point in the same stream (e.g., seeking).

A potential solution for this problem is to let the server signal the client about the performance or QoS of the connection. This enables the client to make a more elaborate choice concerning the initial quality, leading to faster convergence to the optimum. An experiment was performed to test this hypothesis. Its goals are (i) to determine the improvements in quality convergence speed as a consequence of QoS signalling, and (ii) to study any potential (adverse) side effects.

The experiment is executed using a custom built client based on libdash [3]. It implements the MSS rate adaptation algorithm described in [4], with some additional optimisations to take into account the QoS signalling variable. The `minBufferTime` is set to 8 seconds.

The Big Buck Bunny video [5] of the MMSys12 dataset is used for streaming, with segment durations of either 2 or 10 seconds, and 20 quality representations between 50 Kbps and 8 Mbps. The client is directly connected to the delivery server via a traffic shaped link that has an available bandwidth of 5 Mbps and a one-way latency of either 10 or 100 ms.

Figure 2a shows the played video bitrate over time for 5 Mbps available bandwidth and 2-second segments, comparing DASH with (denoted QoS) and without (denoted DASH) QoS signalling. The figure clearly shows that QoS signalling solves the slow quality convergence process, for both low and high latencies. Traditional DASH takes, for respectively 10 and 100 ms latency, 26 and 32 seconds to reach the optimal point, whereas the DASH client with QoS signalling takes 1.8 and 1.92 seconds to start the playback at the optimal point. Playback starts a bit earlier in the traditional DASH client, however, it starts with rendering lower quality segments. If QoS signalling information is provided, the client may safely ignore the `minBufferTime*bandwidth` directive in order to keep the start-up delay comparable to rendering lower quality segments.



(a) 2-second segments

(b) 10-second segments

Figure 2 – Results of the QoS signalling experiment, comparing with QoS signalling and traditional DASH for 5 Mbps available bandwidth and 10 or 100 ms latency.

Figure 2b depicts the same results for 10 instead of 2-second segments. In general, the trend is similar. However, the buffer is filled up faster since there is less throughput loss due to sending fewer requests and waiting for their responses. Consequently, the initial playback delay of the approach with QoS signalling is reduced to 0.02 and 0.2 seconds for 10 and 100 ms latency, respectively. For the same reason, when using longer duration segments quality is less affected by latency in traditional DASH. Finally, the quality convergence process is more extreme without QoS signalling, clocking in at 90 and 190 seconds for 10 and 100 ms latency, respectively. As such, QoS signalling is even more advantageous when longer segments are used. Note that if there were fewer representations available to the clients, the convergence times would be shorter; however, the quality variation during the convergence period would be more drastic.

A NEW PART FOR THE MPEG DASH STANDARD

The SAND Architecture

In the SAND architecture, we have three broad categories of elements. They are (i) DASH clients, (ii) DASH-assisting network elements (DANE), and (iii) regular network elements. Regular network elements are DASH unaware and treat DASH delivery objects as any other object, although they could be present on the media delivery path. Transparent caches are an example of regular network elements. DASH-assisting network elements (DANE) have a minimum intelligence about DASH. For example, a DANE could identify and parse an MPD file and DASH segments to treat them differently or modify them.

The SAND architecture has the following three interfaces that carry various types of messages:

- Client-to-DANE (C2D) Interface: Metrics messages and status messages
- DANE-to-DANE (D2D) Interface: Parameters Enhancing Delivery (PED) messages
- DANE-to-Client (D2C) Interface: Parameters Enhancing Reception (PER) messages

Collectively, PED, PER, metrics and status messages are referred to as SAND messages. In this context, a media origin that serves DASH content, receives metrics messages from the clients and sends PED parameters to other DANEs is also considered a DANE element. Similarly, a third-party analytics server that receives metrics messages from the DASH clients and sends SAND messages to the clients is a DANE element. Note that the third-party server is not necessarily on the media delivery path so it does not see the DASH segments. However, as it understands the DASH metrics and produces SAND messages for the DASH clients to improve delivery, it is still considered a DANE element.

The messages sent by the clients that carry metrics information are called metrics messages. The messages sent by the clients that carry non-metrics information are called status messages. The metrics and status messages have the same structure; however, it is important to distinguish them since these messages carry information of different nature.

The PED messages may flow in one direction or in both directions between two DANEs. This will depend on the functionality of the DANEs. For example, PED messages can be sent by a third-party analytics server to a media origin, packager or an edge router/server to enhance the delivery. However, sending PED messages the other way around (to the third-party analytics server) would not make sense as such a server only digests incoming metrics/status messages and produces PED/PER messages.

In a streaming ecosystem, there are likely several other interfaces that need to be deployed. For example, the media origin or the content provider could need an interface to the DRM servers to exchange keys. The streaming servers could need an interface to the subscription/management servers. Such interfaces are explicitly excluded from the scope of the CE-SAND.

The SAND architecture and message flows are shown in Figure 3. In this figure, metrics and status messages are shown with the same green arrows for simplicity only; this does not mean that these messages are always sent together at the same time. Note that in the figure below, the number and order of the dash-assisting and regular network elements on the media delivery path would depend on the network topology. However, this does not affect how the framework functions.

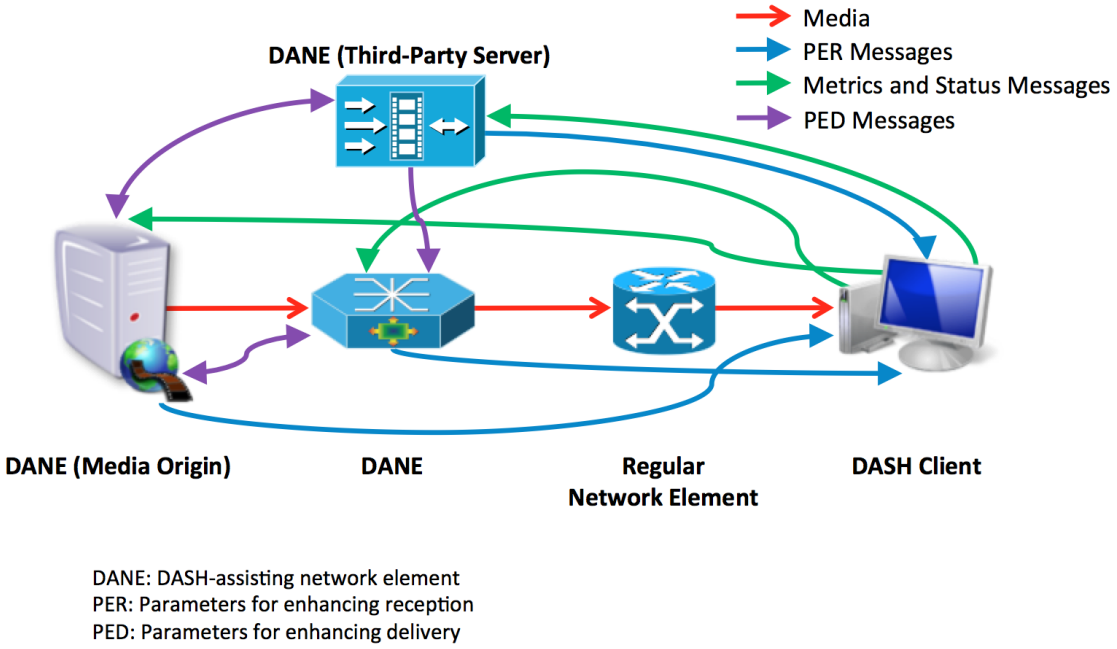


Figure 3 – Proposed SAND architecture and message flows.

The SAND Messages

While the current version of the SAND specification lists a number of PED, PER, metrics and status messages, these messages are subject to change due to the standardization process; i.e., the existing messages can be removed or modified, and new messages can be added. Yet, in this section, we provide examples from each category to demonstrate how SAND can be used in practice. These examples mainly target the use case of bi-directional hinting between servers, clients and the network.

SAND Metrics and Status Messages

Consider a scenario where a DASH client would like to send a hint to the cache server from which it receives media segments about which particular representation(s) of a content it is planning to fetch over a specified number of requests. For example, the DASH client may notify the cache server with a SAND Status message that it plans to request segments 41, 42 and 43 from the 5th representation for movie K. If the cache server is a DANE element that understands such a message, it can try to proactively prefetch these particular segments in advance so that they will likely be ready to be served when the actual request is received from the DASH client. This helps improve the cache hit ratio on the server as well as the streaming quality perceived by the client.

This SAND Status message, which is referred to as `anticipatedRequests`, consists of bunch of URLs (possibly with the byte-range information) for the desired segments and a target duration.

The receiving cache server may or may not be able to process these messages in time if there is an upstream bandwidth or storage shortage, or there are too many such messages all requesting different set of segments. In this case, the messages need to be prioritized appropriately and the caching algorithm may need to be adjusted for proper cache filling and eviction.

SAND PER Messages

Consider a live streaming scenario where a cache is serving a large number of DASH clients. These DASH clients may have different capabilities in terms of connectivity. Prior research [6] has shown that streaming clients that share the same network resource may experience the bitrate oscillation problem, where the clients cannot figure out their fair-share bandwidth and keep requesting segments from different representations (i.e., frequent upshifts and downshifts). Later research [7] has shown that the streaming experience can quickly deteriorate even in the presence of cache servers under certain circumstances due to the lack of ability to prefetch all potential segments that could be requested by the clients.

A solution to this problem is that the cache server sends to clients a SAND PER message, referred to as `resourceStatus`, that lists what segments are available and for how long they will be available on the cache server. This provides the DASH clients a hint for their future requests to maintain a more stable streaming experience.

SAND PED Messages

Consider a live sports event where the content captured in real time is encoded and packaged into a number of representations. Suppose the average representation bitrates are 1, 3 and 5 Mbps, and all the representations are available to the streaming clients. A while after the live event started, the analytics servers will start receiving metrics messages from the clients providing detailed information about their reception quality. If a large portion of the feedback indicates that most clients are oscillating between the 3 and 5 Mbps representations, the decision engine processing the analytics data can speculate that the introduction of a new representation of 4 Mbps will alleviate the problem. This decision can be conveyed to DANE-enabled transcoders and packagers through a PED message, referred to as `alterEncoding`. Alternatively, the PED message can request the transcoder to replace the 5 Mbps representation with the 4 Mbps representation, if there are constraints on processing and storage resources. Following such a change, the manifest has to be revised and the DASH clients fetch the new manifest file through usual means, possibly stimulated by a PER message.

The SAND Transport Protocol

In order to transport the SAND messages, an appropriate message format is necessary as well as a transport protocol. Two types of downlink scenarios are considered relevant:

1. Assistance: A scenario for which the message is provided as auxiliary information for the client, but the service will be continued even if the client ignores the message.
2. Enforcement/Error: A scenario that requires the client to act otherwise the service is interrupted. The DANE cannot or is not willing to respond to the request with a valid resource but provides suitable alternatives.

Both types of communication are relevant and it is up to the service providers to use adequate SAND messages. In addition, both DANE on the media delivery path and outside the media delivery path may use these mechanisms.

To address the use cases described in the CE-SAND, the SAND specification recommends the following HTTP-based communication channels:

- For assistance, a dedicated HTTP header field that indicates an absolute URI pointing to a SAND resource. Upon reception of an HTTP entity containing the SAND header field, the DASH client issues a GET request to the indicated element to receive the SAND message.
- For enforcement, a suitable method is the use of a 300 Multiple Choices response where the response includes an entity containing a SAND message. The entity format is specified by the media type given in the `Content-Type`.
- For error cases, a suitable method is the use of a suitable 4xx error code. The response may include a SAND message from which the client can deduce the reason for the error code and potential resolution of the problem.

The SAND communication channel can also be implemented using other protocols such as Server-Sent Events or WebSockets. In the latter, the signalling of the communication might be achieved by inserting a SAND element in the DASH MPD that advertises the URI endpoint of the communication channel as well as the corresponding protocol to use.

CONCLUSION

Server and Network-assisted DASH (SAND) provides a bridge between the traditional server-controlled streaming and client-controlled HTTP streaming. The technology is introduced in a way that it is expected to assist and enhance the operation of client-centric DASH streaming. The technology addresses messages as well a communication channel in order to fulfil the different requirements and use cases that were collected in the MPEG standardization process. SAND will be another step towards establishing DASH as a format that can be used for a broad set of applications and use cases.

REFERENCES

1. MPEG, Dynamic Adaptive Streaming over HTTP (DASH), ISO/IEC 23009, 2012
2. MPEG/IETF, Joint workshop on session management and control for MPEG DASH, Vösendorf, 28 July 2013, http://mpeg.chiariglione.org/about/events/workshop-session-management-and-control-mpeg-dash
3. Source: https://github.com/bitmovin/libdash
4. Jeroen Famaey, Steven Latré, Niels Bouten, Wim Van de Meerssche, Bart De Vleeschauwer, Werner Van Leekwijck, Filip De Turck, **"**On the merits of SVC-based HTTP Adaptive Streaming," Proceedings of the 13th IFIP/IEEE International Symposium on Integrated Network Management (IM), Ghent, Belgium, 2013
5. Source: http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/
6. Saamer Akhshabi, Ali C. Begen and Constantine Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," Proceedings of ACM Multimedia Systems Conf. (MMSys), San Jose, CA, February, 2011
7. Danny H. Lee, Constantine Dovrolis and Ali C. Begen, "Caching in HTTP adaptive streaming: friend or foe?," Proceedings of ACM International Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Singapore, March, 2014