DASH-IF Position Paper: Server and Network Assisted DASH (SAND)

published December 2016

Abstract

MPEG's Server and Network Assisted DASH (SAND) technology, i.e., specified in ISO/IEC 23009-5 [1] offers standardized messages and exchange protocols for service providers and operators to enhance streaming experience while also improving network bandwidth utilization. In order to enhance the delivery of DASH content, SAND relies on messages between DASH clients and network elements for the purpose to improve efficiency of streaming sessions by providing information about real-time operational characteristics of networks, servers, proxies, caches, CDNs as well as DASH client's performance and status. This whitepaper presents several use cases and applications relevant for SAND, and also presents a few example workflows demonstrating how the various SAND features can help fulfill these use cases.

Comments on this paper are welcome at https://gitreports.com/issue/Dash-Industry-Forum/SAND

1 Introduction

MPEG DASH [2] provides formats that are suitable to stream segmented media content over HTTP. DASH clients follow a client-pull paradigm by adapting their requests based on the available bandwidth and other local resources. This has proven to be easier to deploy over CDN infrastructure than server-push technologies. However, the client-driven and decentralized nature of DASH leads to limits the amount of control and coordination between the client and server / network behaviors leading to suboptimal streaming performance. For instance, service providers may not have influence over the DASH client behavior and may not have the means to share network-level information to assist the client's adaptation logic, which may end up increasing the amount of re-buffering.

To address these challenges, MPEG SAND aims to enable better cooperation between the DASH client and server operations, and provides the standardized interfaces toward realizing the following benefits for streaming services:

- Streaming enhancements via intelligent caching, processing and delivery optimizations on the server and/or network side, based on feedback from clients on anticipated DASH Segments, accepted alternative DASH Representations and Adaptation Sets, and requested bandwidth.

- Improved adaptation on the client side, based on network/server-side information such as cached Segments, alternative Segment availability, and network throughput/QoS.

2 Use Cases and Applications

2.1 DASH Operation with Proxy Caches

2.1.1 Basic Proxy Caching

John, who lives in Europe, has discovered that his DASH-enabled device suffers from frequent playback quality variation and some playback interruptions, both of which he finds annoying. For streaming quality, John prefers to view streaming in a stable playback quality and fewer or preferably no playback interruptions. In addition, higher presentation media quality is preferable. John has also noticed that such quality variations and playback interruptions typically occur when he views media presentations for which origin servers are located at outside of Europe such as Asia. For content providers that deploy HTTP proxy caches worldwide (potentially covered by multiple CDNs), John notices that streaming quality is generally better with more stable playback, since the assistance of proxy help to save bandwidth and reduce delay.

2.2 Partial Representation Caching

John's DASH-enabled device sends HTTP GET segment requests by parsing a specific Media Presentation Description (MPD). Prior to serving John's segment requests, the proxy cache may have served other DASH clients with the same media presentation where they created HTTP GET segment requests by parsing the same MPD as John's DASH-enabled device. The proxy cache may cache segments which have been sent to other clients for serving future clients requests. As DASH clients request segments, but also switch Representations dynamically, the proxy cache may cache multiple Representations, each of which may be completely or only partially cached. A partially cached Representation is defined as a Representation having segment gaps, i.e. not all segments of the Representation are cached.

2.2.1 MBMS-Related Proxy Caching Use Cases

2.2.1.1 Mobility and Coverage Extension for MBMS-based service

The son of Jari, Jarison, is watching the game live in the stadium. Jari picks up his son and drives to the stadium and when he gets there, the same service is provided over DASH+MBMS-based broadcast in HD quality. In addition, multiple views are provided close to the stadium, one being close from the seat where

Jarison sits. After the game, Jari and Jarison leave the stadium, but continue to watch the interviews from the stadium in the car served through a 3G network.

2.2.1.2 Activation / Deactivation of MBMS delivery in an area

Jarison is sitting in a public place and is waiting for the start of a live transmission from a stadium. Jarison is already watching the channel in order to not miss the start of the live transmission. Jarison does not recognize that his phones switching from Unicast DASH to DASH+MBMS. With the start of the live event, the network has activated the MBMS transmission.

2.2.2 HTTP Proxy Cache in a home GW (gateway), capable of serving multiple DASH players in the home

In this use case, an HTTP proxy cache in the home GW is capable to serve multiple DASH players in the house, using HTTP interface over possibly over WLAN, or other Ethernet connection in the home. Moreover, thanks to SAND, DASH clients are able to discover or request which content is already cached at the gateway.

2.2.3 Next Segment Caching

CDNs can optimize the delivery of DASH resources by pre-caching segments and subsegments into the cache. However, if each segment is named and treated independently, the dependency is not recognized by the network and prefetching from the origin is not possible.

This is issue is specifically relevant in the case of using segmented Representations in an On-Demand case. In case a single Representation is used, the use of byte ranges provides sufficient indication for the CDN to prefetch additional data.

One way to accelerate delivery of segmented content over a CDN is to have the edge server pre-fetch the next segment from origin at the same time as it retrieves the current segment. This means that the segment is ready and waiting when the next request arrives from the client. Since the edge server serving the media segment is not necessarily the same server which served the MPD, it has no visibility in to what the next segment might be. Additionally, it is stateless, and retains no knowledge of prior requests or related MPD requests.

2.2.4 Multi-CDN offering

A content provider wants to utilize multiple CDNs for content delivery, e.g. because some CDNs offer better coverage in certain regions. The content offering should include all available delivery choices (e.g. multiple baseUrls). The content provider uses SAND to steer the DASH client to a certain CDN.

2.3 Consistent QoE/QoS for DASH users

2.3.1 Operator Control of DASH in a Cellular Network

A network operator deploying DASH services or a network operator supporting the delivery of DASH services of an OTT service provider has the ambition to provide consistent quality for users in its network. For this purpose, the content provider wants to provide sufficient QoE to all users that have been granted acquisition to the network and the service. It may also have the ambition to provide certain premium users to maintain a certain service quality when the user plane is congested. The operator may want to influence its QoS control and resource allocation to actively support such use cases, e.g., communicate with the UEs to decrease the bitrate for the video to a certain value that would allow the cell to accommodate the load. Here are some more specific examples around this use case:

1) Jari (regular user) and Jarison (premium user) enter a congested radio area. The mobile operator wants to restrict the required bitrate, but ensure that a basic video quality is maintained for its regular users (Jari) and some higher quality for premium users (Jarison). For this purpose, they assign certain bitrate quality levels to different users on their HTTP connections carrying DASH-content.

2) Jari wishes to watch high-definition video content over his tablet, while Jarison would like to watch standard-definition video content over this smartphone. The operator is able to influence its QoS control and resource allocation to ensure that both Jari and Jarison are simultaneously able to watch their desired content with consistent quality of experience, e.g., with sufficient video quality and without any rebuffering or playback interruption.

2.3.2 Network Assistance with DASH Streaming

The use case for network assistance consists of providing the client with better estimates of the short term throughput in a wireless network, so that DASH streaming sessions can better adapt to network conditions and avoid buffer under-run, hence stalling of audio/video playback. MPEG SAND provides a core element to facilitate the envisaged network assistance functionality.

DASH clients typically perform rate adaptation based on their buffer fullness level, available representation rates and estimates of short-term future throughput. In a wireless network the throughput typically varies rapidly, while the client adaptation occurs relatively more slowly, leading to an estimation by the client that may carry an error margin. Accumulated error margins, and/or significant individual estimation errors can lead to buffer under-run and stalling of audio/video content playback during re-buffering.

Essentially there are two parts to the network assistance feature:

1) recommendation of the most appropriate bitrate version of content among those available, according to current and anticipated network conditions, and

2) temporary boost of throughput to the client due to anticipated buffer under-run, hence playback stalling, in an ongoing streaming session.

2.3.3 DASH clients collaboration within the home network

In this use case, a home GW is capable to serve multiple DASH players in the house, using HTTP interface over possibly over WLAN, or other Ethernet connection in the home.

While homes have a limited internet access bandwidth (depending on the plans purchased from the internet access provider), there are more and more DASH clients (phones, tablets, smart TVs, computers …) being used simultaneously in the home and competing over the same bandwidth. Using SAND it becomes possible to have a fair sharing of bandwidth between DASH clients or even set priorities among DASH clients so as to make sure premium services can always be offered with appropriate quality.

2.4 Distribution Overlay and QoE Measurement

End to end content providers use, simultaneously, multiple CDNs, private and public peering arrangements to deliver their content using Internet to the end consumers. The choice via which route the content is send to the player is based, from a technical perspective, mainly on latency / throughput of the network.

This information is available either via monitoring software or via the APIs of the CDNs. The limitation of this approach is the (subjective) info is only available from parts of the network. A more robust way is when the player reports player state information (buffer under run, experienced latency, packet loss) back to the content provider. In a more pro-active scenario the different routes are tested via this mechanism in order not to have the state of the used network only.

As a demonstration of SAND capabilities for this scenario, the website [5] implements a DASH client reporting the buffer length every seconds to a central DANE. Based on this data collection, statistics over the connected clients are computed and presented back on the website.

3 Overview of MPEG SAND in ISO/IEC 23009-5

MPEG SAND defines message formats and exchange protocols between server, client, edge proxy and network elements toward enhancing streaming Quality of Experience (QoE). MPEG SAND messages describe real time operational characteristics of networks, servers, proxies, edge caches, content delivery networks (CDNs) as well as DASH client's performance. In particular, MPEG SAND addresses the following:

- Unidirectional/bidirectional, point-to-point/multipoint communication between servers/CDNs and DASH clients,
- Mechanisms for providing content-awareness and service-awareness towards the underlying protocol stack including server and/or network assistance,
- QoS and QoE signaling and reporting for DASH-based services, and
- Analytics and monitoring of DASH-based services.

The SAND reference architecture is based on four broad categories of elements: i) DASH streaming clients, ii) Regular Network Elements (RNE), which are DASH content unaware and treat DASH-related video delivery objects as any other object, but are present on the path between origin server and DASH clients, e.g. transparent caches. iii) DASH-Aware Network Elements (DANE), which have at least minimum intelligence about DASH; for instance, they may be aware that the delivered objects are DASH-formatted objects such as the MPD or DASH segments, and may prioritize, parse or even modify such objects, and iv) Metrics server, which are DASH aware and are in charge of gathering metrics from DASH clients.

Based on these elements, the SAND reference architecture is defined as shown in Figure 1. Within this architecture, the following four categories of messages, called SAND messages, are exchanged:

- Parameters Enhancing Delivery (PED) messages that are exchanged between DANEs,
- Parameters Enhancing Reception (PER) messages that are sent from DANEs to DASH clients,
- Status messages that are sent from DASH clients to DANEs,
- Metrics messages that are sent from DASH clients to Metrics servers.

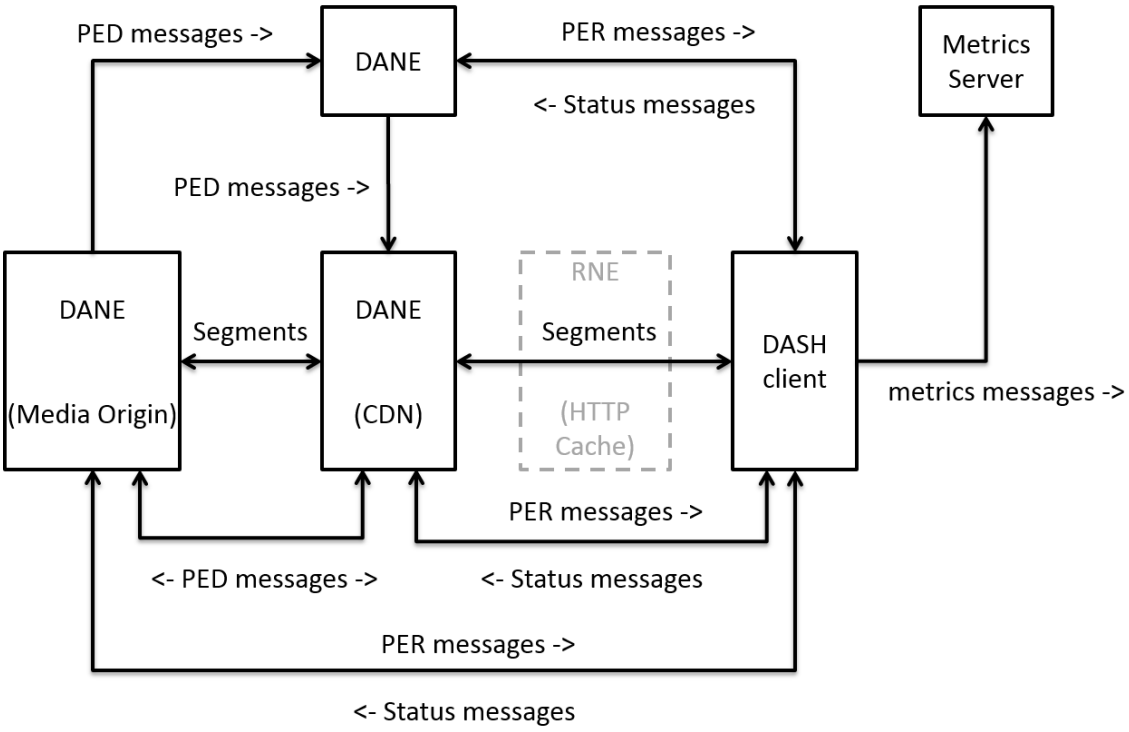

Figure 1 - SAND reference architecture (taken from ISO/IEC 23009-5 [1])

Most of the SAND messages are delivered in Extensible Markup Language (XML) format using HTTP protocols with the detailed syntax of each message defined in the SAND specification [1]. In case of small metric messages, status messages or PED messages, the DASH client may attach the SAND message in a plain text format to the uplink via HTTP header extension in (GET or POST) message.

Using the metrics and status messages, the DASH clients can inform the network (i.e., DANE) about requested bandwidth / quality, anticipated DASH segments, acceptable alternative content, etc. This leads to intelligent caching and real-time media processing at the server or proxy. As defined in the SAND specification [1], Metrics and Status Messages are comprised of the following:

- *QoE metrics from DASH Part 1*, i.e., ISO IEC 23009-1 [2], including average throughput, buffer level, initial playout delay, HTTP request/response transactions, representation switch events, and playlist, as also described in 3GPP TS 26.247 [3]. QoE metrics are beneficial for detecting and debugging failures, managing streaming performance, and allowing for QoE-aware network adaptation and service provisioning useful for the network operator and content/service provider.

- *SharedResourceAllocation*, allows a DASH client to provide information on a set of operating points (such as desired bandwidth and quality) to one or several DANE(s) with an intent to share network resources.

- *AnticipatedRequests*, allows a DASH client to announce to a DANE which specific set of segments it is interested in. The intent is to signal the set of segments in representations that the DASH client is likely to select and request soon.

- *AcceptedAlternatives*, allows DASH clients to inform DANEs on the media delivery path (typically caching DANEs) when they request a given DASH segment that they are willing to accept other DASH segments alternatives.

- *AbsoluteDeadline*, allows DASH clients indicating the DANE the absolute deadline in wall-clock time by when the requested DASH Segment needs to be completely received. As such, further action

can be taken by the network, e.g., the DANE can pre-fetch content to ensure the timely delivery to the client.

- *MaxRTT*, allows DASH clients indicating the DANE the maximum round trip time (RTT) of the request from the time when the request was issued until the request needs to be completely available at the DASH client.

- *NextAlternatives*, allows DASH clients to inform a DANE about which alternatives they are willing to accept for the request of the next segment.

- *ClientCapabilities*, allows DASH clients to share their SAND capabilities, i.e., the set of SAND messages they support, with the DANE.

Using the PER Messages, the DANE can inform the client about cached segments, alternative segment availability, timing information for delivery, network throughput/QoS, etc., which leads to intelligent DASH client adaptation behavior. As defined in the SAND specification [1], the PER Messages are comprised of the following:

- *ResourceStatus*, allows for a DANE to inform a DASH client – in advance – about knowledge of segment availability including the caching status of the segment(s) in the DANE. The DASH client adaptation can take advantage of this information and potentially prefer accessing the content cached at the edge due to faster download times.

- *DaneResourceStatus*, allows DANEs to signal the available and possibly anticipated to be available data structures to the DASH client and also signal which data structures are unavailable. This method is complementary to the ResourceStatus message mentioned above as it allows to express the available segments at the time of the status message.

- *SharedResourceAssignment*, allows the DANE to send to DASH clients competing for bandwidth over the same network information about how much bandwidth they should use in order to stay in a fair sharing of the total bandwidth. This message is usually send to DASH clients as a response to a SharedResourceAllocation message and is usually sent by a DANE who acts as a resource allocation entity.

- *MPDValidityEndTime*, provides the ability to signal to the client that a given MPD, whose @type is set to 'dynamic' and @minimumUpdatePeriod is present, can only be used up to at a certain wall-clock time.

- *Throughput*, allows a DASH client to have – in advance – knowledge of the throughput characteristics and the guarantees along with this from the DANE to the DASH client.

- *AvailabilityTimeOffset*, allows a DASH client to have – in advance – knowledge of the availability time offset from the DANE to the DASH client. The status may be different for different baseURLs or different Representation IDs used, allowing to signal availability time offset dependent on the network delivering it.

- *QoSInformation*, signals to a DASH client about the available QoS information, including parameters such as guaranteed bitrate (GBR), maximum bitrate (MBR), delay and packet loss rate. A DASH client can take the available network QoS information into consideration when requesting media segments such that the consumed content bandwidth remains within the limits established by the signaled QoS information.

- *DeliveredAlternatives* serves as a response to an AcceptedAlternatives message sent by a DASH client, where a DANE may deliver an alternative segment rather than the requested segment. If so, the DANE also sends a DeliveredAlternatives message to the DASH client to inform him that the response contains a segment alternative and not the requested segment.

- *DANECapabilities*, allows DANEs to share their SAND capabilities, i.e., the set of SAND messages they support, with the DASH clients.

SAND as defined in ISO/IEC 23009-5 [1] mandates HTTP as the minimum transport protocol to be supported by SAND-enabled elements. In addition, SAND also defines an optional transport protocol based on WebSocket [4]. Nevertheless, it does not preclude that other additional transport protocols could also be implemented.

The use of HTTP as a minimum transport protocol to implement is defined in [1] for:

a) Metrics messages (from DASH client to DANE)

b) Status messages (from DASH client to DANE)

c) PER messages (from DANE to DASH client)

Depending on the nature of SAND messages, the use of HTTP protocol by SAND network elements varies. Table 1 summarizes which HTTP usages is mandatory in [1] for a SAND element (in bold in the table) or may be optional depending on the nature of the SAND message.

Table 1 – Mandatory usages of HTTP for carrying SAND messages (taken from ISO/IEC 23009-5 [1])

Metrics messages	**HTTP POST** HTTP headers may be used for small metrics messages.
Status messages	**HTTP headers**
PER messages	**HTTP GET**

4 Example Workflows for SAND

4.1 Architectural Considerations

Figure 2 depicts the assumed system architecture for the example SAND workflows described in this section:

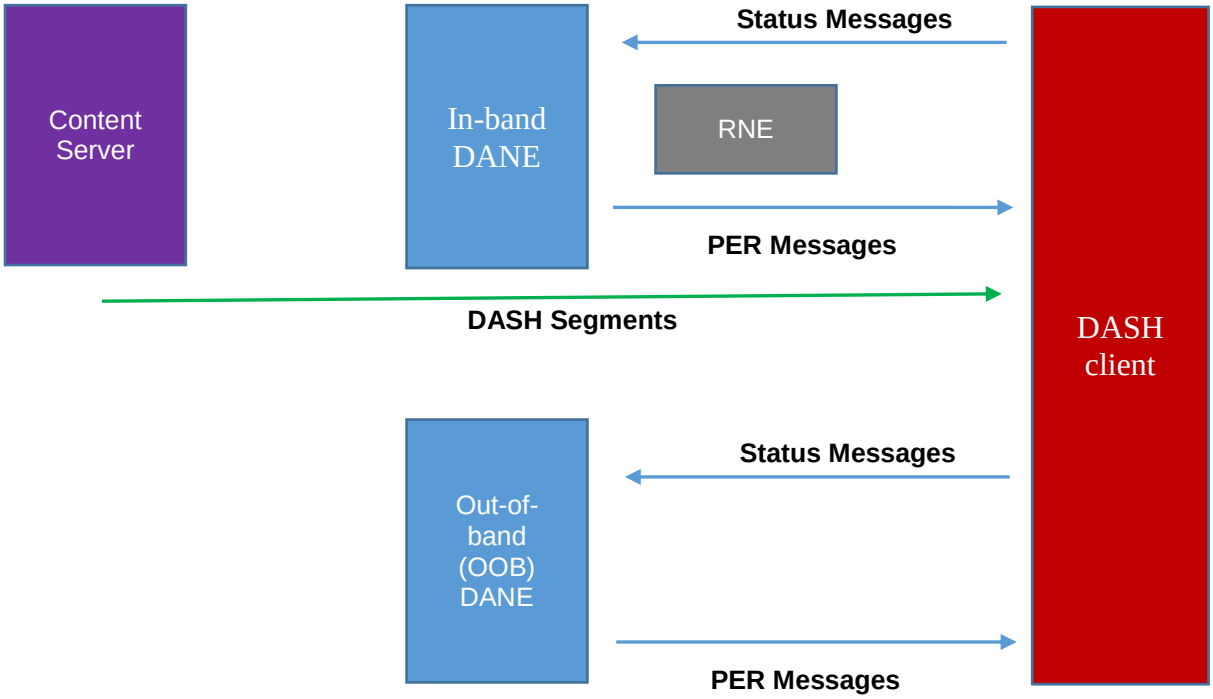

Figure 2 – Assumed system architecture for SAND workflows

As such, the content server is where the BaseURLs in the MPD are hosted, and can be the origin server or a CDN node.

The in-band DANE is considered to be a CDN node in the media path between the content server and DASH client and is where (i) the status messages from the DASH client are processed, and (ii) PER messages to the DASH client are generated. The out-of-band (OOB) DANE is considered to be a CDN node that is not on the media path, but is able to process the status messages from the DASH client and generate PER messages for the DASH client. Unless otherwise mentioned, all DANEs mentioned in the workflows presented in the subsequent sections are in-band DANEs.

If present, the Regular Network Entity (RNE) is assumed to be a proxy cache (regular or transparent) in the media path. As such, RNE receives and forwards HTTP messages from DASH clients and it also can cache media. If RNE has the requested object in cache, it serves it to the DASH client immediately, and it otherwise forwards the request to the DANE. It is assumed that the RNE does not understand the HTTP headers carrying SAND messages, and that it forwards them without any modification.

4.2 Generic Work Flow for SAND Capability Exchange / Negotiation

4.2.1 General Description

SAND capability exchange refers to the procedures where the DANE and DASH client exchange SAND messages describing their respective capabilities, i.e., the status message on client capabilities, and PER message on DANE capabilities. There are two alternatives for conducting SAND capability exchange / negotiation:

1- **Client-triggered:** In the header of an HTTP request, client includes the SAND header that contains the status message to advertise client capabilities

2- **DANE-triggered:** In an HTTP response, DANE includes SAND header to advertise capabilities with the URI hosted at the DANE for the corresponding PER message on DANE capabilities

4.2.2 Work Flow for Client-triggered SAND Capability Exchange / Negotiation

Figure 3 depicts the workflow for client-triggered SAND capability exchange / negotiation.

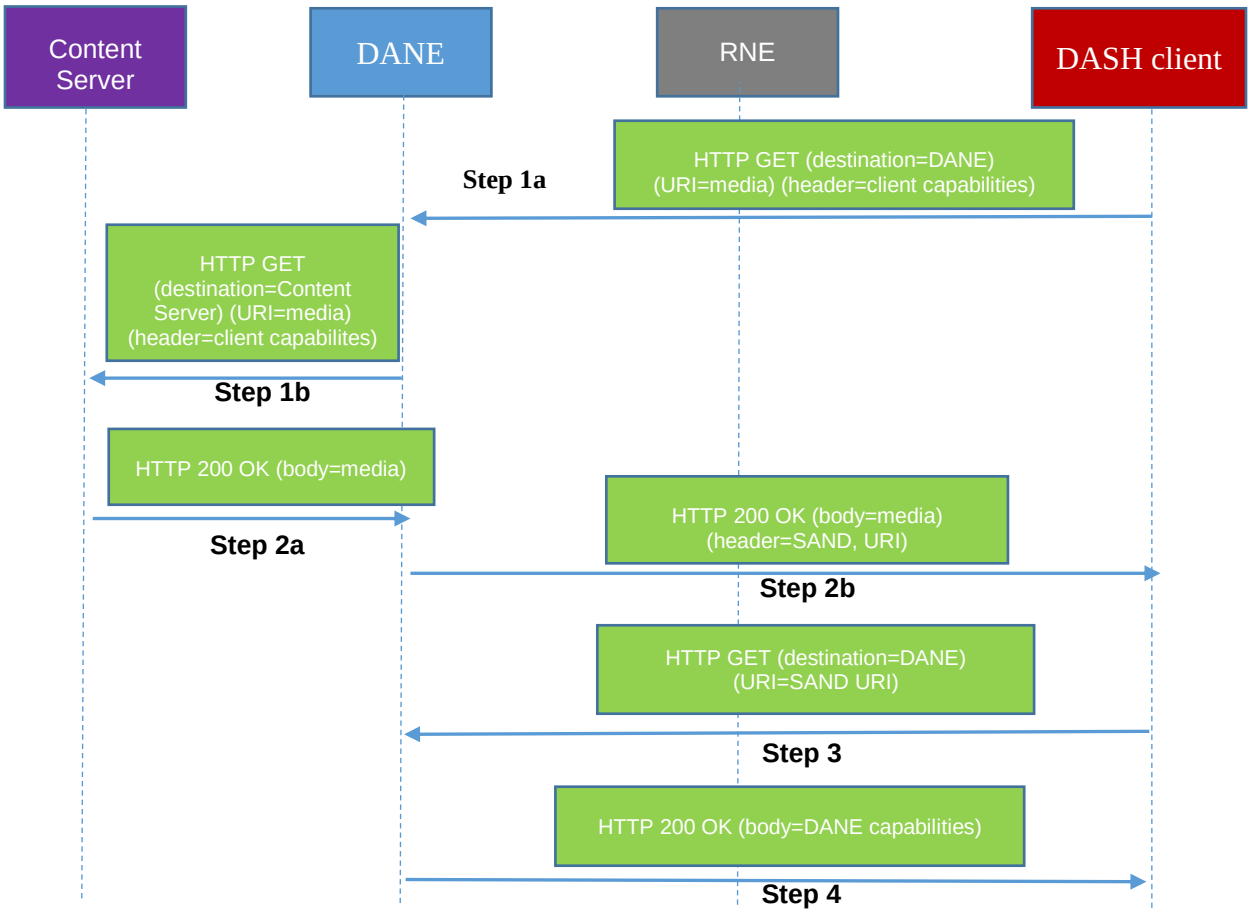

Figure 3 - Workflow for client-triggered SAND capability exchange / negotiation.

Step 1a: Client issues an HTTP GET and sends request for media to the DANE (CDN node). In the header of the HTTP request, client includes the SAND header that contains the status messages on client capabilities. Any Regular Network Entity (RNE) in the path do not have the object in cache, so all HTTP transactions are forwarded to the DANE.

Step 1b: DANE processes the SAND header that contains the status messages on client capabilities. DANE forwards the HTTP headers carrying SAND messages to without any modification. Since DANE does not have the desired media object in cache, DANE issues an HTTP GET and sends request for media to the content server.

Step 2a: Content server responds with HTTP 200 OK with body containing media.

Step 2b: In the HTTP response, DANE includes SAND header to advertise availability of PER message on DANE capabilities with the URI hosted at the DANE for the corresponding PER message.

Step 3: Client issues an HTTP GET request targeting the URI hosted at the DANE to fetch the PER message on DANE capabilities.

Step 4: DANE responds with the HTTP OK with body containing the PER message on DANE capabilities.

Note 1: When the RNE finds the object in the cache, the RNE serves immediately.

Note 2: While the workflows in this specification consider DANEs to forward SAND messages without modification, there is no recommendation or guideline suggested herein regarding DANE forwarding behavior for SAND messages. The workflows presented in this specification are still valid even if DANEs were to modify or remove SAND messages.

4.2.3 Work Flow for Client-triggered SAND Capability Exchange / Negotiation with Out-of-Band (OOB) DANE

This case relies on the assumption that the out-of-band (OOB) DANE is placed on the MPD delivery path, which is assumed to be different from the delivery path of the media segments. Figure 4 depicts the workflow for client-triggered SAND capability exchange / negotiation with OOB DANE.

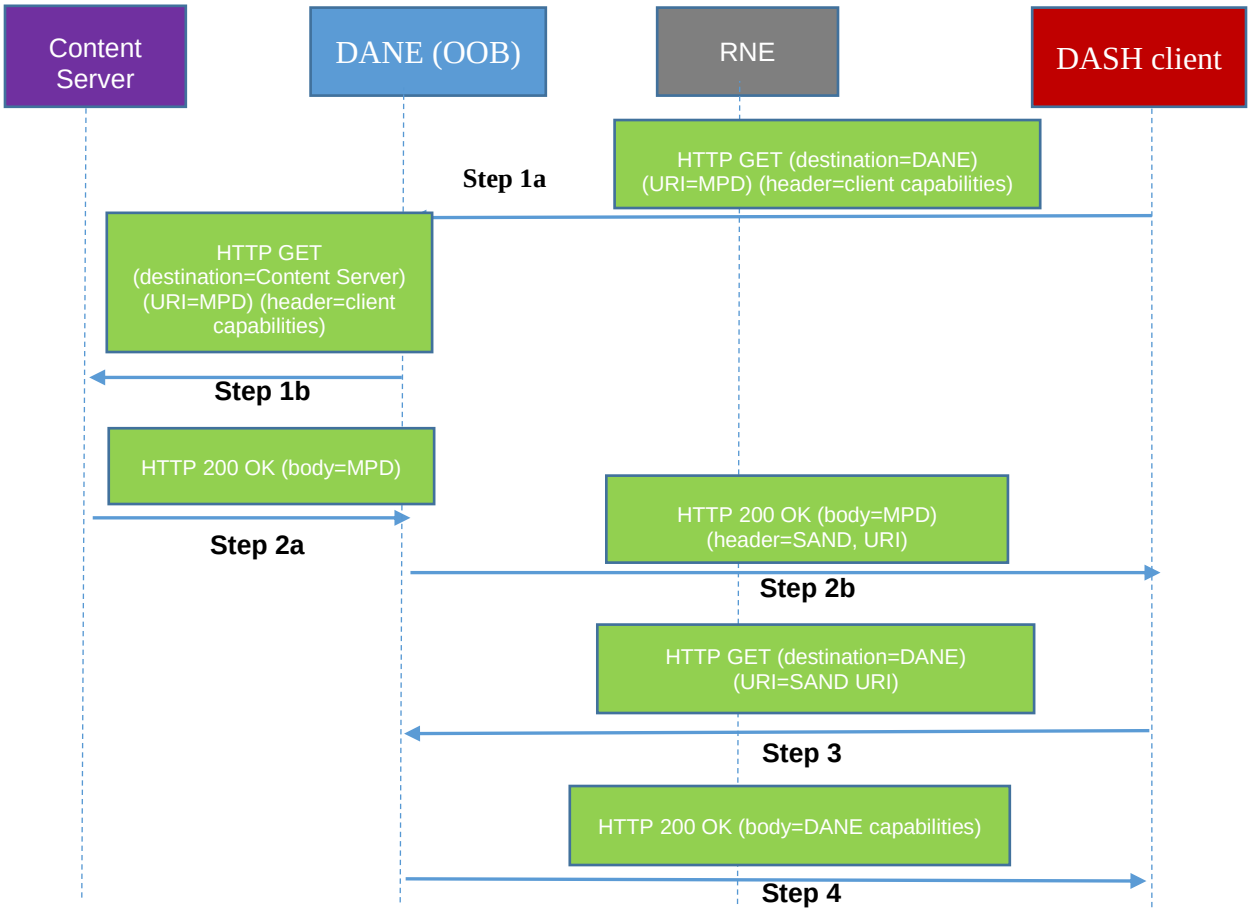

Figure 4 - Workflow for client-triggered SAND capability exchange / negotiation with OOB DANE.

Step 1a: Client issues an HTTP GET and sends request for MPD to the OOB DANE (CDN node). In the header of the HTTP request, client includes the SAND header that contains the status messages on client capabilities. Any Regular Network Entity (RNE) in the path do not have the object in cache, so all HTTP transactions are forwarded to the OOB DANE.

Step 1b: The OOB DANE processes the SAND header that contains the status messages on client capabilities. The OOB DANE forwards the HTTP headers carrying SAND messages to without any modification. Since the OOB DANE does not have the desired MPD object in cache, OOB DANE issues an HTTP GET and sends request for MPD to the content server.

Step 2a: Content server responds with HTTP 200 OK with body containing the MPD.

Step 2b: In the HTTP response, the OOB DANE includes SAND header to advertise availability of PER message on DANE capabilities with the URI hosted at the OOB DANE for the corresponding PER message.

Step 3: Client issues an HTTP GET request targeting the URI hosted at the OOB DANE to fetch the PER message on DANE capabilities.

Step 4: The OOB DANE responds with the HTTP OK with body containing the PER message on DANE capabilities.

4.2.4 Work Flow for Client-triggered SAND Capability Exchange / Negotiation with Out-of-Band (OOB) DANE using WebSocket advertised via MPD

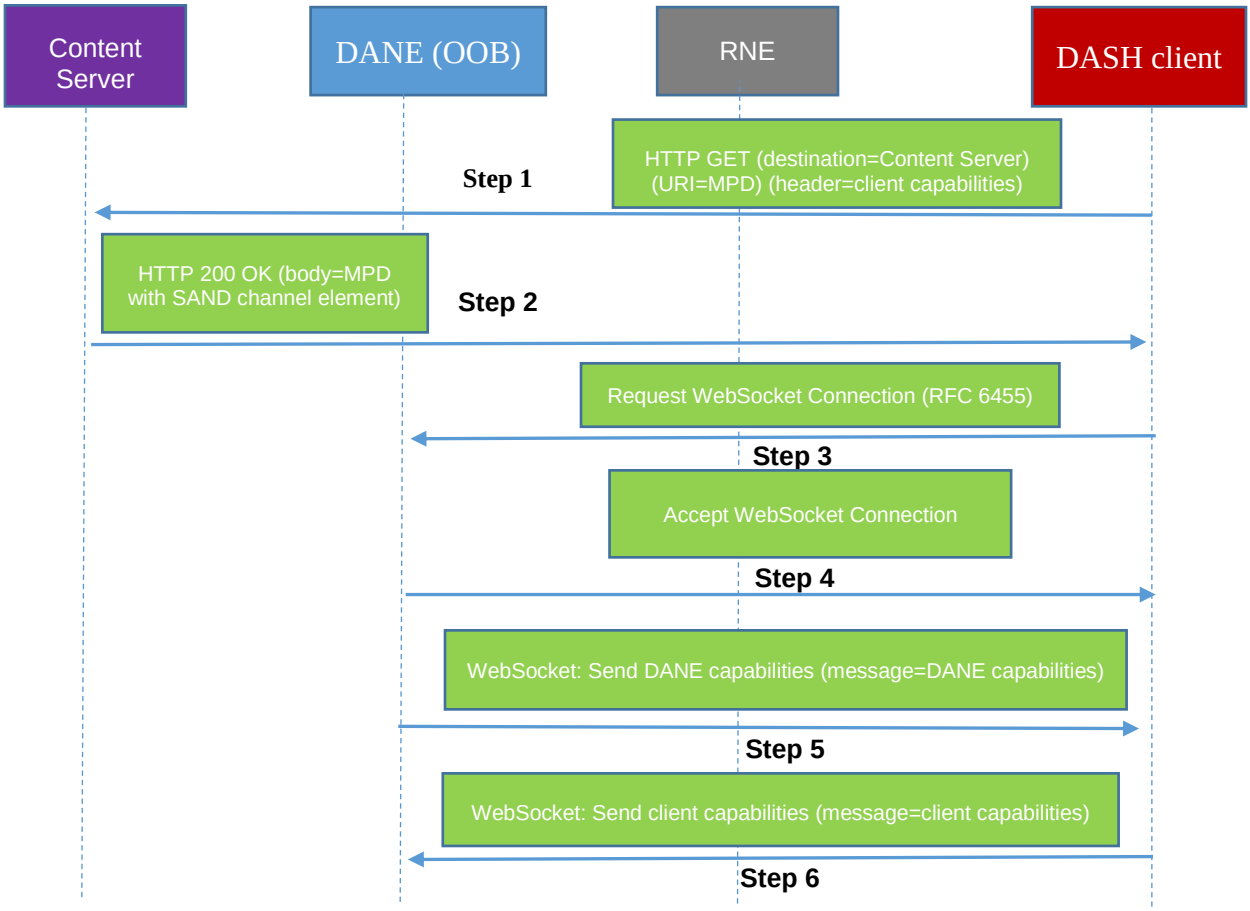

Figure 5 - Workflow for client-triggered SAND capability exchange / negotiation using WebSocket.

Figure 5 depicts the workflow for client-triggered SAND capability exchange / negotiation with OOB DANE using WebSocket advertised via MPD.

Step 1: Client issues an HTTP GET and sends request for MPD to the content server. In the header of the HTTP request, client includes the SAND header that contains the status messages on client capabilities.

Step 2: Content server responds with HTTP 200 OK with body containing the MPD. As specified in ISO/IEC 23009-5 [1], the MPD contains a sand:Channel element whose @schemeIdUri is "urn:mpeg:dash:sand:channel:websocket:2016" and WebSocket URI in the @endpoint attribute.

Steps 3, 4: The DASH Client parses the MPD starts downloading the segments. In addition, the **sand:Channel** element is located in the **MPD** element. Using this information, the DASH client initiates the WebSocket connection with the OOB DANE as specified in RFC 6455 [4].

Steps 5, 6: Upon successful establishment of the WebSocket connection between a DASH Client and OOB DANE, the DASH client starts listening for incoming PER messages and may send metrics and status messages when needed. Since the WebSocket Protocol establishes a full-duplex connection, the OOB DANE and the DASH client may exchange SAND messages travelling simultaneously in opposite directions over the channel.

The DANE sends the DANE capabilities message to the DASH Client (Step 5). When received, the DASH Client replies with a client capabilities message (Step 6). As specified by ISO/IEC 23009-5 [1], the messages are WebSocket messages sent in the text format and formatted in XML.

When the capabilities messages have been exchanged, the WebSocket connection stays open and any other SAND message may be exchanged.

4.3 DASH Operation with Proxy Caches

4.3.1 SAND Features

The relevant SAND messages to fulfill the proxy caching use cases in Section 2.1 are the following:

- PER:
  - ResourceStatus
  - DaneResourceStatus
  - DeliveredAlternatives
- Status Messages:
  - AnticipatedRequests,
  - AcceptedAlternatives
  - NextAlternatives,

4.3.2 Work Flow for Proxy Caching (on-the-fly caching)

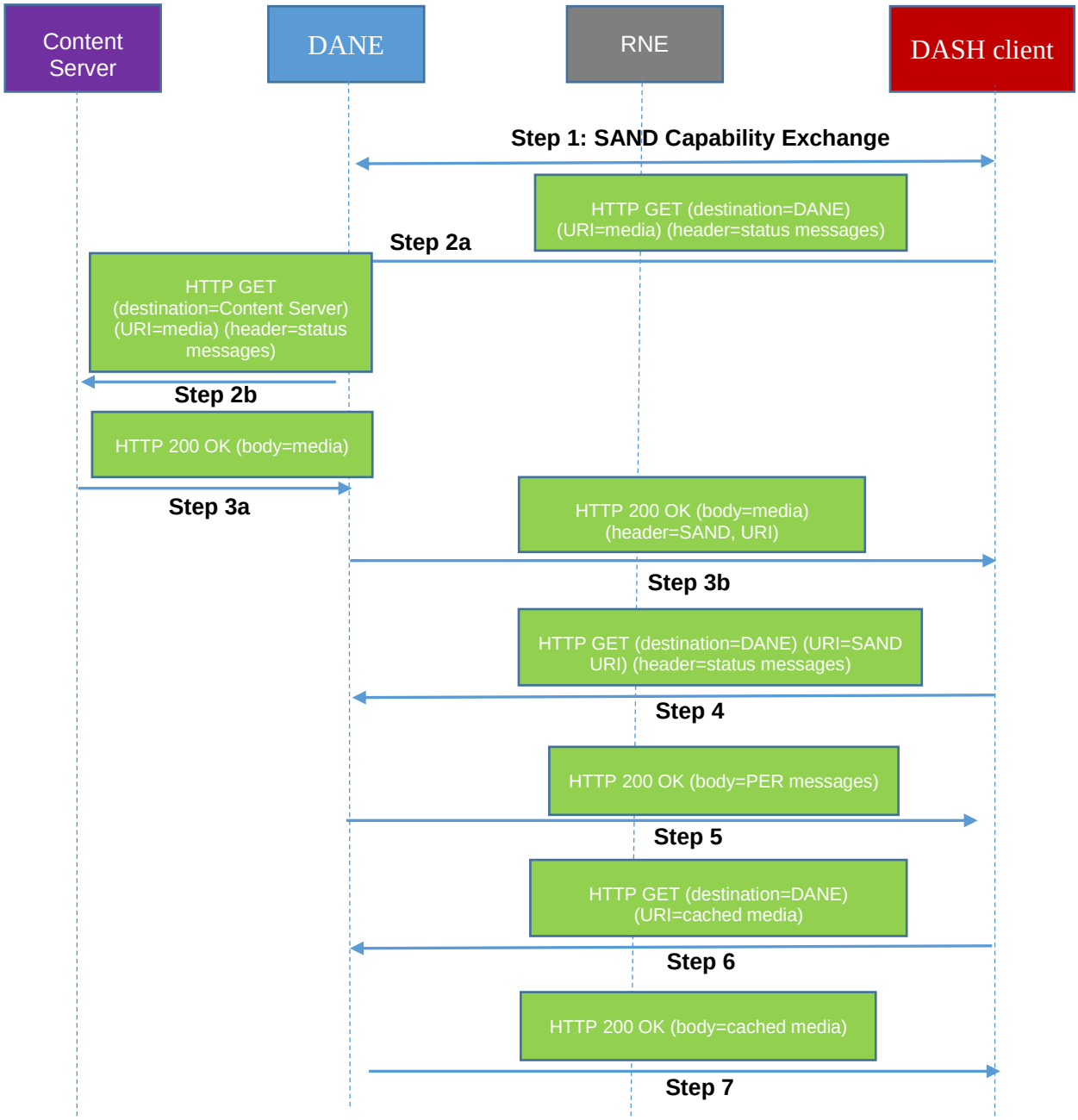

Figure 6 - Workflow for Proxy Caching for the case of on-the-fly caching.

This workflow is depicted in Figure 6 and addresses the case of on-the-fly caching, where the DANE caches content based on SAND-based status messages received from the client.

Step 1: For the use cases on proxy caching, the SAND capability exchange between the DANE and client will negotiate the use of the related SAND messages for proxy caching, namely the PER and status messages in Section 4.3.1.

Step 2a: Client issues an HTTP GET and sends request for media to the DANE (CDN node). In the header of the HTTP request, client includes the SAND header that contains the status messages on proxy caching, namely on anticipated requests, accepted alternatives and/or next alternatives. DANE receives these status messages, processes them and then forwards the SAND header that contains the status messages.

Step 2b: The DANE forwards the HTTP request for the desired media to the content server, since the DANE does not have a cached version of the media. DANE forwards the HTTP headers carrying SAND messages to without any modification.

Step 3a: Content server responds with HTTP 200 OK with body containing media

Step 3b: In the HTTP response, DANE includes SAND header to advertise availability of PER messages on proxy caching with the URI hosted at the DANE for the corresponding PER messages, namely on resource status, DANE resource status and/or delivered alternatives.

Step 4: Client issues an HTTP GET request targeting the URI hosted at the DANE to fetch the PER messages on proxy caching, namely on resource status, DANE resource status and/or delivered alternatives. In the header of the HTTP request, client may include the SAND header that contains further status messages on proxy caching, namely on anticipated requests, accepted alternatives and/or next alternatives.

Step 5: DANE responds with the HTTP OK with body containing the PER message on proxy caching, namely on resource status, DANE resource status and/or delivered alternatives.

Steps 6,7: Client requests and downloads cached media from DANE.

4.3.3 Work Flow for Proxy Caching (advanced caching case I)

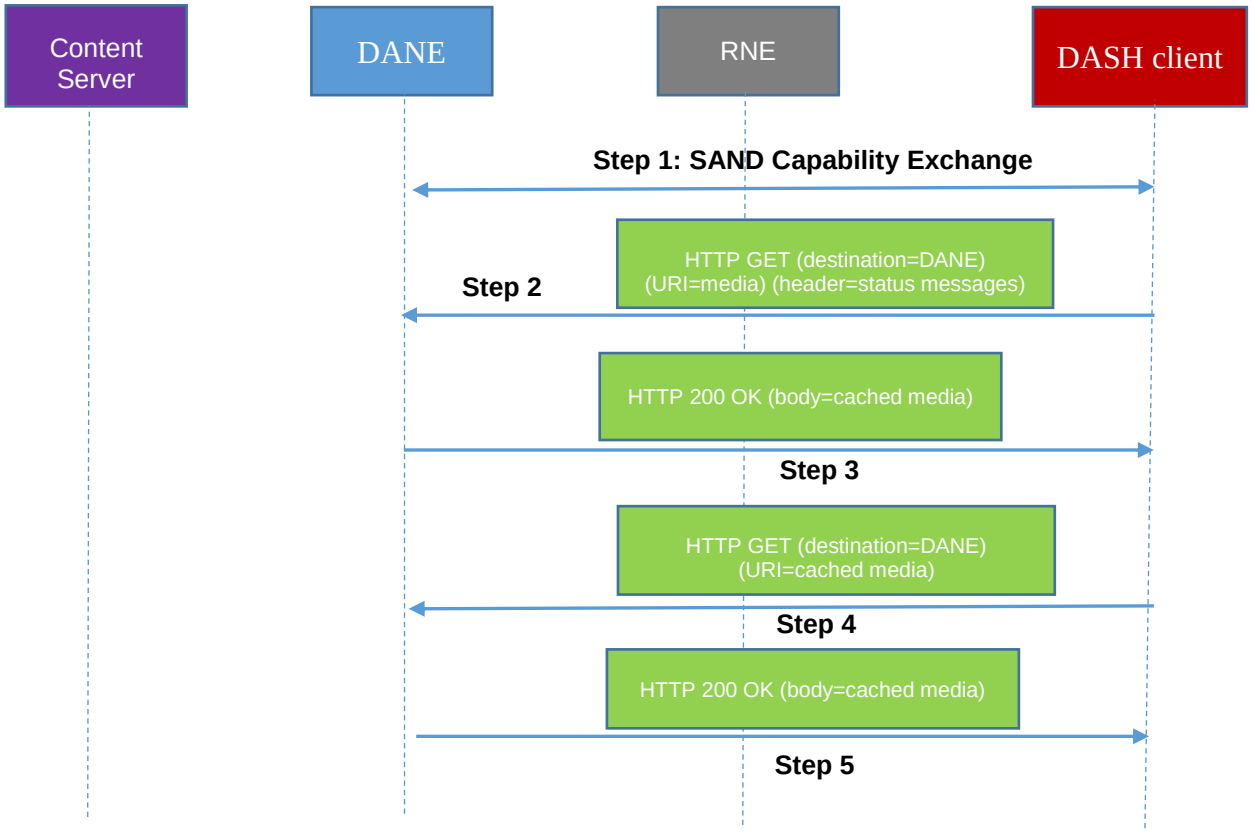

Figure 7 - Workflow for Proxy Caching for the case of advanced caching.

This workflow, depicted in Figure 7, addresses the case of advanced caching, where the DANE already has content cached and responds the media requests with the cached content as soon as it identifies the relevance of its cached content for the DASH client after processing the SAND-based status messages.

Step 1: For the use cases on proxy caching, the SAND capability exchange between the DANE and client will negotiate the use of the related SAND messages for proxy caching, namely the PER and status messages in Section 4.3.1.
Step 2: Client issues an HTTP GET and sends request for media to the DANE (CDN node). In the header of the HTTP request, client includes the SAND header that contains the status messages on proxy caching, namely on anticipated requests, accepted alternatives and/or next alternatives. DANE receives these status messages.
Step 3: As DANE has already cached the representations desired by the DASH client, DANE issues an HTTP response directly and sends cached media to the client.
Step 4, 5: Client continues to request and downloads cached media from DANE.

4.3.4 Work Flow for Proxy Caching (advanced caching case II)

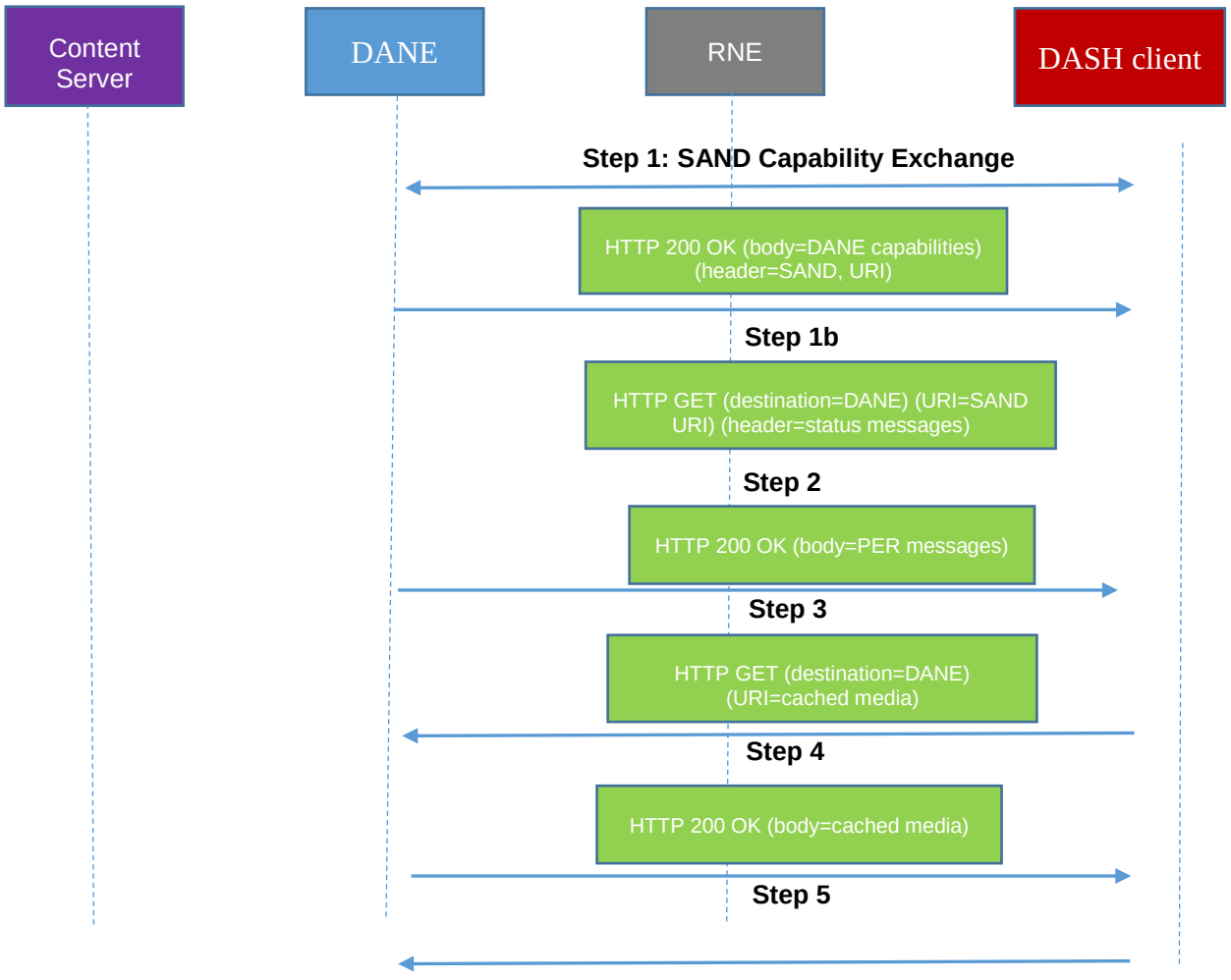

Figure 8 – Another workflow for Proxy Caching for the case of advanced caching.

This workflow, depicted in Figure 8, addresses the case of advanced caching, where the DANE already has content cached and advertises their availability using SAND-based PER messages as early as during the SAND capability exchange procedures.

Step 1: For the use cases on proxy caching, the SAND capability exchange between the DANE and client will negotiate the use of the related SAND messages for proxy caching, namely the PER and status messages in Section 4.3.1.

Step 1b: When DANE responds with the HTTP OK with body containing the PER message on DANE capabilities, DANE also includes SAND header to advertise availability of PER messages on proxy caching with the URI hosted at the DANE for the corresponding PER messages, namely on resource status, DANE resource status and/or delivered alternatives.
Step 2: Client issues an HTTP GET request targeting the URI hosted at the DANE to fetch the PER messages on proxy caching, namely on resource status, DANE resource status and/or delivered alternatives. In the header of the HTTP request, client may include the SAND header that contains status messages on proxy caching, namely on anticipated requests, accepted alternatives and/or next alternatives.
Step 3: DANE responds with the HTTP OK with body containing the PER message on proxy caching, namely on resource status, DANE resource status and/or delivered alternatives.
Steps 4,5: Client requests and downloads cached media from DANE.

4.4 Consistent QoE/QoS for DASH users

4.4.1 SAND Features

The relevant SAND messages to fulfill the consistent QoE/QoS use cases in Section 2.2 are the following:

- PER:
  - SharedResourceAssignment
  - Throughput
  - QoSInformation
- Metrics:
  - BufferLevel
  - PlayList
- Status Messages:
  - SharedResourceAllocation

4.4.2 Work Flow for Consistent QoE/QoS

Figure 9 depicts the example SAND workflow for fulfilling consistent QoE/QoS use cases in Section 2.2.

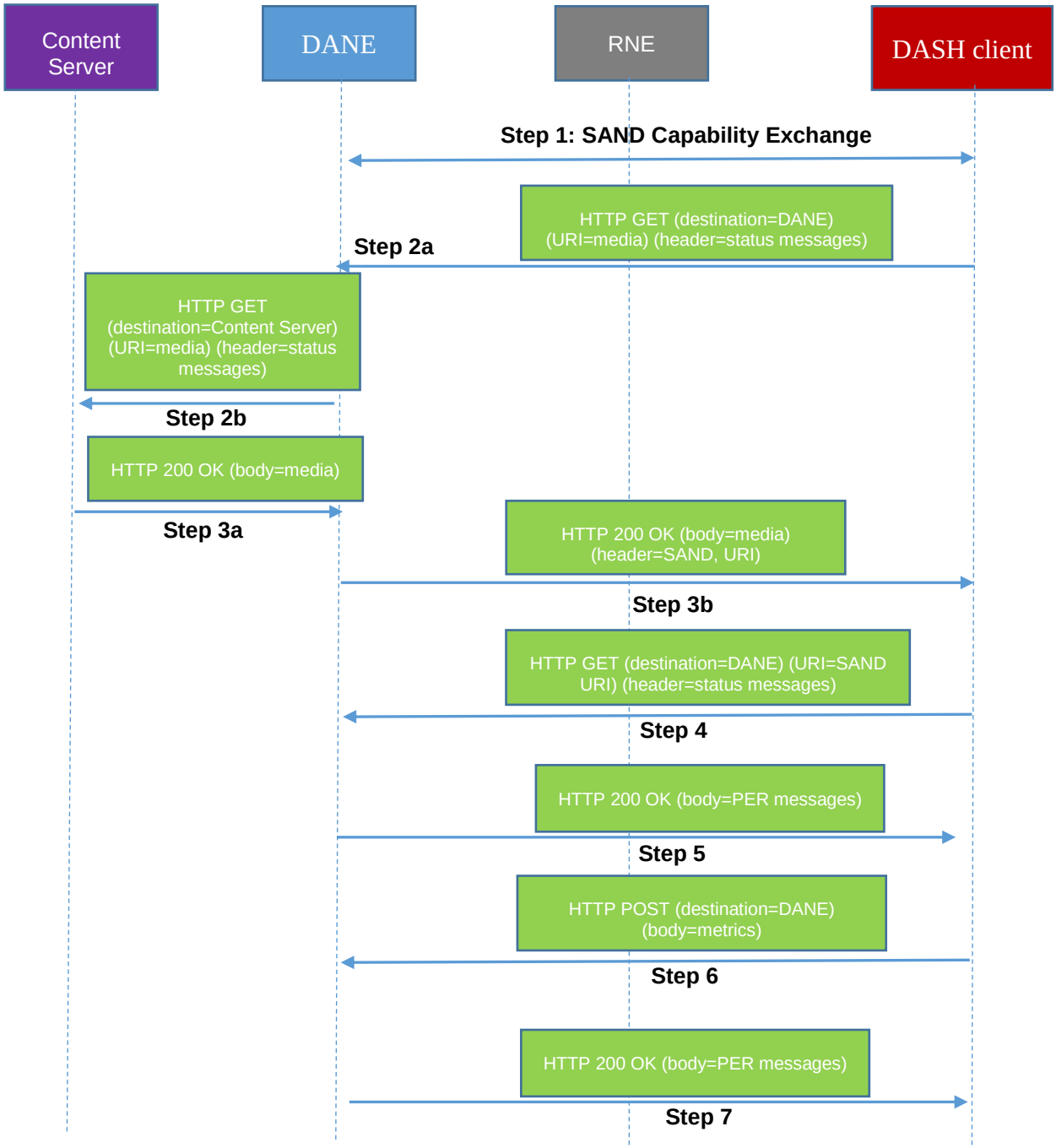

Figure 9 – Example workflow for Consistent QoE/QoS

Step 1: The SAND capability exchange between the DANE and client will negotiate the use of the related SAND messages for consistent QoE/QoS, namely the PER and status messages in Section 4.4.1.

Step 2a: Client issues an HTTP GET and sends request for media to the DANE (CDN node). In the header of the HTTP request, client includes the SAND header that contains the status message on consistent QoS/QoE, namely on shared resource allocation. DANE receives these status messages, processes them and then forwards the SAND header that contains the status messages.

Step 2b: The DANE forwards the HTTP request for the desired media to the content server, since the DANE does not have a cached version of the media. DANE forwards the HTTP headers carrying SAND messages to without any modification.

Step 3a: Content server responds with HTTP 200 OK with body containing media

Step 3b: In the HTTP response, DANE includes SAND header to advertise availability of PER messages on consistent QoS/QoE with the URI hosted at the DANE for the corresponding PER messages, namely on shared resource assignment, QoS information and throughput.

Step 4: Client issues an HTTP GET request targeting the URI hosted at the DANE to fetch the PER messages on consistent QoS/QoE, namely on shared resource assignment, QoS information and throughput. In the header of the HTTP request, client may include the SAND header that contains further status messages on consistent QoS/QoE, namely on shared resource allocation.

Step 5: DANE responds with the HTTP OK with body containing the PER message on consistent QoS/QoE, namely on shared resource assignment, QoS information and throughput.

Step 6: Client optionally sends QoE metrics on buffer level and playlist (it is assumed that the QoE reporting is already triggered via the MPD and DANE is designated as the QoE reporting server). In this workflow, it is assumed that DANE and QoE metrics server are co-located and therefore the server hosting the DANE also receives QoE metrics reports.

Step 7: After collecting QoE metrics from the clients, the DANE may further update shared resource assignment, QoS information and throughput and issue new PER messages to update the DASH clients.

5 References

[1] ISO/IEC FDIS 23009-5:2016: "Information Technology — Dynamic adaptive streaming over HTTP (DASH) — Part 5: Server and network assisted DASH (SAND)"

[2] ISO/IEC 23009-1:2014: " Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats".

[3] 3GPP TS 26.247: “3GPP TS 26.247: "Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH)".

[4] IETF RFC 6455: “The WebSocket Protocol”.

[5] "SAND Real-time metrics reporting via WebSocket", TNO Medialab, http://tnomedialab.github.io/sand/

6 Abbreviations

3GPP	Third Generation Partnership Project
API	Application Program Interface
CDN	Content Delivery Network
DANE	DASH-Aware Network Element
DASH	Dynamic Adaptive Streaming over HTTP
FDIS	Final Draft International Standard
FFS	For Further Study
GBR	Guaranteed Bitrate
GW	Gateway
HD	High Definition

HTTP	Hypertext Transfer Protocol
MBR	Maximum Bitrate
MBMS	Multimedia Broadcast and Multicast Service
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
OOB	Out Of Band
OTT	Over-The-Top
PED	Parameters Enhancing Delivery
PER	Parameters Enhancing Reception
QoE	Quality of Experience
QoS	Quality of Service
RNE	Regular Network Element
RTT	Round Trip Time
SAND	Server and Network Assisted DASH
TCP	Transmission Control Protocol
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
WLAN	Wireless Local Area Network
XML	Extensible Markup Language