A Simulation Framework for QoE-Aware Real-Time Video Streaming in Multipath Scenarios

Manlio Bacco[1], Pietro Cassarà[1], Alberto Gotta[1], and Massimo Puddu[2]

[1] Institute of Information Science and Technologies (ISTI), National Research Council (CNR), Pisa, Italy - e-mails: [name.surname]@isti.cnr.it
[2] University of Pisa, e-mail: massimo.puddu5@studenti.unipi.it

Abstract. This work presents the ongoing development of a simulation/emulation framework for real-time multimedia transmissions in multichannel scenarios. The proposed software engine can be used to simulate the aggregation of multiple network links delivering a video flow from a video source, such as a camera mounted on-board an Unmanned Aerial Vehicle (UAV), to the indented receiver, such as the pilot on the ground. The software engine can also be used in real testbeds exploiting multiple physical links, such as 4G/5G cellular connections. The main aim of this novel software framework is to support the design, development, and test of different scheduling strategies to achieve real-time, good quality and fluidity, energy-efficient multimedia transmissions, by selecting the optimal subset of network channels able to meet the target Quality of Experience (QoE) at the receiver.

Keywords: video quality · PSNR · simulator · scheduling · multipath

1 Introduction

UAVs are attracting the attention of both the scientific community and the industrial world. Their use in science is mainly directed to UAV-assisted base-station nodes [1] in 5G networks or more complex heterogeneous networks such as space information networks [2]. In civil applications, UAVs are nowadays employed in a variety of application scenarios, thanks to the versatility and low cost. Application fields of interest vary from supervising danger situations in smart environments, remote monitoring of buildings and infrastructures, treatments in precision agriculture, rescue procedures, etc. [3–5]. In those scenarios, an UAV can provide both a fast deployment and a live visualization to remote operators through on-board cameras [6]. The advantage of having a reliable live video-feedback from an UAV is twofold: on the one hand, it opens to the use of Augmented Reality (AR) / Virtual Reality (VR) techniques to support operators; on the other hand, it enables the remote management of the UAV. Indeed, remote piloting leverages Global Navigation Satellite System (GNSS)-based services through way-points, which trace a route to the destination. However, remote visual feedback for the pilot is a requirement to perform adjustment and emergency maneuvers, thus constantly monitoring the surrounding environment and avoiding possible accidents. The extreme heterogeneity of conditions, such as time- and space-varying channels characteristics, require very careful testing of the communication subsystem, and real testbeds are both costly and risky. For this reason, a simulation/emulation tool is an effective solution in this regard. To fill the gap between a simulation platform and a real environment, we propose in this work an emulation tool, able to reproduce the communication chain between a remote UAV and a visualization interface used by the pilot. Both simulated and real links between the transmitter and the receiver can be used and tested. The proposed tool differs from other renowned ones, like EvalVid [7], because it aims at designing and implementing multipath scheduling policies; for instance, exploiting multiple cellular connections jointly, so meeting the requirement of a minimum QoE [8] (measured according to the Mean Opinion Score (MOS)) when subject to real or simulated Quality of Service (QoS) parameters, such as Packet Loss Rate (PLR), jitter, and delay. Furthermore, the emulation tool will implement a man-in-the-loop feedback chain to allow the end-user (e.g. the pilot) to provide subjective feedback (e.g. perceived QoE) to the scheduler. The feedback provided by the end-user can be used jointly with objective feedback metrics (e.g. QoS), such as Peak Signal-to-Noise Ratio (PSNR). This additional subjective feedback can be leveraged through a reinforcement approach to map QoE onto QoS in different scenarios, also opening to the use of proactive [9] and reactive coding [10, 11] to counteract impairments.

In Section 2, we present a brief survey of analytical models to map QoS onto QoE. In Section 3, the proposed simulator framework is presented. The conclusions and the future works are in Section 4.

2 QoE Models for Video Quality Assessment

As mentioned in Section 1, reliable video feedback from an UAV to a remote operator on the ground allows developing real-time remote command and control systems for UAVs. In this scenario, quantifying the feeling of the operator about the received video, in terms of QoE, becomes crucial to design a control system adapting the video transmission to the quality of the network links, expressed in terms of QoS. Appropriate metrics are then needed to evaluate the mapping of the variations of the QoS parameters into the user-level QoE perception. Once done, such a mapping can be used by a control system to keep the user-perceived quality above an acceptable threshold. A reference metric used to measure the feeling of a user about a video is the MOS [8]. MOS evaluation is performed using the statistical inference on the opinion scores, usually within a five-point interval. For the scope of this work, both video quality and video fluidity are considered, thus entailing the need for control policies. Video artifacts, missing frames, poor fluidity, and so on, mainly depend from QoS parameters such as packet loss, delay, jitter, or reordering.

Several works in the literature, e.g. [12–14], faced with the definition of mapping QoS onto QoE. In [13, 14], the authors discuss learning approaches for both online and offline mapping. All the proposed mappings build on quality comparisons between the undistorted video and the received video, namely *reference* and *outcome*, respectively. The quality of the outcome can be rated in terms of MOS using the reference. Whether the reference is available or not defines the following types of metrics: Full Reference (FR), No Reference (NR), and Reduced Reference (RR). In the case of FR, subjective and objective comparisons of the outcome with the reference can be performed, because both are available at the receiver. Hence, an accurate metric can be derived, performing the offline estimation of the model parameters. Offline estimations can be very precise, but also rather complex and strictly dependant from the considered settings. In the case of NR, a quality score must be derived directly from the outcome, because no reference video is available at the receiver. An alternative comes from online estimations of QoE, for instance through the end-user's feedback. NR-based metrics lack the possibility of discerning between quality issues and disturbance due to the network [13,14]. On the one hand, the latter cannot be precisely captured; on the other hand, obtained metrics are rather simple (low complexity) and thus suitable for online use in resource-constrained and/or real-time settings. In the case of RR, man-in-the-loop techniques are used, thus users' feedback is collected at the sender in addition to network statistics. Such a feedback can be used to evaluate the perceived QoE and consequently mapped onto QoS. The estimation of the mapping parameters can be performed using online approaches [14]. Also, the RR metric is rather simple, thus opening to its use in resource-constrained and/or real-time settings, as in our reference scenario [6].

Figure 1 shows a qualitative mapping between QoE (y-axis) and QoS (x-axis); QoS degrades when moving from QoS_1 to QoS_2. The first region, to the left of the threshold QoS_1, represents the case of a slight degradation of QoS with negligible effects on the QoE. When the QoS degradation falls within $[QoS_1, QoS_2]$ (i.e., the second region), the QoE start decreasing and, as the line color turns from green to orange, it increasingly becomes uncomfortable for end-users. After the threshold QoS_2 (third region), QoE can be considered as unacceptable, with end-users potentially giving up on the service. In [15,16], the authors discuss the results of a metric having a similar trend to that in Figure 1. They show that a model based on Eq. (1) is suitable for a scenario using cellular networks.

$$QoE = k_1 - \frac{k_2}{(1 + \frac{k_3}{QoS k_4})^{\eta}} \tag{1}$$

The maximum value of QoE is imposed through the parameter $k1$. Instead, the parameters $\{k2, k3, k4, \eta\}$ depend on the network and on the video codecs used for the video streaming, and must be evaluated empirically. The model in Eq. (1) is also adopted in [17] to analyze the mapping between QoE and QoS, taking into account the correlation of both PLR and packet size. The authors use those two parameters to evaluate the theoretically expected Decodable Frame Rate. Q can be defined as the fraction of the decodable frame rate, i.e., the ratio of the number of theoretically expected decodable frames to the total number of frames sent by a video source. Then, the authors rewrite the model in Eq. (1) as

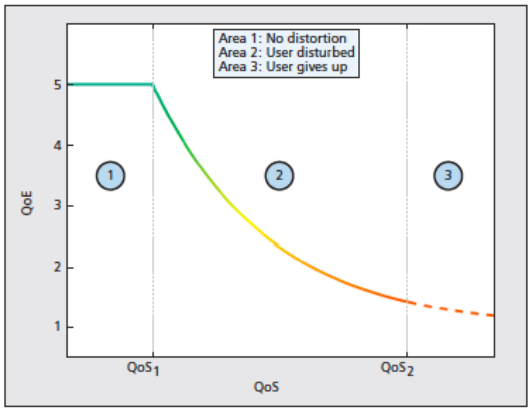

Fig. 1: Quantitative trend of the QoE vs QoS parameter

a function of Q with values ranging within $[0, 1]$, also providing an estimation for the set of parameters $\{k1, k2, k3, k4, \eta\}$. The estimation in [17] is a good match for our reference scenario [6] because the cross-layer approach proposed therein puts the network parameters, i.e., QoS, in relation with the perceived QoE. In details, the latter is influenced by the packet error rate, in turn affecting the decoding rate of video frames (I, P, and B frames).

3 The novel software tool

The logical architecture of the tool proposed in this work can be seen in Figures 2 and 3. Three main blocks can be identified: sender, channel, and receiver. The sender side takes a video stream as input, captured through the use of the PCAP interface (e.g., employing *tcpdump*/*libpcap*). The input video flow must be a RTP/UDP stream, which is then sent to a TEE[3] module (see Figure 2) responsible for the creation of N copies of the stream, with N being the instantiated source modules, corresponding to N physical channels. Each source module src_i is responsible for delivering the video to the corresponding receiver module dst_i via the physical channel ch_i. Thus, the multimedia stream to be delivered can be replicated up to N times. The logical multipath channel is composed of N physical channels, jointly used to meet the desired QoE. The scheduler module is responsible for implementing the scheduling policy, exploiting periodic feedbacks. In fact, the transmission probability pt_i is estimated by the scheduler per source module by applying a strategy aiming at maximizing the QoE of the multimedia flow. The scheduler relies on periodical feedback to adjust the transmission probability per channel according to the implemented policy, which uses the QoS parameters (see Section 2) as inputs. In more words, the copies of each packet of the stream are sent (or not) on the corresponding physical channel ch_i

[3] Name inspired by the Gstreamer *tee* module performing the same function.

according to its transmission probability pt_i; thus, it is possible to finely control the use of each physical channel, ranging from 0% to 100%.

In order to better understand how the simulator works, we should distinguish between the use of simulated physical channels (see Section 3.1) and real ones (see Section 3.2). In fact, each source element src_i opens a network socket to send data to the corresponding socket dst_i at destination: thus, it is alternatively possible to: (i) rely on the simulated channel implementation we developed to assess the impact of different channel statistics on the perceived QoE; (ii) use a custom tool to simulate/emulate physical channels, such as NetEm[4]; (iii) transmit the video via real physical channels. At this time, options (i) and (ii) are under testing, and (iii) under active development. The destination side is described in Section 3.3.

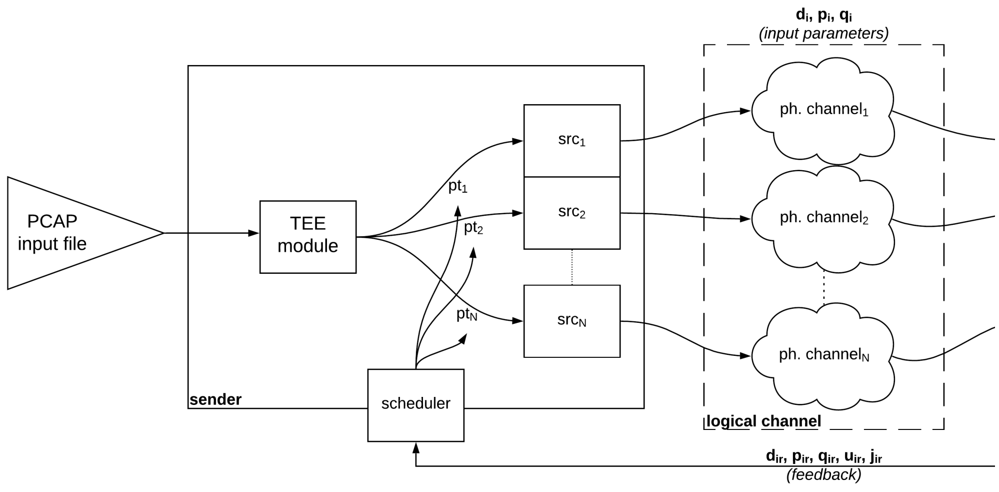

Fig. 2: sender side and channel

3.1 Simulated physical channels

When multiple physical channels are simulated, it is possible to set the channel statistics by means of an external configuration file. Three parameters must be set: the delay d_i introduced by each channel; the average rate of loss events p_i, and the average rate of packet losses q_i per channel. In more words, each time a loss event is triggered according to the rate p_i, the number of lost packets within the loss event is governed by the rate q_i, thus accounting for bursty losses. The parameters d_i, p_i, and q_i must be provided for each channel as a tuple (α_i, β_i), representing shape and scale of a Gamma distribution $\mathit{\Gamma}_i(\alpha_i, \beta_i)$.

[4] Details at `http://man7.org/linux/man-pages/man8/tc-netem.8.html`

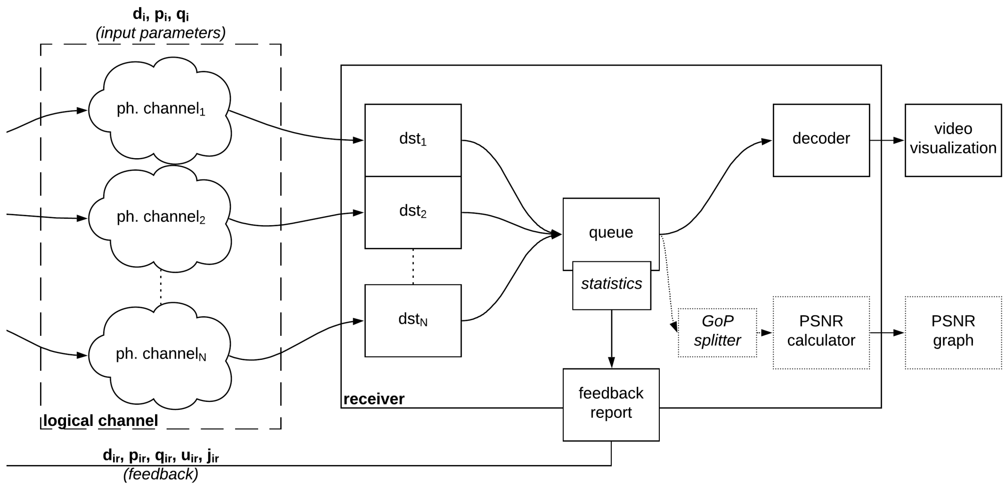

Fig. 3: channel and receiver side

3.2 Real physical channels

If a real testbed is considered, then the video is sent to a receiver running on another machine. In the case of emulated channels, external tools can be exploited (e.g. NetEm). In both cases, the configuration parameters presented in Section 3.1 are ignored. Thus, actual delay, rate of loss events, and average rate of packet losses depend on external parameters or factors.

3.3 Destination side

At the destination side, each module dst_i receives the packet stream on the listening socket and forwards them to the queue module. The latter is responsible for aggregating the N flows and calculating per-flow statistics. The tuple of values that provides the per-flow statistics are: (i) the average delay d_{ir}; (ii) the average rate of loss events p_{ir}; (iii) the average rate of packet losses q_{ir}; (iv) the average rate of out-of-order packets u_{ir}; (v) and the jitter j_{ir}. A feedback report is built by concatenating the N tuples, and then the report is sent to the scheduler module at the sender side. The queue module forwards the received stream to the video decoder, then to a video player. We assume real-time video, i.e., a maximum overall delay of $m = 200$ [ms] as in [6]. A Gstreamer-based[5] pipeline is used for playing the video, composed of the following ordered Gstreamer plugins: `udpsrc` | `rtpbin` | `rtph264depay` | `h264dec` | `audiovideosink`. Such a solution can be substituted by other solutions taking an UDP/RTP stream as input.

In the case of emulated/simulated channels with both sender and receiver running on the same machine, the three modules in dotted squares in Figure 3,

[5] Details at `https://gstreamer.freedesktop.org`

i.e. the Group of Pictures (GoP) splitter, the PSNR calculator, and the PSNR graph, are automatically enabled. The GoP splitter recognizes the end of a H.264 GoP, then it waits additional m [ms] to account for delayed or out-of-order packets, and finally writes to disk the still images (video frames) composing the GoP. Assuming that still images, representing the original video sequence, are also available at the sender side (or in a local directory whose path is set in the configuration file), PSNR can be computed. The resulting data series is then graphically visualized in the *PSNR graph* module, providing a real-time objective evaluation of the received video.

4 Conclusions

We plan to conclude soon the development and test phases of the proposed software tool, which will be freely released as open source code to the scientific community. Future works see the development of a web interface to test and use the simulator, along with the possibility of providing subjective feedback to the scheduler as regards the perceived video quality. Multiple scheduling strategies will be designed, developed, and tested, taking into account video quality, video fluidity, channels use, and energy efficiency.

References

1. B. Li, Z. Fei, and Y. Zhang, "UAV Communications for 5G and Beyond: Recent Advances and Future Trends," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2241–2263, 2018.
2. M. Bacco, L. Boero, P. Cassará, M. Colucci, A. Gotta, M. Marchese, and F. Patrone, "IoT Applications and Services in Space Information Networks," *IEEE Wireless Communications Magazine*, vol. PP, pp. 1–7, 2019.
3. M. Bacco, S. Chessa, M. Di Benedetto, D. Fabbri, M. Girolami, A. Gotta, D. Moroni, M. A. Pascali, and V. Pellegrini, "UAVs and UAV swarms for Civilian Applications: Communications and Image Processing in the SCIADRO Project," in *International Conference on Wireless and Satellite Systems.* Springer, 2017, pp. 115–124.
4. M. Bacco, F. Delmastro, E. Ferro, and A. Gotta, "Environmental Monitoring for Smart Cities," *IEEE Sensors Journal*, vol. 17, no. 23, pp. 7767–7774, 2017.
5. M. Bacco, P. Barsocchi, P. Cassará, D. Germanese, A. Gotta, G. R. Leone, D. Moroni, M. A. Pascali, and M. Tampucci, "Monitoring Ancient Buildings: Real Deployment of an IoT System Enhanced by UAVs and Virtual Reality," *IEEE Access*, vol. 8, pp. 50 131–50 148, 2020.
6. M. Bacco, P. Cassará, A. Gotta, and V. Pellegrini, "Real-Time Multipath Multimedia Traffic in Cellular Networks for Command and Control Applications," in *Vehicular Technology Conference (fall), Honolulu, Hawaii, USA.* IEEE, September 2019, pp. 1–5.
7. J. Klaue, B. Rathke, and A. Wolisz, "Evalvid–A Framework for Video Transmission and Quality Evaluation," in *International conference on modelling techniques and tools for computer performance evaluation.* Springer, 2003, pp. 255–272.

8. ITU-T, "REC-P.800.2 SERIES P: Terminals and Subjective and Objective Assessment Methods," July 2016, www.itu.int/rec/T-REC-P.800.2/en.
9. M. Wu, S. Makharia, H. Liu, D. Li, and S. Mathur, "IPTV Multicast over Wireless LAN using Merged Hybrid ARQ with Staggered Adaptive FEC," *IEEE Transactions on Broadcasting*, vol. 55, no. 2, pp. 363–374, 2009.
10. N. Celandroni and A. Gotta, "Performance Analysis of Systematic Upper Layer FEC Codes and Interleaving in Land Mobile Satellite Channels," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 4, pp. 1887–1894, 2011.
11. A. Gotta and P. Barsocchi, "Experimental Video Broadcasting in DVB-RCS/S2 with Land Mobile Satellite Channel: a Reliability Issue," in *International Workshop on Satellite and Space Communications.* IEEE, 2008, pp. 234–238.
12. U. Engelke and H.-j. Zepernik, "Perceptual-Based Quality Metrics for Image and Video Services: A Survey," in *Next Generation Internet Networks Conference.* IEEE, May 2007, pp. 190–197.
13. M. Alreshoodi, E. Danish, J. Woods, A. Fernando, and C. De Alwis, "Prediction of Perceptual Quality for Mobile Video Using Fuzzy Inference Systems," *Transactions on Consumer Electronics*, vol. 61, pp. 546–554, November 2015.
14. E. A. A. Riker, M. Mu, and S. Zeadally, "Real-time QoE Prediction for Multimedia Applications in Wireless Mesh Networks," in *IEEE Int. Conference CCNC'12*, January 2012, pp. 592–596.
15. J. Poncella, G. Gomez, A. Hierrezuelo, F. J. Lopez-Martinez, and M. Aamir, "Quality Assessment in 3G /4G Wireless Networks," *Business Media, Wireless Personal Communications*, vol. 76, pp. 363–377, November 2014.
16. V. F. Monteiro, D. A. Sousa, T. F. Maciel, F. R. M. Lima, E. B. Rodrigues, and F. R. P. Cavalcanti, "Radio Resource Allocation Framework for Quality of Experience Optimization in Wireless Networks," *Network*, vol. 29, pp. 33–39, November/December 2015.
17. M. Sidibe, H. Koumaras, I. Kofler, A. Mehaoua, A. Kourtis, and C. Timmerer, "A Novel Monitoring Architecture for Media Services Adaptation Nased on Network QoS to Perceived QoS Mapping," *Signal, Image and Video Processing*, vol. 2, pp. 307–320, October 2008.