Network Quality-Aware Architecture for Adaptive Video Streaming from Drones

Jesús Molina, David Muelas, Jorge E. López de Vergara, José Javier García-Aranda

Abstract— Video streaming over IP networks presents several challenges for remote drone piloting. To achieve a high Quality of Experience, minimal latency is mandatory. However, wireless links usually impose dynamic changes to Quality of Service conditions. Moreover, bandwidth limitations can increase both the final perceived latency and packet loss during video streaming. These circumstances require an architecture capable of estimating network performance and applying corrective actions in a timely manner to optimize application-level quality. In this paper, we present such an architecture, and discuss the results of its application in video streaming for remote drone piloting. Our proposal offers a framework with low coupling between its functional blocks and high adaptability to dynamic scenarios. Accordingly, we aim to pave the way for reactive applications that leverage edge-computing elements and adapt to network conditions.

Keywords—Quality of Service, Quality of Experience, Quality Measurement, Q4S Protocol, VMAF.

INTRODUCTION

THE development of both computational and network performance is fostering the deployment of ambitious, real-time applications that serve content from the network edge [1]. These applications are intended for diverse use cases, such as connected homes, health monitoring, connected cars, industry 4.0, and smart cities. However, the assurance of adequate Quality of Service (QoS) levels for such applications is posing significant challenges in network and performance monitoring. Specifically, low latency levels and high bandwidths are common application requirements [2] that can compromise service quality as they vary over time.

These challenges—particularly amidst the increased connectivity provided by the Internet of Things (IoT)—are fueling novel solutions that alleviate the burden of variable network conditions. As devices are transmitting more multimedia content—i.e., the Internet of Multimedia Things (IoMT) [3]—this matter is becoming increasingly urgent because users' Quality of Experience suffers when network performance is insufficient. This deteriorates because IoT devices usually have modest capabilities, which calls for solutions that are both fitted to such devices and capable of timely service degradation detection.

In this work, we present a framework to improve adaptive video streaming from drones over IP networks. Its design is intended to provide the highest-achievable QoE with respect to network performance indicators. For this, our solution integrates monitoring elements to assess network performance between application endpoints; modules to define data-driven adaptive policies that maximize users' QoE with respect to network QoS; and agents that adapt the application to actual network conditions in real-time.

Two main contributions stand out in this paper: First, policy learning is conducted in a separate phase (prior to the application setup) via the maximization of a QoE metric when network performance indicators are known, which reduces the computational requirements during operation. Second, decoupling among functions provides a flexible and cost-effective approach, adjusting services to the actual end-to-end performance. To highlight the usefulness of this approach, we present a use case in which video is streamed in real time from an IoT node aboard a drone to a remote receiver. Subsequently, we offer an empirical analysis according to strict bandwidth and latency restrictions.

The remainder of this paper is organized as follows: First, we describe our proposed architectural design and its operation. Subsequently, we present the experimental use case, which highlights the main advantages of our approach. We then analyze and discuss the implications and applicability of our results to broader scenarios. Finally, we conclude this paper by highlighting the key findings of our work.

ARCHITECTURAL DESIGN AND OPERATION

The design principles applied in this work are intended to maximize QoE for actual network conditions with high flexibility and adaptability. Given a configurable application, our solution adapts its parameters (e.g., video streaming frame rate) to the most optimal setting with respect to network QoS indicators. As QoS varies over time, the system periodically and unobtrusively monitors network performance, and reacts when indicators change.

Figure 1 displays the architectural design of our solution, including its main entities and streams. We differentiate two phases to achieve a high degree of decoupling between QoE optimization and application control. This strategy reduces the computational burden during operation. This is critical when functions are placed in reduced IoT devices, such as a System-on-Chip (SoC) aboard a drone, at the cost of having immutable

First three authors are with the Department of Electronics and Communication Technologies, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Spain. Last author is with Nokia Spain. *Manuscript received April 26th 2019, revised July 30th 2019, revised December 3rd, 2019, accepted January 5th, 2020. Cite as: J. Molina, D. Muelas, J.E. López de Vergara, J.J. García-Aranda, "Network Quality-Aware Architecture for Adaptive Video Streaming from Drones", IEEE Internet Computing, 24(1), January/February 2020, pp. 5-13. DOI:10.1109/MIC.2020.2965492.*

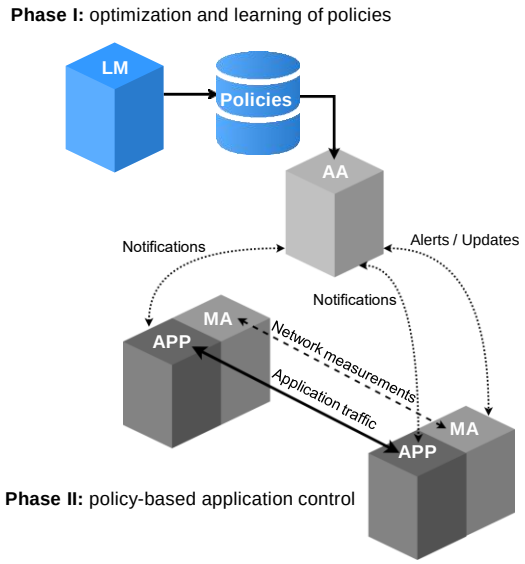


Figure 1. Diagram displaying the entities and streams of our solution. Phase I: Learning Module (LM) and policy database. Phase II: Application (APP), Monitoring Agents (MA) and network measurements, Actuation Agent (AA), and control flows.

optimization rules. However, and given the input of the models, we overcame this apparent shortcoming using extensive emulation of possible cases during the learning phase.

Phase I: optimization and learning of policies

In short, *Phase I* seeks the set of policies—i.e., application settings and update rules—that achieves the optimal QoE, given the network QoS. The main entity in this phase is a Learning Module, which takes the following: (i) a quality estimator or utility function; (ii) a sample of application-specific traffic under different configurations; and (iii) a representative grid of the network conditions as inputs. In this manner, the Learning Module maps application conditions into quality estimation, network restrictions into quality estimator, and application network requirements into network conditions. The outcome of this phase is a set of policies derived from the optimization of quality/utility. For this, we applied grid search, as it suited the complexity of the parametric space of the optimization. Moreover, and given the sources of uncertainty in QoE and QoS estimations, the optimization process searches for the maximum-expected QoE with respect to the application settings and QoS levels.

The Learning Module relies on an analysis of the traffic generated by the target application with varying configurations and network performance. Based on the principles in [4], this module emulates impairments to the application-plane data according to a variety of configurations, and obtains an estimation of their effect on the utility/quality value. This approach offers a rich source for testing diverse scenarios, and has proven to be an adequate method for assessing quality in this type of application [5]. Particularly, it provides the configuration to consider for the maximization of quality or utility when network performance indicators are known.

Policies are stored in a database from which they are provided to the entities operating in the second phase. In this setup, we used a straightforward management approach, whereby policies are assumed stationary, and we consider them as immutable after the learning phase is complete. However, this design can be extended to include on-line learning schemes—even using reinforcement to improve the overall application response with time [6]. Hence, this framework enables the introduction of feedback loops, although these were not supported in the studied case.

Phase II: policy-based application control

Entities in *Phase II* follow a modular design to improve flexibility and facilitate network slicing, which may reduce interference between functions. They are classified into three planes, based on the functions that they include: monitoring, application, and actuation.

1) **Monitoring plane:** This plane oversees the testing of network conditions. It comprises two Monitoring Agents, which are associated with their respective application endpoints. The agents conduct active measurements through message exchange. With this, the plane determines if requirements imposed by the application are met. If network conditions change, one of the Monitoring Agents alerts the actuation plane. To reduce intrusiveness, the Monitoring Agents thin out the number of indicators that represent the network status. Additionally, their levels adapt to the application requirements to reduce intrusiveness. Finally, the actuation plane can update the target performance levels monitored by the agents if the application is reconfigured.
2) **Application plane:** This includes the application functions and streams with QoE needs. The application plane must report its QoS requirements for the measurements to be adapted.
3) **Actuation plane:** This plane encompasses an Actuation Agent that functions to adapt the application parameters to the current network conditions. The Actuation Agent receives notifications from the Monitoring Agents and updates the application with the current network quality information. The Actuation Agent links the measured levels to the application parameters that maximize QoE using the policies defined in the previous phase. The actuation plane may be placed in a separate network element or slice from that of the application and monitoring entity.

The Monitoring Agents receive feedback from the Actuation Agent and update the performance levels that application demands. The Actuation Agent controls the application configuration, and it decides if the application can be launched (or when it is finished) by notifying the Monitoring Agents.

Streams in the architecture induce an implicit trade-off between the decoupling of functions and the overhead of communications among them: Although different functional blocks may be placed in different nodes, locality can reduce the computational costs, delays, and intrusiveness of certain flows. Moreover, the actuation plane can be deployed in a separate network slice, whereas the application and monitoring planes must be attached to the same one. This allows the Monitoring

Agents to estimate the actual service provided to the application, and can reduce reconfiguration discontinuities caused by network performance breakdowns.

EMPIRICAL EVALUATION

Use case description

Our use case encompassed video streaming from a SoC using a logarithmic hop encoding (LHE) implementation. This experimental video codec [7] is resilient to packet loss, has a reasonable compression rate, and has been successfully deployed in devices with hardware accelerators, such as field-programmable gate arrays (FPGAs) [8]. The video is transported over IP networks via RTP and is used to pilot a drone, which involves severe real-time restrictions with strong latency constraints and heavy penalties if the video stalls. Consequently, the application prioritizes low latency and ongoing service over video quality. Other priority rules could be implemented if the multimedia streaming had a different objective from piloting a drone.

Here, the Learning Module maps network QoS indicators into optimal application parameters: Application QoE is defined by the QoE of video transmission and drone piloting fluency. Regarding video QoE, video multimethod assessment fusion (VMAF) [9] was used for quality assessment, given its apparent performance in recent empirical comparisons [10]. For the emulation of network conditions during the Learning Phase, we used standard Linux `netem` filters to apply a grid of delays and packet loss to multimedia traffic transmitted from a network interface (see https://wiki.linuxfoundation.org/networking/netem for further details). This process was automated using shell scripting to explore the expected VMAF value in a uniform grid of network performance parameters.

The Monitoring Agent was based on the Quality for Service (Q4S) protocol [11]. Q4S is a text-based protocol with a client-server architecture that defines messages to negotiate and conduct QoS measurement sessions. Q4S evaluates the network in two phases: *Negotiation*—initial network assessment to determine if QoS indicators fulfill application requirements; and *Continuity*—measurements during application execution to verify that negotiated requirements are still met. Q4S includes a third phase, *Termination*, which occurs when network conditions are below the minimum requirements.

Q4S starts in the *Negotiation* phase, when Monitoring Agents estimate the end-to-end bandwidth, jitter, packet loss, and latency. If constraints are unfulfilled, measurements are repeated after reducing the target service level. As soon as the constraints are met, the Q4S server notifies the Actuation Agent, which starts the application with the configuration that maximizes the QoE with respect to the negotiated conditions. Otherwise, application may not start. Subsequently, Q4S enters the *Continuity* phase, in which the Monitoring Agents periodically measure latency, jitter, and packet loss. If any Monitoring Agent detects a violation of the QoS requirements, it sends an alert to the Actuation Agent via a simple Representational State Transfer (REST) interface.

The Actuation Agent receives Q4S alerts and updates the application parameters according to the predefined policies. To do so, the Actuation Agent must know Application network requirements for each setting to compare them with the received alerts. If network conditions improve, the server-side Monitoring Agent sends a *Recovery* message to the Actuation Agent, which elevates the application QoS. However, if the QoS targets are not eventually met, the Monitoring Agents notify the Actuation Agent, which may close the application. Our implementation of the Monitoring and Actuation agents is available at https://github.com/hpcn-uam/q4s/.

Testbed and experimental design

The scenario and hardware details for the proof-of-concept evaluation are summarized in Figure 2. The figure shows the placement of the following functional blocks within the different nodes: a SoC capturing and transmitting the video, a PC receiving and displaying the video, and a router connecting the two.

Phase I was executed in the PC. Software modules obtained subjective quality estimations based on reference and impaired video sequences under varied network conditions. An evaluation of element behavior in *Phase II* was conducted with commodity and modest-cost hardware elements. We used a ZynqBerry (datasheet available at https://wiki.trenz-electronic.de/display/PD/TE0726+-+ ZynqBerry) as SoC for video streaming using LHE offloaded in the FPGA [8]. As the SoC is the content provider, it was chosen to place the server Monitoring Agent and Actuation Agent; the PC assumed the client role and corresponding Monitoring Agent.

Given the particularities of IoMT use cases, our evaluation was based on the following experimental inquiries:

1) Which effects have relevant QoS factors and their implications for policy definitions.
2) To what extent the framework can adapt the application behavior to network QoS.
3) How resource consumption increases after introducing Actuation Agent and Monitoring Agent in the IoMT device.

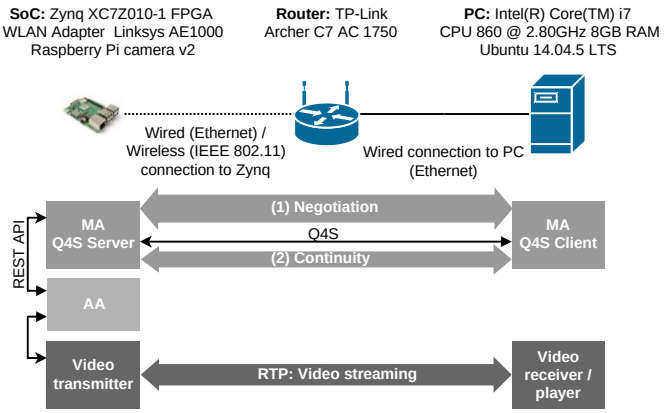


Figure 2. Hardware testbed for evaluation. Entity placement and streams are identified. Unlabeled black solid arrows represent connectivity for notifications, alerts, and actuation.

The analysis of QoS influence on the perceived QoE focused on packet loss and latency, given their prominence in the degradation of multimedia and real-time services over IP networks. The tests were oriented to determine how packet loss affected the video QoE—latency, jitter, and bandwidth limitations usually produce the same effect—, and how latency influenced drone piloting.

The assessment of video QoE relied on the analysis of the main and interaction effects of packet loss and Application settings; specifically, the luminance-bandwidth-chrominance (YUV) profile, frames per second (FPS), packet size, and codified block size. The Learning Module emulated the transmission of a first-person drone video (coded with varied parameters) across a network. The video was restored to the original settings and compared with the original when received. The setup included strict real-time bounds (less than 100 ms) for the delay between a driving action and the first frame presenting the result. Note that decreasing the frame rate reduces generated traffic, but increases the time between frames, which leads to a trade-off between lower latency and network bandwidth requirements.

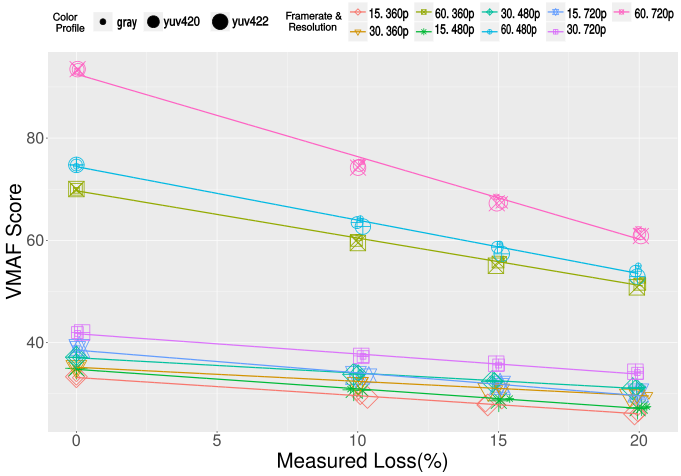
Subsequently, the system was tested to determine how it behaved under variable network conditions, and its overall performance was evaluated. To accomplish this, we monitored the application settings, CPU, and memory usage in several executions lasting 90 s each. Resource consumption was measured in the ZynqBerry, as it was the most limited device.

ANALYSIS OF RESULTS

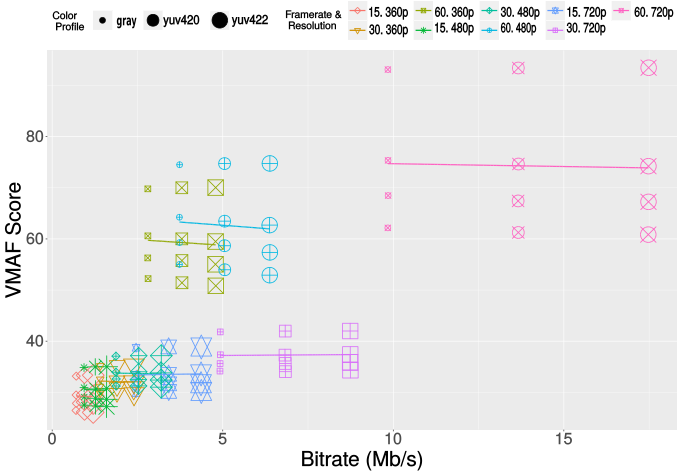
Network performance and video QoE

Figure 3 summarizes the video QoE analysis. Figure 3 (a) shows the test results for how packet loss influenced the VMAF score under different application configurations—i.e., which factors were most significant for QoE, and how QoE degraded when the loss increased. Figure 3 (b) complements the analysis by linking QoE to bandwidth consumption.

Frame rate produced the most significant impact on the QoE scores, followed by resolution; whereas color profile was essentially irrelevant to the scores. Additionally, it became apparent that the packet loss effect was higher in the upper QoE levels. Finally, frame rate was found to be the most influential factor for bandwidth consumption. These relations are the basis for the definition of policies, as they link application settings, network conditions, and expected QoE.



(a) VMAF score in different setups with respect to packet loss.



(b) VMAF score in different setups with respect to bitrates.

Figure 3. Analysis of QoE with diverse application configurations. Point size shows three different color profiles (gray (YUV 4:0:0), YUV4:2:0, and YUV4:2:2). Color and shape of the points show different frame rates (15, 30 and 60 frame per seconds) and resolutions (360p, 480p and 720p). Straight lines show the trend in each case.

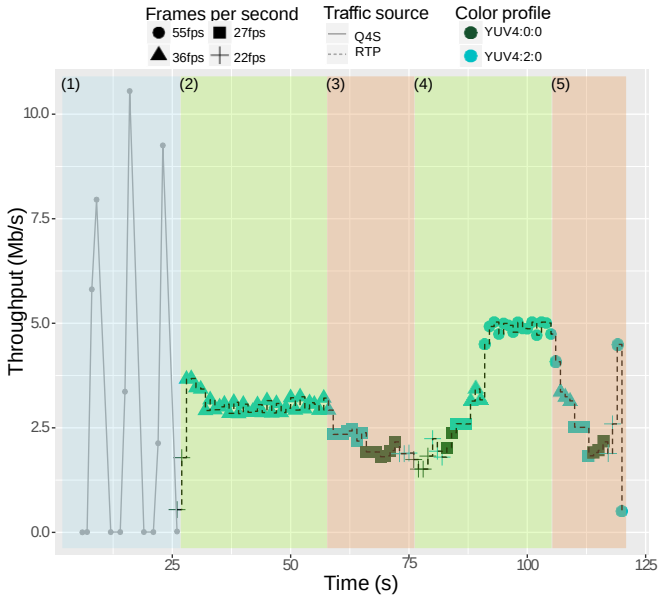
System behavior

Figure 4 (a) shows how the system reconfigured the multimedia application under varying network conditions; the background color of each numerated region distinguishes a different operational condition.

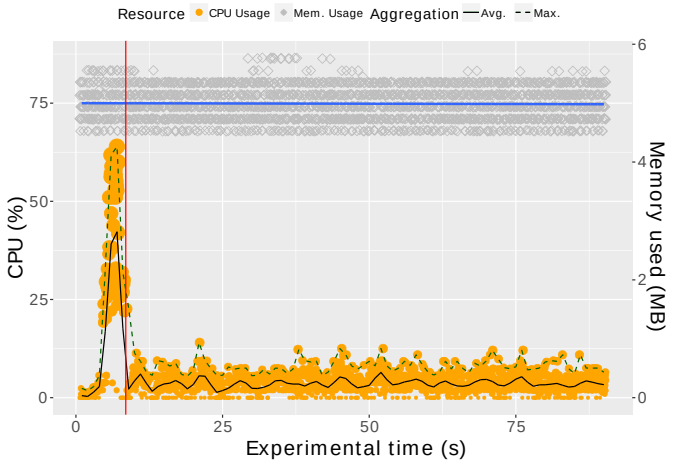
Initially, the `netem` filter restricted bandwidth to 10 Mb/s, which lies below the highest application requirements (1). Measurements were then repeated until both Monitoring Agents negotiated the initial conditions—this explains the observable traffic peaks in the figure. The Actuation Agent then launched the application with the negotiated configuration. Subsequently, the bandwidth limitation was removed (2), and latency was increased to 100 ms (3) to force a reconfiguration of the application as the requirements were no longer met. The Monitoring Agent detected the threshold violation and notified the Actuation Agent, which reduced the QoE and generated traffic. Subsequently, latency was restored (4), which caused the server Monitoring Agent to notify the Actuation Agent of the network performance recovery to increase quality. Finally, the latency constraint was reestablished (5), which triggered a QoE decrease until the application closed due to insufficient QoS—this behavior was defined for testing purposes, despite being inadvisable during real drone piloting.

Regarding computational requirements, Figure 4 (b) displays the CPU and memory footprints during the experiments, as these are typically the most limiting resources for a SoC. Specifically, large overhead in either would reduce the applicability of the tested solution, and may impact the overall behavior of the drone, by substantially increasing power consumption.

In the figure, the CPU usage presents in two regions, which is consistent with the main monitoring phases. There is a remarkable peak during the Negotiation phase (the region before red line), in



(a) Adaptation to operational conditions. (1) Before application starts, Q4S protocol is used to probe the network. (2) Stable QoS: video is transmitted over RTP at 36 frames per second in color. (3) QoS degrades: video is transmitted at 27 frames per second, first in YUV4:2:0 (color) and next in YUV4:0:0 (gray), finally reducing the frame rate to 22 frames per second. (4) QoS recovers: video is transmitted from 22 to 55 frames per second, switching also from gray to color. (5) QoS below requirements: video changes again the frame rate and color profile, until no transmission policy fits in network conditions.



(b) Resource usage as CPU percentage (left axis) and memory Megabytes (right axis). Vertical red line separates negotiation and continuity phases, points (orange for CPU and gray for memory) show the measured consumption for the set of experiments executed, solid lines (black for CPU and blue for memory) are average values, and dashed lines indicate maximum values for CPU percentage.

Figure 4. System behavior: (a) actuation in application configuration; (b) resource consumption.

which the average CPU usage was approximately 40%. Subsequently, it fell below 5%, where it remained for most of the duration. Although CPU usage during the Negotiation phase was intensive, it is worth noting that the application is launched later; hence, there was no interference with its operation. Once the application was launched, the CPU usage decreased to below 10%, leaving enough space for the application, even when the video quality was switched due to changes in network conditions. Latency on quality switching is negligible, as it is done by writing in a register of the SoC. Memory usage remained fairly constant at approximately 5 MB.

DISCUSSION AND RELATED PROPOSALS

Our framework separates policy learning and application control, as it decomposes the entire workflow into two phases: a complex one with no latency constraints, and a simpler one with real-time constraints. With this, we aim at taking advantage of user-centered policies based on perceived QoE estimations while reducing the computational burden.

According to the results, we can state two immediate facts. First, the methodology for policy definition seems to provide a suitable framework for optimizing QoE according to QoS indicators. Second, our solution can detect performance changes and adapt the application configuration to network conditions in a timely manner.

Regarding applicability to broader contexts, our proposal focuses on maximizing QoE instead of minimizing consumption of network resources. Hence, this framework should be carefully tuned and deployed in scenarios that require more control on constrained networks.

Our proposal offers an alternative to narrower or less flexible solutions, such as custom modifications to streaming protocols [12], or applications that couple network measurement and configuration—e.g., adaptive video encoding. Although our solution shares the main objectives of such proposals, we focus on the consumer side instead of assuming control of the infrastructure. Moreover, we point to potential extensions of our work by introducing more complex learning methods and network control appliances. For instance, some proposals [6], [13] have integrated mean opinion score (MOS) and other network-related factors to improve the user's experience via topological optimization and bandwidth allocation. We believe that combining these approaches will improve future applications served from drones and IoMT nodes at the network edge.

CONCLUSIONS

This article presented an architecture for promptly adapting applications to actual network conditions. The strength of our architecture is to offer the best QoE to the user according to the available network QoS. Its design is user-oriented, focusing on satisfaction instead of network performance.

Our approach offers the possibility of changing the content dynamically, depending on the measurements taken. If network performance decreases, the Actuator Agent reconfigures the application and reduces requirements while maximizing the provided QoE. Conversely, the application service level rises when network conditions recover.

We have presented supporting evidence for the usefulness of our method, providing empirical insights into a use case related to multimedia streaming for remote drone piloting. Our results illustrate how policies are obtained from the maximization of QoE given diverse network conditions and how the architecture adapts

video streaming via its reconfiguration under dynamic network performance, and demonstrate the low computational burden of our design. We believe our findings can lead to new research in the optimization of smart strategies for drones and IoMT nodes, which dimensions significantly affect the operation of such strategies, and how they can be fairly compared.

ACKNOWLEDGMENT

This research received funding from the projects RACING DRONES (MINECO/FEDER RTC-2016-4744-7) and TRÁFICA (MINECO/FEDER TEC2015-69417-C2-1-R).

REFERENCES

[1] M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Dustdar, O. Scekic, T. Rausch, S. Nastic, S. Ristov, T. Fahringer, "A deviceless edge computing approach for streaming IoT applications," *IEEE Internet Computing*, 23(1), pp. 37-45, January 2019.

[2] X. Zuo, Y. Cui, M. Wang, T. Xiao, X. Wang, "Low-latency networking: Architecture, techniques, and opportunities," *IEEE Internet Computing*, 22(5), pp. 56-63, September 2018.

[3] S.A. Alvi, B. Afzal, G.A. Shah, L. Atzori, W. Mahmood, "Internet of multimedia things: Vision and challenges," *Ad Hoc Networks*, 33, pp. 87-111, 2015.

[4] H.D. Moura, D. Fernandes Macedo, M.A.M. Vieira, "Automatic Quality of Experience Management for WLAN Networks using Multi-Armed Bandit," in Proc. 16th IFIP/IEEE International Symposium on Integrated Network Management, April 2019.

[5] Z. Duanmu, A. Rehman, Z. Wang, "A Quality-of-Experience Database for Adaptive Video Streaming," *IEEE Trans. on Broadcasting*, 64(2), pp. 474-487, June 2018.

[6] X. Huang, T. Yuan, G. Qiao, Y. Ren, "Deep Reinforcement Learning for Multimedia Traffic Control in Software Defined Networking," *IEEE Network*, 32(6), pp. 35-41, November 2018.

[7] J.J. García Aranda, M. González Casquete, M. Cao Cueto, J. Navarro Salmerón, F. González Vidal, "Logarithmical hopping encoding: a low computational complexity algorithm for image compression," *IET Image Processing*, 9(8), pp. 643-651, 2015.

[8] T. Alonso, M. Ruiz, A.L. García-Arias, G. Sutter, J.E. López de Vergara, "Submicrosecond Latency Video Compression in a Low-End FPGA-based System-on-Chip," in *Proc. 28th Int. Conf. Field-Programmable Logic and Applications*, August 2018.

[9] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, M. Manohara, "Toward a practical perceptual video quality metric," *The Netflix Tech Blog*, 6, 2016.

[10] N. Barman, M. G. Martini, S. Zadtootaghaj, S. Möller, S. Lee, "A Comparative Quality Assessment Study for Gaming and Non-Gaming Videos," in *2018 Tenth Int. Conf. on Quality of Multimedia Experience*, May 2018, pp. 1-6.

[11] J.J. García Aranda, M. Cortés, J. Salvachúa, M. Narganes, I. Martínez Sarriegui, "The Quality for Service Protocol," IETF Internet-Draft, July 2019.

[12] O. Said, Y. Albagory, M. Nofal, F.A. Raddady, "IoT-RTP and IoT-RTCP: Adaptive Protocols for Multimedia Transmission over Internet of Things Environments," *IEEE Access*, 5, pp. 16757-16773, 2017.

[13] X. Huang, K. Xie, S. Leng, T. Yuan, M. Ma, "Improving Quality of Experience in multimedia Internet of Things leveraging machine learning on big data," *Future Generation Computer Systems*, 86, pp. 1413-1423, 2018.

Jesús Molina (j.molina.merchan@gmail.com) received his B.Sc. in Telecommunication Technologies and Services (2016) and M.Sc. in Telecommunication Engineering (2018) from Universidad Autónoma de Madrid, where he did his thesis in the scope of the Racing Drones project. His research interests are in the monitoring and analysis of multimedia networks.

David Muelas (dav.muelasr@gmail.com) received his M.Sc. degrees in Mathematics and Applications and Information and Communications Technologies (2015), and a Ph.D. in Computer and Telecommunication Engineering (2019) from Universidad Autónoma de Madrid. He was a researcher in the HPCN-UAM group, with interests in network traffic analysis, SDN, and applied mathematics. Currently, he is a data scientist in BBVA Data & Analytics.

Jorge E. López de Vergara (jorge.lopez_vergara@uam.es) is an associate professor at Universidad Autónoma de Madrid and founding partner of Naudit HPCN. He received his M.Sc. and Ph.D. degrees in Telecommunication Engineering from Universidad Politécnica de Madrid in 1998 and 2003, respectively. He researches network and service management and monitoring, having co-authored more than 100 papers on this topic.

Jose Javier García Aranda (jose_javier.garcia_aranda@nokia.com) is an innovation project leader at Nokia Spain. He received his M.Sc. and Ph.D. degrees in Telecommunication Engineering from Universidad Politécnica de Madrid in 1996 and 2015, respectively. He is the main author of the LHE codec and Q4S protocol, and led the Racing Drones project.