Computer Networks 231 (2023) 109808

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Measuring and estimating Key Quality Indicators in Cloud Gaming services[☆]

Carlos Baena, O.S. Peñaherrera-Pulla, Raquel Barco, Sergio Fortes[*]

Telecommunication Research Institute (TELMA), Universidad de Málaga, E.T.S. Ingeniería de Telecomunicación, Bulevar Louis Pasteur 35, Malaga, 29010, Andalucia, Spain

A R T I C L E   I N F O

Dataset link: https://ieee-dataport.org/documents/e2e-dataset-video-streaming-and-cloud-gaming-services-over-4g-and-5g

Keywords:
Cloud gaming
Mobile networks
Quality of experience
Key Quality Indicators
End-to-end
Machine learning

A B S T R A C T

The gaming industry has proposed the concept of Cloud Gaming (CG), a paradigm that enhances the gaming experience on reduced hardware devices. However, this paradigm puts a lot of pressure on the communication links that connect the user to the cloud. As a result, the service experience becomes highly dependent on network connectivity.

In this context, the present work proposes a framework for measuring and estimating the most important E2E (end-to-end) metrics of the CG service, namely Key Quality Indicators (KQIs). Therefore, different machine learning (ML) techniques are evaluated to predict KQIs related to the CG user experience. For this purpose, the most important KQIs of the service, such as input lag, freezes or perceived video frame rate, are collected in a real network deployment. The results show that ML techniques can be used to estimate these indicators solely from network-related metrics. This is seen as a valuable asset for the delivery of CG services over cellular networks, even without access to the user's device, as it is expected for telecom operators.

1. Introduction

The entertainment sector has recently experienced tremendous growth thanks to multimedia services such as gaming. This industry has been positioned in recent years as one of the main sources of profit in the leisure sector [1].

The gaming sector has been able to catch people's attention by extending the original goal pursued since its foundation: offering a virtual world where players can perform a myriad of actions as well as interacting [2] and socialising with other users [3]. These virtual worlds are getting closer and closer to reality, ranging from sharper 2D environments to 3D augmented and virtual scenarios. Nonetheless, rendering these environments to the player requires computationally demanding tasks. This means that users are forced to invest large amounts of resources in devices that can immerse them in such realistic environments.

As a result, the gaming industry is trying to take advantage of the high performance capabilities of cloud computing through the Cloud Gaming (CG) paradigm [4]. This concept aims to move the execution of rendering tasks to the cloud, turning user devices into thin clients. In this way, the only role of the user device is to be the interface between the user and the virtual world. This means the collection of user input actions and the display of video scenes generated by the remote server, where the environment is being rendered.

In this way, users can play almost any game, anywhere, anytime, on any device, since decoding is supported by the huge mass-produced chips included in the cheapest devices on the market. Moreover, this paradigm allows the introduction of Game on Demand (GoD), which, in addition to avoiding the installation of games, opens up new ways of commercialisation (e.g. via subscription) [5].

When it comes to game companies, CG implies a strong asset for combating piracy, since a game copy will never be downloaded in users' devices. Additionally, this concept eases platform compatibility issues, which might also reduce game production costs.

As its main drawback, CG makes the service to become completely network dependent. Conversely, traditional games only required a simple network connection to enable online multiplayer, allowing users to interact, despite their geographical location. In this sense, the information usually sent in classic online games ranges from the player's action and position to the clothing or object that each character is wearing at any given moment. This information typically involves exchanging small packets.

☆ This work has been partially funded by : Ministerio de Asuntos Económicos y Transformación Digital and European Union - NextGenerationEU, Spain within the framework "Recuperación, Transformación y Resiliencia y el Mecanismo de Recuperación y Resiliencia" under the project MAORI. This work has been also supported by Junta de Andalucía, Spain through Secretaría General de Universidades, Investigación y Tecnología, Spain with the predoctoral grant (Ref. PREDOC_01712) as well as by Ministerio de Ciencia y Tecnología, Spain through grant FPU19/04468. Funding for open access charge: University of Málaga / CBUA.

* Corresponding author.
E-mail addresses: jcbg@ic.uma.es (C. Baena), sppulla@ic.uma.es (O.S. Peñaherrera-Pulla), rbm@ic.uma.es (R. Barco), sfr@ic.uma.es (S. Fortes).

https://doi.org/10.1016/j.comnet.2023.109808
Received 23 December 2022; Received in revised form 6 April 2023; Accepted 3 May 2023
Available online 8 May 2023

Conversely, in CG, the fact that the environment is rendered in a remote server leads to a huge increase in network traffic demand: the demanding latency requirements hinder the application of content compression techniques, increasing the volume of data compared to traditional video services. In addition, the interactive nature of video games makes this paradigm more sensitive to network latency than other streaming services [6]. Consequently, the network is the most critical element in CG performance.

In this context, fifth generation (5G) mobile networks can boost CG services. The high data rates and low latency values expected with the service categories introduced and supported by 5G (i.e. Enhanced Mobile Broadband (eMBB) and Ultra Reliability and Low Latency Communications (URLLC)) place this new technology as the key to enabling the CG paradigm [7].

This, together with improvements in cloud infrastructure, has attracted the interest of many of the largest technology companies, which have launched their own CG services, such as Playstation Now (Sony) [8], Nvidia GeForceNow [9], Amazon Luna [10] or Microsoft xCloud [11]. At the same time, telecom operators have identified these platforms as a potential way to offer exclusive services to their customers, which could be a key differentiator in the market.

However, network management becomes extremely complex in the scenarios introduced by 5G. Consequently, the use of Self-Organising Networks (SON) algorithms together with Machine Learning (ML) techniques are highlighted as the main solution to cover these tasks [12–15]. Together with SON, network operators are beginning to adopt a new management strategy based on Quality of Experience (QoE). In this trend, the network is configured and optimised based on service Key Quality Indicators (KQIs) [16,17], with the aim of improving the quality of service perceived by the user.

Nevertheless, the collection of these End-to-End (E2E) metrics is a challenging task. On the one hand, access to device data is generally limited and implies a potential threat to the confidentiality of user data. On the other hand, the increasingly adoption of security protocols hinders their calculation from traditional techniques such as packet inspection.

Therefore, this paper aims to contribute in two ways. Firstly, to provide a measurement framework to solve the difficult task of extracting E2E metrics from the CG service. In this respect, the main factors affecting the user experience are described, highlighting some of the most important KQIs for the service. Secondly, to facilitate the integration of service information into network management tasks. To this end, an ML-based approach is presented that enables the estimation of KQI metrics from available network information.

The rest of the present article is organised as follows. Section 2 presents related works and exposes the main contributions of this work. Section 3 introduces the most important KQIs for CG services. Based on that, Section 4 describes a framework for gathering these metrics. Then, Section 5 presents a ML approach to building regression models that estimate CG KQIs. Subsequently, Section 6 provides an evaluation of the ML regression techniques considered. Finally, Section 7 shows the conclusions and future research lines.

2. Related work and contributions

Since interaction is the main attraction of games, the QoE of online and CG is highly dependent on system response, thus on network latency. This has been extensively covered in the literature. In [18] the authors present a Mean Opinion Score (MOS) study in which they show the high impact of network latency and packet loss on the QoE of CG. Following this line, the authors in [19] show the lineal correlation between network latency and MOS degradation using both an objective and subjective study. Raeen et al. [20] show in their study that the majority of gamers are unable to distinguish response times below 40 ms. However, they also indicate that about half of casual gamers cannot tolerate service response times above 100 ms.

In the same line, the authors in [21] study how gamers adapt to different delay variations using both objective and subjective methods. The results show that users can adapt to a constant delay, while frequent delay variations annoy gamers. Furthermore, [22] examines the influence of user strategy on delay sensitivity. It concludes that the effect of delay sensitivity on QoE is independent of the strategy chosen by the gamer to complete a task.

Other works such as [23–25] have studied the impact of network latency on these services depending on the type of game, e.g. action, puzzle, etc. Focusing on online games, the authors in [23,25] show the importance of latency in user performance, suggesting that latency perception is determined by the precision and deadline of the game action. Actions with high precision and tight deadlines (e.g. first-person shooters) are more sensitive to latency than those that require low precision and do not have immediate deadlines (e.g. strategy games). Similarly, [26] shows the pronounced effects that first-person shooters have on player performance.

Using OnLive CG platform, Quax et al. [24] provide a qualitative comparison among action, strategy, puzzle, and racing types, pointing to action-oriented games as the most critical in terms of latency. Likewise, authors in [27] develop a model to predict different game strictness based on the rate of players' inputs and the game screen dynamics, easing the detection of a CG's latency sensibility. An extended prediction model is provided in [28], where the authors use other game characteristics such as temporal and spatial accuracy, degree of freedom, consequences, importance of actions, number or required actions among others.

Conversely, Slivar et al. [29] show through an empirical QoE study that CG services are more sensitive to network conditions than online games. By testing both paradigms under different network conditions, they conclude that CG services suffer more degradation in terms of user experience than a traditional gaming architecture. They also highlight the importance of ensuring adequate video quality. The latter fact is taken into account in the MOS study conducted by the authors in [30], where video quality is positioned together with input sensitivity as a primary feature to calculate CG QoE.

Authors in [31] assess the performance of some thin clients such as UltraVNC and TeamViewer within the CG concept. Based on the graphic quality as well as the image fluency, they show that this kind of widely extended platform for remote desktop purposes is not able to support CG, since low values of frame rate are reached. Additionally, Claypool et al. [32] present the important role that the network plays in streaming parameters through their study of CG traffic features. Based on bitrate, frequency and volume of data, their study shows that a poor network quality (in terms of high packet loss rate and insufficient bandwidth) leads to poor quality of the service.

In this context, network operators pursue the provision of CG services with the best QoE possible. Nonetheless, the complex architecture presented by the latest cellular networks, together with the provision on the same infrastructure of multiple services with heterogeneous requirements, make network management tasks extremely complex. In this sense, optimised network management is becoming increasingly important to offer a good service.

Here, authors in [33] provide an algorithm for the adaptive provision of CG at edges. Along the same lines, in [34], deep-reinforcement learning is used for an adaptive resource allocation in the edge. Authors in [35] analyse the cross-correlation between network parameters and the subjective and objective QoE . This study concludes that the level of end-user satisfaction based on QoS information can lead service providers to wrong conclusions.

In order to avoid that, some studies have focused on KQIs, which provides an E2E vision of the quality of the service. Herrera et al. [16] focus on FTP (File-Transport-Protocol) services to present a useful KQI modelling for the management of new generation mobile networks. Similarly, in [17] the authors propose a system based on video KQI estimation to support network slicing negotiation.

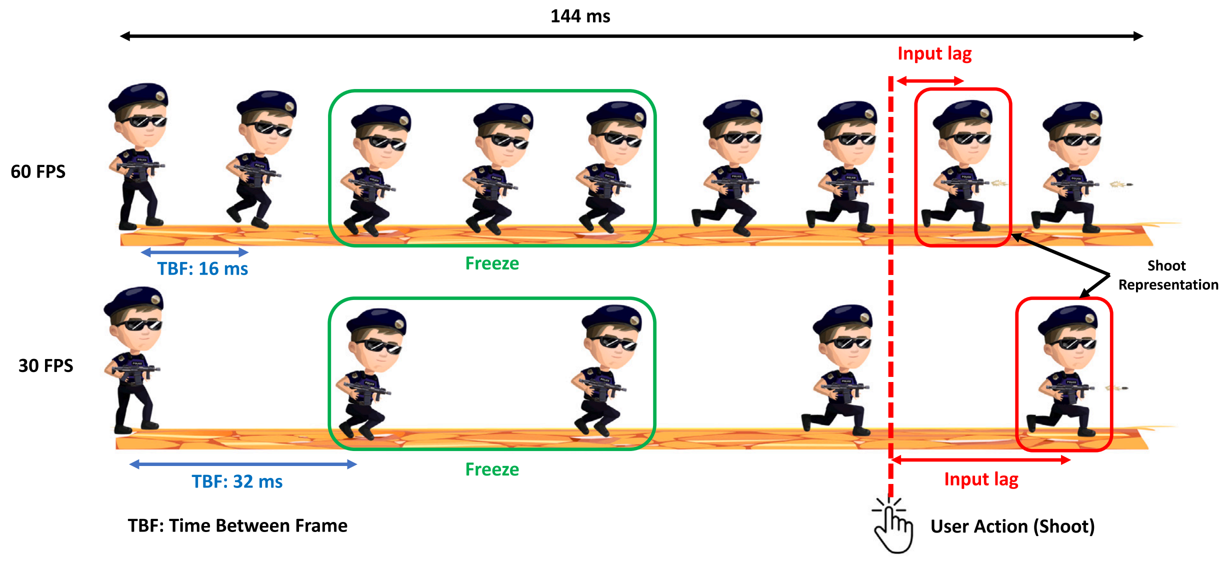


Fig. 1. Quality indicators in an image sequence.

To the best of our knowledge, there are no works in the literature that help the provision of CG services based on objective E2E metrics (i.e. KQIs). In this scope, the contribution of this paper is twofold.

Firstly, it is proposed a framework for measuring KQI from CG services. The novelty here resides on the use of the device's graphical driver for collecting data. This avoids complex and expensive setups based on external devices, such as high-speed cameras, to collect information from the service [26]. Likewise, it allows the collection of KQIs at the end point of service delivery, providing a very accurate view of E2E service performance. This differs from the framework proposed in [36], where metrics are only collected from the CG client.

Secondly, the use of regression models to facilitate the integration of E2E data into the CG architecture. In this respect, standard ML techniques have been assessed for estimating KQIs from readily available radio network metrics.

3. Cloud gaming quality indicators

The gaming industry has been able to offer a new type of entertainment thanks to the immersive nature of its services. This differentiation from other services results in a variety of factors affecting service QoE [37]. On the other hand, the definition of KQIs provides information on the E2E performance of the service. Thus, KQIs are metrics that allow to objectively measure and evaluate different aspects of perceived service quality.

Although QoE and KQI are different concepts, they are closely related. They allow the quality of a service to be assessed in different network contexts. For example, high input lag may negatively affect the QoE of a CG user. However, the degree to which the value of this KQI (i.e. input lag) degrades QoE is subjective and depends on other factors external to the network, such as the type of game played [37].

In this regard, based on visualisation and interaction with the game, this work considers three of the most important KQIs that provide valuable information about the CG service. Each of them are described in the following subsections.

3.1. Visualisation

The various challenges posed by games, as well as the events taking place in the virtual world, are perceived by users through film scenes. In this way, the visualisation of the scenario is fundamental to the gaming experience. It is often based on image resolution, frame rate and freezing.

Firstly, image resolution describes the granularity of the digital image. This means that a higher resolution usually results in a more detailed scene, hence, a better user experience. Nowadays, the most popular resolutions are 720p (High Definition — HD), 1080p (Full High Definition — FHD), 1440p (Quad High Definition — QHD) and 4K (Ultra High Definition — UHD).

The number of frames per unit of time (e.g. frames per second — fps) used to represent the scenario sequences then becomes one of the most important elements considered by players. Fig. 1 shows the different effects that using different numbers of frames in a scene has on the CG QoE services. Therefore, as can be seen in Fig. 1, a higher number of frames means that they are updated in a shorter period of time. This improves the user experience by providing a more fluid and therefore realistic moving display. This is usually described by the frame rate metric, whose values indicate the number of frames per second (fps) used in a video rendering.

Finally, the term "freeze" is used when the same frame is displayed several times. This is caused by the lack of new frames to display. These events usually have an extremely negative impact on the game experience. They give the feeling of a frozen image, which breaks the fluidity of movement. They also tend to be very annoying, as users may lose control of important events in the environment and lose the game.

Traditionally, all of these indicators are usually dependent on the CPU (Central Processing Unit) and GPU (Graphics Processing Unit) capabilities of the user's devices. These are responsible for rendering the virtual environment. Nonetheless, in CG, the network connection takes the leading role in the values of these indicators, since all the rendering tasks are moved to the cloud. This means that all content displayed to the player is sent over the network.

3.2. Interaction

The main pillar of games lies in the attractive and addictive possibility given to users to interact with the virtual world. As a result, one of the most critical parameters to take into account in gaming services is the responsiveness of the system, i.e. the time it takes from the user sending an action to its visualisation (see Fig. 1). This indicator is commonly known as input lag or latency.

In traditional games, input lag is basically related to the time taken by the user's devices to render the different scenes. In addition, as can be seen in Fig. 1, the frame rate also affects the responsiveness of the service. A higher frame rate allows a new frame to be seen sooner. This means that the response time of the system is reduced, which is the main reason why regular gamers always try to achieve high data rates. Similarly, input lag has been linked to the network with the introduction of online multiplayer. In this context, most of the response time of the system is caused by sending the user's metadata to a common server, which must be accessible to all players sharing the virtual world.

CG response time is generally monopolised by the delay introduced by the network into the data trading process. However, the volume of data exchanged in CG scenarios is much greater than in multiplayer, putting more pressure on the network link. This means that latency values are highly dependent on the capabilities of the link. In addition, visualisation settings also affect the input latency of CG systems: sending content at a higher resolution or frame rate puts more stress on the network, which can become congested and increase system latency.

4. Measuring key quality indicators

KQI metrics are presented as a quantitative alternative to QoE for obtaining information about service E2E performance. Using them provides a numerical way to represent the quality perceived by the end user. Nonetheless, obtaining these high-level metrics is usually a difficult task, mostly triggered by the limited access to the application data. With this in mind, this section describes a Python-based framework for extracting various KQIs from CG services.

4.1. Metrics extraction

For the extraction of the data, a controlled environment has been created over Moonlight client [38]. This platform is an open-source implementation of Nvidia's GameStream protocol. Besides, it enables content streaming up to 4K resolution at 120 fps over the RTP protocol (Real-time Transport Protocol).

In this sense, the controlled environment consists of a one-minute League of Legends game in which the user's screen is captured by the graphics driver. This is done using a Python script that also timestamps each frame it captures. Here, only the specific area where the action took place is monitored. This is known as the Area of Interest (AoI). By taking it into account, the amount of processing required to capture what is on the screen is reduced, while the frame rate is increased to 144 frames per second.

Along the gameplay, 5 actions are automatically sent by the automatic action tool presented in [36], which allows to replicate the actions of the user. Furthermore, at the beginning of each session, the visualisation settings used to stream the game (i.e. resolution and frame rate) are set.

At the end of the session, the RGB matrices of each frame are stored. They are then processed to provide high-level service information (i.e. KQIs) such as effective frame rate, freezes or system input delay. Note that the RGB matrices correspond to the decoded frames, so all the processes described below are independent of the codec.

4.1.1. Effective frame rate

Although the server renders and transmits the scene at the frame rate set at the beginning of the session, the transmission of the content over the network may differ from the frame rate visualised in the user's device. In this paper, this is referred to as the effective frame rate (EFPS). To measure EFPS, the session is analysed frame by frame, deleting those that are not different from their predecessors. Then, once the entire session has been examined, the total number of frames is divided by the session time. This gives the frame rate at which the game has been displayed on the user's device.

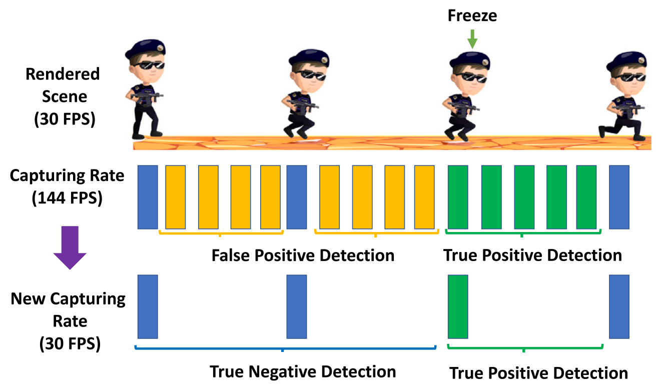


Fig. 2. Decimation process.

4.1.2. Freeze occurrences

As with the effective frame rate, sending rendered content over the network can affect the fluidity of the scene. Some events, such as packet loss or jitter, can affect the rendering of all the frames in the scene, causing it to freeze. In this respect, the calculation of freeze events is very similar to the effective frame rate method: each frame of the scene is analysed to identify successive frames that are identical.

This analysis is based on game animations, which are continuous movements that are part of the game experience. The aim of animations is to provide the user the feeling that the scene continues moving even when no action is being performed or nothing is happening. This means avoid the false feeling of a frozen scene. Consequently, animations causes small differences between frames, so freeze will be detected when two consecutive frames do not differ in terms of RGB matrix. Besides, by using the timestamp on each frame, it will be easy to determine how long the same frame has been displayed on the user's device.

Likewise, capturing rate plays a paramount role in the detection of freeze events. As shown in Fig. 2, a higher capture rate than the rendered scene rate will cause some frames to be the same between them, leading the algorithm to trigger a false freeze (false positive detection). To avoid this, a decimation process is required. This means that frames are discarded until the capturing frame rate matches the rendered one (see Fig. 2). In this way, by adjusting the capturing frame rate, the algorithm is able to correctly detect the freeze that has occurred in the scene (i.e. true negative and positive detection).

4.1.3. Input lag

The measurement of the input delay is a challenging task due to the inclusion of the user action in its calculation. In this sense, it is necessary to consider the time between the user performing the action and the time of its display, namely $t1$ and $t2$ respectively.

For the former (i.e. $t1$), the timestamps of the actions are collected by an automatic action tool [36]. The latter (i.e. $t2$) is obtained through the detection of motion on the screen, which is done by processing the content that is displayed on the AoI.

In this sense, all the frames of the session are analysed one by one in order to find those that differ by 25% from their predecessor. The value of this threshold is determined by studying the impact of all possible actions of the game on the decoded frames. This threshold hence provides the optimum level of sensitivity for the identification of the first frame in which the action is represented. Likewise, it also avoids false detections caused by game animations.

To speed up the process, the frames are divided into different subsets corresponding to each action performed during the session. This parallelises the process and facilitates correlating the motion detected with the action performed, i.e. finding $t2$ for each $t1$. In this way, it is possible to correctly identify the exact frame and time in which each action is represented, and therefore to obtain the absolute response time of the system through the difference between the two timestamps.

Table 1
Dataset summary.

Source	Indicator	Unit	Min	Mean	Max	Std.Variation	Description
CG session's quality	CGlatency	ms	30.59	87.43	498.65	36.35	50th-tile of the input lag of the whole session.
	FreezePercent	%	0	8.7	100	17.6	Percentage of the session with a frozen image.
	EFPS	fps	0.1	57.09	116.17	29.75	User's perceived frame rate.
CG server	Resolution	–	720p	–	4K	–	Resolution set in the server for the CG session
	fps	fps	30	–	120	–	Frame rate set in the server for the CG session
UE	PING_avg	ms	1	74.88	895	84.63	Round-trip-time between UE and server during the session
	PING_Radio_Loss%	%	0	0.23	25	1.27	Ping percentage loss in the radio part.
	PING_Host_Loss%	%	0	0.61	25	2.02	Ping percentage loss in the whole path.
	RSRP	dBm	−104	−71	−50	13.06	Avg. received power from the reference signal.
	RSRQ	dB	−8	−4.1	−3	1.1	Quality of the received reference signal.
	RSSI	dB	−95	−56.51	−51	8.5342	Strength of the received radio signal.
	SINR	dBm	6	17.52	26	6.85	Signal-to-interference-plus-noise ratio.
BS	n_rb_dl	RB	25	–	100	–	No. available resource blocks (RB) to assign.
	cqi	–	0	12	15	2.53	Channel quality indication reported.
	pucch_snr	dBm	−11.39	14.66	44.14	15.92	PUCCH SNR reported to the Base Station.
	pusch_snr	dBm	−25.78	15.57	36.65	8.67	PUSCH SNR reported to the Base Station.

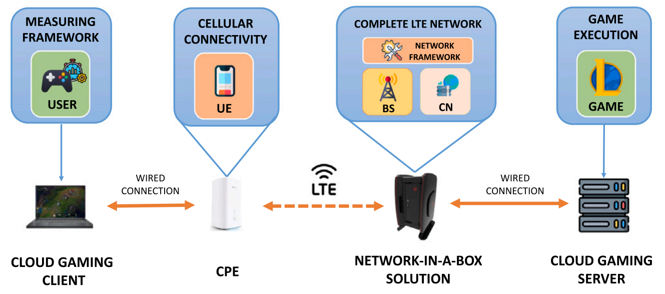


Fig. 3. Considered CG scenario.

4.2. Dataset building

A dataset of CG KQI metrics has been built by considering the measuring framework previously described. Fig. 3 shows the CG scenario used to generate the data.

The scenario uses a laptop as a CG thin client. It runs the controlled environment introduced in Section 4.1. This laptop uses a Customer Premises Equipment (CPE) to reach the CG server via an LTE connection. In terms of network, the CPE is seen as an User Equipment (UE). The CG server is located on the backhaul of the LTE network and is equipped with an Nvidia RTX 2070 Super GPU, which enables content streaming using Moonlight Gamestream.

The LTE network supporting the CG service is deployed via a network-in-a-box solution. This is based on Software-Defined Radio (SDR) and General Purpose Processor Platforms (GPPP) to integrate all the entities that comprise a cellular network into a single device. This includes the Base Station (BS) and a complete Core Network (CN). In this respect, Amarisoft software is used for the deployment of all network elements in accordance with Release 15.

Simultaneously, the network-in-a-box device is complemented by the network framework presented in [39]. This framework increases the configurability of the deployed network by running on top of the device. It also enables the creation of different radio scenarios, as well as the collection of network information.

So, considering only one BS and one UE, the network has been configured with 4 radio Physical Resource Blocks (PRBs) allocations (25, 50, 75 and 100) and different radio conditions (e.g. different power transmissions or interference). This results in up to 16 network scenarios. For each of the network scenarios, the CG controlled environment described in Section 4.1 has been run with 4 resolutions (720p, 1080p, 1440p and 4K) and 3 different frame rate values (30, 60 and 120 fps). Here, a single session minute is sufficient, thanks to the control of the network supporting the service. This makes it possible to guarantee the same network behaviour for all game sessions in a given network scenario.

Table 1 gives a summary of the 16 parameters that make up the dataset, including the source of each metric. Moreover, it shows the minimum, maximum and mean values, together with the standard deviation, that each parameter takes along the 3840 samples that make up the dataset. This dataset is publicly available in [40] and more information about it can be found in [41].

5. Estimation of key quality indicators

As seen in Section 4.1, measuring the CG KQI involves several processes that consume large amounts of computing resources (e.g. screen capture, frame decimation, or input lag calculation). Although these processes do not affect the performance of the service, they do introduce some delay in the data collection phase. In addition, the execution of these processes is restricted to a controlled environment where access to the CG client is available.

These two aspects hinder the integration of service information (i.e. KQIs) into network management processes. This encourages the use of ML techniques to obtain these CG KQIs in other environments without involving the game client. Fig. 4 shows the pipeline of this ML approach. The goal is to create regression models that are able to estimate the CG KQIs from easily accessible network parameters. For this purpose, the models are trained in an offline phase. Then, in an online phase, they will provide service information based on the estimation of current network parameters.

In this way, the first step is to pre-process the data injected from the measuring framework presented in Section 4. This involves standardising the values and dividing them into training and test subsets.

Based on the training subset, several models following different ML approaches are created for the estimation of each KQI. To do this, a grid search is used to find the configuration (i.e. hyperparameter) with which the models best fit the data to be predicted.

Using these configurations, the different models are trained using both all the input features of the dataset and a meaningful subset of them using Feature Selection (FS). This selection is carried out according to the highest scores with which the different features are weighted. In addition, the source of the different features (e.g. UE or BS) is taken into account. This allows the evaluation of predictions from only one part of the overall architecture.

Once all the models have been trained, their performance in terms of accuracy and prediction time is evaluated using the test subset. Then, in the online phase, the selected models could then be integrated into different parts of the CG architecture, such as the UE or BS, where they are expected to have a direct impact on the service.

In this context, following subsections will provide a more detailed description of the offline stage.

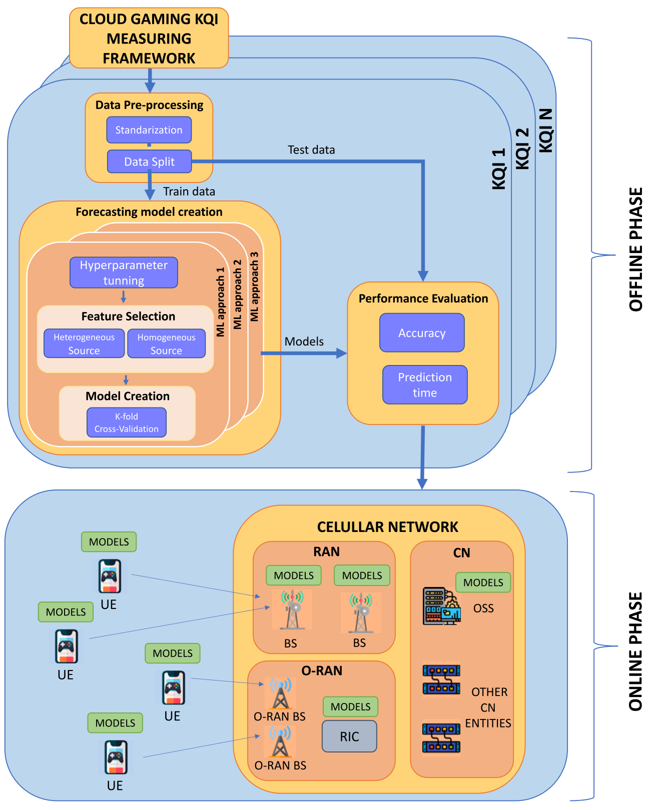


Fig. 4. KQI estimation pipeline.

5.1. Data pre-processing

As it can be seen in the Table 1, the different indicators come from heterogeneous sources (e.g. UE and BS). Furthermore, they show different ranges (e.g. *SINR* from UE shows values from 6 to 26, while *pusch_snr* takes values between −25 and 36) and units (e.g. *PING_avg* and *RSRP* are measured in ms and dBm respectively).

In this context, the application of standardisation techniques plays an important role. These techniques allow the dataset to be overlaid in order to put all the inputs on the same scale. This facilitates the analysis of the features used for modelling, leading to faster convergence of the different ML algorithms.

Accordingly, in this work, the data are standardised according to a Min-Max Scaler. This method regularises the data to unit variance as follows:

$$x_{i-scaled} = \frac{x_i - x_{min}}{x_{max} - x_{min}}, \tag{1}$$

where x_i is the original value of the parameter, x_{max} is the maximum value along the entire dataset, and x_{min} is the minimum value.

Additionally, the dataset is randomly split into training and test subsets, containing 70% and 30% of the samples respectively.

5.2. Estimation model building

Model building is the most important step in performing a high-accuracy estimation. This section outlines the techniques considered, as well as the steps taken to build various regression models.

5.2.1. Machine learning techniques

As this approach aims to estimate high-level metrics (i.e. KQI), which are difficult to obtain in uncontrolled situations, from other metrics, which are easily obtained in any situation, supervised learning (SL) techniques are considered.

SL approaches map one or more inputs to an output by analysing a set of examples. These examples, which are used to train the different techniques, are known as labelled data.

In this scope, six of the most extended SL techniques are considered in this approach:

- Linear Regression (LR) is a mathematical procedure commonly used to approximate the relationship between a dependent variable and one or more independent variables. This is done by finding the linear combination of features that minimises the residual sum of squares between the actual and predicted values.
- K-Neighbours Regression (KNR) is a regression based on the K-Nearest Neighbours (KNN) algorithm. This algorithm uses the similarity of the features to predict the values. This means that the prediction is assigned to a point based on how similar it is to the k points (or neighbours) in the training set. Finally, the value of the prediction is assigned by an average (or weighting) of the different neighbours.
- Support Vector Regression (SVR) follows to obtain a hyperplane that fits most of the data samples. This is done by transforming the data into a higher dimensional space using a set of mathematical functions (also called *kernel*). This makes it easier to define and search for an optimal regression hyperplane. At the same time, based on a tolerance parameter ϵ, bounds are defined around the hyperplane with the aim of neglecting any variation within these margins. In this way, unlike other regression methods, SVR attempts to minimise the generalised error within a range.
- Kernel Ridge (KRR) is a regression approach, like SVR, that uses the kernel trick to project the data into other dimensional spaces and facilitate the search for a regression function. However, the main difference between SVR and KRR resides in the loss function. Here, as in LR, the technique is to minimise the squared error loss to find the optimal model. Instead, a regularisation parameter α is used to reduce the variance of the estimate.
- Random Forest (RF) consists of a large number of decision trees whose outputs are averaged to produce predictions. Such decision tree models are created by bootstrap aggregation (bagging), which consists of replacing the training data with random subsets. This process eases the creation of uncorrelated models, which leads to overcoming the sensitivity of the decision tree to the trained data. In this way, their assembly operation allows to obtain robust regression models, since the trees protect each other from individual errors, thereby increasing the performance of the model.
- Artificial Neural Network (ANN) is an ML technique based on the architecture of the human brain. It consists of several nodes, also called artificial neurons, distributed along different layers (input, hidden and output layers). These artificial neurons are responsible for calculating and weighting the data through activation functions and sending their output to the next layer of the network. Thus, this approach achieves the creation of a complex regression model.

5.2.2. Hyperparameter tunning

ML algorithms are open to multiple designs through different *hyperparameters*. Their setting depends strongly on the nature of the problem and the data. Therefore, an inappropriate configuration of them can lead to a degradation of the model prediction. For this reason, it is essential to carry out a hyperparameter tuning process that provides the optimal settings for each model.

In this context, an exhaustive search of the most important hyperparameters has been carried out separately for the different KQIs that we follow to predict (i.e. *CGlatency*, *FreezePercent* and *EFPS*).

Table 2 briefly describes the different hyperparameters tuned by grid search in this work. Here, the hyperparameters fixed are those

Table 2
Hyperparameters tunning summary.

	Hyperparameter	Description	Latency	Freeze	EFPS
KNR	n_neighbours	Number of neighbours	12	4	4
	p	Power parameter (only for Minkowski metric)	–	–	–
	weight	Prediction weight function	Distance	Distance	Distance
	metric	Distance metric for finding neighbours	Manhattan	Manhattan	Manhattan
SVR	kernel	Function to transform the data	Poly	–	RBF
	epsilon	Tolerance parameter within which no penalty is associated	3.5	–	2.5
	degree	Degree of the polynomial function (only for poly kernel)	5	–	–
	C	Regularisation parameter through a squared l2 penalty	10	–	300
KRR	alpha	Regularisation strength	1.0	1.0	1.0
	kernel	Function to transform the data	Poly	Poly	Poly
	degree	Degree of the polynomial function (only for poly kernel)	6	6	6
RF	n_estimators	Number of trees conforming the forest	70	80	90
	max_depth	Maximum depth of the trees	10	20	20
	criterion	Figure of merit to measure the quality of a split	MSE	MSE	MAE
ANN	No. Layers	Number of layer conforming the Neural Network	4	3	6
	No. Neurons	Number of neurons in hidden layers	(20,14)	(10,5)	(120, 40, 20, 10)
	Activation function	Function used to determine the output of each layer	Relu	Relu	Tanh
	Optimiser	Function to optimise ANN attributes (e.g., weights)	Adam	RMSprop	Adam

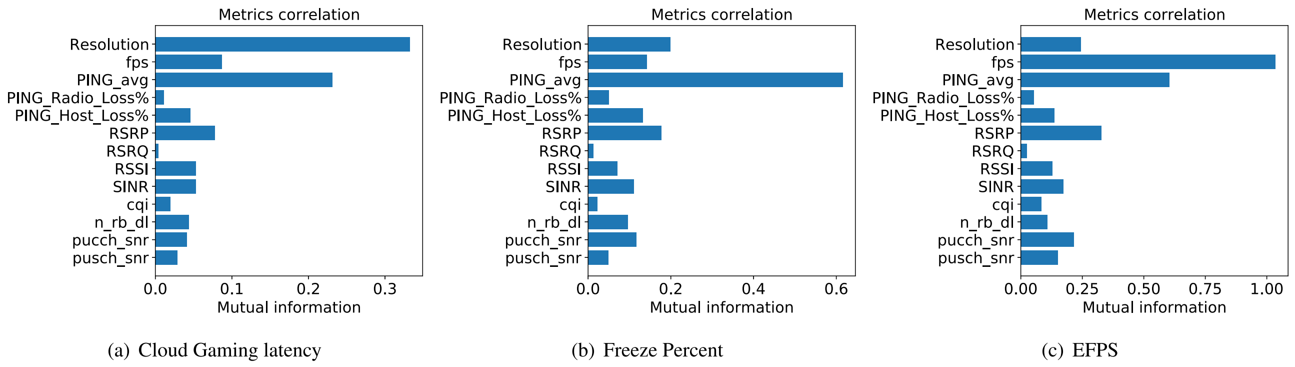


Fig. 5. Metric correlation through mutual information.

that allow to minimise the Mean Absolute Error (MAE) of the training dataset. This figure of merit is defined as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{2}$$

where $\hat{y}_i$ represents the predicted value, and y_i is the original one.

5.2.3. Feature selection

Generally, in ML, the more input features a model has, the better the accuracy of its estimation. Nonetheless, the use of a large number of predictors leads to increased computational cost and time spent on pre-processing and training. Consequently, a trade-off between accuracy and number of predictors is required.

The use of features from different sources could enrich the performance of the estimation. In this work, the integration of the KQI measurement framework with a controlling network environment eases the collection of features from different elements of the architecture (i.e. UE and BS).

However, it can be difficult to obtain all these features in a non-controlled scenario. On the one hand, the collection of metrics from heterogeneous sources is not trivial, as it is necessary to carefully manage the collection of these metrics in the different elements in order to merge them based on timestamps. Besides, this process might lead to the fact that not all indicators are available at the same time, hence causing a delay in the prediction. On the other hand, network metrics are usually secured at operator level. This complicates their availability and leads their provision to external parties based on high-level network information together with a time slot. Moreover, the high computational cost of storing a large number of metrics leads operators to minimise the number of parameters to be saved.

In this scope, the effect of using different predictors is investigated by selecting different features. In this way, different models are created taking into account different subsets of features from the whole dataset. In order to do this, an automatic feature selection is applied according to two criteria: features from the whole dataset or features from a single source together with the configuration of the CG stream.

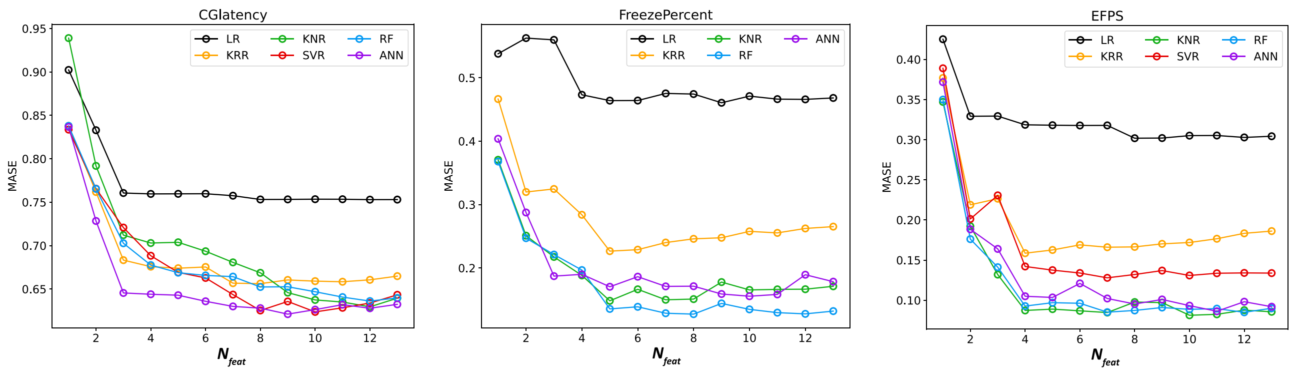
In order to perform an automatic selection, the features are taken according to the k-highest scores assigned to each of them. These scores are assigned by analysing the dependency between the input and target features, which is usually determined by Pearson's correlation. Nevertheless, this work proposes Mutual Information (MI) as a score function. MI provides the degree of certainty between two variables as follows:

$$I(X,Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j)\cdot log\frac{P(x_i, y_j)}{P(x_i)\cdot P(y_j)} \tag{3}$$

In this sense, MI defines the scores based on relationships beyond the linear ones obtained with Pearson's correlation, such as non-monotonic relationships. Fig. 5 shows the mutual information between the different input features of the dataset and the KQIs to be predicted.

Here, although the values are mainly dependent on the output, it is generally observed that the configured resolution, the frame rate of the session and the ping between client and server are the most influential features for each KQI. Besides, other parameters such as RSSI, SINR and pucch_snr seem to have the same influence on the different KQIs. Conversely, RSRQ and CQI seem to have very slight dependence on the target parameters.

In this way, the automatic selection will take the k-highest influential features for each KQI. Instead, the source-based selection of the



(a) Cloud Gaming latency (b) Freeze Percent (c) EFPS

Fig. 6. MASE evolution.

features takes into account the element from which the features are generated. For instance, a model can be created to estimate a KQI by taking only the parameters of the session configuration and those available in the UE (see Table 1).

This criteria allows to analyse the performance of the models when only indicators (or a part of them) of one element of the whole communication architecture are available. In order to also take into account that not all the indicators of an element may be available, MI has also been used to analyse the performance of the estimation through the different number of input features.

5.2.4. *Model creation*

In this process, models are created based on the information obtained in the previous blocks. This means that each ML algorithm is configured according to the hyperparameters that best fit the nature of the data. Likewise, the models are trained taking into account the previous feature selection.

In addition, a k-fold cross-validation is carried out in order to avoid overfitting or underfitting. This enables validating the effectiveness of the model on new data input.

5.3. *Performance evaluation*

Once the models have been built using the techniques described above, their performance is assessed using the Mean Absolute Scaled Error (MASE) [42].

This metric provides a measure of how well a model estimates compared to a naive estimation. The latter is a simple and basic method of estimation, which consists in assuming that the future value is equal to the average of the values obtained in the past. The MASE then shows how the use of regression models improves the estimation obtained without identifying causal factors. It is referred to as:

$$MASE = mean\left(\frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{\sum_{i=1}^{N}|y_i - \bar{y}|}\right) \tag{4}$$

From the equation it is seen that MASE gives a scaled error of the model prediction with respect to a naive prediction. In this sense, a MASE value of 0.1 means that the estimation error of the regression model is 10% of the average error obtained with the naive estimation. In other words, the model is 90% more accurate than a naive estimation. Therefore, the lower the MASE value, the better the accuracy of the model.

Like Mean Absolute Percentage Error (MAPE), MASE is a scale-free figure of merit that allows estimation performance to be compared regardless of the scale of the data. Nevertheless, MASE offers several advantages over MAPE. Unlike MAPE, it applies the same penalty to both positive and negative errors. This issue often leads to the use of symmetric MAPE (sMAPE) instead of MAPE. Nonetheless, MASE never returns infinite or undefined values, even when there are zero values in the dataset, as is the case with MAPE and sMAPE.

As can be seen in Table 1, some of the CG KQIs could have values equal or close to zero (i.e. *FreezePercent* and *EFPS*). For this reason, and thanks to the characteristics described above, MASE is positioned as the most appropriate metric for assessing models accuracy.

Finally, in addition to the estimation error, the computational cost of each model in terms of prediction time is also considered. These two aspects are of primary importance for the online phase, where accurate and fast predictions are required.

6. Evaluation

In order to evaluate the KQI estimation, this section provides a performance analysis of the different models. Table 2 shows the hyperparameters that minimise the prediction error for each model. From this point on, all models evaluated will follow these settings, taking into account the algorithm used and the KQI to be predicted.

For this purpose, as described in Section 5, each regression approach is tested with different predictors. Thus, models are built using predictors from the whole dataset or considering their source (i.e. combining CG server parameters with metrics from the UE or BS). These have been trained following 10-fold cross-validation.

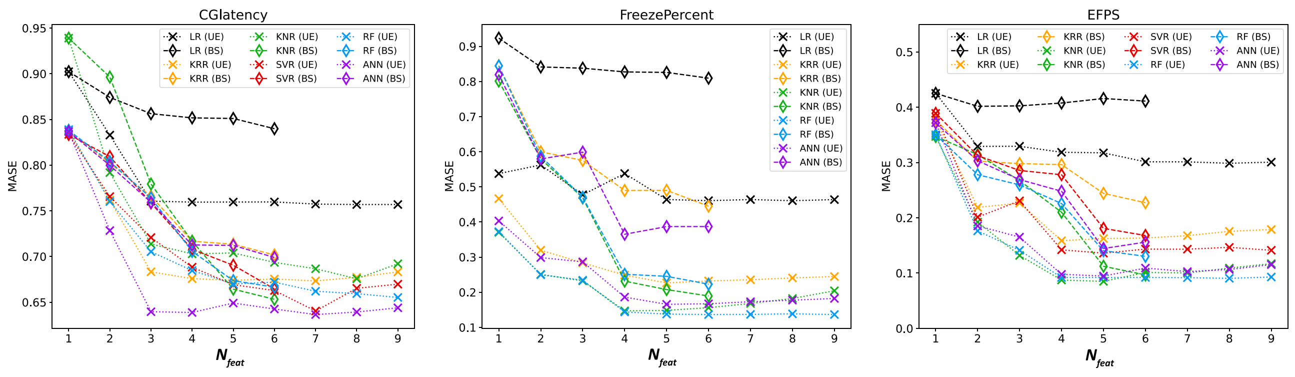
The following subsections present the evaluation of such models for the different KQIs. As aforementioned, both estimation error and prediction time are taken into account.

6.1. *Estimation error*

Fig. 6 shows the estimation error in terms of MASE of the different ML algorithms for estimating the three KQIs introduced in the previous sections: input lag (*CGlatency*), freeze percent (*FreezePercent*) and effective frame rate (*EFPS*). In this way, the MASE obtained with each regression model is represented as a function of the number of input features (N_feat). These features are selected according to their influence on each KQI, i.e. the metrics shown in Fig. 5 with the k-highest MI values.

Similarly, Fig. 7 shows the evolution of the MASE along the number of input features, but in this case taking into account the source of the features (i.e. UE or BS). This means that only information from one source, together with data from the CG server, is used to estimate the KQIs. For example, for cases marked as BS, only metrics whose source field in the 1 table corresponds to BS (i.e. *n_rb_dl, cqi, pucch_snr, pusch_snr*) are used together with those from the CG server (*Resolution* and *fps*).

In order to facilitate the analysis, all models are categorised according to the value of the MASE as poor ($MASE > 0.7$), acceptable ($0.5 < MASE \leq 0.7$), good ($0.2 < MASE \leq 0.5$) or excellent ($MASE \leq 0.2$).



(a) Cloud Gaming latency (b) Freeze Percent (c) EFPS

Fig. 7. MASE evolution considering the source.

6.1.1. *Latency*

Focusing on Fig. 6(a), it can be observed that all regression techniques generally achieve MASE values below 0.8. Here, the LR technique, which is used as a baseline in this paper, produces poor regression models, as the best estimation is made with a MASE of approximately 0.76. Furthermore, its evolution indicates that only three metrics provide information, as no improvement in prediction is shown by adding more metrics (i.e. $N_{feat} > 3$). This means that only these predictors have a significant linear relationship with this KQI. Regarding the MI values shown in Fig. 5, these metrics correspond to *Resolution*, *fps* and *PING_avg*.

Conversely, acceptable models (i.e. $MASE < 0.7$) are obtained by the KRR, KNR, SVR, RF and ANN techniques. Unlike LR, these techniques manage to find a relationship with more than three features, which leads to an improvement in the accuracy of the models. The estimation by KNR and RF models is richer when a greater number of features are taken into account, although this is ultimately seen as a slight deterioration. In this sense, the introduction of 12 features ($N_{feat} = 12$) allows to obtain values of MASE around 0.65.

Nevertheless, this behaviour is not replicated by the other approaches. The KRR technique needs seven features to estimate with the smallest error (i.e. MASE $\approx$ 0.66). Beyond this number of features (i.e. $N_{feat} > 7$), the accuracy of KRR models begins to degrade. The same applies for the SVR and ANN algorithms: the best performance (i.e. MASE $\approx$ 0.64) is obtained with eight and nine features respectively, becoming worse as the number of inputs increases.

In this case, ANN is positioned as the most accurate technique for the estimation of *CGlatency*. Furthermore, it can be seen that the best performance is obtained with a low number of features (i.e. $N_{feat} = 8$), outperforming the other techniques up to the inclusion of eight predictors.

Similarly, in Fig. 7(a), the evolution of MASE considering CG server and UE inputs (labelled UE) shows a similar trend. This behaviour is due to the fact that, as seen in Fig. 5, metrics from the CG server and UE have more MI than BS metrics. This is why the models with the fewest number of inputs are consistent. This is shown in the case of the LR technique: the models show the same behaviour for the LR and LR(UE) cases. As seen before, this algorithm is not able to find strong links beyond the metrics *Resolution*, *fps* and *PING_avg*.

Similarly, the remaining models (i.e. KRR(UE), SVR(UE), KNR(UE), RF(UE) and ANN(UE)) achieve closer results compared to the use of inputs from heterogeneous sources. As before, ANN is retained as the most accurate model when only metrics from the UE are available. Furthermore, it requires only three inputs to perform estimation with smaller error.

Focusing now on the BS metrics, the results show a general deterioration for all cases. This trend is very noticeable for the LR(BS), ANN(BS) and KRR(BS) cases, whose difference in terms of MASE with respect to the best case goes up to approximately 0.2, 0.1 and 0.05, respectively. This means that these models estimate with 20%, 10% and 5% more error, respectively, than the most accurate models for each ML technique.

For their part, the SVR and RF models are capable of estimating with similar performance with both UE and BS metrics. In this respect, both approaches achieve a MASE of about 0.67 with $N_{feat} = 6$.

Finally, considering the source of the features, KNR can achieve a more precise estimation with BS inputs than with UE metrics. Moreover, with all the BS features, the model performs with an error close to the ANN(UE), which are the most accurate models with UE metrics. Thus, when only metrics from the CG session and the BS are available, the KNR models can be confirmed as the best option for estimating *CGlatency*.

6.1.2. *Freeze percent*

Fig. 6(b) shows the evolution of the MASE obtained with the freeze percent estimation models. It can be seen that all the models obtained for this KQI are good or excellent. However, it is noteworthy that it has not been possible to obtain a model with SVR that converges with the data (i.e. $MASE = 4.6$). This is because it is not possible to find a set of hyperparameters that fit this metric and hence obtain a model that converges. Therefore, in order to facilitate the analysis of the models, the SVR is not shown in Figs. 6(b) and 7(b).

In this case, the traditional LR technique achieves MASE values around 0.47. Moreover, from their evolution it can be deduced that four predictors have a strong relationship with the KQI. Considering the MI values shown in Fig. 5, these are *PING_avg*, *Resolution*, *RSRP* and *SINR*.

As in the case of *CGlatency*, the use of advanced techniques leads to a significant improvement in the accuracy of the prediction. Here, KRR allows building good regression models, resulting in estimation with about 25% less error than naive estimation. Moreover, it is noted that the peak accuracy is reached for $N_{feat} = 5$, as the performance degrades as the number of features increases.

Conversely, excellent models (i.e. $MASE < 0.2$) are built using KNR, RF and ANN techniques. Considering ANN models, the best results (i.e. MASE $\approx$ 0.16) are obtained for $N_{feat} = 9$, getting worse when more than 11 features are introduced. A slight improvement is achieved with the KNR technique, which results in an error that is 5% lower than the one obtained with the ANN models. In addition, the best results for KNR are obtained with fewer predictors (i.e. $N_{feat} = 5$). These results are downgraded when more than eight input features are considered. Here, the prediction error is even slightly better than that obtained with ANN models.

However, the best prediction is reached with RF models. The use of these models guarantees a small estimation error for any number of inputs, with the exception of $N_{feat} = 3$. Furthermore, it is observed that from 5 input features, the error values are very close to this obtained by the most accurate case (i.e. $N_{feat} = 8$).

With regard to the source of the metrics, it is again evident that the use of data from the UE gives similar results to a mixed use. As with *CGlatency*, the most influencing parameters for this KQI come from the CG session and the UE.

Thus, Fig. 7(b) shows similar results to those described above. On the one hand, LR(UE) models are able to halve the estimation error of a naive prediction (i.e. $MASE \approx 0.5$). Unlike before, this value is now obtained with $N_{feat} = 5$, keeping constant beyond the use of more input predictors.

On the other hand, these results are improved by the other regressions. Thus, KRR(UE) and ANN(UE) show the same performance as before, although their MASE gradually decreases when more than five inputs are used. Furthermore, RF(UE) still has the best MASE, but in this case the minimum is reached for $N_{feat} = 5$. Here KNR(UE) achieves the same behaviour as RF(UE) for $N_{feat} \leq 4$. However, it can be observed that beyond this number of features, their performance starts to get progressively worse as the number of input features increases. Therefore, both KNR(UE) and RF(UE) are considered excellent for estimating freeze percent using only data from the UE.

When it comes to metrics from BS, LR(BS) gets MASE values above 0.82. This means that LR(BS) is only 18% more accurate than a naive estimation, so they are considered poor models. Moreover, these values are far from the most accurate models (i.e. KNR (UE) and RF (UE)), which achieve a MASE around 0.15.

For their part, KRR(BS) and ANN(BS) models improve the prediction accuracy, obtaining MASE values of 0.5 and 0.4 respectively. However, although they are considered good models, the estimation error is still higher than that obtained by KNR (UE) and RF (UE).

In this context, as can be seen in Fig. 7(b), RF(BS) and KNR(BS) are the models that provide estimations with a lower degree of error. However, the accuracy of KNR(BS) is slightly better than that of RF(BS).

From this it can be concluded that KNR is the most reasonable technique to use when metrics are available from only one part of the architecture. Otherwise, the best estimation of the freeze percent would be achieved by RF models.

6.1.3. EFPS

The estimation error with which each model estimates the EFPS is represented by the MASE in Fig. 6(c).

Firstly, it is noted that this KQI has a stronger linear relationship with some inputs than in the case of *CGlatency* and *Freeze Percent*. For example, the best MASE values for the traditional LR are around 0.32. Furthermore, it can be seen that the greatest improvement is obtained with two features. Concerning the MI information results shown in Fig. 5, these are the configured frame rate in the client (i.e. *fps* parameter) and the average ping between UE and BS (i.e. *PING_avg*).

Following the trend previously observed for the other KQIs, a better estimation is achieved with the advanced techniques, which are able to establish strong relationships between the metrics. This leads to the creation of excellent models (i.e. $MASE < 0.2$).

KRR and SVR show a similar evolution for $N_{feat} \leq 5$, albeit the latter achieves better values. However, for $N_{feat} > 5$ KRR begins to show a negative trend, while SVR models continue to improve up to seven predictors.

In the case of ANN, the models estimate with smaller errors compared to the previous techniques. This algorithm uses 11 of the 13 available metrics to achieve the lowest error (i.e. $MASE \approx 0.1$). RF and KNR also achieve similar values, but with fewer features than ANN. On the one hand, RF needs seven features to obtain the most accurate model. After that, there are no significant differences. On the other hand, KNR models give the best estimation for $N_{feat} = 4$. Later, some fluctuations are noticed, although eventually MASE gets closer to the previous values.

Regarding the source of the features, the MASE of all these ML techniques for estimating the EFPS metric is shown in Fig. 7(c). Here, the estimation error of the EFPS metric is similar to that seen above. LR(UE), KRR(UE) and SVR(UE) follow the same trend as before. The same applies to the RF(UE) models, which continue to behave as before, even when only UE metrics are considered. Finally, for ANN(UE) and KNR(UE), a gradual decrease in MASE is observed for $N_{feat} > 5$.

Focusing on the use of metrics from the BS, all models perform with a MASE below 0.40, so they are considered good estimation models. Among these models, the less accurate estimations are obtained by LR(BS) and KRR(BS), with a MASE of 0.40 and 0.23 respectively. For their part, the KNR(BS), SVR(BS), ANN(BS) and RF(BS) models provide excellent estimates, obtaining low estimation errors (i.e. $MASE \leq 0.2$). Here, KNR(BS) has the best performance (i.e. $MASE \approx 0.1$), which is consistent with the most accurate models using features from the UE.

Thus, it can be seen that KNR is the best technique for estimating *EFPS*, since excellent models are obtained for each of the three cases analysed. Nevertheless, it is noteworthy that the vast majority of the models considered achieve a substantially good estimation for this KQI, regardless of the sources of the predictors.

6.2. Prediction time

In addition to the estimation error, the models are also evaluated on their prediction time, i.e. the time it takes each model to estimate values. This evaluation has been carried out on an Intel Xeon Silver with 12 CPU cores, 2.2 GHz clock frequency and 128 GB RAM. Table 3 summarises the mean prediction time (in milliseconds) of 1000 predictions of each KQI using different regression techniques and input source (i.e. All, UE or BS).

For simplicity, models using the maximum available number of features for input source have been considered. This means that for All, UE and BS source, the models evaluated for each technique take 13, 9 and 6 input features, respectively. As expected, models based on LR technique are the fastest thanks to its simplicity, requiring less than 1 ms in any case. When it comes to more advanced techniques, the prediction times become much higher. KNR needs about 17 ms to get an estimate for both *FreezePercent* and *EFPS*, being double in the case of *CGlatency*. These values decrease for UE or BS input metrics due to fewer inputs considered.

These values are higher for the SVR case, which takes about 47 ms and 31 ms to predict *CGlatency* and *EFPS* respectively. The difference between these two values lies in the kernel set used for the models. Here, similar prediction times are obtained when only UE or BS metrics are considered. Note that SVR based models could not be trained for *FreezePercent*. Similarly, KRR models need about 55 ms to estimate any KQI with any type of input (i.e. All, UE or BS).

ANN models spend up to 4 times as much time providing an estimation. This huge increment resides on the high complexity of these models. Thus, ANN needs about 200 ms to estimate *CGlatency* using any of the sources. These times are higher for the estimation of *FreezePercent* and *EFPS*, reaching values up to 234 ms and 268 ms respectively.

Finally, RF takes the shortest time for the prediction of the latency (i.e. 11.69 ms). It also shows similar prediction times to KNR in the case of *FreezePercent* and *EFPS*. Similarly, RF achieves low prediction times when considering only metrics from UE or BS. In this case, these times are only improved by the KNR models.

In this respect, the RF and KNR models require less time to provide an estimation. This means that they are less computationally demanding in the online phase of the approach. These models are therefore suitable for use in devices with hardware limitations. Besides, RF models (UE) have shown excellent performance in terms of accuracy, resulting in the best balance between error and prediction time.

Conversely, if the models are focused to be used in the network, other alternatives could be more adequate. For example, although the ANN model shows the highest prediction time, it could be interesting for estimating CGlatency when less than 8 metrics are available from the whole system (i.e. UE and BS). In these cases, ANN shows less error than the other models (see Fig. 6(a)). Nonetheless, RF still shows a

Table 3
Model prediction times (ms).

Source	Tech	CGlatency	FreezePercent	EFPS
ALL	LR	0.079	0.086	0.079
	KNR	30.89	16.84	17.03
	SVR	46.94	–	31.37
	KRR	52.96	53.52	52.98
	RF	11.69	16.07	20.59
	ANN	208.23	234.09	165.6
UE	LR	0.1	0.1	0.1
	KNR	15.33	10.09	10.57
	SVR	55.84	–	28.96
	KRR	58.16	58.21	60.74
	RF	13.1	20.6	26.47
	ANN	196.39	216.08	268.15
BS	LR	0.1	0.11	0.11
	KNR	8.72	6.87	6.21
	SVR	50.32	–	27.22
	KRR	55.7	55.13	56.98
	RF	14.67	21.3	26.77
	ANN	207.53	197.4	248.38

very good performance in both aspects (i.e. accuracy and prediction time). Likewise, when only the metrics available in the BS are taken into account, the KNR (BS) models achieve the best balance in the estimation of the CG KQIs.

It can be concluded then that a balance must be reach depending on the place where the models will be integrated. Higher accuracy is preferred when it is expected to be deployed on a network element (e.g. BS, OSS). Meanwhile, simpler models (ranked here by their prediction time) are advisable for environments with low computational capabilities.

7. Conclusions

This work has presented a framework for measuring KQIs in CG services, in particular the input lag, the percent of session freezes or the frame rate perceived by the user. The framework avoids the use of expensive and complex setups involving high speed cameras by monitoring the user's peripherals through their graphics driver. In addition, an ML-based approach is presented. It facilitates the integration of service information into network management tasks through regression models. These are expected to be used in different parts of the service architecture.

To this end, a comprehensive study of several ML techniques has been carried out in order to estimate such KQIs from the session configuration and from the network metrics. Traditional linear regression was evaluated together with five of the most extended ML techniques (i.e. KNR, SVR, KRR, RF and ANN) in terms of accuracy and prediction time. The results have highlighted the good balance in terms of accuracy and prediction time offered by RF and KNR. Likewise, others such as ANN can be appropriated for some case, although they require higher prediction time.

Obtaining KQIs from readily network metrics opens up a number of possibilities for optimising service experience. On the one hand, ML models could be run in the UE to enable autonomous selection of the wireless network technology (e.g. WiFi, 4G or 5G). For example, the UE could choose the best option to support service delivery based on KQI requirements. Similarly, it could support streaming configuration settings according to the current network conditions.

On the other hand, these models can be instantiated in the Operational Support System (OSS) to enable network performance optimisation to guarantee the Service Level Agreement (SLA) based on KQIs. Similarly, in OpenRAN network architectures, these models can be hosted in the Radio Intelligent Controller (RIC) as part of an xApp. This would support Radio Access Network (RAN) management in a more effective way than over-provisioning resource approaches. For example, network slicing scenarios could benefit from these estimates to allocate resources to each slice.

Future works will focus on the development of network management algorithms based on this approach. Likewise, these models are expected to be improved with more information that affects the QoE of the service, such as the device used or the type of game.

For all these reasons, this measuring and estimation approach could be considered a powerful solution for supporting CG services in next-generation mobile networks.

CRediT authorship contribution statement

Carlos Baena: Conceptualization, Methodology, Software, Formal Analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **O.S. Peñaherrera-Pulla:** Software, Investigation, Writing – original draft. **Raquel Barco:** Resources, Supervision. **Sergio Fortes:** Conceptualization, Writing – review & editing, Project administration, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data from the authors available on https://ieee-dataport.org/documents/e2e-dataset-video-streaming-and-cloud-gaming-services-over-4g-and-5g

References

[1] A. Marchand, T. Hennig-Thurau, Value creation in the video game industry: Industry economics, consumer benefits, and research opportunities, J. Interact. Mark. 27 (3) (2013) 141–157, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1094996813000170.

[2] N. Ducheneaut, R.J. Moore, The social side of gaming: A study of interaction patterns in a massively multiplayer online game, in: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, CSCW '04, Association for Computing Machinery, New York, NY, USA, 2004, pp. 360–369, [Online]. Available: https://doi.org/10.1145/1031607.1031667.

[3] M. Szell, S. Thurner, Measuring social dynamics in a massive multiplayer online game, Social Networks 32 (4) (2010) 313–329, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378873310000316.

[4] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V.C. Leung, C.-H. Hsu, A survey on cloud gaming: Future of computer games, IEEE Access 4 (2016) 7605–7620.

[5] S. Baek, J. Ahn, D. Kim, Future business model for mobile cloud gaming: the case of South Korea and implications, IEEE Commun. Mag. (2023).

[6] M. Abdallah, C. Griwodz, K.-T. Chen, G. Simon, P.-C. Wang, C.-H. Hsu, Delay-sensitive video computing in the cloud: A survey, ACM Trans. Multimedia Comput. Commun. Appl. 14 (3s) (2018) [Online]. Available: https://doi.org/10.1145/3212804.

[7] C. Baena, Gaming in the Cloud: 5G as the pillar for future gaming approaches, 2022, http://dx.doi.org/10.36227/techrxiv.21665645.v1, [Online]. Available: https://www.techrxiv.org/articles/preprint/Gaming_in_the_Cloud_5G_as_the_pillar_for_future_gaming_approaches/21665645.

[8] PS Now: On-demand PlayStation games on PS5, PS4 or PC, [Online]. Available: https://www.playstation.com/en-us/ps-now/.

[9] GeForce now Cloud Gaming, [Online]. Available: https://www.nvidia.com/en-us/geforce-now/.

[10] Play your favorite games straight from the cloud with Amazon Luna, [Online]. Available: https://www.amazon.com/luna/landing-page.

[11] Xbox Cloud Gaming with Xbox Game Pass, [Online]. Available: https://www.xbox.com/en-GB/xbox-game-pass/cloud-gaming.

[12] P.V. Klaine, M.A. Imran, O. Onireti, R.D. Souza, A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2392–2431.

[13] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. Yau, Y. Elkhatib, A. Hussain, A. Al-Fuqaha, Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges, IEEE Access PP (2017).

[14] D. Palacios, S. Fortes, I. de-la Bandera, R. Barco, Self-Healing Framework for Next-Generation Networks through Dimensionality Reduction, IEEE Commun. Mag. 56 (7) (2018) 170–176.

[15] S. Fortes, C. Baena, J. Villegas, E. Baena, M.Z. Asghar, R. Barco, Location-Awareness for Failure Management in Cellular Networks: An Integrated Approach, Sensors 21 (4) (2021) [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1501.

[16] A. Herrera-Garcia, S. Fortes, E. Baena, J. Mendoza, C. Baena, R. Barco, Modeling of key quality indicators for end-to-end network management: Preparing for 5G, IEEE Veh. Technol. Mag. 14 (4) (2019) 76–84.

[17] C. Baena, S. Fortes, E. Baena, R. Barco, Estimation of Video Streaming KQIs for Radio Access Negotiation in Network Slicing Scenarios, IEEE Commun. Lett. 24 (6) (2020) 1304–1307.

[18] M. Jarschel, D. Schlosser, S. Scheuring, T. Hoß feld, Gaming in the clouds: QoE and the users' perspective, Math. Comput. Modelling 57 (11–12) (2013) 2883–2894.

[19] M. Claypool, D. Finkel, The effects of latency on player performance in cloud-based games, in: 2014 13th Annual Workshop on Network and Systems Support for Games, IEEE, 2014, pp. 1–6.

[20] K. Raaen, R. Eg, C. Griwodz, Can gamers detect cloud delay? in: 2014 13th Annual Workshop on Network and Systems Support for Games, 2014, pp. 1–3.

[21] S.S. Sabet, S. Schmidt, C. Griwodz, S. Möller, Towards the impact of gamers' adaptation to delay variation on gaming quality of experience, in: 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), 2019, pp. 1–6.

[22] Sabet, et al., Towards the Impact of Gamers Strategy and User Inputs on the Delay Sensitivity of Cloud Games, in: International Conference on Quality of Multimedia Experience, IEE Inc., 2020.

[23] M. Claypool, K. Claypool, Latency and player actions in online games, Commun. ACM 49 (11) (2006) 40–45.

[24] P. Quax, A. Beznosyk, W. Vanmontfort, R. Marx, W. Lamotte, An evaluation of the impact of game genre on user experience in cloud gaming, in: 2013 IEEE International Games Innovation Conference (IGIC), IEEE, 2013, pp. 216–221.

[25] M. Claypool, K. Claypool, Latency can kill: precision and deadline in online games, in: Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems, 2010, pp. 215–222.

[26] S. Liu, M. Claypool, The impact of latency on navigation in a first-person perspective game, in: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, CHI '22, Association for Computing Machinery, New York, NY, USA, 2022, [Online]. Available: https://doi.org/10.1145/3491102.3517660.

[27] Y.-T. Lee, K.-T. Chen, H.-I. Su, C.-L. Lei, Are all games equally cloud-gaming-friendly? An electromyographic approach, in: 2012 11th Annual Workshop on Network and Systems Support for Games (NetGames), IEEE, 2012, pp. 1–6.

[28] S.S. Sabet, S. Schmidt, S. Zadtootaghaj, C. Griwodz, S. Möller, Delay Sensitivity Classification of Cloud Gaming Content, in: Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems, MMVE '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 25–30, [Online]. Available: https://doi.org/10.1145/3386293.3397116.

[29] I. Slivar, M. Suznjevic, L. Skorin-Kapov, M. Matijasevic, Empirical QoE study of in-home streaming of online games, in: Annual Workshop on Network and Systems Support for Games, Vol. 2015, 2015.

[30] S. Möller, D. Pommer, J. Beyer, J. Rake-Revelant, Factors influencing gaming QoE: Lessons learned from the evaluation of cloud gaming services, in: Proceedings of the 4th International Workshop on Perceptual Quality of Systems (PQS 2013), 2013, pp. 1–5.

[31] Y.-C. Chang, P.-H. Tseng, K.-T. Chen, C.-L. Lei, Understanding the performance of thin-client gaming, in: 2011 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR), IEEE, 2011, pp. 1–6.

[32] M. Claypool, D. Finkel, A. Grant, M. Solano, Thin to win? Network performance analysis of the OnLive thin client game system, in: 2012 11th Annual Workshop on Network and Systems Support for Games (NetGames), 2012, pp. 1–6.

[33] T. Cao, Y. Jin, X. Hu, S. Zhang, Z. Qian, B. Ye, S. Lu, Adaptive provisioning for mobile cloud gaming at edges, Comput. Netw. 205 (2022) 108704, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912862100565X.

[34] X. Deng, J. Zhang, H. Zhang, P. Jiang, Deep-Reinforcement-Learning-Based Resource Allocation for Cloud Gaming via Edge Computing, IEEE Internet Things J. 10 (6) (2023) 5364–5377.

[35] N. Ahmad, A. Wahab, J. Schormans, A.A. Arnab, Significance of Cross-Correlated QoS Configurations for Validating the Subjective and Objective QoE of Cloud Gaming Applications, Future Internet 15 (2) (2023) [Online]. Available: https://www.mdpi.com/1999-5903/15/2/64.

[36] O.S. Peñaherrera-Pulla, C. Baena, S. Fortes, E. Baena, R. Barco, Measuring Key Quality Indicators in Cloud Gaming: Framework and Assessment Over Wireless Networks, Sensors 21 (4) (2021) 1387.

[37] ITU-T Study Group 12, Recommendation ITU-T G.1032: Influence factors on gaming quality of experience, in: ITU-T G-series Recommendations, International Telecommunication Union - Telecommunication Standardization Sector, Geneva, Switzerland, 2019.

[38] Moonlight. An open source Nvidia gamestream client, Nvidia, 2021, [Online]. Available: https://moonlight-stream.org/, Accessed July 2021.

[39] C. Baena, S. Fortes, O. Peñaherrera, R. Barco, A Framework to boost the potential of network-in-a-box solutions, in: 2021 12th International Conference on Network of the Future (NoF), 2021, pp. 1–3.

[40] C. Baena, O.S. Peñaherrera-Pulla, L. Camacho, R. Barco, S. Fortes, E2E dataset of video streaming and cloud gaming services over 4G and 5G, 2022, http://dx.doi.org/10.21227/k0w8-qz67, [Online]. Available: http://dx.doi.org/10.21227/k0w8-qz67.

[41] C. Baena, O. Peñaherrera, L. Camacho, R. Barco, S. Fortes, Video Streaming and Cloud Gaming services over 4G and 5G: a complete network and service metrics dataset, 2022, http://dx.doi.org/10.36227/techrxiv.21456267.v1, [Online]. Available: https://www.techrxiv.org/articles/preprint/Video_Streaming_and_Cloud_Gaming_services_over_4G_and_5G_a_complete_network_and_service_metrics_dataset/21456267.

[42] R. Hyndman, A. Koehler, Another look at measures of forecast accuracy, Int. J. Forecast. 22 (2006) 679–688.

Carlos Baena received his M.Sc. degree in telematic engineering from the University of Malaga, Spain. Since 2018, he has worked as a research assistant with the Department of Communications Engineering, where he is currently pursuing his Ph.D. focused on E2E network optimisation for video and gaming services.

O.S. Penaherrera-Pulla (M.Sc.) is a research assistant at Universidad de Malaga. He received his engineering degree in Electronics with a mention in Telecommunications in 2017 from the Universidad Politecnica Salesiana in Ecuador and his master's degree in Telematics and Telecommunication networks in 2020 from the University of Malaga, Spain. Currently, he is pursuing his Ph.D. degree focused on gaming and AR/VR applications, their performance monitoring, and the assessment and optimisation of these services for wireless networks.

Raquel Barco received her M.Sc. and Ph.D. degrees in telecommunication engineering from the University of Malaga, Spain. In 2000, she joined the University of Malaga, where she is currently a Full Professor. She has worked on projects with major mobile communications operators and vendors and is an author of more than 100 high-impact papers.

Sergio Fortes received his M.Sc. and Ph.D. degrees in telecommunication engineering from the University of Malaga, Spain. He began his career in the field of satellite communications, holding positions in European space agencies where he participated in various research and consultant activities on broadband and aeronautical satellite communications. In 2012, he joined the University of Malaga, where his research is focused on the application of machine learning techniques for the advanced management of cellular, satellite, and IoT.