OMAF4CLOUD: STANDARDS-ENABLED 360° VIDEO CREATION AS A SERVICE

Yu You, Ari Hourunranta, Emre Aksu

Nokia Technologies, Finland

ABSTRACT

Omnidirectional MediA Format (OMAF) is a media standard for 360° media content developed by the Moving Picture Experts Group (MPEG). More complex and tailored multimedia services are needed for advanced media processing and delivery such as virtual reality content stitching, packaging, and adaptive streaming. To achieve the desired result, these complex workflows require many advanced functionalities to work together on the media content. In order to address the needs of advanced services, MPEG is also developing a new standard called Network based Media Processing (NBMP), a standard that aims at increased media processing efficiency, faster and lower cost deployment of interoperable media processing functions and the ability to provide large scale deployment by leveraging the public, private or hybrid cloud services. This paper covers both OMAF and NBMP standards. Additionally, an end-to-end design and proof of concept is provided to enable an immersive virtual reality experience to the end users.

INTRODUCTION

Omnidirectional MediA Format (OMAF) [1] is a systems standard developed by MPEG. OMAF defines a media format for omnidirectional content with three degrees of freedom (3DOF) such as 360° video, images, audio and timed text. OMAF also supports viewport-dependent streaming, where a user's viewport is transmitted with higher picture quality than the remaining areas of the viewing sphere. Moreover, the next version of the standard is under development, enabling standardized features for multiple viewpoints and media overlays.

Converting viewport-agnostic 360° video to OMAF compliant content requires file format and transport protocol level modifications only (e.g. fragmented MP4 and DASH based streaming). However, adaptive bit-rate (ABR) is needed for real-world use cases, requiring several encoded versions of the video. Furthermore, 360° videos need to be transcoded to enable viewport-dependent operation, which puts certain constraints on the video encoding process. This requires several transcoding instances to run in parallel during OMAF compliant content creation.

Video processing is a computing resource intensive process. Traditionally content providers used dedicated in-house hardware to transcode their content. Such an approach may introduce high capital expenditure related costs and provide limited scalability.

Network-based solutions are more scalable in terms of the computing resources and provide remote access to users, for instance, over the web, as well as programmable APIs for various integration needs.

NETWORK-BASED MEDIA PROCESSING

Multi-Access Edge Computing (MEC) [2] and 5G enables more programmability and flexibility for the development of new service platforms. Microservices and serverless architectures use container-based deployments and enable services to be deployed easily and quickly in the cloud, with the help of flexible software defined networking (SDN). In the domain of media processing, dynamic or on-demand computing capacity is required in such architectures. In the meantime, media processing continues to evolve to address ever more complex tasks and services, ranging from image upscaling to real-time augmented reality, immersive virtual reality use cases such as VR content stitching, pre-rendering or point cloud aggregation. Such processing pipelines commonly involve media processing with advanced algorithms for on-the-fly media conversion or content composition in the network. In the meantime, as an evolution of cloud computing, MEC enables hosting of independent entities from centralized data centers down to the network edge, closer to consumers with reduced latency and high bandwidth requirements for live or real-time media processing.

Multimedia service providers and network/cloud service providers work together to offer customized immersive services to their customers. Unfortunately, this approach is being hampered by fragmentation. Multimedia service providers are faced with the challenge of adapting their services to multiple cloud and network service providers to reach their customers. These cloud and network service providers often define their own APIs to assign computing resources to their customers.

The Network-based Media Processing (NBMP) standard has been under development as ISO/IEC 23090 Part 8 [3]. It is developed to address problems like fragmentation and to offer a unified standard way to perform media processing on top of any cloud platform. NBMP defines interfaces, as well as media and metadata formats to facilitate instantiating any type of media processing. It will be complementary, providing guidance to current Edge and Cloud computing architectures. Example use cases for the NBMP format include ingest and egress of timed meta data and auxiliary information towards media processing functions, enabling complex media processing such as spatial content stitching for virtual reality, or temporal content composition for customized media presentations. Another functionality for NBMP evolves around network media processing-based delivery which reduces network redundancy in the delivery phase by on-the-fly conversion in the network. This is useful for Augmented Reality (AR) where computer graphics and natural content are combined, or for user centric presentations where broadcast and

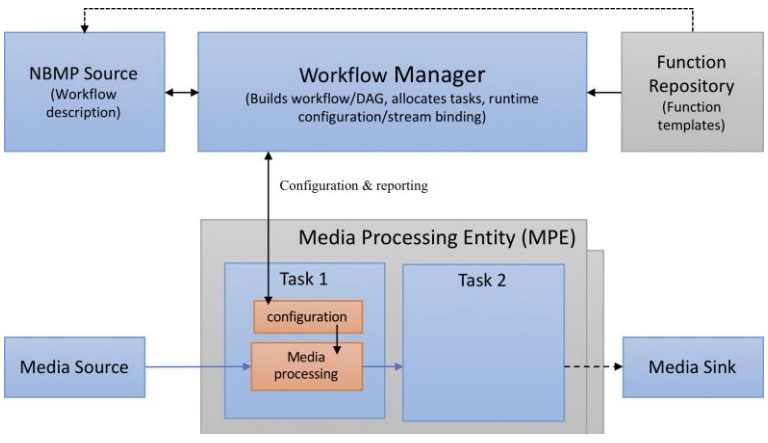

Figure 1 NBMP reference architecture

user-related contents are composed in a single presentation. In addition, the immersive virtual reality media and virtual reality use cases for content stitching, pre-rendering and point cloud aggregation in the cloud are within scope.

Figure 1 shows the NBMP architecture with the annotations for the specific use case. To distribute the processing functions efficiently and improve the interoperability and portability in different cloud solutions, NBMP defines standard APIs and formats such as Function templates and Workflow Description Document (WDD) consisting of a number of logic descriptors. A minimal set of RESTful APIs are defined as the control plane to the workflow manager to build the processing workflows and link function instances, that is, the tasks together. The manager is the central management entity that provides the actual data plane finicalities and routes the data traffic to tasks, guided by the input and output ports defined by the standard. Functions vary from low-level encoders to full-blown ones like VR stitcher and image upscaler.

There are other workflow or flow-chart languages varying from conceptual business process modelling such as BPMN [4], to language specific ones such as Apache Airflow for Python [5], and down to virtual machine management at the infrastructure level such as Argo for Kubernetes [6] and Mistral for OpenStack [7]. NBMP WDD lies in between the BPMN and Airflow and connects the input and output ports of media processing functions into a directed acyclic graph (DAG). NBMP workflow represents the data flow between functions rather than logic dependencies between functions.

We realized the NBMP workflow by using World Wide Stream (WWS) [8], a stream processing platform developed by Nokia Bell Labs. It compiles the NBMP WDD into optimized task description in the JSON format, with appropriate configurations. Optimized deployment involves system-level concerns, such as deployment and operation resource cost, computing resource or data storage capacity constraints, or non-functional service requirements, such as latency, stream quality and reliability. Thanks to the underlying event brokers and media server for multimedia streams, the actual source streams can be bound to a task (function instance) dynamically at run-time by the workflow manager. To handle different types of streams, a RabbitMQ event broker is used to transport structured streams by Advanced Message Queuing Protocol (AMQP), the standard messaging protocol, among tasks across the execution entities. For opaque media bitstreams, a media server is used to support raw YUV bitstreams between tasks directly or indirectly over the socket connections.

USE CASE: OMAF COMPLIANT VIDEO AND OVERLAY COMPOSITION

Content creation workflow for VR 360° video consists of at least an omnidirectional fish eye camera, a stitcher to combine the camera inputs to equirectangular or cubemap projection format, possibly some video editing and postprocessing steps, one or more video codecs, file format generator and a DASH segmenter. The video format between postprocessing and transmission blocks may be a mezzanine format, which then needs to be decoded before encoding to transmission format.

MPEG-DASH is a streaming standard based on ISO Base Media File Format (ISOBMFF). It is based on media tracks, each encapsulated in randomly accessible ISOBMFF segments. OMAF adds VR-related metadata to ISOBMFF and to DASH manifest, enabling

players to identify 360° video, but also providing enablers to optimize the 360° video experience.

Generally, DASH streaming with adaptive bitrate support requires encoding video at several bitrates, hence typically several video encoder instances are utilized in parallel. This is also the case with OMAF HEVC-based Viewport Independent profile. However, VR video requires high resolutions, for example, 4K or more. It is very likely that the requirements for the encoder may be higher than with traditional video streaming. Furthermore, OMAF HEVC-based Viewport Dependent profile sets further requirements for the video encoding, as it expects to have the foreground (viewport) and background encoded at different qualities and/or resolutions.

Viewport-adaptive operation is a known technique for reducing the needed bandwidth for 360° video, by focusing the bits on the area that the user is watching. However, as the user has the freedom to turn his head and change the active viewport, the system must be able to react quickly to provide different areas of the video in high quality. As there is always some transmission latency, all the areas of 360° video need to be covered with the appropriate video content, but the quality can be lower than for the viewport area.

The OMAF HEVC-based Viewport Dependent profile can be realized in several ways [11]. In this context, we concentrate only on the Region-Wise Mixed Resolution (RWMR) scheme. It is based on encoding video with HEVC Motion Constrained Tile Sets (MCTS), where the video picture is divided into rectangular, independent tiles, enabling the tiles to be used as subpictures, and mixing them flexibly in the decoder. The RWMR scheme allows the tiles outside the current viewport to have a lower resolution than the tiles inside the viewport. This reduces the overall 360° video resolution, which further reduces bandwidth and video decoding capacity requirements. For example, a 4K-capable video decoder can decode a RWMR video that provides an effective 6K resolution for the viewport.

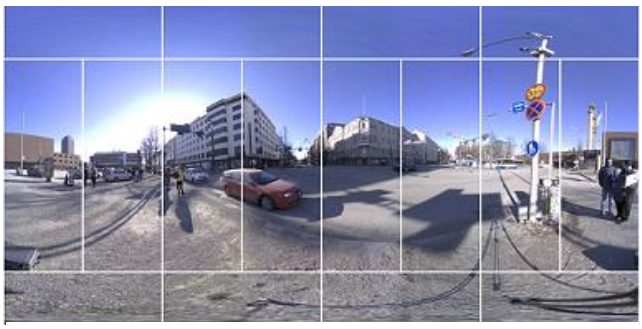

Figure 2 Tile grids for effective 6K scheme

Figure 3 An example tile composition in player

The effective 6K equirectangular scheme, informatively defined in OMAF Annex D.6. [1] and in VR-IF guidelines [10], uses four different video resolutions: 6K and 3K resolutions for the vertically center area, and 3K and 1.5K resolutions for the polar areas. Figure 2 presents tile grids combined in one picture. The pole tiles are always in lower resolution and hence the player can combine the tiles as illustrated in Figure 3 for one viewing direction.

OMAF specifies the use of extractor tracks to assist in merging subpictures into a single MCTS bitstream. Consequently, a player [9] only needs to follow the instructions of an

extractor track to obtain a decodable bitstream from tracks containing MCTSs.

Features under definition for OMAF version 2 include for example overlays and multiple viewpoints. Video overlays can be either 2D or omnidirectional, and be either relative/anchored to the viewing sphere or viewport. They do involve quite a lot of new static metadata, as well as timed metadata tracks. Viewpoints, on the other hand, represent additional omnidirectional cameras in the scene, enabling the player to switch from one viewpoint to another, hence multiplying the processing requirements for 360º video processing.

SYSTEM DESCRIPTION

Our NBMP system prototype is built on top of a generic stream processing platform named World Wide Stream (WWS) [8]. WWS can ingest, process and deliver large numbers of data and media streams in both online and offline modes between geographically distributed sources and sinks. We use WWS as the environment to experiment NBMP functionalities and as a reference implementation.

Figure 4 shows the building blocks of the OMAF use case mapping to current NBMP architecture. Green blocks are OMAF specific components that are end-user facing interfaces. Users interact with the whole system through those 2 components. The rest of the building blocks (blue color) are not directly visible to the end-users.

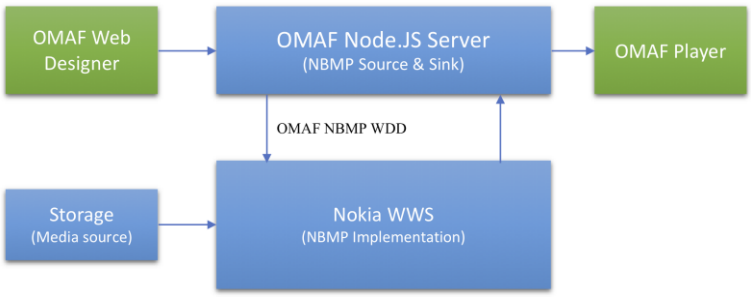

Figure 4 System building blocks

The user starts with the OMAF Web designer, the frontend, and uploads the traditional 360° movie to the Storage via the Node.JS Server, the backend. Followed by the overlay design stage, the user can edit the positions and relative depths of the overlays in an intuitive way. The designer utilizes the Three.JS library [12] for visualization. The actual processing takes place in the NBMP system after the design is finished. The Node.JS server outputs an NBMP workflow description document (WDD) and defines itself as a NBMP Sink where the workflow result is consumed. In addition, we implemented a dashboard UI to receive and monitor the status of workflows in the system, since media processing typically takes a relatively long time to complete. A monitoring interface is very useful to track the status of executed tasks and understand the source of any exceptions, if they occur.

OMAF Workflow Realization

The frontend has a Web user interface covering common video generation operations such as selecting the input content, specifying video transcoding settings such as ABR bitrates and tiling scheme, and adding overlays and viewpoint definitions. The frontend is connected to our Node.JS server via the REST API. Once the user is ready to apply the settings, either for previewing or final processing, the Node.JS server creates the workflow description documents (WDDs) by selecting what processing functions to set up based on

user selections. It also converts the user input to workflows and detailed codec, file format, and/or DASH creation parameters. The generated WDDs are further submitted by the Node.JS server to the NBMP Workflow Manager to start the actual workflow instances. The Node.JS and NBMP systems can reside in different locations, e.g. nodes in the cloud.

Figure 5 shows one typical setup of OMAF version 2 content creation with options to have multiple overlays. It presents a single viewpoint only but the same setup can be applied and duplicated to multiple viewpoints. The setup includes three ABR variants for the high-resolution tiles, and two 3K variants with a different tile setup for center and polar areas. Our design does the HEVC MCTS encoding for full video pictures with a tiling grid, and then splits the output bitstream to subpictures. Transcoding the overlays is not always necessary, but in some cases, it may be beneficial to do so, e.g. if the original video has much larger resolution than required for the overlay. If all the video encoders are run in separate cloud nodes, a decoder and a down-scaler could be coupled with each encoder, to avoid high bandwidth raw data transmissions between nodes. The OMAF Creator [9] takes care of video bitstream post-processing from full pictures to tile-based subpictures, with extractor track generation. It also creates DASH/ISOBMFF segments, inserting OMAF specific metadata and creating timed metadata e.g. for initial viewing direction track and for overlays. It runs as a single instance, since the tracks have file format level associations, e.g. different viewpoints are grouped in file format level. The operations under OMAF Creator's responsibility are much less resource demanding compared to video decoding and encoding, and hence the single creator should not become an issue from a load balancing point of view.

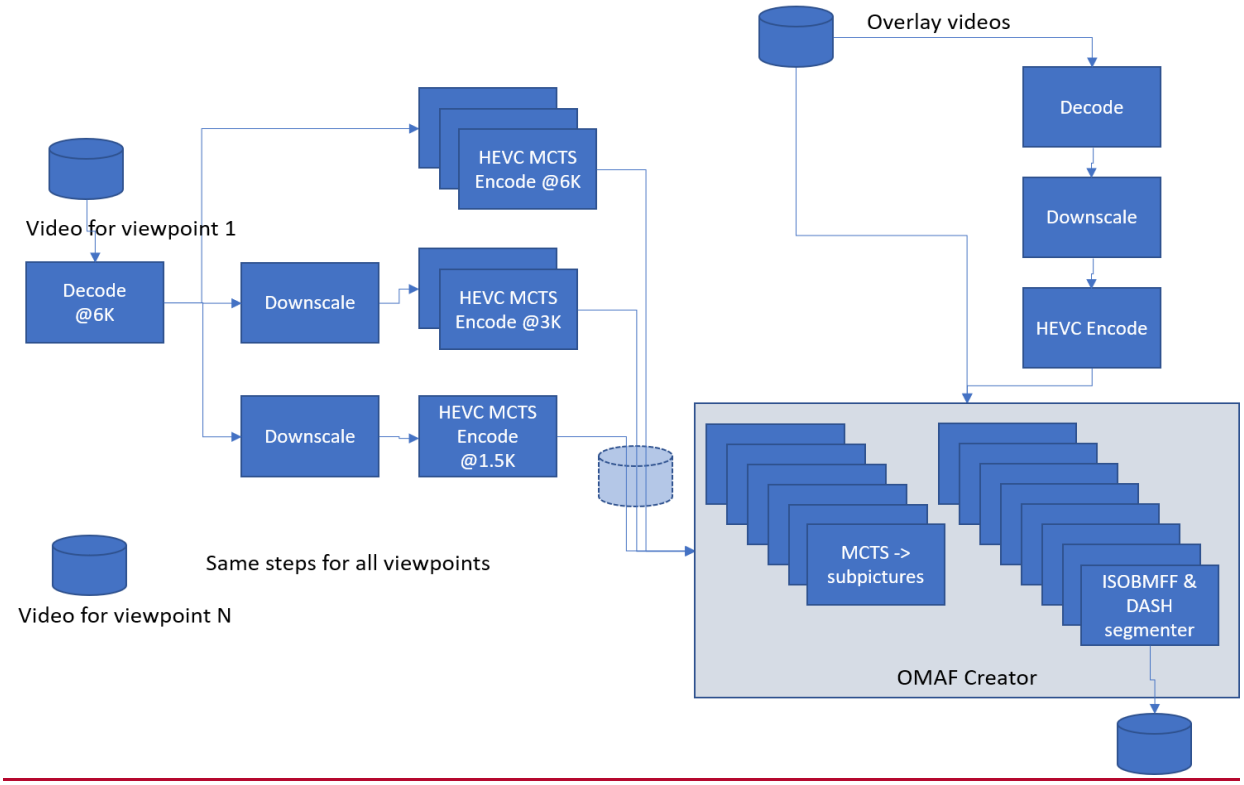

Figure 5 Typical setup of an OMAF effective 6K viewport dependent DASH generation with overlays

That being said, an NBMP WDD is required to be generated with the information of the input 6K video and single or multiple overlay sources. The Node.JS server acts as the NBMP Source to generate or update the NBMP WDD files (multiple workflows) and send to the Workflow Manager by using the REST Workflow API. Inside the WDD, a Processing Descriptor [3] defines the processing functions and the ConnectionMap object that represents the workflow graph (see Figure 6). All function information can be found from a NBMP Function Repository. The repository provides APIs to retrieve NBMP Function

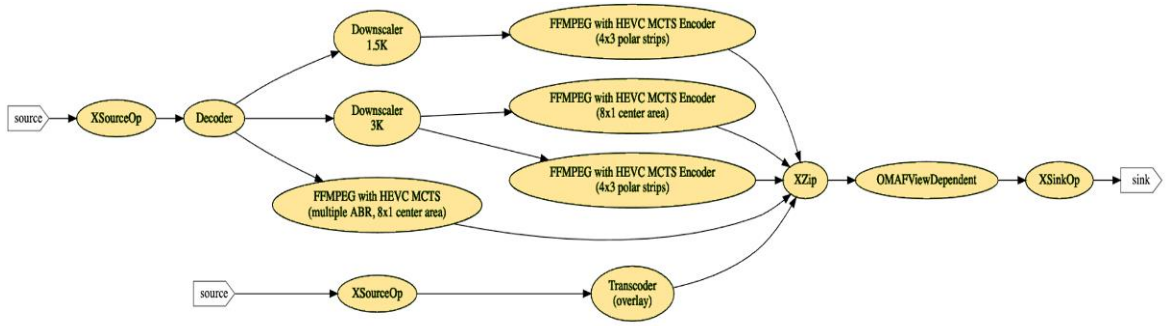

Figure 6 One workflow graph deployed

descriptions that contain function implementation (e.g. in the format of Docker [13] container images). NBMP tasks, instances of functions, run as containers re-scheduled and re-deployed easily to different cloud hosts. The state of the workflow is the data persisted in the storage through the workflow edges. To make small updates like changing the positions of the overlays, the workflow is designed to allow temporal caching to speed up the processing, as long as the workflow is not killed on purpose.

At the end of the workflow, the Node.JS server again acts as an NBMP Sink and informs the OMAF players as soon as the workflow produces any output, for instance, the metadata about the readiness of final media or DASH MPD. Instead of streaming the video content to the NBMP Sink, the workflow is designed to produce lightweight metadata to the Node.JS server, the Sink. It is known that this design requires extra support of a shared storage and may not be ideal for real-time use cases where low latency is critical. Considering the overall cost of the transcoding, we chose this design in favor of its simplicity and extensibility.

The XZip operation in Figure 6 is a built-in function provided by the system to combine two or more input streams into a single output. The function synchronizes multiple inputs and emits an output whenever all of its input streams have produced and signaled at least one output (e.g. the five inputs in Figure 6).

OMAF Players do not need to be tightly integrated to the pipeline. They only need to know the URL of the DASH manifest to start the playback. However, to enable efficient content creation, a preview option is required. We implemented it in two steps. First, the web user interface can play 360° videos in overlay editing phase, making it easy for the users to visualize where the overlays are placed. This is implemented using Three.JS library [12]. However, as current HTML5 video players do not support OMAF playback yet, previewing of the final content is provided via Nokia OMAF Player [9] on Android or Windows platforms. Further, an integrated mode of the player is made to connect to the Node.JS server and to listen for indication events on new content becoming available from the NBMP workflows.

SUMMARY

Media processing continues to evolve to become ever more complex and to involve more tasks and services provided by different vendors. Cloud computing promises a distributed architecture to offload the processing by scaling the workflows horizontally to multiple computing entities in the Cloud or 5G Multi-Access Edge facilities. To ease these kinds of deployments, interfaces between media processing entities in the network should be defined. NBMP, the standard under development by MPEG (ISO/IEC 23090-8), defines formats and APIs for good interoperability and portability of processing functions. With the help of the function discovery API, more complex media processing jobs can be composed as workflows. Current NBMP workflow is made up with a set of descriptors covering the media source and meta-data, instructions for media processing, as well as supportive features like requirements and reporting. NBMP will stick to currently defined standards as much as possible, enabling re-use of many of the existing tools and content for network-based media processing.

This paper presents one realization of NBMP in OMAF compatible content creation. A prototype NBMP workflow implementation for converting traditional 360° videos to OMAF compliant format is presented to demonstrate a full end-to-end interactive media service for an immersive user experience. At the time of writing, NBMP, as a standard, has been under development. Moreover, OMAF v2 standard which introduces VR overlays is also under development. Our implementation did not take full advantage of the NBMP standard, for example, the features such as: reporting for status tracking, requirements for cloud resource management and orchestration with auto-scaling. As a next step, we plan to conduct some experiments and focus on metrics such as availability and scalability, reliability and resiliency.

REFERENCES

1. Omnidirectional Media Format (OMAF), ISO/IEC JTC1/SC 29/WG 11 MPEG, ISO/IEC 23090-2, 2019. http://wg11.sc29.org/
2. Multi-access Edge Computing, ETSI ()
3. Network-based Media Processing (Committee Draft), ISO/IEC JTC1/SC 29/WG 11 MPEG, ISO/IEC 23090-8, 2019. http://wg11.sc29.org/
4. Business Process Model and Notation (BPMN) Version 2.0, OMG (Object Management Group), v2.0, 2011. http://www.bpmn.org/
5. Airflow Workflow Engine, Apache, 2019. https://airflow.apache.org/
6. Argo Container Workflow for Kubernetes, Argo project. https://argoproj.github.io/argo/
7. Mistral for OpenStack, OpenStack Foundation, https://docs.openstack.org/mistral/latest/
8. World Wide Stream, Nokia Bell Labs, 2019. https://www.worldwidestreams.io/
9. Nokia OMAF Player and Creator source code, Nokia Technologies Github, 2019. https://github.com/nokiatech/omaf
10. VR-IF Guidelines https://www.vr-if.org/guidelines/
11. M. M. Hannuksela, Y-K. Wang, A. Hourunranta, “An Overview of the OMAF Standard for 360° Video”, Data Compression Conference 2019, March 2019
12. Three.JS - JavaScript 3D library, https://Three.JS.org
13. Docker technologies, Docker Inc. https://www.docker.com/