# Serverless streaming for emerging media: towards 5G network-driven cost optimization

## A real-time adaptive streaming FaaS service for small-session-oriented immersive media

**Konstantinos Konstantoudakis[1]** [ID] **· David Breitgand[2] · Alexandros Doumanoglou[1] · Nikolaos Zioulis[1] · Avi Weit[2] · Kyriaki Christaki[1] · Petros Drakoulis[1] · Emmanouil Christakis[1] · Dimitrios Zarpalas[1] · Petros Daras[1]**

## Abstract

Immersive 3D media is an emerging type of media that captures, encodes and reconstructs the 3D appearance of people and objects, with applications in tele-presence, teleconference, entertainment, gaming and other fields. In this paper, we discuss a novel concept of live 3D immersive media streaming in a serverless setting. In particular, we present a novel network-centric adaptive streaming framework which deviates from the traditional client-based adaptive streaming used in 2D video. In our framework the decisions for the production of the transcoding profiles are taken in a centralized manner, by considering consumer metrics vs provisioning costs and inferring the expected consumer quality of experience and behavior based on them. In addition, we demonstrate that a naive application of the serverless paradigm might be sub-optimal under some common immersive 3D media scenarios.

## 1 Introduction

With the emergence of immersive media, applications such as teleconferencing and info-tainment are becoming even more attractive and real, since much better a quality of

---

✉ Konstantinos Konstantoudakis
   k.konstantoudakis@iti.gr

Extended author information available on the last page of the article.

experience (QoE) can be be provided to consumers. However, next-generation immersive media types [81] —such as omni-directional (360°), multi-view or three-dimensional video —impose exacerbated bandwidth and latency requirements that vastly surpass even the high-end profiles (i.e. Ultra High Definition - UHD) of traditional media.

Several approaches aim at mitigating this issue by optimizing the use of resources to better deal with the dynamic consumer populations often encountered in the media services. For the real-time media streaming setting, resource management should extend beyond the standard resource scaling and elasticity [73]. Such finer-grained decisions might include the optimized selection of transcoding profiles for improved cost-efficiency from the service provider perspective [90]. Recent standards, such as MPEG-DASH SAND [40], leverage the knowledge obtained from the network to manage media services collaboratively, in order to optimize the users' QoE [52].

The aforementioned approaches have evolved for the centralized cloud model and are limited by each cloud provider's infrastructure, functionalities, and billing schemes. The emerging 5G networks, with their targeted key performance indicators for ultra-fast, ultra-reliable, and high-bandwidth capable edge, are becoming an attractive option for immersive media service developers [56].

A softwarized network, powered by Software Defined Networks (SDN) and Network Function Virtualization (NFV) technologies, is much more amenable to collaborative application and infrastructure optimization via optimized workload placement, application demand adaptation, and network optimization across cloud and edge, based on elaborate monitoring of the infrastructure and service behavior analytics. The finer-grained approaches are able to factor in the information provided by the network into the optimization schemes [90] and are better suited to address the central challenge of developing a network architecture able to dynamically adapt to fluctuating traffic patterns [21].

Serverless computing was first introduced in the end of 2014 and in the last few years it has become an extremely popular cloud native pattern used to build highly granular, yet very cost-efficient, micro-services. Broadly speaking, a serverless application scales to zero in the absence of load and automatically scales out when load is applied. A serverless execution model, where a unit of work is a *function* provided on demand (e.g. in response to some *event*) is called Function-as-a-Service (FaaS). FaaS is a sub-model in a broader serverless paradigm. However, exempting instances where clarity demands a specific term, we will use the terms serverless and FaaS interchangeably in this paper.

These type of workloads characterize many immersive media applications. However, to the best of our knowledge, FaaS is not being widely applied to media intensive services yet. The reasons are many-fold: First, in the typical serverless paradigm, inter-function communication is performed via a shared storage database, which is way too slow and inadequate for media. Second, FaaS frameworks do not support Day 1, Day 2 configuration of functions. Complex management flows—in which serverless functions should be executed in response to events, get configured to connect to the rest of an application and then terminate while the rest of an application continues to execute—are not supported in current frameworks. Third, the use of specialized hardware, like Graphical Processing Units (GPUs), which is often used for real-time transcoding in many media-intensive applications, is not supported out of the box either by open source FaaS, or by commercial offerings. Fourth, since this model is relatively new, it is largely unknown to the broader community involved with immersive media.

This paper is intended to fill this void. We present a novel architectural approach to developing cost-efficient immersive media applications using the FaaS approach. The overall architectural framework and standards for deploying applications in 5G edge is being

evolved by standardization organizations, such as ETSI (the European Telecommunication Standards Institute)", which stipulates that application of FaaS technology in 5G Multi-Access Edge Computing (MEC) [63]. In our previous work, we applied the 5G MEC principles to media-intensive applications at the cloud edge [2] and presented an overall architectural framework that pioneered the use of FaaS in media-intensive applications. That work applied for the first time FaaS to NFV orchestration, utilizing a FaaS VIM integrated with the ETSI MANO framework.

In this paper we continue this line of work by considering session-based workloads typical in immersive media streaming related to infotainment and tele-presence. We develop a fully functional prototype of a tele-immersive gaming service, where time-varying multi-view textured meshes of two players are being produced in real time (a watertight geometry of a player is being produced from four camera streams) and embedded into the virtual environment, where the players can freely move in all 6 degrees of freedom. The players communicate with each other via a broker that is being placed in the 5G MEC in geographical proximity to the players to leverage the 5G latency and bandwidth for the sake of the application. Spectators can join from any edge location and also from a non-5G access network. The spectators tolerate some small lag (much like it is the case for sport events broadcasting).

In contrast to the players, who directly exchange immersive media frames via the broker, the spectators consume 3D streams that are being transcoded to match the capabilities of their terminals. It should be noted that we take an approach different from a typical media streaming architecture. Rather than letting spectators request a specific transcoding profile, our application automatically considers the capabilities of the spectators' terminals and the network conditions and allocates the most cost-efficient transcoding scheme, trying to balance the trade-off between the cost of transcoders, revenue produced by the spectators and the total benefit for spectators in the form of QoE, that motivates them to stay longer in the sessions. Cheaper transcoding profiles are being accommodated on Central Processing Units (CPUs), with less RAM and perhaps lower-quality configuration, while more expensive ones utilize GPUs (using our extended FaaS framework based on Apache Open-Whisk and Kubernetes).

When in-application events of interest occur (e.g. a score point in the immersive game), a replay serverless function can be executed on demand. The function uses some buffered media to produce a replay clip on and stores it in a low-cost cloud storage from which spectators can retrieve it at any time. The number of events happening during the session serves as an indicator by which to estimate the session's popularity with spectators.

For remote spectators joining at the edges where no broker is present, a broker is started on demand, connected to the main broker that is being used by the players and each of the transcoded 3D streams is transmitted only once to the remote broker, to reduce overall traffic load on the network.

Several important points should be noted about our approach. First, each serverless function in our application has one well-defined functionality and a single configuration profile. This greatly simplifies design and operation. Second, thanks to the inbuilt auto-scaling, the application is elastic by design. Third, FaaS is an excellent match for the session-based nature of the application and its fine granularity (a single function level) allows to optimize cost-efficiency of resource allocation at the level of individual sessions, paying only for what is actually being used. These advantages are not available out of the box in any other cloud-native model.

We validate our approach via extensive experimentation, contrasting our network-centric optimization approach with a naive serverless implementation (which would always start

transcoders on demand irrespective of the predicted accrued benefit), and a traditional Virtual Machine (VM)-based approach. Since some features (e.g. support for GPUs) are not yet available in commercial offerings, the billing schemes necessary for experimentation on cost efficiency are not available. To that end, we examine how GPUs are being offered in the cloud today and examine conditions for their cost efficiency in FaaS offerings in 5G MEC. We then use the billing schemes extrapolated from this study as a proxy to obtain preliminary figures illustrating a comparative cost-efficiency of the proposed approach.

In summary, our main contributions are as follows:

– We expand the range of applications for serverless architectures to media streaming, addressing its requirements and introducing the concept of serverless streaming;
– We apply this concept to a demanding use-case of next-generation media by implementing and deploying an adaptive streaming service to 5G-enabled network infrastructure, in the context of a real-time and interactive media scenario;
– We show how a serverless architecture within a 5G framework can also enable in-network service optimization and network-centric adaptation for the media-intensive verticals;
– We demonstrate the cost-effectiveness of serverless streaming compared to traditional solutions, taking into account the balance between the total QoE and cost of production;
– Our findings also serve as a guideline to how serverless should be used in similar use cases and indicate that naively applying serverless would be sub-optimal.

The rest of this paper is organized as follows: In Section 2 we discuss related work and the present work's relation and connection to it. In Section 3 we outline our extensions to the Apache OpenWhisk serverless framework, while in Section 4 we present our serverless adaptive streaming service. In Section 5 network-centric cost optimization is discussed and in Section 6 experimental results are given. Finally, Section 7 concludes the paper.

## 2 Related work

In this work we expand on the novel concept of network-centric 3D immersive media real-time adaptive streaming in a serverless setting. The concept is multidisciplinary and therefore has several partial overlaps with various topics in the literature.

To facilitate the reader's comprehension, we split this section into a small number of more focused subsections covering different sub-topics. In Section 2.1, we briefly describe and provide examples of 3D immersive media production platforms. In Section 2.2, we present the principal ideas and some of the more recent advancements in the area of video adaptive streaming. In Section 2.3, we focus on immersive media, namely 360° video and 3D representations. Next, in Section 2.4, we go over other adaptive streaming solutions, including some of the more recent works in the area of server-based, network-assisted adaptive streaming and cloud-based streaming solutions. In Section 2.5 we provide an overview of the serverless computing model focusing on the features that are more relevant to the context of this work. Finally, Section 2.6 summarizes our innovations.

### 2.1 3D immersive media production platforms

The key enabler of 3D immersive media production is a volumetric capturing system. A volumetric capturing setup is usually comprised of a 360° arrangement of inward looking camera sensors, defining a capturing space with specific boundaries. Despite the fact that

volumetric capturing systems most commonly output a multi-view color plus depth (RGB-D) [22] representation of the captured scene, the most common 3D immersive media format is colored point clouds or textured 3D meshes. The latter are produced by 3D reconstruction algorithms [43, 44] run on the 3D points of the spatially aligned captured views. In general, the 3D reconstruction process can be performed either offline, or in real time which — given sufficient computational and network resources — can additionally allow for live streaming.

One of the earliest low-cost 3D immersive media production platforms [91] utilized 4 consumer grade RGB-D sensors, enabling tele-immersion at interactive rates, utilizing a single GPU. Roughly at the same period, Holoportation [58] utilized 16 infrared-stereo pairs for depth estimation along with 8 color cameras for texturing, to produce and stream high-quality 3D textured meshes. Even though Holoportation produces stunning 3D reconstructions, its computational complexity and requirements for processing power is high, as it requires 20 GPUs in total, while for a 30 frames-per-second real-time streaming scenario it would require over 1 Gbps of bandwidth.

A significant improvement on the volume of the streamable content, which has been kept below 16 Mbps without compromising quality, has been demonstrated by the offline immersive media platform in [18]. This remarkable performance was achieved by utilizing 61 12-core Intel Xeon machines for dedicated processing, which takes 25-29 sec/frame. Similar to [58] and [18], other 3D immersive media platforms, like [23, 67], require an increased number of processing resources to maintain high quality content and keep-up with the extreme bandwidth requirements for streaming.

Finally, recently, [75] has been an open, free-to-use, state-of-the-art, low-cost and portable volumetric capture system, based on consumer-grade RGB-D sensors, offering a balanced option between quality and cost. However, [75] allows for multi-view capturing of point clouds but not an integrated mesh 3D reconstruction solution.

## 2.2 Adaptive streaming

Consumers of media content over the internet are highly heterogeneous. A consumer is characterized by device capabilities, available processing power and network quality (bandwidth, latency, and loss rate). The most common way that the contemporary technology optimizes QoE for consumers, is through HTTP Adaptive Streaming (HAS) [10]. The objective of HAS is to maintain the viewer's QoE at high levels, countering the negative impact of the network bandwidth fluctuations. In video HAS, prior to the distribution, the video needs to be available in segments and encoded in multiple qualities. The most popular HAS protocols today are MPEG-Dynamic Adaptive Streaming over HTTP (DASH) [72] and Apple's HTTP Live Streaming (HLS) [59].

There exist multiple studies on QoE in video adaptive streaming [9, 36, 61, 69]. Some of the more important factors affecting QoE include: initial delay, stalling frequency, stalling duration, adaptation (quality change) interval, adaptation frequency, adaptation direction, adaptation amplitude, video's spatial resolution, video's frame-rate and video's visual quality [69]. Due to the multiplicity of factors affecting a consumer's QoE, there is no single QoE model that different studies converge on and which can serve as a common reference framework.

HAS leaves encoding schemes and the adaptation strategy without a specification. According to [11], and based on the location of the adaptation logic inside the HAS system, HAS schemes can be split into four categories: i) client-based ii) server-based iii) network-assisted and iv) hybrid. The most common scheme is (i), in which the adaptation logic runs on the client with the video player fetching the video segments based on a manifest. In most

implementations, the adaptation logic relies on monitoring internal buffer levels and measuring throughput [12]. Current state-of-the-art client-based bit-rate adaptation algorithms are presented in [74] and [49]. A cutting-edge reinforcement-learning approach is provided by [51], while [87] describes ensemble algorithms tailoring different network conditions.

## 2.3 Immersive media adaptive streaming

### 2.3.1 Omnidirectional media

In 360° video streaming [34], the most common adaptation strategy is viewport-based, in which the equirectangular image is split into tiles which are encoded at different bitrates based on the viewing direction of the client [33, 64, 65, 70, 71] often exploiting tiling support in video coding algorithms, like HEVC (High Efficiency Video Coding) [55, 77].

In [38], a tile-based approach is described using MPEG-DASH SRD (Spatial Relationship Descriptor) and tile-over-viewport prioritization. In [82] and [14] two probabilistic approaches are presented for, tile pre-fetching and viewer tile navigation prediction, respectively.

In [35], the authors attempt to provide a solution to 360° video streaming to smartphones, overcoming their processing power limitations compared to desktop PCs. Finally, [8] presents a real-time streaming system of 360° video relying on GPU-based HEVC [77] coding. A survey on 360° video streaming can be found in [30].

### 2.3.2 3D media

Due to a higher complexity of 3D representations, the 3D Immersive media coding and streaming approaches are less mature compared to 360° or standard 2D video. To begin with, there exist very few 3D immersive media codecs exploiting inter-frame redundancy in time-varying mesh sequences (the mesh sequences of varying geometry and connectivity like the ones produced by real-time 3D reconstruction systems) [24, 84]. Thus, for the 3D mesh geometry, only static 3D mesh codecs are utilized [25]. Furthermore, there is very little literature regarding QoE for 3D immersive media streaming, which could drive adaptive streaming systems [26].

On the other hand, for the point-cloud representations, more options exist. In [47] and [16], point clouds are compressed exploiting volumetric function representations, while in [54] point cloud sequences are intra-frame and inter-frame coded based on octrees and motion prediction. A detailed survey summarizing works in 3D geometry compression can be found in [50]. Finally, the accompanied textures used to colorize the 3D mesh are often compressed using standard 2D image or video compression algorithms like Motion-JPEG (MJPEG) or HEVC.

One of the first works for real-time adaptive streaming of textured 3D time-varying meshes is [19], which is based on a dynamic rule adaptation strategy modifying compression parameters of the real-time stream. For point-cloud streaming, in [37] and [60], multiple 3D objects of the same scene are streamed with adaptation relying on content's proximity to the viewer, along with the viewer's looking direction and distance to content. Recently, in [80], a DASH-compliant HAS system for dynamic point clouds is demonstrated, introducing rate adaptation heuristics that are based on viewer's position and looking direction, network bandwidth and buffer status. At the same time, the encoding scheme utilizes the recently introduced MPEG Video Point Cloud Coding (V-PCC) algorithm [20].

## 2.4  Other adaptive streaming solutions

Server-based, network-assisted and hybrid approaches to adaptive streaming used to be less popular, but recently they started attracting an increased interest with the emergence of SDN and 5G Networks. In [28], a DASH-based server-client adaptive streaming system for standard 2D video is proposed, while in [53] a SAND-DASH network-assisted approach is described, for adaptive video streaming to mobile devices, in MEC scenarios. In [79], the authors propose Cloud Live Video Streaming (CLVS), a model that exploits Amazon S3's storage capabilities in order to enable cost-efficiency in a live video streaming scenario oriented towards small streaming sessions.

The solution in [79] eliminates the need for a constantly up-and-running streaming server (and in that sense it is serverless). Rather, the source video is recorded, segmented and encoded by a mobile device, while the encoded video segments are pushed into a designated Amazon S3 bucket. On end-user devices, the client program of CLVS directly retrieves the most recent video segments from the S3 bucket subsequently proceeding with decoding and playback. While being inventive and accruing cost-efficiency advantages compared to a typical solution, in which a cloud based video streaming server can have idle periods, CLVS will not scale to support real-time 3D adaptive streaming neither from the latency, nor from the bandwidth, nor from the cost-efficiency perspectives. Also this design does not allow a network-centric adaptation of QoE.

Finally, in [78], the authors try to exploit the 5G network infrastructure to offer better QoE in 360° video streaming.

## 2.5  Serverless computing for media

Broadly speaking, FaaS, a specifically popular serverless computing model, refers to structuring applications as stateless functions that are being called on demand (e.g. in response to events). Recently FaaS has been applied by practitioners to video streaming [32, 66]. Little scientific literature exists on the topic. In [88] a measurement study of transcoding tasks has been performed to explore how different lambda function configurations (in terms of memory and proportionally allocated CPU) affect performance and cost. The study reveals that the memory configuration for cost-efficient serverless functions is non-trivial. The best memory configuration is influenced by the task type or even the video content. More work is needed to design an efficient and adaptive system to find the best configuration for serverless functions in video processing pipelines. In [3], a serverless framework facilitating development of video processing pipelines is described. Common to all these solutions is rising serverless functions (e.g. AWS Lambda) for performing operations (e.g. transcoding) on a video file that is uploaded to the cloud storage (e.g. S3 bucket). Upon the file upload, an event is being generated by the storage, which triggers the execution of a serverless function, whose output is either stored in the cloud storage again (potentially creating another trigger for another function execution) or propagated to a Content Distribution Network (CDN), such as AWS CloudFront. The reader is advised to consult [46] for a comprehensive review of serverless frameworks.

## 2.6  Summary of our innovations

In this paper, we expand on a preliminary design [27] of a novel multiplayer tele-immersive game application [17] where players are embedded inside the game environment via their 3D reconstructed avatars. The gaming application is supported by a 3D immersive media

production platform which uses [75] for volumetric capturing and a re-implementation of the 3D reconstruction algorithm found in [1]. This platform is low-cost, portable, real-time and produces streamable content of $\sim$ 50 Mbps, at interactive rates (25 frames per second). The application offers live spectating of the game action and on-demand viewing of replay clips. It is deployed on 5G serverless infrastructure and employs adaptive streaming techniques to stream the 3D appearance of the players to spectators.

Our adaptive streaming algorithm is based on [25] for compressing geometry and MJPEG for compressing textures. Adaptation is achieved by varying compression parameters to produce different profiles at various bit-rates. Further, apart from costs, adaptation optimization is driven by a variant of the QoE model in [86].

To the best of our knowledge, prior to this work, no attempt has been made to use serverless functions for adaptive transcoding of a live 3D immersive media stream. In our implementation we used open source Apache OpenWhisk project [4], leveraging its capability of executing functions on top of Kubernetes to provide features such as direct network communication among serverless functions (as opposed to the communication via storage or database, which is typical in other frameworks), support for Day 0, Day 1, and Day 2 configuration, as well as support for GPUs. Previously, all these features were not being provided out of the box to developers, which hinders serverless adoption for media-intensive applications. In this paper we demonstrate how adding these features might open up new opportunities to achieve cost-efficient immersive media implementations.

Finally, this work is also among the first to provide a network-centric novel adaptive streaming algorithm which takes into account the serverless benefits in order to minimize service costs while offering high QoE to spectators.

## 3 FaaS extensions

FaaS frameworks and offerings are rapidly proliferating, yet few industrial-grade open-source FaaS platforms are available. One such framework is Apache OpenWhisk [4] (OW), which powers the IBM Cloud Functions commercial offering [39]. Presently, FaaS commercial offerings do not offer usage of GPUs in serverless functions. The reason for that is that GPU sharing is a relatively new topic that poses a number of challenges. Since NVIDIA has introduced Multi-Process Service (MPS) in its Volta GPU architecture [57], GPU sharing has become a hot research topic [89].

The most common computing virtualization technology powering FaaS is containers, managed by orchestrators, such as Kubernetes [48]. However, current container orchestrators do not leverage architectures such as NVIDIA MPS. Thus, the solution that we adopt for extending FaaS to use GPUs is the time-sharing of GPUs rather than collocating workloads on the same GPU. Another reason for preferring time-sharing to spatial collocation is that the latter may require re-writing of the application code.

Another problem that current FaaS frameworks do not address is supporting both inbound and outbound network traffic to and from serverless functions. Usually, only the outbound traffic is being supported seamlessly. For the inbound traffic, an image of the serverless function container should include some communication service, which might be difficult to do due to inability to expose the function as a service to the outside world or intricate firewall settings. Our solution relies on a Container Network Interface (CNI) to connect serverless functions to a logical network maintained by the orchestrator. Finally, in the context of 5G MEC, a FaaS framework is provided as part of the MEC platform. Figure 1 shows the ETSI reference architecture for 5G MEC. In this architecture variant,
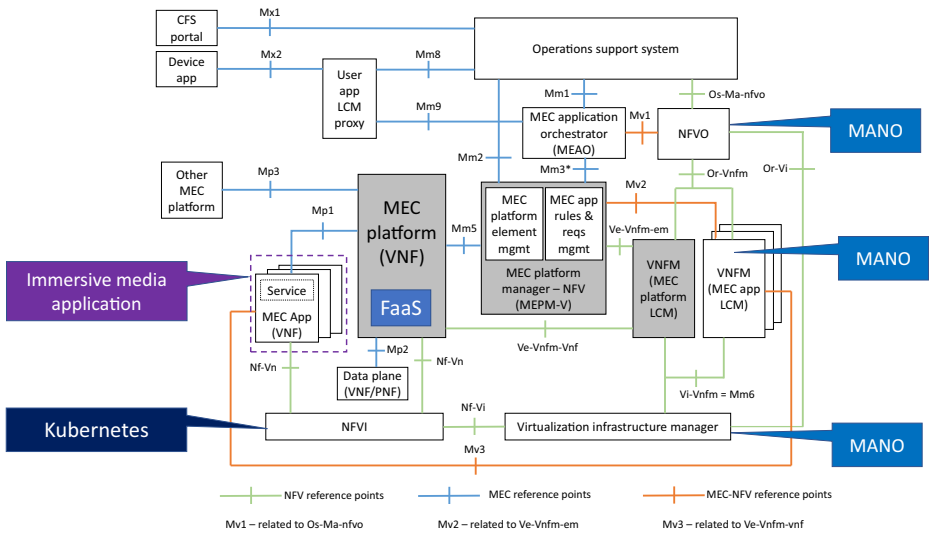
**Fig. 1** Serverless Tele-Immersive Media juxtaposed on the ETSI 5G MEC architecture [29]. Image adapted from https://www.researchgate.net/figure/ETSI-MEC-referencearchitecture-in-an-NFV-environment_fig1_328540778

termed *MEC in NFV*, the application components (serverless functions) are required to be packaged as Virtual Network Functions (VNFs) to be managed by the ETSI Management and Orchestration Stack (MANO), via either a Virtual Network Function Manager (VNFM) or a Network Function Virtualization Orchestrator (NFVO). Finally, the actual container allocation should be performed by a Virtual Infrastructure Manager (VIM) MANO component. Therefore, a challenge arises in how to harmonize ETSI MANO standards with FaaS. We partially addressed this problem in our previous work [2], where we described an ETSI-compatible FaaS VIM. In this paper, we deal with additional problems related to harmonizing orchestration of serverless functions with ETSI MANO to implement the tele-immersive media application.

We will now briefly discuss the challenges mentioned above, and outline how we deal with them in the proposed solution.[1]

### 3.1 Orchestrating serverless applications in 5G MEC

One of the more important challenges for integration between MANO and serverless technology that we faced was the inability to model a FaaS-based service using ETSI VNF Descriptors and VNF Packages. A FaaS-based network service includes components that should not be started upon service instantiation, but created and deleted based on custom events. Some of these events are possibly happening inside the application itself. This pattern cannot be reduced to what ETSI MANO already handles well — auto-scaling. Rather it requires additional flexible orchestration mechanisms, which are application specific. We

---

[1]A reference implementation of our extended FaaS framework and its integration with MANO, can be found in https://github.com/5g-media/faas-vim-plugin.
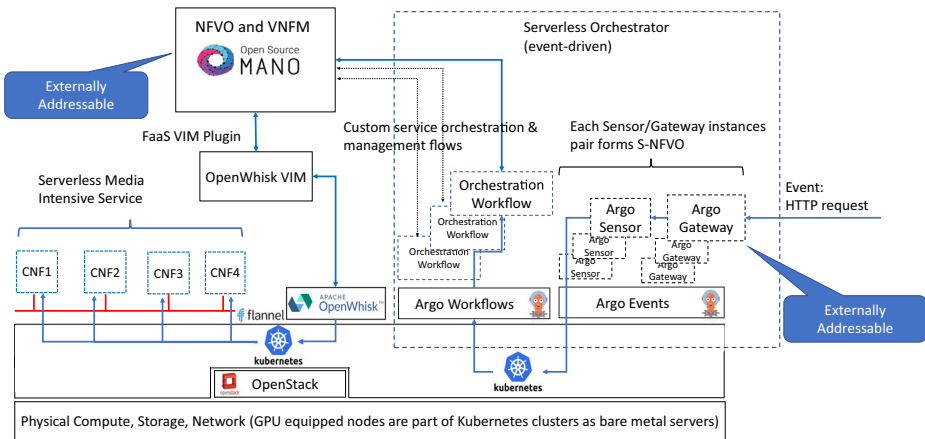
**Fig. 2** Proposed serverless orchestration architecture complementing MANO in 5G MEC

have developed such serverless orchestration, which generalizes to any custom orchestration scenarios and across multiple use cases.

In Fig. 2 we show how we combine serverless orchestration with MANO for the sake of managing serverless tele-immersive media in 5G MEC. We use Kubernetes as our NFVI because of it being a de facto container orchestration standard. It also provides out-of-the-box capabilities for networking and GPU consumption by the OW serverless functions, as we discuss in Sections 3.2 and 3.3, respectively.

To harmonize serverless functions with the standard ETSI network service modeling and life cycle management cycle, we add key/value pairs to the optional information field of a Virtual Network Function Descriptor (VNFD), indicating whether a VNF is serverless and whether is should be started upon instantiation of the network service.

Each serverless VNF in our system is an OW action[2] that is pre-registered with the OW FaaS system, which is provided as part of the 5G MEC Platform. This pre-registration is part of the onboarding into a VIM mechanism prescribed by MANO. For more details see our previous work [68].

This metadata is interpreted by our OW VIM, which then does not invoke an action (does not start the serverless VNF) upon the initial instantiation driven by ETSI MANO (shown on the left).

To handle event-driven instantiation and configuration of serverless functions, we developed a novel orchestration subsystem, which is shown on the right side of Fig. 2. We use CNCF Argo Workflows [6] and Argo Events [5] as the basic mechanism for the proposed serverless orchestration. The former is a Kubernetes-native workflow management engine, while the latter is a Kubernetes-native event dependency resolution system that can trigger Argo Workflows in response to external events. We include a special *bootstrap* function with every network service that uses serverless functions that should be started on demand. In particular, in our implementation of the tele-immersive gaming, transcoders and replay functions are started on demand in response to in-application events rather than upon the initial instantiation.

---

[2]In Apache OpenWhisk parlance, functions are termed *actions*. We will use the terms interchangeably, wherever this does not cause ambiguity

The bootstrap function contains *yaml* definitions for two Kubernetes *Custom Resources (CRs)*: Gateway and Sensor, which are specific to this network service. The CRs comprise the standard Kubernetes mechanism to extend its resources' ecosystem to manage external resources like native ones (pods, jobs, etc). For this work, CRs are essentially yaml files adhering to the Argo dialect. Each such file is an instance of a schema called Custom Resource Definition (CRD). Gateway and Sensor lifecycles are managed by respective controllers, which watch for the new CR instances of Gateway and Sensor CRDs. When such instances appear as a result of applying the CR document to the Kubernetes API server, the Gateway controller sets up a new Gateway instance and connects it to an external event source and a Sensor target, as specified in the CR specification. Likewise, when a new Sensor CR is applied, a Sensor controller that watches the Sensor CRD creates a new Sensor instance and makes itself available to receiving events from the appropriate gateways.

In our implementation, the Argo Gateway, Sensor, and Workflow controllers are part of the pre-deployed services provided by the MEC (see Fig. 1). A bootstrap function is always started upon service instantiation and, immediately after starting, it applies yaml CR definitions of Gateway and Sensor for this service instance, thus creating a session-level event-driven orchestration control plane. This control plane exists for the duration of the service and once the service is deleted (or naturally comes to a termination, e.g. if the game time is up), it is purged from the system.

Our implementation uses an out-of-the-box Webhook Gateway that can receive external HTTP requests that it passes to the Sensor.

The sensor is more intricate. Based on the payload of the HTTP request (i.e., an event that it receives from the Gateway), it conditionally executes lifecycle management actions, such as starting a serverless VNF, stopping a VNF, Day2 configuration related actions, etc. A service developer has to program the Sensor to enable this event-driven orchestration at runtime.

We use this novel orchestration mechanism as follows: When our network-centric optimization decides to reallocate specific transcoding profiles, the control plane of the application that performs the optimization of a specific session sends an HTTP request to the Gateway of that session (previously set up by the bootstrap function upon instantiation of the service), requesting termination of some transcoder profiles and allocation of some other profiles (i.e. terminating some running OW actions and invoking some other OW actions in the Kubernetes NFVI through the OW API). Likewise, when an event of interest happens during the session, an HTTP request to start a replay function is sent to the Gateway of the session, triggering a management workflow in the Sensor of the session that invokes the replay action, configures it and connects it to the rest of the running service.

## 3.2 Networking for serverless applications in the 5G MEC

FaaS frameworks do not support direct network communication between functions out of the box. In our prototypical implementation, we use Kubernetes as the backend for OW actions (containers) execution, which provides a number of networking solutions through its Container Network Interface (CNI) standard. These solutions differ in the level of maturity and sophistication. In our proposed solution, we use Flannel [31], a simple pod-level overlay network that can be used to enable containers running in these pods to communicate directly. The challenge in using Flannel for our work was in devising the orchestration workflows in the Sensor to set up the network just in time upon the service instantiation and then connecting the newly invoked OW actions (which eventually run as pods) to get connected to this network.

A typical hard problem associated with this is port mapping. For each pod in Kubernetes, the IP address of the pod is the address of the Kubernetes Master (also known as the address of the cluster). However, ports should be allocated dynamically and without conflicts. For internally addressable components (i.e. within the same Kubernetes cluster), the port mapping is automatically solved by using a NodePort resource that exposes a pod as a service. However, in our case, if a service component should be accessed externally, a more elaborated Ingress resource should be defined. We omit the technical details of setting up and configuring the Ingress resource and Ingress Controller. It is important to stress that in our system this is being done on demand using our serverless orchestration mechanism described in the previous subsection.

### 3.3 GPU allocation for serverless applications in 5G MEC

Some transcoding profiles require GPUs for efficiency. In fact, a large part of this work is devoted to optimizing usage of GPUs for serverless tele-immersive media applications in 5G MEC, where these resources might be scarce and relatively expensive. However, before we can optimize usage of GPUs by serverless frameworks, we need a basic support for consuming them. Apache OpenWhisk proved to be an easily extensible framework in this respect. OpenWhisk contains an extensible dictionary of action *kinds* that defines their runtimes. We created a new runtime that uses NVIDIA's CUDA framework. For example, a generic CUDA action can be defined as shown in Listing 1:

Adding an entry to the action kinds dictionary is not sufficient to make OW interpret this new action kind. We use the OW Kubernetes Client, which — when OW is configured to use Kubernetes as a container management environment for the actions — creates a Kubernetes pod yaml definition out of the action metadata. This yaml definition is then applied by the Kubernetes Client to the Kubernetes API Server and the action starts executing as a Kubernetes pod. A Kubernetes yaml definition for the action shown in Listing 1 would look as shown in Listing 2:

We modified the OW Kubernetes Client to recognize GPU action kinds that we defined for the GPU-based transcoding profiles. When such an action is on-boarded on OW that is configured to work with Kubernetes, which has GPU equipped worker nodes in its cluster, the action will be placed by the Kubernetes scheduler to a node that has NVIDIA GPU (this

```
1
2  "cuda":[
3    {
4          "kind": "cuda:8@gpu",
5          "default": true,
6          "image": {
7                "prefix": "docker5gmedia",
8                "name": "cuda8action",
9                "tag": "latest"
10        },
11        "deprecated": false,
12        "attached": {
13        "attachmentName": "codefile",
14              "attachmentType": "text/plain"
15        }
16   }
```

**Listing 1** CUDA action

```
 1  apiVersion: v1
 2  kind: Pod
 3  metadata:
 4    name: cuda8action
 5  spec:
 6    containers:
 7      - name: cuda8action
 8        image: "docker5gmedia/cuda8action:latest"
 9        resources:
10          limits:
11            nvidia.com/gpu: 1
```

**Listing 2** CUDA action Kubernetes yaml definition

functionality is being supported in Kubernetes as an experimental feature since Kubernetes 1.8). A full implementation is available in [13].

## 4 Serverless adaptive streaming service

*Tele-immersive* media streaming services are usually sporadic in nature, with long periods of idleness interspersed with short sessions of activity (e.g. gaming or conferencing). Under a traditional VM-based design, apart from the increased service complexity, a constant sizing problem would manifest when seeking to optimize the service's costs. FaaS offers a cost-efficient alternative as it automatically scales to the number of active sessions.

As media streaming consumers can have very different bandwidth or processing capabilities and network conditions can fluctuate, a crucial part of an effective media streaming service is adaptation. The original content is transcoded into a number of media profiles, each targeting a different bandwidth and media quality, allowing each consumer to receive the profile most suited to their needs. Lack of an appropriate profile can lead to frequent buffering events for on-demand consumption, or make meaningful reception completely impossible for live streaming. Hence, adaptation is especially important in live streaming media services.

An apparent advantage of a serverless adaptive media streaming service is more efficient utilization of the available resources, such as different transcoding profiles. Indeed, for smaller consumer size sessions, not all profiles might be relevant, which allows for cost-optimized resource use. Thus, apart from inter-session scaling, serverless streaming offers finer-grained intra-session scaling and adaptation.

This is more pronounced for emerging media services, which due to their immaturity have to support a wider repertoire of profiles. Specifically for 3D immersive media [17, 42], the profile selection problem is more complex [25] due to the simultaneous availability of various profiles (joint 2D and 3D), and their suitability to highly heterogeneous consumer types (e.g. mobile, workstations, VR headsets) that, in turn, have different requirements for the received profiles.

This type of immersive media delivers two payloads simultaneously, the 3D mesh media stream and the multi-view textures media streams. While the latter are encoded with traditional flat/2D media encoders, the former use distinct 3D codecs. This effectively renders each immersive media stream profile to be a tuple of a video (i.e. 2D), $P_{2D}$, and 3D, $P_{3D}$ profiles, leading to a more complex visual quality formulation [26]. Furthermore, emerging

consumption means that VR and AR (Virtual and Augmented Reality, respectively) accompany traditional displays (i.e. desktop/laptop and mobile), creating a far more complex landscape for profile selection that depends on each consumer type's computing and viewing characteristics. We argue that for sessions with relatively few consumers, which will require only an optimal subset of profiles, a serverless streaming model is more appropriate, because it opens up more opportunities for optimization.

As explained in the previous section, our extended FaaS framework allows GPU consumption. This adds another dimension to our profiles, expanding the two-tuple to a three-tuple ($P_{3D}^a$, $P_{2D}^b$, $R^c$), containing the 3D and video profiles, in addition to the computing resource type $R^c$ (i.e. CPU: $c = 0$; or GPU: $c = 1$). Thus, profiles with similar bit-rates, may reduce processing latency at the expense of using higher cost resources. For conciseness, we denote a transcoder's joint 3D media profile as $P^n$, with $n$ encoding a unique combination of $a$, $b$ and $c$.

Serverless design follows the single responsibility principle: each function is responsible only for a single task, instantiated as the need arises and destroyed when the task is completed. In the context of media streaming adaptation, this translates to having one transcoding function for every combination of profile and source (i.e. a player).

A general scenario for tele-immersive media streaming includes a population of producers ($K$), which generate live 3D video streams; and a population of spectators ($S$) who need to receive the streams of all producers and reconstruct them in the virtual environment. Our service then comprises a broker function (*vBroker*) and $|N| \cdot |K|$ transcoder functions (*vTranscoder*), where $N$ is the set of transcoding profiles, as each vTranscoder is responsible for transcoding the stream of one specific player to one specific profile.

Producers send their production streams to the vBroker, while vTranscoders receive these streams from the vBroker, transcode them according to predefined profiles, and upload them back to the vBroker. Consumers then are served either the production stream or a transcoded one from each producer, based on an adaptation logic. In the context of Kubernetes, this means allocating a set of transcoding actions for each production stream.

This serverless adaptive streaming design lends itself to optimization. Transcoder functions can be deployed on demand while monitoring the service's behaviour, as events in response to the monitoring analysis. Typically this relates to monitoring its cost, and seeking to minimize it, and monitoring the QoE of its consumers, seeking to maximize it. Taking into account the cloud-native transformation happening thanks to the emergence of 5G and the virtualization and softwarization of the network, it is possible to perform service optimization in an integrated manner with the network itself.

Instead of relying on a local client-based adaptation, service adaptation and optimization can take a more global approach.

Our streaming service is entirely dynamic, with the vBroker action deployed at the start of each session, for that specific session. This allows for edge proximity placements and a flexible vBroker interconnection scheme that unifies edge and core resources, allowing our session-based services to span multiple infrastructures. Transcoders are deployed on the fly according to the network-centric session optimization logic. The service has a choice either starting with zero transcoders and subsequently adding them on demand as guided by the optimization, or starting with a default transcoder profiles configuration, and then adapting it to the actual consumers. This is similar to client-based adaptation that starts either on the lowest/highest profile, and then adapts to that which results in higher QoE.

Application-specific events (e.g. replaying highlights) trigger processing functions that are deployed on the serverless infrastructure and are responsible to synchronize media and
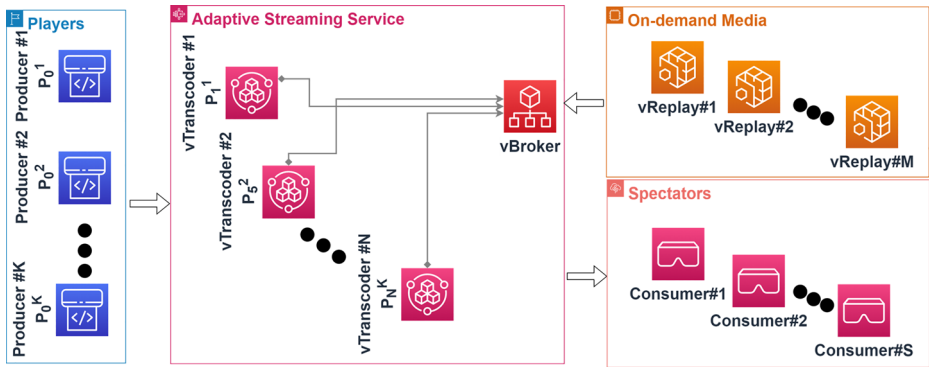
**Fig. 3** An abstracted architecture of our service: vTranscoders are instantiated or destroyed as needed, each one responsible for transcoding the media stream of one producer to one profile. vReplay functions are similarly triggered by certain events. All streams flow through the vBroker, from which consumers receive the allocated media streams

game-state streams to produce replay clips that they can later be served to spectators on demand.

In Fig. 3, the service components are depicted. On the left, producers in the 3D immersive media production platform produce high-quality profile 3D media streams, denoted as $P_0^k$. The adaptive streaming components are comprised of a set of vTranscoders, each one being responsible for transcoding an input 3D media stream from a single player to a single profile. Those transcoded streams become available to the consumers via the vBroker instance.

Additionally, vReplay instances are instantiated on the FaaS infrastructure in response to specific events, as described in Section 3. Upon the completion of replay clip processing, the processed media clips become available to the application consumers on demand.

In more detail, our network-centric real-time adaptive streaming service drives an AugmentedVR [42] gaming application. The application manages gaming sessions supporting $K$ players and $S$ spectators, where $|S| \gg |K|$. Each player is captured with a volumetric capturing station [76] and 3D-reconstructed in real time [1], producing a live 3D media stream. The players' live media traffic, along with the application game state metadata are transmitted and synchronized among the playing users (more details regarding the application's architecture can be found at [17]). In this way, players are emplaced within the same shared virtual environment, and interact within it in a capture-the-flag context. Through the aforementioned adaptive streaming service back-end, the application allows for remote party spectating of each gaming session.

The spectators $S$ receive the synchronized game state and all $|K|$ players' media streams, faithfully reproducing the current session, with example screenshots presented in Fig. 4. While the players' communication is based on stringent real-time requirements, the spectators' media consumption relies on broadcast traffic, and thus requires consistent streaming with relaxed latency constraints.

This is driven by a centralized control plane of the application, which oversees the production and delivery of appropriate profiles to each spectator for smooth playback. The control plane is extensible and new optimization algorithms can be plugged in as needed. In Section 5.5 we present our proposed network-centric optimization to drive the control plane and in Section 6 we compare this smart optimization with a more naive baseline algorithm to quantify the benefits of the network-centric optimization. The control plane of the

**Fig. 4** Screenshots of the AugmentedVR immersive media game where the playing users real-time 3D media streams are embedded into the same shared virtual environment. The screenshots' viewpoints are those of spectating users that can freely navigate the scene in order to spectate the action around the virtual arena

application interacts with the Serverless Orchestration mechanism described in Section 3 to actuate the transcoder profile allocation plans calculated via optimization.

These profiles are selected from a set of profiles $N$, with each spectator receiving one profile $P_n^k \in N$ (with $n \in |N|$) for each player $k \in K$. Each profile is served by a single transcoding action, spawned and managed by the service, that re-encodes the originally produced profile $P_0^k$ from a specific playing user, to a lower bit-rate profile $P_n^k$, which is made available on the broker. At the same time, the application orchestrates the production of on-demand media in the form of highlight replay clips. These are event-driven processing actions that produce finite media streams of previously captured live traffic. Once produced, these too are available on the service's broker for on-demand consumption by the spectators. Finally, the orchestration and management of the transcoding actions are handled by our service's optimization logic that has a dual role. On one hand, to optimize the application's costs, while preserving the resulting QoE by making scaling decisions for its elastic components (i.e. the transcoding actions); and, on the other hand, to apply network-centric adaptation by collectively deciding each spectator's consumed profile.

One important design concern is dealing with the fixed maximal life time of FaaS executions. In cases, when the session time is about to exceed the lifetime of the functions involved, a shadow FaaS invocation can be started and configured. As explained in Section 3, we use NodePort to expose serverless functions as Kubernetes services. This means that we can transparently switch one FaaS invocation with another without disturbing the service. Therefore, while any concrete serverless function cannot execute beyond its maximal life time, collectively an intensive media session can be extended as needed at fine granularity.

## 5 Network-centric cost optimization

When considering the optimization of a serverless live streaming delivery network, there are two conflicting objectives: to maximize the QoE of every individual spectator and to minimize the cost to the service provider. Maximizing the QoE entails making the streams

available in multiple versions differentiated in visual quality and bitrate, so that each specta-tor can consume a version most suited to their device type, processing power and connection capabilities. The production of multiple transcoding profiles, however, involves running more transcoder FaaS functions, thus increasing cost.

In order to balance a tradeoff between QoE and cost, both must be expressed in common units. Providing a certain QoE level can be naturally connected to generating revenue for the stream producer, either directly or indirectly. Our proposed optimization maximizes profit for the stream provider (i.e. the revenue minus cost objective). This section describes the components involved in modelling revenue and cost.

## 5.1 Spectator behavior

During the course of a session of live-streamed media, individual spectators may be con-suming the stream from its start, or join at any later point in time. Streams of different characteristics (e.g. popularity) may attract new spectators at different rates and numbers. Similarly, spectators may stay online until the stream ends, or quit before that, for reasons which may or may not be related to the stream characteristics.

### 5.1.1 Spectator arrival

Traditionally, an arrival process of people to stores, facilities, telephone calls, etc, has been modeled using Poisson distribution [41, 45]. The Poisson distribution calculates the proba-bility of $k$ events (e.g. arrivals) occurring in a specified interval, given the average number $\lambda$ of events per interval [85]:

$$P(k \text{ events occurring in interval}) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{1}$$

Although outstanding circumstances and external events (e.g. a social media post) can cause spectator arrival to diverge from the Poisson distribution, these are considered extreme cases and they are impossible to model. Hence, in this work we follow the mainstream approach and model spectator arrival as a Poisson distribution. We define the distribution's interval as the ten-second time step and set the average number $\lambda$ of arrivals per interval to a range of values from 0.25 to 1, with a default of 0.5 arrivals per ten-second time step. In addition, we assume a starting spectator base corresponding to 20 time steps (hence, a default of 10 starting spectators) for each session.

### 5.1.2 Spectator quitting

Once spectators join, they may remain online until the end of the stream or quit before then. Chen et al. [15] model spectator quitting probability as a function of their QoE: a spectator with very bad QoE is certain to quit, while a spectator with very good QoE is likely to remain but still has some 20% probability of quitting before the session ends, for non-QoE related reasons. Between these two extremes, the decrease of quitting probability with QoE is assumed to be linear.

In a scenario with a diverse mix of spectators, QoE may vary significantly depending on device type, processing power and connection bandwidth. Spectators with powerful PCs and a good connection will have a better QoE than spectators with mobile devices, which would lead to mobile spectators quitting much more frequently. In this work we consider that each spectator is aware of their own hardware and connection capabilities, and will be happy

with the best QoE possible for that configuration. Hence, in estimating quitting probability, we consider the *difference* between the maximum QoE possible for each spectator and their actual QoE.

Other factors that might impact quitting for non-QoE related reasons include the interest level of a given session: spectators may abandon a boring or slow session more easily than a very active or thrilling session. This will also impact QoE-based quitting probability modeling, as spectators may be reluctant to leave an interesting stream despite QoE being mediocre. QoE-related quitting can be further altered by how demanding a spectator population is.

Hence, based on the findings of [15] and these considerations, we build a linear quitting model for each 10-second time step. The probability that during a time step $t$ a spectator experiencing $QoE_t$, will quit is:

$$q_t = q(QoE_t) = b + (QoE_{max} - QoE_t) \cdot d = b + dQoE \cdot d \tag{2}$$

where:

– $b$ is the base quitting probability per time step for non-QoE related reasons, with a default value of 0.37%, corresponding to a cumulative probability of 20% to quit at some point in the course of a 10-minute session.
– $dQoE$ is the difference from the maximum possible QoE for that spectator.
– $d$ is a factor denoting how much QoE impacts quitting, which is dependent on the QoE value range produced by the QoE model and the session parameters (i.e. how interesting or important a session is, and consequently how likely spectators are to leave because of QoE dissatisfaction). The QoE model we adopt (see Section 5.3) produces values usually within the range of 2.8–3.8. Accordingly, $d$ ranges from 10% (an interesting session that spectators won't quit easily) to 50% (very demanding spectators), with a default value of 20%.

The probability that a spectator remains online in a given time step is $p_t = 1 - q_t$. The probability for a spectator to remain online from $t_0$ to $t_1$ would be the product of remaining at each individual time step in between, which, naturally, is decreasing over time:

$$p_{t_0 \rightarrow t_1} = \prod_{t=t_0}^{t_1} p_t = \prod_{t=t_0}^{t_1} (1 - q_t) \tag{3}$$

Equation (3) assumes that quitting events during different time steps are independent and identically distributed. While this may not always be the case, a more sophisticated model of spectator behavior is currently outside of the scope of this work, because more field data should be collected on immersive media spectators' online behavior as these services become mainstream. Presently, this is still a new area and we believe that using simpler modelling is justified for initial exploration of cost/QoE trade-offs.

To calculate the probability of a spectator remaining active from the beginning of the session to its end, $t_0$ and $t_1$ can be set to 0 and $|T|$, respectively. For a 10-minute session comprised of 10-second time steps, $|T| = 60$.

Hence, for example, a demanding spectator in a boring session, with a $dQoE$ of 0.5, might have a 5.37% probability to quit every 10 seconds, meaning she may soon leave unless her QoE improves. Note that in the relatively narrow QoE range produced by the QoE model (see Section 5.3), a $dQoE$ of 0.5 represents a significant decrease from the optimal QoE for this spectator. Conversely, for an undemanding spectator in an interesting session

with a $dQoE$ of only 0.1, quitting probability would be 0.57% per ten-second time step, and he is 71% likely to remain until the end of a 10-minute session.

## 5.2 Revenue

Depending on the use-case and the marketing approach, revenue for the media stream service provider can range from direct (e.g. a subscription-based or pay-per-use service) to indirect (e.g. a service supported by ads).

In general, the provider is interested in keeping spectators engaged for longer time periods, because it might generate more revenue. In an ad-supported service, spending more time watching the stream results in greater exposure to the advertisements. In a subscription service, spectators who don't spend so much time watching the stream may reconsider renewing their subscription. Spectator QoE may also impact the revenue they are generating, or not, depending on the specific use-case. In a pay-per-use service, the revenue generated is directly proportional to the time spent in the service.

In this work, we consider an ad-supported use-case as a baseline scenario, and correspondingly assume that each active spectator generates indirect revenue per time unit, so long as they remain active. Revenue generated per time step can be constant, or a function of the spectator's QoE, considering that spectators happier with their QoE may be more receptive to ads. As revenue modelling varies by use-case and is outside the scope of this paper, we consider the generic case that revenue is a function of QoE. This can be modelled by any monotonically non-decreasing function, e.g. constant, linear or logistic:

$$r_t = r(QoE_t) = a_1 \qquad \text{(constant)} \qquad (4)$$

$$\text{or} \quad r_t = r(QoE_t) = a_2 \cdot QoE_t \qquad \text{(linear)} \qquad (5)$$

$$\text{or} \quad r_t = r(QoE_t) = a_3 \cdot \frac{1}{1 + e^{-QoE_t}} \qquad \text{(logistic)} \qquad (6)$$

Over the course of a streaming session, the revenue generated by a spectator during each time step accumulates to produce the total revenue over time:

$$r_{t_{join} \rightarrow t_{quit}} = \sum_{t=t_{join}}^{t_{quit}} r_t \qquad (7)$$

However, the time that spectators remain active, and therefore generate revenue, is directly affected by the QoE they are experiencing, as mentioned in Section 5.1.2 and (2). For a given future time step $t$, the *average expected* revenue generated by a spectator with $QoE_t$ and $q_t$ probability of quitting will be dependent on the probability they remain active until $t$. Taking into consideration (3):

$$E(r_t) = r_t \cdot p_{t_0 \rightarrow t_1} = r_t \cdot \prod_{t=t_0}^{t_1} (1 - q_t) \qquad (8)$$

Therefore, taking into consideration (8) and the dependency of $r$ and $q$ on QoE for every time step, the total expected revenue from a spectator, from the current time $t_0$ until time $t_1$ is:

$$
\begin{aligned}
E(r_{t_0 \to t_1}) &= \sum_{k=t_0}^{t_1} E(r_k) \\
&= \sum_{k=t_0}^{t_1} \left[ r_k \cdot \prod_{t=t_0}^{k} (1 - q_t) \right] \\
&= \sum_{k=t_0}^{t_1} \left[ r(QoE_k) \cdot \prod_{t=t_0}^{k} \Big( 1 - q(QoE_t) \Big) \right]
\end{aligned}
\tag{9}
$$

Equation (9) highlights how QoE can impact revenue both directly, by altering the revenue an active spectator generates per time unit, and indirectly, by affecting their probability of quitting early.

### 5.3 QoE model

In order to keep spectators from quitting the stream early, thus maximizing generated revenue, an optimization algorithm would need to know what each spectator's QoE is at present, and how it may change depending on the network-centric optimization decisions. Although a number of video streaming QoE models exist (e.g. [61, 62, 83]), there is none, to our best knowledge, that regards textured 3D meshes viewed in a free viewpoint environment. However, for testing purposes, a suitable 2D video QoE model may be adopted.

In this paper we derive our QoE model from Zadtootaghaj et al. [86]. In that work the authors consider cloud gaming, which is a close match to our own use-case. Using subjective mean opinion score (MOS) measurements, they derive QoE as a second degree function of image PSNR and frame rate (FR), fitted to the MOS:

$$
\begin{aligned}
QoE = &-8.97 + 0.056 \cdot \text{FR} + 0.41 \cdot \text{PSNR} - 0.0038 \cdot \text{PSNR}^2 \\
&-0.001 \cdot \text{FR}^2 + 0.00079 \cdot \text{FR} \cdot \text{PSNR}
\end{aligned}
\tag{10}
$$

Knowing the average PSNR and frame size for each transcoding profile and each spectator's bandwidth, we use this model to calculate each spectator's QoE at present and estimate their QoE in the future for different profiles.

In a tele-immersive game, a spectator will be receiving each player's 3D representation in a transcoding profile. For each profile, the average PSNR is known, calculated from the PSNR of the textures used to color the 3D mesh, considering that part of the screen occupied by the 3D reconstruction. Although the latter of course varies by a spectator viewpoint, in the vast majority of cases the 3D reconstruction will occupy an area of 1-5% of the total screen area. Given that the area not occupied by the 3D reconstruction is computer-generated and suffers no loss of quality with different transcoding profiles, we offset average texture PSNR to obtain an estimate of average spectator view PSNR.

Depending on a spectator's maximum bandwidth, they may be unable to receive the incoming stream at its full framerate. Equation (10) considers the actual framerate experienced by a spectator, which will depend on that spectator's connection bandwidth and the average frame size of the received profile.

In a tele-immersive game for $|K|$ players, spectators will receive one transcoded media stream for each player, each with possibly different transcoding profiles and resulting QoE. The total QoE for each spectator, which aims to reflect their satisfaction with the whole

immersive experience, will be a function of the individual QoEs corresponding to each player. The simplest approach is to simply average the QoE of each player's 3D reconstruction. A more thorough modelling, which is beyond the scope of the present work, might take into account the relative position and orientation of the spectator and the players inside the virtual space, and assign greater weights to the 3D representations of players closer to the spectator and nearer the center of their field of view. However, position and orientation would likely not remain constant in an immersive environment, even for a ten-second time step.

In this work we opt for the simple averaging approach, assuming that spectators can see both players equally in the virtual space. This in no way limits the generality of the methodology and outcomes, as it considers the most generic case.

## 5.4 Costs

The costs of delivering live media to a population of spectators are comprised of two separate categories: the cost of running the necessary software to transcode and buffer the data, and the cost of delivering the data to the consumers.

In the serverless approach we examine in this work, each transcoded media quality is being produced by its own dedicated FaaS transcoder. We assume that such transcoders are being deployed in a 5G MEC FaaS (e.g. using our extended FaaS framework). Since MEC is, essentially, a cloud deployed at the edge (also referred to by telcos as a *cloud edge*) the business model is similar to that of the cloud, but the resources are more scarce and therefore are likely to be priced differently. Applications (such as our tele-immersive gaming) rent these resources on a pay-as-you-go basis.

In addition to the regular resources available to FaaS in the current commercial offerings, video transcoders may also require the use of a GPU for real-time processing, which will incur additional costs, as described in the following subsection.

Besides the transcoders, a broker function, active throughout the session, is also necessary to facilitate the media stream traffic. The core broker function also facilitates communication between the players. Therefore it will always be placed in the MEC.

In cloud-deployed functions, only outgoing (not internal) traffic is usually charged, with typical prices ranging from $ 0.05 to $ 0.10 per GB. In 5G MEC deployments it is too early to reason about the pricing plans for inside-edge traffic and in-bound and out-bound traffic between 5G MEC and the cloud, because commercial offerings are still being formulated.

### 5.4.1 GPU pricing model

Over the last few years, GPUs have become essential to a multitude of applications. Cloud vendors have recognized this market potential and have started providing new virtual server families that include GPUs. However, GPUs are not easily amenable to sharing among different workloads. This dictates a time-sharing approach and drives up the cost of the cloud-based GPU servers.

Limitations to GPU sharing are especially challenging for serverless computing. If time-sharing is used, then only one serverless function consuming GPU can run at a GPU-equipped virtual server at a time, with the rest of the server resources (CPU, RAM) being wasted. As we go to press, we are not aware of any commercial offering for serverless

computing with GPUs. This does not preclude such offerings in the near future as GPU sharing improves (Nvidia,[3] Nuweba[4]). Furthermore, we believe that a significant progress with building commercial cloud offerings for serverless GPUs will only become possible when shareable GPU architectures will become ubiquitous and this programming model will be consumable at the application level.

In our previous work, we developed a first-of-its-kind prototype for using GPUs with serverless functions. Our prototype uses Apache OpenWhisk and Kubernetes [13]. To enable quantitative reasoning about using serverless computing for tele-immersive gaming in the 5G MEC's FaaS, we need to develop an estimation of a realistic pricing model for GPU usage in serverless computing. The MEC business model is essentially the same as the public cloud business model, but with scarcer resources, which justifies their higher pricing. Essentially, the supposition of MEC is that it behaves like a cloud on the edge, allowing to leverage proximity to users and higher KPIs at possibly higher price points for providers, but overall making more profit by enabling new application capabilities and providing much better QoE that would attract a larger customer base.

We therefore derive our hypothetical pricing plan for MEC using public clouds as a starting point. To that end, we consider a typical CPU-based cloud functions pricing, and CPU-based virtual server pricing vs GPU-based virtual server pricing and develop a speculative model for the GPU based serverless costs. It should be stressed that our intention is neither to propose an actual pricing model for GPU-based serverless computing nor to argue that the profit margins should necessarily be the same as for the CPU-based one. Rather, our intention is to provide an educated guess for what this model might look like and use it to study the pros and cons of our proposed approach quantitatively.

Our methodology is to assume the same profit margins ratio between the GPU- and CPU-based serverless computing as between GPU- and CPU-based *virtual servers*. The latter is directly observable from the publicly advertised cloud vendors pricing plans. Note that while this assumption can deviate from the actual ratios in practice, a proportionality between the internal cost of production and the profit should exist. Hence, as long as we preserve the directly observable ratios in our estimations, they should serve as a reasonable proxy.

As an example pricing reference point, we consider pricing plans for IBM Cloud Functions[5] and Virtual Server Instances.[6] Similar results can be obtained for other cloud vendors. ACL1.8x60 and M1.8x64 are the two models of virtual servers with and without GPU, respectively. These two models have the same number of CPUs (8) and approximately the same amount of RAM (60 and 64GB, respectively). Billing is on a monthly basis. At full time utilization (i.e. 720 hours per month up time), M1.8x64 costs \$362.88 at hourly rate \$0.504 while ACL1.8x60 costs \$1402.56. This means that leasing a GPU-enabled server with other parameters being equal to a CPU-based one is about of 3.8 times more expensive. Note that what is important in this study is the internal cost.

With the time-shared GPU-based serverless computing, the server can run only one GPU-based function at a time. Typically, GPU-enabled servers are large. Therefore, running a single GPU-based function is tantamount to fully occupying a large server for the duration of the function lifetime. The number of 15 minutes long serverless functions per month per

---

[3]https://www.nvidia.com/en-us/data-center/virtual-gpu-technology/

[4]https://www.nuweba.com/

[5]https://cloud.ibm.com/functions/learn/pricing

[6]https://www.ibm.com/cloud/virtual-servers/pricing/

server will be $2,880 = 720 \cdot 4$. Therefore, the cost of a single 15 min execution can be assumed to be $0.487 = \frac{1,402.56}{2,880}$.

To verify this calculation, one can observe that exactly the same number can be obtain by simply dividing the hourly rate of ACL1.8x60 ($1.9472) by 4 (number of 15 minute long functions per hour). This would give a base rate of $0.00054 = \frac{1.94}{3,600}$ GPU seconds (we assume the same usage of RAM as for the CPU case).

Note that while running a GPU-based serverless function, the same host can be used to also run CPU-based functions. Otherwise the CPUs and RAM of the GPU based host will be just wasted. As we observed above, the cost ratio between a CPU- and a GPU-based VM is 3.8. IBM Cloud Functions are being priced at the base rate of $0.000017 per second of execution, per GB of memory allocated (we abbreviate this to per GB seconds). This implies a base rate of $0.000064 = 3.8 \cdot 0.000017$ per GB seconds for CPU based functions (when running on a GPU-enabled host). Of course, it is unreasonable that a CPU function will become more expensive in the public cloud just because we introduced GPU-based functions. This means that to keep the CPU functions at the current base rate, GPU-based functions should be made even more expensive, which will increase the ratio between the GPU and CPU serverless computing costs beyond 3.8 (alternatively, GPU sharing architecture should be developed and deployed to reduce the GPU price when consumed via serverless functions). However, in the 5G MEC, the users can be more receptive to higher price points, because it is expected for the MEC resources to be scarcer and, therefore, more expensive.

A detailed pricing modeling for the time-shared GPU model is outside of the scope of this paper. For the sake of modeling serverless GPU functions costs in this work, we assume that the FaaS is provided on top of the GPU-enabled servers, similar to, say, ACL1.8x60 with the base rates of $0.000064 per GB seconds for CPU-based functions and $0.00054 per GB seconds for GPU-based functions (i.e. an order of magnitude difference in the cost). With this choice, we will be able to avoid inflating the estimated benefits of our proposal while still be able to demonstrate its usefulness.

## 5.5 Optimization

Our goal is to maximize the profit that an immersive game provider accrues from offering the service on the 5G MEC using FaaS. While there are multiple costs involved with provisioning (e.g. storage for replay clip files, databases for managing service subscription, monitoring subsystem, FaaS charges for replay clips, etc.), in this paper we focus on minimizing the overall payment for serverless transcoders allocated to spectators to maximize their QoE.

Revenue is assumed to be generated by active spectators, who have a greater probability to remain active for longer if they experience a QoE close to the maximum afforded by their terminal and bandwidth, as discussed in Section 5.2.

However, catering to the needs of different and diverse spectators will require more, and possibly more expensive, transcoding profiles. Since our goal is to maximize profit for the provider, the network-centric optimization should serve spectators a better QoE only if this increase in QoE is expected to produce added revenue that exceeds any extra cost. Conversely, worsening QoE to save costs is justified only when this does not impact revenue too much by triggering too many spectators to quit the stream. In the course of our network-centring cost-efficiency optimization two sets of decisions must be taken *on-line,* based on the metrics reported by active spectators and models for spectator behavior, cost, revenue and QoE developed in previous subsections:

1. Which transcoding profiles should be deployed in production for each player, at each point in time, to minimize production costs?
2. Which of the produced profiles of each player should be allocated by the service provider to each spectator to maximize their QoE, thus maximizing revenue?

We now define our optimization problem more rigorously. Table 1 summarizes the notations that we use in problem formulation.

**Table 1**  Notation summary

| Notation | Description |
|---|---|
| *Sets* | |
| $K$ | playing users (players), $k \in K$ |
| $S$ | spectators $s \in S$; $S_t$ is the set of spectators at time $t$ |
| $N$ | transcoding profiles, $n \in N$ |
| | |
| *Sequences* | |
| $T = \{t_i\}_{i=0}^{|T|}$ | equidistant time steps $t_0, t_1, \ldots, |T|$, where $|T|$ is the maximal session lifetime |
| | |
| *Parameters* | |
| $q^s(QoE) = q_t^s$ | an estimated probability that spectator $s$ quits at time $t$ for a given $QoE$ value (See (2)) |
| $p_t^s = 1 - q_t^s$ | an estimated probability that spectator $s$ stays in the session at time $t$ |
| $\vec{n} = (m, g)$ | a resource demand vector of a transcoding profile $n$, where: |
| | $m$ is memory in GB, |
| | $g = \begin{cases} 1, & \text{if GPU should be allocated} \\ 0, & \text{otherwise} \end{cases}$ |
| $b_n$ | the average outbound bandwidth (GB/sec) of a transcoding profile $n$ |
| $b_t^s$ | the average inbound bandwidth (GB/sec) that spectator $s$ can contain at time $t$ |
| $c(\vec{n})$ | the cost (per second) of hosting a transcoding profile $n$ using FaaS |
| $o(b_n)$ | the cost (per GB) of the outbound traffic produced by a transcoding profile $n$ |
| $r_t^s(QoE)$ | the revenue generated by spectator $s$ at time $t$ for $QoE$ level (see (4), (5), (6)) |
| $QoE_{K \to N}^s$ | QoE of a spectator $s$ when consuming transcoding profiles allocation $K \to N$, $\forall k \in K$, $\forall n \in N$ |
| | |
| *Decision Variables* | |
| $y_{k \to n}^s$ | spectator consumption assignments: |
| | $y_{k \to n}^s = \begin{cases} 1, & \text{if spectator } s \text{ consumes profile } n \text{ for player } k \\ 0, & \text{otherwise} \end{cases}$ |
| $f_{k \to n}^s \in (0, 1)$ | a fraction of the nominal bandwidth consumption $b_n$ of $s$ reduced to match capacity |
| | |
| *Auxiliary Variables* | |
| $x_n^k$ | transcoding profile active status: |
| | $x_n^k = \begin{cases} 1, & \text{if transcoding profile } n \text{ is produced for player } k \\ 0, & \text{otherwise} \end{cases}$ |

**Given** a set of transcoding profiles $N$, a set of players $K$, and a set of spectators $S_{t_0}$ at time $t_0$,

**Determine** the transcoding profiles $x_n^k$ that should be produced and assign which of those produced each spectator should consume $y_{k \to n}^s$, so as to

**Maximize** an expected total profit (ETP) for the immersive gaming service provider.
Based on all of the above, the expected total profit is given by:

$$
\begin{aligned}
ETP = \; & \sum_{s \in S_t} E\left( r_{t \to |T|}^s \left( QoE_t^s(y_{\mathbf{K} \to \mathbf{N}}^s) \right) \right) \\
& - \sum_{k \in \mathbf{K}} \sum_{n \in \mathbf{N}} \left( c(\vec{n}) \cdot \left\lceil \frac{\sum_{s \in S_t} y_{k \to n}^s}{|S|} \right\rceil \right) \\
& - o \cdot \sum_{n \in \mathbf{N}} \left[ b_n \cdot \sum_{k \in \mathbf{K}} \sum_{s \in S_t} \left( y_{k \to n}^s - f_{k \to n}^s \right) \right]
\end{aligned}
\tag{11}
$$

**Subject to constraints**:

$$
\sum_{n \in \mathbf{N}} y_{k \to n}^s = 1, \; \forall s \in \mathbf{S_t}, \; \forall k \in \mathbf{K}, \; \forall t \in \mathbf{T}
\tag{12}
$$

$$
\sum_{k \in \mathbf{K}} \sum_{n \in \mathbf{N}} b_n \cdot \left( y_{k \to n}^s - f_{k \to n}^s \right) \leq b_t^s, \; \forall s \in \mathbf{S_t}, \; \forall t \in \mathbf{T}, \; f_{k \to n}^s \in (0, 1)
\tag{13}
$$

$$
y_{k \to n}^s \geq f_{k \to n}^s, \; \forall s \in \mathbf{S_t}, \; \forall k \in \mathbf{K}, \; \forall n \in \mathbf{N}, \; \forall t \in \mathbf{T}
\tag{14}
$$

Equation (11) gives the expected total profit (ETP) of the provider as the difference between the expected revenue and the costs. It consists of three terms: The **first term**, deriving from (8), sums the expected revenue for all spectators, which is a function of their QoE, which, in turn, depends on the profiles each of them is assigned to consume. The **second term** represents transcoding costs, summed for all players and profiles. For each player/profile combination, the ceiling function returns 1 if at least one spectator consumes that profile (and hence it is actually in production), and 0 if none do. The **third term** calculates traffic costs, summed for all spectators, players and profiles. Each profile $n$ has an average bandwidth requirement of $b_n$, which is the maximum consumed by a spectator $s$ who is receiving that profile from player $k$ (i.e. $y_{k \to n}^s = 1$). However, some of these spectators may be receiving $n$ at a lower framerate and thus consume less bandwidth; this reduction is expressed by $f_{k \to n}^s$.

Constraint (12) ensures that each spectator is allocated exactly one transcoding profile per each player. Constraint (13) makes sure that the total effective bandwidth consumed by any spectator at any given time instance does not exceed the maximum bandwidth that this spectator can contain. Finally, Constraint (14) prevents negative outbound traffic allocation.

---

**Algorithm 1** The overall smart network-centric optimization algorithmic framework.

---

**Input**: Transcoding Profiles $N$, Players $K$
**Output**:
      (1) Optimized Deployment of Transcoding Profiles $\forall t \in T$
      (2) Optimized Allocation of Transcoding Profiles to spectators $\forall t \in T$
QoE model $\leftarrow$ (10);
Costs model $\leftarrow$ pricing plan for GPU and CPU FaaS and outbound bandwidth;
Revenue model $\leftarrow$ e.g. 4;
**for** $t \leftarrow t_0$ **to** $|T|$ **do**
    | $Metrics \leftarrow$ collect metrics from the active spectators $S_t$;
    | Infer $b_t^s$ and compute power $\forall s \in S_t$;
    | Estimate $QoE_{K \to N}^s$, $\forall s \in S_t$ and $\forall K \to N$;
    | Estimate $q_t^s$ $\forall s \in S_t$;
    | $y_{k \to n}^s$, $f_{k \to n}^s \leftarrow$ solve the optimization, considering (11), (12), (13), (14);
    | Determine which transcoding profiles to activate for each player:

$$x_n^k = \begin{cases} 1, & \text{if } \exists\, y_{k \to n}^s = 1 \\ 0, & \text{otherwise} \end{cases}$$

    | Activate transcoders (via FaaS), as required;
    | Inform spectators about the new profile allocation;
**end**

---

Algorithm 1 depicts how we solve the provider problem in the on-line setting. Since in this setting the future is not known, we solve the optimization problem at every time step, at the end of which each spectator is assigned a set of transcoding profiles to consume. These directly influence their QoE and hence their probability of quitting. Future revenue is estimated based on the expected quitting probability of currently active spectators, assuming it will remain constant after the upcoming assignation of transcoding profiles to spectators. In each next time step we collect new metrics, note the arrival or departure of spectators, and solve the optimization problem again. Since the network conditions (as well as availability of the compute resources) might change from one time window to another for spectator $s$, the transcoding profile allocation for $s$ can also change. As we use FaaS, there are no additional costs associated with releasing serverless transcoders and starting new ones. Since in practical settings the optimization problem is relatively small it can be solved exactly either using linear solvers like CPLEX or even through brute force. In [7] this problem is approached using reinforcement learning, reducing its complexity.

Achieving cost-efficiency depends on an accurate modelling of the costs and revenues. The former depends on the available cloud and 5G MEC commercial offerings for FaaS. The latter depends on the spectators' behavior. The modeling approach of Sections 5.1–5.4 is relatively simple and generic, developed with the use-case of immersive 3D media live streaming in mind. Naturally, each use-case will have its own peculiarities, which will need to be modelled accurately and possibly fine-tuned using real data.

In this paper, our focus is on demonstrating that even for the relatively simple model, the serverless computing paradigm might result in significant benefits to the provider.

# 6 Experiments and results

We performed a series of experiments to validate the proposed optimization approach and quantify its benefits in different scenarios and conditions. Our experiments consider the aforementioned Augmented VR game use-case, in which spectators must receive two 3D video streams, one for each player.

## 6.1 Experimental setup

To develop and test the application functionality, all components of the service, as described in Section 4 and Fig. 3, were implemented, deployed and tested in the infrastructure provided by the 5G-MEDIA project[7] that offered Kubernetes NFVI with worker nodes equipped with NVIDIA GTX Geforce 1650 GPUs and Open Sorce MANO (OSM) R5.05 with FaaS VIM plugin installed.[8] Players, spectators, and the control plane have been deployed locally on PCs, while the broker, transcoders and replay application components have been deployed as FaaS VNFs via OSM/FaaS Plugin and orchestrated by the control plane in an event-driven manner using our Serverless Orchestration mechanism with OSM being a unified entry point.

However, the infrastructure we had access to has been relatively small and imposed hard limits on both the number of spectators and the number of concurrent transcoders that can use GPUs. Hence, after initial tests on the actual infrastructure, a more extensive study of cost optimization was conducted using simulation.

### 6.1.1 Simulated spectators

The simulated spectators adhere to joining and quitting behavior described in Sections 5.1.1 and 5.1.2. The experiments feature a diverse set of spectators, varying in connection bandwidth and processing capabilities, to reflect a mixture of real-life user profiles. For each spectator a set of metrics is collected every 10 seconds, reporting their bandwidth, processing power, the transcoding profiles they are currently receiving, and the framerate for each. Based on those metrics, each spectator's current QoE is calculated (from (10)), as well as an estimate of the QoE they would experience if they were to receive different transcoding profiles.

We consider relatively small sessions and assume that GPUs are available as needed in 5G MEC's NFVI.

Spectator bandwidth is subject to a small degree of random fluctuation, to simulate changing network conditions. Likewise, the processing power that can be allocated to the video processing is varied to simulate changing workload conditions of the user equipment. Processing power can impose a limit to the maximum framerate a spectator can decode.

### 6.1.2 Network optimization

The control plane receives metrics from all spectators and, based on them, decides the optimal set of transcoding profiles that must be produced, and which one of them each spectator should consume. The algorithm makes decisions at 10-second time-steps, corresponding to the 10-second monitoring intervals.

---

[7] http://www.5gmedia.eu/

[8] https://github.com/5g-media/faas-vim-plugin

We compare two optimization algorithms:

– *Naive Optimization* greedily optimizes spectator QoE. Based on the QoE modelling described in Section 5.3, it determines which transcoding profile will result in the optimal QoE for each spectator and allocates transcoders to produce this set of profiles, regardless of production costs.
– *Smart Network-Centric Optimization* optimizes cost-efficiency. It balances the trade-off between the cost and QoE. It considers spectator QoE, the quitting probability as a function of QoE, revenue generated by the spectators remaining online, and production and delivery costs, and determines the set of transcoding profiles to be produced to maximize cost (see (11), (12), (13), (14)). In particular, expected revenue (8) is calculated on the assumption that profiles assigned to spectators during the current time step will also persist for future time steps.

In the following text, we will refer to these two algorithms simply as *Naive* and *Smart*.

### 6.1.3 Transcoder parameters

For each of the two players' 3D video streams five transcoding profiles are supported, in addition to the production streams, which are also available for consumption by the spectators and require no transcoding.

The production stream and all still image profiles encode textures as JPEG images of quality 30, in various resolutions. The video profiles encode textures as a HEVC video of fixed resolution, targeting various bit-rates. Production framerate is set to be 25 frames per second. Table 2 lists the specifications of transcoding profiles used in our experiments.

Still image profiles produce slightly worse image quality and, naturally, a significantly larger average frame size. They can be transcoded in real time on a CPU node. Since they have no inter-frame decoding dependency, spectators who fail to receive or decode the frames at the production framerate can skip frames to always display the most current frame.

On the other hand, video profiles achieve better image quality with much higher compression rates, and require a GPU node for real-time transcoding. In the test implementation, they do not support skipping frames, due to inter-frame compression. Spectators who cannot match the production framerate may start lagging behind, so the optimization algorithm running in the control plane will never assign a video profile to such a spectator.

**Table 2** Transcoding profiles' specifications

| Name | Node | Frame size | Texture | | Mesh | |
| | | | Resolution | PSNR | Geometry | Blend weights |
| --- | --- | --- | --- | --- | --- | --- |
| Production | None | 200 KB | 960x540 | 32.02 dB | 10 bits | 6 bits |
| Images Mid | CPU | 170 KB | 864x486 | 28.78 dB | 9 bits | 5 bits |
| Images Low | CPU | 135 KB | 768x432 | 28.02 dB | 8 bits | 4 bits |
| Video Low | GPU | 55 KB | 960x540 | 28.66 dB | 8 bits | 4 bits |
| Video Mid | GPU | 70 KB | 960x540 | 30.00 dB | 9 bits | 5 bits |
| Video High | GPU | 85 KB | 960x540 | 31.59 dB | 10 bits | 6 bits |

## 6.1.4 Experiment timeline

Each experiment considers a streaming session of ten minutes. The stream starts with a number of spectators equal to the average arrival in 20 time-steps, the default being 10 spectators. New spectators join the stream according to the Poisson process (See Section 5.1.1) and leave the stream according to their probability of quitting (see Section 5.1.2).

Figure 5 shows the timeline of a sample experiment featuring the Smart optimization in the control plane. Cost-related quantities (revenue, cost, and profit) correspond to the upper left vertical axis and are shown in a per-time-step basis. Initially revenue is low, due to a low number of initial spectators, resulting in low profit, at times dipping into the negative, for the first couple of minutes. As the session progresses, more spectators join, gradually increasing revenue and profit.

Node costs are significant in the beginning of the session, becoming less so as more spectators join and revenue increases. They remain more or less constant, while random changes in spectator bandwidth or processing power occasionally result in the production of an extra set of GPU profiles, when it is deemed profitable. Towards the end of the session node costs dip lower, as the expected future revenue of individual spectators diminishes, capped by the diminishing remaining duration. The higher cost of video profiles is partially offset by their lower bitrate, which result in lower traffic costs. The latter naturally increase as more spectators join, but at a much lower rate, reaching a plateau after about four minutes, when the active spectator population can justify the production of more video profiles.

Average QoE follows an upward trend. As more spectators join, the increased revenue can support the production of more transcoding profiles, able to satisfy a more diverse population.
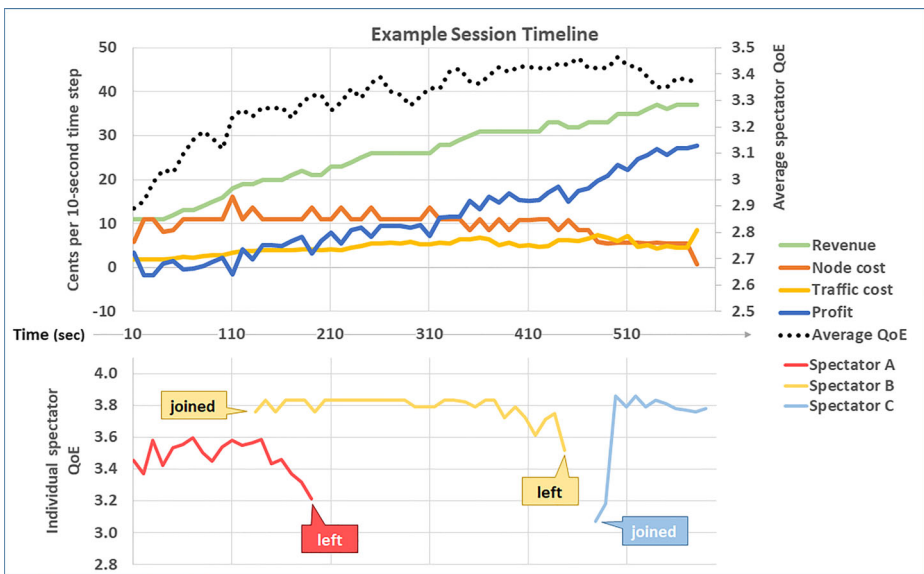


**Fig. 5** A sample timeline of an experiment. Above the horizontal axis: revenue, node cost, traffic cost, profit (left vertical axis) and mean QoE (right vertical axis). Below the horizontal axis: Individual QoE progress for three example spectators

The part of the graph below the horizontal axis shows the QoE progress of three sample spectators. Although the behavior of individual spectators has very small impact to the total revenue and profit in a session of about 20 active spectators, these examples can provide some intuition about the progress of a session. Spectator A has joined from the start; she has limited bandwidth with significant fluctuations. Sometimes she is unable to receive the better quality profiles, and her QoE drops as a result. Finally, when it drops too low, she decides to leave. Her departure can be seen marking a small decrease in the revenue and profit of the next couple of time steps. Spectator B joins in the middle of the stream and experiences only very minor fluctuations. After some time he leaves, perhaps for non-QoE-related reasons, as his QoE is not so low. When spectator C joins, she is experiencing a quite low QoE. However, the optimization algorithm quickly assigns an appropriate transcoding profile that maximizes her QoE, and so she remains active until the end of the stream.

Abrupt changes in individual spectators' QoE are not always reflected in the average QoE or the profit, meaning that the optimizer made a decision to lower some spectators' QoE and raise that of the others, aiming for maximum expected profit.

### 6.1.5 Experiment variables

In order to obtain general results and compare the Smart and Naive optimization adaptability under different client conditions, experiments were performed using a range of values for different experiment variables. Each set of experiments measured the impact of changing one variable while keeping the others constant at their default values. Experiment variables included:

– **Spectator arrival rate**, following the Poisson distribution. Default value of 0.5 new spectators per time step.
– **Revenue** generated by each spectator. Default value of 0.2 cents per time step.
– **Numbers of GPUs available**. The three video transcoding profiles require a GPU to perform in real time. A limitation on the number of available GPUs implies that some profiles may not be able to be produced concurrently. The default number is 6, i.e. no limitation on the concurrent use of GPUs, for 2 players at 3 GPU profiles each.
– **GPU costs**. As mentioned before, there is currently no commercial option to rent GPU processing for FaaS. Based on calculations derived from current CPU and GPU pricing for VMs, and considering the implementation obstacles in GPU sharing, we estimate a default value of 10 times that of an otherwise equivalent CPU node, and test for factors between x5 and x20, which seem reasonable, considering the analysis in Section 5.4.1.
– **Spectator population**. We identify 5 broad types of spectators:

  - Mobile devices on Wi-Fi.
  - Mobile devices on 4G data.
  - Standard PC on basic DSL connection.
  - Standard PC on faster connections (e.g. VDSL)
  - High-end PC on a fiber optic connection.

In the preliminary experiments, average decoding timings for all transcoding profiles were measured for each device type, and these are used to calculate a maximum frame rate from a hardware perspective. In addition, each connection type is associated with a bandwidth typical for it, which provides a frame rate cap from a connection perspective. The default population consists of a balanced mix of the above spectator types, while we also conduct experiments where specific types of spectators are dominant.

– **Quitting behavior**. As mentioned in Section 5.1.2, quitting probability is a function of QoE dissatisfaction and non-QoE-related causes, such as how interesting a specific session is. As the default, derived from [15], we assume a 20% probability to quit before the session's end at maximum QoE. We conduct experiments for relatively boring (50% to quit) or interesting sessions (10% to quit), and also for more demanding spectators, in which case QoE dissatisfaction weighs more.

## 6.2 Results

In the following graphs we present a comparison between Smart and Naive optimizations. Each graph displays a number of experiments differing at one experiment variable (see Section 6.1.5), shown on the horizontal axis while keeping the others constant. The graphs show aggregate measurements of the entire 10-minute sessions, averaged across several runs of the same experiment.

Quantities denoted with an *S* refer to Smart and are shown in a lighter shade, while those denoted with an *N* refer to the Naive. Across all graphs, money-related quantities (revenue, cost, and profit) are shown as bars and correspond to the left vertical axis. An additional quantity, relevant to each graph, is shown as a line corresponding to the right vertical axis, including:

– Spectators: The average number of spectators active during the session. This directly impacts revenue.
– dQoE: the average difference between spectators' actual QoE and the maximum QoE they could possibly achieve, given their network connection and hardware. This directly impacts the quitting probability, which indirectly affects revenue. dQoE is shown in an inverted vertical axis.

### 6.2.1 Arrival rate

Figure 6 compares performance of Naive and Smart under different spectator arrival rates. As the rate increases, so does the average number of spectators staying in the session and
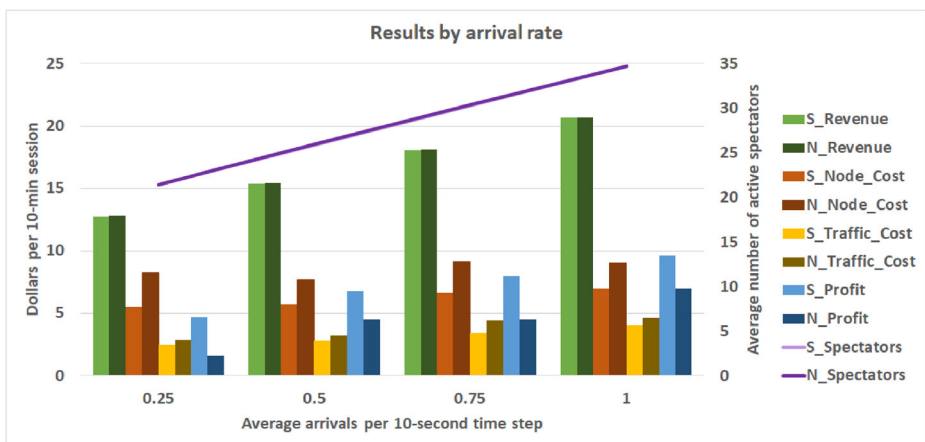


**Fig. 6** Total session revenue, costs and profit (on the left vertical axis) and average spectator number (on the right vertical axis) for different rates of new spectator arrival

the revenue they generate. Traffic costs also naturally increase, as there are more spectators downloading the streams. With more spectators, Smart's spending on transcoding nodes also increases slightly, as the higher costs of producing more profiles is offset by a larger number of spectators who will benefit from them. It can be noted that, as expected, the Smart's advantage is more pronounced when less spectators are active.

### 6.2.2 Revenue rate

Figure 7 presents the experimental results for different revenue rates around the default of 1 cent per 10-second time step. This set of experiments follows the default arrival rate of 0.5, meaning that revenue is generated by an average of about 25 spectators per session.

Naturally, as the revenue generated by each active spectator increases, so too does the overall revenue. The costs also increase, as it is becomes more profitable to keep spectators satisfied. This is also reflected on the decreasing dQoE, shown on the right vertical axis. At lower revenue rates Smart makes a greater difference in profit. A similar behavior of Smart is observed with the increasing arrival rate as shown in Fig. 6.

### 6.2.3 Available GPUs

This series of experiments, shown in Fig. 8, considers the case where production GPUs availability is limited, capping the number of video profiles that can be transcoded simultaneously. As the number of available GPUs drops, so do the options and versatility of Smart, limiting its benefit.

### 6.2.4 GPU node cost

As mentioned before, to the best of our knowledge there is currently no commercial option to rent GPU nodes for FaaS processing. However, it is entirely possible that such options will be available in the near future, especially if demand for it rises. GPU processing will certainly cost more than CPU processing. This set of experiments examines the impact of
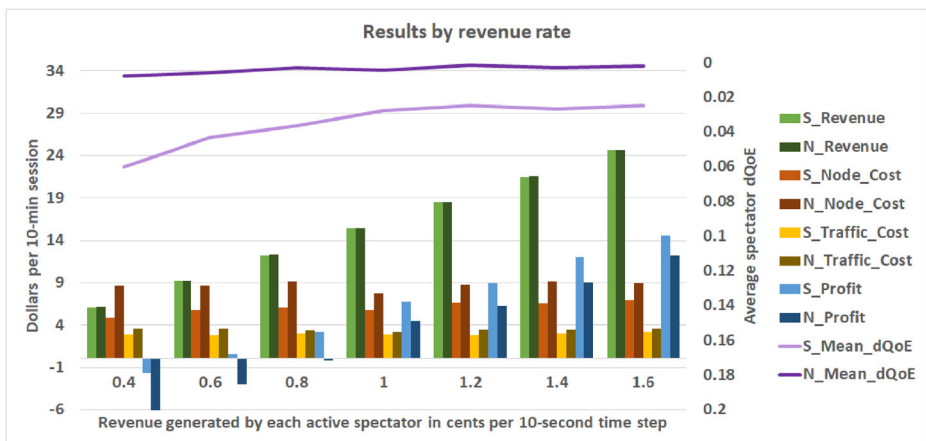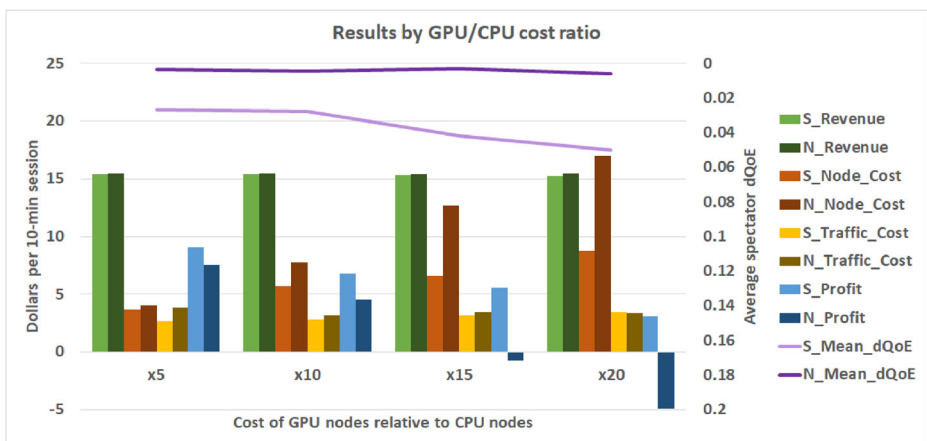


**Fig. 7** Total session revenue, costs and profit (on the left vertical axis) and average difference from the maximum possible QoE (on the right vertical axis) for the different (constant) rates of revenue per active spectator and per time step

**Fig. 8** Total session revenue, costs and profit (on the left vertical axis) and average difference from the maximum possible QoE (on the right vertical axis) when different numbers of GPUs are available for the transcoding of streams to video in real time

the price ratio between GPU and CPU nodes. As seen on Fig. 9, as GPU processing becomes more expensive, Smart becomes more frugal with GPU-dependent profiles, letting spectator QoE drop away from the optimal. In this way, it can keep running costs manageable and generate a profit even when GPU utilization is priced high, with a small decrease in QoE. It can be noted that although at low GPU pricing optimization offers a relatively small benefit, this becomes much more emphasized when GPU usage is more expensive. Also note how Smart's traffic costs get higher as GPU cost increases and still-image CPU profiles, which have a lower compression rate and thus higher bitrate, are preferred.



**Fig. 9** Total session revenue, costs and profit (on the left vertical axis) and average difference from the maximum possible QoE (on the right vertical axis) if we assume different ratios in the prices of GPU nodes to CPU nodes. Ratios of x5 - x20 seem reasonable, based on current VM rental prices
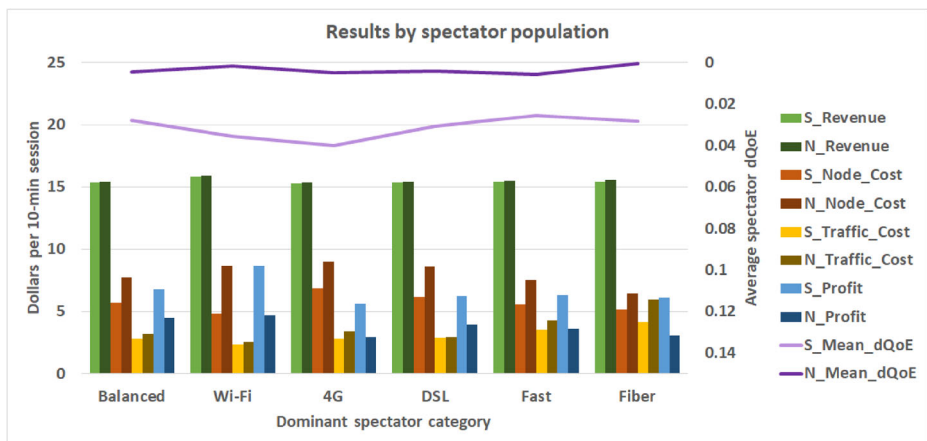
### 6.2.5 Spectator population

Figure 10 regards experiments with different spectator populations. Although all experiments contain all types of spectators, in this set we examine the impact of having different dominant spectator types. Smart holds a steady advantage across all cases. The two rightmost sets of bars, corresponding to a greater percentage of faster connections, shows a marked decrease in node costs, offset by an increase in traffic costs, as many of those spectators can consume the high-bitrate production stream, obviating the need for (and cost of) transcoding.
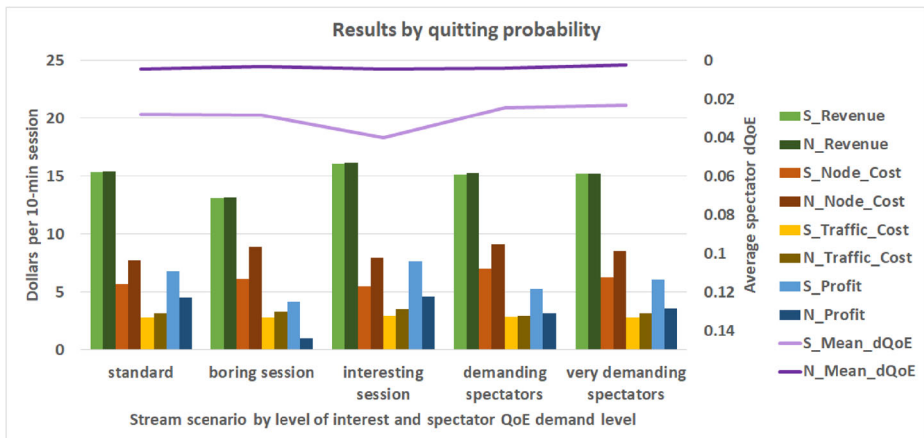
### 6.2.6 Quitting probability

This set of experiments considers how quitting probability impacts Smart's decisions. Figure 11 shows measurements for standard boring and interesting sessions, in which quitting probability is respectively higher or lower; and also how more demanding spectators (in which case QoE dissatisfaction weighs more in their probability of quitting) affect the process. In a boring session, Smart increases spending in an effort to keep spectators from quitting via providing a better QoE. On the contrary, in the interesting session Smart reduces costs and allows QoE to drop, as spectators are less likely to leave anyway. With more demanding spectators, Smart targets higher QoE to keep them engaged, resulting in higher transcoding node costs.

### 6.2.7 Summary of the results

Having conducted experiments with parameters spanning numerous different assumptions and cases, some overall conclusions may be reached. Smart QoE-cost optimization can reduce transcoding costs by up to 60% and traffic costs by about 20%, while keeping revenue and QoE very close to the optimum. Optimization's benefits are especially pronounced in cases with few spectators, low revenue and high GPU costs. When GPUs become available for FaaS, possibly in the near future, they can be expected to start at higher prices,



**Fig. 10** Total session revenue, costs and profit (on the left vertical axis) and total aggregate QoE (on the right vertical axis) for spectator populations where different types are more frequent

**Fig. 11** Total session revenue, costs and profit (on the left vertical axis) and average difference from the maximum possible QoE (on the right vertical axis) when different factors affect spectator quitting probability, including how interesting or boring the session is and how demanding the spectators

gradually dropping as their use becomes more common. Live streaming platforms, especially those dealing in emerging media, will likely need to start with a small spectator base before gaining momentum and scaling. Hence, smart QoE-cost optimization will be indispensable in the live media streaming landscape of the near future.

## 7 Conclusion

5G networks will disrupt the way media intensive services are being developed and operated by unlocking a plethora of opportunities to both service developer and service operator. In this work, we studied one such capability that integrates modern serverless technology with real-time adaptive media streaming in 5G MEC. To the best of our knowledge, this is the first work that does this.

Apart from the conceptual, architectural and technical contributions, our work further examined the potential of this option in terms of network-centric service cost optimization. Our findings indicate that for small user populations and finite duration sessions, serverless adaptive streaming can offer reduced operating expenditure (OPEX) while preserving the service's QoE.

Also, through our extensive modelling and analysis we concluded that naively applying serverless will not necessarily offer these gains. We hope that our work will inspire further research and development towards adapting services not originally suited for lighter-weight virtualization to serverless architectures, and unify them with the advanced capabilities that 5G networks offer to capitalize on its advantages in novel ways. Finally, taking into account the recent introduction and the emerging availability of GPUs specifically designed for data centers our work can be extended to accommodate these developments. Specifically for media services, GPU slicing can allow for even finer-grained cost optimization, something that was not possible before.

One interesting future work direction is to explore more sophisticated placement schemes for transcoders and other components, in which they can be spread across the full compute

spectrum across cloud and edge to leverage differentiated pricing for compute, storage and network resources to meet demanding KPIs at lower price points.

**Compliance with Ethical Standards**

**Conflict of interests** The authors declare that they have no conflict of interest.

# References

1. Alexiadis D, Chatzitofis A, Zioulis N, Zoidi O, Louizis G, Zarpalas D, Daras P (2017) An integrated platform for live 3D human reconstruction and motion capturing. IEEE Trans Circ Syst Video Technol 27(4):798–813. https://doi.org/10.1109/TCSVT.2016.2576922
2. Alvarez F, Breitgand D, Griffin D, Andriani P, Rizou S, Zioulis N, Moscatelli F, Serrano J, Keltsch M, Trakadas P et al (2019) An edge-to-cloud virtualized multimedia service platform for 5g networks. IEEE Trans Broadcast 65(2):369–380
3. Ao L, Izhikevich L, Voelker GM, Porter G (2018) Sprocket: a serverless video processing framework. In: Proceedings of the ACM symposium on cloud computing (SoCC '18). https://doi.org/10.1145/3267809.3267815, http://par.nsf.gov/biblio/10098946
4. Apache OpenWhisk: Open Source Serverless Cloud Platform. https://openwhisk.apache.org/
5. Argo Events Team: Argo Events - The Event-driven Workflow Automation Framework. https://github.com/argoproj/argo-events
6. Argo Team: Argo Workflows. https://github.com/argoproj/argo
7. Athanasoulis P, Christakis E, Konstantoudakis K, Drakoulis P, Rizou S, Weit A, Doumanoglou A, Zioulis N, Zarpalas D (2020) Optimizing qoe and cost in a 3d immersive media platform: a reinforcement learning approach. In: MMEDIA 2020: the twelfth international conference on advances in multimedia. IARIA
8. Ballard T, Griwodz C, Steinmetz R, Rizk A Rats: adaptive 360-degree live streaming. In: Proceedings of the 10th ACM multimedia systems conference, MMSys '19, p 308–311. Association for Computing Machinery (2019). https://doi.org/10.1145/3304109.3323837. Event-place: Amherst, Massachusetts
9. Barman N, Martini MG (2019) QoE modeling for HTTP adaptive video streaming–a survey and open challenges. IEEE Access 7:30831–30859. https://doi.org/10.1109/ACCESS.2019.2901778. Conference Name: IEEE Access
10. Bentaleb A, Taani B, Begen AC, Timmerer C, Zimmermann R (2018) A survey on bitrate adaptation schemes for streaming media over http. IEEE Commun Sur Tutor 21(1):562–585
11. Bentaleb A, Taani B, Begen AC, Timmerer C, Zimmermann R (2019) A survey on bitrate adaptation schemes for streaming media over HTTP. IEEE Commun Surv Tutor 21(1):562–585. https://doi.org/10.1109/COMST.2018.2862938. Conference Name: IEEE Communications Surveys Tutorials
12. Bhargava A, Martin J, Babu SV Comparative evaluation of user perceived quality assessment of design strategies for HTTP-based adaptive streaming (2019). https://doi.org/10.1145/3345313
13. Breitgand D, Weit A (2019) Using gpus with apache openwhisk. https://medium.com/openwhisk/using-gpus-with-apache-openwhisk-c6773efcccfb
14. Chakareski J, Aksu R, Corbillon X, Simon G, Swaminathan V (2018) Viewport-driven rate-distortion optimized 360° video streaming. In: 2018 IEEE International conference on communications (ICC), pp 1–7. https://doi.org/10.1109/ICC.2018.8422859
15. Chen Y, Zhang F, Wu K, Zhang Q (2015) Qoe-aware dynamic video rate adaptation. In: 2015 IEEE Global communications conference (GLOBECOM). IEEE, pp 1–6
16. Chou PA, Koroteev M, Krivokuća M (2020) A volumetric approach to point cloud compression—part i: attribute compression. IEEE Trans Image Process 29:2203–2216. https://doi.org/10.1109/TIP.2019.2908095
17. Christaki K, Apostolakis KC, Doumanoglou A, Zioulis N, Zarpalas D, Daras P (2019) Space wars: an augmentedvr game. In: International conference on multimedia modeling. Springer, pp 566–570

18. Collet A, Chuang M, Sweeney P, Gillett D, Evseev D, Calabrese D, Hoppe H, Kirk A, Sullivan S High-quality streamable free-viewpoint video (2015). https://doi.org/10.1145/2766945
19. Crowle S, Doumanoglou A, Poussard B, Boniface M, Zarpalas D, Daras P (2015) Dynamic adaptive mesh streaming for real-time 3d teleimmersion. In: Proceedings of the 20th international conference on 3D web technology, Web3D '15. Association for Computing Machinery, pp 269–277. https://doi.org/10.1145/2775292.2775296. Event-place: Heraklion, Crete, Greece
20. Cui L, Mekuria R, Preda M, Jang E (2019) Point-cloud compression: moving picture experts group's new standard in 2020. IEEE Consum Electron Mag 8(4):17–21. https://doi.org/10.1109/MCE.2019.2905483
21. Ding AY, Janssen M (2018) 5g applications: requirements, challenges, and outlook. arXiv:1810.06057
22. Domański M, Stankiewicz O, Wegner K, Grajek T (2017) Immersive visual media — mpeg-i: 360 video, virtual navigation and beyond. In: 2017 International conference on systems, signals and image processing (IWSSIP), pp 1–9
23. Dou M, Khamis S, Degtyarev Y, Davidson P, Fanello SR, Kowdle A, Escolano SO, Rhemann C, Kim D, Taylor J, Kohli P, Tankovich V, Izadi S (2016) Fusion4D: real-time performance capture of challenging scenes. https://doi.org/10.1145/2897824.2925969
24. Doumanoglou A, Alexiadis DS, Zarpalas D, Daras P (2014) Toward real-time and efficient compression of human time-varying meshes. IEEE Trans Circ Syst Video Technol 24(12):2099–2116. https://doi.org/10.1109/TCSVT.2014.2319631
25. Doumanoglou A, Drakoulis P, Zioulis N, Zarpalas D, Daras P (2019) Benchmarking open-source static 3d mesh codecs for immersive media interactive live streaming. IEEE J Emerg Selected Topics Circ Syst 9(1):190–203. https://doi.org/10.1109/JETCAS.2019.2898768
26. Doumanoglou A, Griffin D, Serrano J, Zioulis N, Phan TK, Jiménez D, Zarpalas D, Alvarez F, Rio M, Daras P (2018) Quality of experience for 3-D immersive media streaming. IEEE Trans Broadcast 64(2):379–391. https://doi.org/10.1109/TBC.2018.2823909. Conference Name: IEEE Transactions on Broadcasting
27. Doumanoglou A, Zioulis N, Griffin D, Serrano J, Phan TK, Jiménez D, Zarpalas D, Alvarez F, Rio M, Daras P (2018) A system architecture for live immersive 3d-media transcoding over 5g networks. In: 2018 IEEE International symposium on broadband multimedia systems and broadcasting (BMSB). IEEE, pp 11–15
28. El Marai O, Taleb T, Menacer M, Koudil M (2018) On improving video streaming efficiency, fairness, stability, and convergence time through client–server cooperation. IEEE Trans Broadcast 64(1):11–25. https://doi.org/10.1109/TBC.2017.2781146
29. ETSI: Multi-access Edge Computing (MEC); Framework and Reference Architecture; GS MEC 003 V2.1.1 (2019-01) (2019). https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf
30. Fan CL, Lo WC, Pai YT, Hsu CH (2019) A survey on 360° video streaming: acquisition, transmission, and display. ACM Comput Surv, 52(4). https://doi.org/10.1145/3329119
31. Flannel, Kubernetes Networking. https://github.com/coreos/flannel#flannel
32. Girinathan J, Breckinridge R (2018) Simple serverless video on demand (vod) workflow. https://aws.amazon.com/blogs/networking-and-content-delivery/serverless-video-on-demand-vod-workflow/
33. Graf M, Timmerer C, Mueller C (2017) Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP: design, implementation, and evaluation. In: Proceedings of the 8th ACM on multimedia systems conference - MMSys'17. ACM Press, Taipei, pp 261–271. https://doi.org/10.1145/3083187.3084016
34. Hannuksela MM, Wang YK, Hourunranta A An overview of the OMAF standard for 360° video. In: 2019 Data compression conference (DCC), pp 418–427 (2019). https://doi.org/10.1109/DCC.2019.00050. ISSN: 2375-0359
35. He J, Qureshi M, Qiu L, Li J, Li F, Han L (2018) Rubiks: practical 360-degree streaming for smartphones, p 482–494. https://doi.org/10.1145/3210240.3210323
36. Hoßfeld T, Seufert M, Sieber C, Zinner T, Tran-Gia P (2015) Identifying QoE optimal adaptation of HTTP adaptive streaming based on subjective studies. Comput Netw 81:320–332. https://doi.org/10.1016/j.comnet.2015.02.015, http://www.sciencedirect.com/science/article/pii/S1389128615000626
37. Hosseini M (2019) Adaptive rate allocation for view-aware point-cloud streaming. arXiv:abs/1911.00812
38. Hosseini M, Swaminathan V (2016) Adaptive 360 vr video streaming: divide and conquer. In: 2016 IEEE International symposium on multimedia (ISM), pp 107–110. https://doi.org/10.1109/ISM.2016.0028
39. IBM Cloud Functions. https://cloud.ibm.com/functions/
40. ISO/IEC: Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 5: Server and network assisted DASH (SAND). https://www.iso.org/standard/69079.html

41. Jain R (1990) The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling. Wiley. https://books.google.co.il/books?id=eOR0kJjgMqkC

42. Karakottas A, Papachristou A, Doumanoqlou A, Zioulis N, Zarpalas D, Daras P (2018) Augmented vr. In: 2018 IEEE Conference on virtual reality and 3d user interfaces (VR), pp 1–1

43. Kazhdan M (2005) Reconstruction of solid models from oriented point sets. In: Proceedings of the third eurographics symposium on geometry processing. Eurographics Association, p 73

44. Kazhdan M, Bolitho M, Hoppe H (2006) Poisson surface reconstruction. In: Proceedings of the fourth eurographics symposium on geometry processing, vol 7

45. Kleinrock L (1976) Queueing systems. Wiley

46. Kritikos K, Skrzypek P (2018) A review of serverless frameworks. In: 2018 IEEE/ACM international conference on utility and cloud computing companion (UCC companion). IEEE, pp 161–168

47. Krivokuća M, Chou PA, Koroteev M (2020) A volumetric approach to point cloud compression–part ii: geometry compression. IEEE Trans Image Process 29:2217–2229. https://doi.org/10.1109/TIP.2019.2957853

48. Kubernetes: Production-grade container orchestrator. https://kubernetes.io/

49. Li Z, Zhu X, Gahm J, Pan R, Hu H, Begen AC, Oran D (2014) Probe and adapt: rate adaptation for HTTP video streaming at scale. IEEE J Selected Areas Commun 32(4):719–733. https://doi.org/10.1109/JSAC.2014.140405. ArXiv:1305.0510

50. Maglo A, Lavoué G., Dupont F, Hudelot C (2015) 3d mesh compression: survey, comparisons, and emerging trends. ACM Comput Surv 47(3). https://doi.org/10.1145/2693443

51. Mao H, Netravali R, Alizadeh M (2017) Neural adaptive video streaming with pensieve. In: Proceedings of the conference of the ACM special interest group on data communication, SIGCOMM '17. Association for Computing Machinery, Los Angeles, pp 197–210. https://doi.org/10.1145/3098822.3098843

52. Mehrabi A, Siekkinen M, Ylä-Jääski A (2017) Joint optimization of qoe and fairness through network assisted adaptive mobile video streaming. In: 2017 IEEE 13th International Conference On Wireless And Mobile Computing, Networking And Communications (WiMob). IEEE, pp 1–8

53. Mehrabi A, Siekkinen M, Yla-Jaaski A (2019) Edge computing assisted adaptive mobile video streaming. IEEE Trans Mob Comput 18(4):787–800. https://doi.org/10.1109/TMC.2018.2850026

54. Mekuria R, Blom K, Cesar P (2017) Design, implementation, and evaluation of a point cloud codec for tele-immersive video. IEEE Trans Circ Syst Video Technol 27(4):828–842. https://doi.org/10.1109/TCSVT.2016.2543039

55. Misra K, Segall A, Horowitz M, Xu S, Fuldseth A, Zhou M (2013) An overview of tiles in HEVC. IEEE J Selected Topics Signal Process 7(6):969–977. https://doi.org/10.1109/JSTSP.2013.2271451. Conference Name: IEEE Journal of Selected Topics in Signal Processing

56. New European Media (NEM): 5G-MEDIA Slice Definition. https://bscw.5g-ppp.eu/pub/bscw.cgi/d322688/NEM%20Networld2020%205G%20media%20slice%20V1-2_24092019.pdf (2019)

57. NVIDIA: Multi-process service. https://docs.nvidia.com/deploy/mps/

58. Orts-Escolano S, Rhemann C, Fanello S, Chang W, Kowdle A, Degtyarev Y, Kim D, Davidson PL, Khamis S, Dou M, Tankovich V, Loop C, Cai Q, Chou PA, Mennicken S, Valentin J, Pradeep V, Wang S, Kang SB, Kohli P, Lutchyn Y, Keskin C, Izadi S (2016) Holoportation: virtual 3D teleportation in real-time. In: Proceedings of the 29th annual symposium on user interface software and technology, UIST '16. Association for Computing Machinery, Tokyo, pp 741–754. https://doi.org/10.1145/2984511.2984517

59. Pantos R, May W (2017) HTTP live streaming. Tech. Rep. RFC8216, RFC Editor. https://doi.org/10.17487/RFC8216. https://www.rfc-editor.org/info/rfc8216

60. Park J, Chou PA, Hwang JN (2018) Volumetric media streaming for augmented reality. In: 2018 IEEE Global communications conference (GLOBECOM), pp 1–6. https://doi.org/10.1109/GLOCOM.2018.8647537

61. Paudyal P, Battisti F, Carli M (2016) Impact of video content and transmission impairments on quality of experience. Multimed Tools Applic 75(23):16461–16485. https://doi.org/10.1007/s11042-015-3214-0

62. Robitza W, Garcia MN, Raake A (2017) A modular http adaptive streaming qoe model—candidate for itu-t p 1203 ("p nats"). In: 2017 Ninth international conference on quality of multimedia experience (QoMEX). IEEE, pp 1–6

63. Sami Kekki EA (2018) MEC in 5G networks, ETSI White Paper No. 28. https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

64. Schatz R, Sackl A, Timmerer C, Gardlo B (2017) Towards subjective quality of experience assessment for omnidirectional video streaming. In: 2017 Ninth international conference on quality of multimedia experience (QoMEX), pp 1–6. https://doi.org/10.1109/QoMEX.2017.7965657. ISSN: 2472-7814

65. Schatz R, Zabrovskiy A, Timmerer C (2019) Tile-based streaming of 8K omnidirectional video: subjective and objective QoE evaluation. In: 2019 Eleventh international conference on quality of multimedia experience (QoMEX), pp 1–6. https://doi.org/10.1109/QoMEX.2019.8743230. ISSN: 2472-7814

66. Schoeffelen T (2020) Designing a serverless video streaming pipeline. https://medium.com/@tschoffelen/designing-a-serverless-video-streaming-pipeline-2d3828f3ccf8

67. Schreer O, Feldmann I, Renault S, Zepp M, Worchel M, Eisert P, Kauff P (2019) Capture and 3D video processing of volumetric video. In: 2019 IEEE International conference on image processing (ICIP), pp 4310–4314. https://doi.org/10.1109/ICIP.2019.8803576. ISSN: 2381-8549

68. Service Development Kit for Media-Type Virtualized Network Services in 5G Networks (to appear). IEEE Communication Magazine (2020)

69. Seufert M, Egger S, Slanina M, Zinner T, Hoßfeld T, Tran-Gia P (2015) A survey on quality of experience of HTTP adaptive streaming. IEEE Commun Surveys Tutor 17(1):469–492. https://doi.org/10.1109/COMST.2014.2360940. Conference Name: IEEE Communications Surveys Tutorials

70. Singla A, Göring S, Raake A, Meixner B, Koenen R, Buchholz T (2019) Subjective quality evaluation of tile-based streaming for omnidirectional videos. In: Proceedings of the 10th ACM multimedia systems conference, MMSys '19. Association for Computing Machinery, Amherst, pp 232–242. https://doi.org/10.1145/3304109.3306218

71. Skupin R, Sanchez Y, Podborski D, Hellge C, Schierl T (2017) Viewport-dependent 360 degree video streaming based on the emerging Omnidirectional Media Format (OMAF) standard. In: 2017 IEEE international conference on image processing (ICIP), pp 4592–4592. https://doi.org/10.1109/ICIP.2017.8297155. ISSN: 2381-8549

72. Sodagar I (2011) The MPEG-DASH standard for multimedia streaming over the internet. IEEE MultiMedia 18(4):62–67. https://doi.org/10.1109/MMUL.2011.71. Conference Name: IEEE MultiMedia

73. Soltanian A, Naboulsi D, Salahuddin MA, Glitho R, Elbiaze H, Wette C (2018) Ads: adaptive and dynamic scaling mechanism for multimedia conferencing services in the cloud. In: 2018 15th IEEE Annual consumer communications & networking conference (CCNC). IEEE, pp 1–6

74. Spiteri K, Urgaonkar R, Sitaraman RK (2016) BOLA: near-optimal bitrate adaptation for online videos. arXiv:1601.06748 [cs]

75. Sterzentsenko V, Karakottas A, Papachristou A, Zioulis N, Doumanoglou A, Zarpalas D, Daras P (2018) A low-cost, flexible and portable volumetric capturing system, 200–207. https://doi.org/10.1109/SITIS.2018.00038

76. Sterzentsenko V, Karakottas A, Papachristou A, Zioulis N, Doumanoglou A, Zarpalas D, Daras P (2018) A low-cost, flexible and portable volumetric capturing system. In: 2018 14th International conference on signal-image technology & internet-based systems (SITIS). IEEE, pp 200–207

77. Sullivan G, Ohm JR, Han WJ, Wiegand T (2012) Overview of the high efficiency video coding (HEVC) standard. IEEE Trans Circ Syst Video Technol 22(12):1649–1668. https://doi.org/10.1109/TCSVT.2012.2221191

78. Sun L, Duanmu F, Liu Y, Wang Y, Ye Y, Shi H, Dai D (2018) Multi-path multi-tier 360-degree video streaming in 5g networks. In: Proceedings of the 9th ACM multimedia systems conference, MMSys '18. Association for Computing Machinery, pp 162–173. https://doi.org/10.1145/3204949.3204978. Eventplace: Amsterdam, Netherlands

79. Tian Y, Babcock R, Taylor C, Ji Y (2018) A new live video streaming approach based on amazon s3 pricing model. In: 2018 IEEE 8th Annual computing and communication workshop and conference (CCWC), pp 321–328. https://doi.org/10.1109/CCWC.2018.8301615

80. van der Hooft J, Wauters T, De Turck F, Timmerer C, Hellwagner H (2019) Towards 6DoF HTTP adaptive streaming through point cloud compression. In: Proceedings of the 27th ACM international conference on multimedia, MM '19. Association for Computing Machinery, Nice, pp 2405–2413. https://doi.org/10.1145/3343031.3350917

81. Wien M, Boyce JM, Stockhammer T, Peng WH (2019) Standardization status of immersive video coding. IEEE J Emerg Selected Topics Circ Syst 9(1):5–17

82. Xie L, Xu Z, Ban Y, Zhang X, Guo Z (2017) 360probdash: improving qoe of 360 video streaming using tile-based http adaptive streaming. In: Proceedings of the 2017 ACM on multimedia conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017. ACM, pp 315–323. https://doi.org/10.1145/3123266.3123291

83. Xin Z, Fu S (2019) User-centric qoe model of visual perception for mobile videos. Vis Comput 35(9):1245–1254

84. Yamasaki T, Aizawa K (2010) Patch-based compression for time-varying meshes. In: 2010 IEEE International conference on image processing, pp 3433–3436. https://doi.org/10.1109/ICIP.2010.5652911

85. Yates RD, Goodman DJ (2014) Probability and stochastic processes: a friendly introduction for electrical and computer engineers. Wiley

86. Zadtootaghaj S, Schmidt S, Möller S (2018) Modeling gaming qoe: towards the impact of frame rate and bit rate on cloud gaming. In: 2018 Tenth international conference on quality of multimedia experience (QoMEX). IEEE, pp 1–6
87. Zhang G, Lee JYB (2019) Ensemble adaptive streaming - a new paradigm to generate streaming algorithms via specializations. IEEE Transactions on Mobile Computing, pp 1–1. https://doi.org/10.1109/TMC.2019.2909202. Conference Name: IEEE Transactions on Mobile Computing
88. Zhang M, Zhu Y, Zhang C, Liu J (2019) Video processing with serverless computing: a measurement study. In: Proceedings of the 29th ACM workshop on network and operating systems support for digital audio and video, pp 61–66
89. Zhang W, Chen Q, Fu K, Zheng N, Huang Z, Leng J, Li C, Zheng W, Guo M (2020) Towards QoS-aware and resource-efficient GPU microservices based on spatial multitasking GPUs in datacenters. arXiv:2005.02088
90. Zheng Y, Wu D, Ke Y, Yang C, Chen M, Zhang G (2016) Online cloud transcoding and distribution for crowdsourced live game video streaming. IEEE Trans Circ Syst Video Technol 27(8):1777–1789
91. Zioulis N, Alexiadis D, Doumanoglou A, Louizis G, Apostolakis K, Zarpalas D, Daras P (2016) 3D tele-immersion platform for interactive immersive experiences between remote users. In: 2016 IEEE International conference on image processing (ICIP), pp 365–369. https://doi.org/10.1109/ICIP.2016.7532380. ISSN: 2381-8549

**Publisher's note**  Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Affiliations

**Konstantinos Konstantoudakis[1]** [ID] **· David Breitgand[2] · Alexandros Doumanoglou[1] · Nikolaos Zioulis[1] · Avi Weit[2] · Kyriaki Christaki[1] · Petros Drakoulis[1] · Emmanouil Christakis[1] · Dimitrios Zarpalas[1] · Petros Daras[1]**

David Breitgand
davidbr@il.ibm.com

Alexandros Doumanoglou
aldoum@iti.gr

Nikolaos Zioulis
nzioulis@iti.gr

Avi Weit
weit@il.ibm.com

Kyriaki Christaki
kchristaki@iti.gr

Petros Drakoulis
petros.drakoulis@iti.gr

Emmanouil Christakis
manchr@iti.gr

Dimitrios Zarpalas
zarpalas@iti.gr

Petros Daras
daras@iti.gr

[1]   Visual Computing Lab (VCL), Information Technologies Institute (ITI), Centre for Research and Technology - Hellas (CERTH), Thessaloniki, Greece

[2]   Hybrid Cloud, Cloud and Data Technologies, IBM Research, Haifa, Israel