Performance Analysis of WebRTC-based Video Streaming over Power Constrained Platforms

M. Bacco*, M. Catena*, T. de Cola†, A. Gotta* and N. Tonellotto*

*CNR, Institute of Information Science and Technologies, Italy
†DLR, Institute of Communications and Navigation, Oberpfaffenhofen, Germany

***Abstract*—This work analyses the use of the WebRTC framework on resource-constrained platforms. WebRTC is a consolidated solution for real-time video streaming, and it is an appealing solution in a wide range of application scenarios. We focus our attention on those in which power consumption, size and weight are of paramount importance because of size, weight and power requirements, such as the use case of unmanned aerial vehicles delivering real-time video streams over WebRTC to peers on the ground. The testbed described in this work shows that the power consumption can be reduced by changing WebRTC default settings while maintaining comparable video quality.**

I. Introduction

Nowadays, enabling efficient mobile multimedia delivery on portable devices is one of the key objectives of the wireless industry in the continuous upgrade of current networks and in the design of new ones. As a matter of fact, smartphones, Internet of Things (IoT) devices or other resource-constrained platforms can be in principle used to run a large gamma of applications, but the main limitation they suffer from is the energy efficiency because of the short battery duration. Therefore, the possibility to deploy services on *energy-efficient* mobile platforms is a very important topic for both industrial and research communities. In particular, the case of video-streaming application is particularly appealing because of the numerous applications, such as environmental monitoring [1] and remote surveillance, just to cite a few. In the light of this motivation, this paper focuses on a video-streaming service running on resource-constrained platforms.

Amongst all the possible protocol solutions enabling video-streaming services, this work considers the use of Web Real-Time Communications (WebRTC), a recent IETF standard [2] based on the use of the Real-time Transport Protocol (RTP). In more detail, WebRTC is a suite of communication protocols that enables real-time communications over peer-to-peer connections, such as video streaming, web conferencing, chat and data exchange provided by web applications building on JavaScript and HTML5 software technologies. As such, WebRTC is already supported by the most common web browsers for desktop and mobile platforms. In fact, the ambition of WebRTC embraces a broader number of possible application scenarios: security (e.g., the Amaryllo camera with integrated WebRTC support), remote presence on work sites [3], e-health [4], and emergency scenarios [5]. More recently, its applicability has been also extended towards IoT domains to monitor and control smart objects, as discussed in [6].

In particular, the WebRTC framework can be considered as an enabling technology in resource-constrained hardware platforms, although a deep understanding of all protocol performance implications is actually still missing. More precisely, a comprehensive characterization of energy efficiency has been not yet provided by the scientific community, to the best of authors' knowledge. As a first attempt to shed some more light on this aspect, this work experimentally analyses the power consumption of WebRTC on a Raspberry board, a small-sized low-power system powered by an ARM processor, which in spite of its dimensions is capable of video encoding and streaming at a good quality. Thus, the main aim of this work is in providing a simple but comprehensive case study on the feasibility of WebRTC-based video streaming on a Raspberry platform, by analysing the trade-off between achievable video quality and energy efficiency. To this end, we extensively tested WebRTC under both automatic and manual configurations, i.e., when framework-dependent input parameters are automatically or manually set, respectively, in order to meet a given set of requirements.

The rest of this paper is organized as follows: Section II illustrates the current state of the art of video-streaming applications running on low power systems. Section III introduces the scenario taken as reference in this work, whose experimental characterization is provided in Section IV by means of a dedicated testbed. The conclusions are drawn in Section V.

II. Related Works

The problem of power consumption in multimedia mobile devices (e.g., implementing streaming capabilities) has been quite explored by the scientific community, which has come up with different solutions to compensate the limited duration of batteries. For example, reference [7] explores the problem of energy conservation from a communication standpoint, analyzing the possible battery savings with respect to the various functionalities provided by the proposal stack. A seminal work about energy management in handsets from different angles is contained in [8], which is also the baseline for the work in [7], where both cross-layer and per-layer approaches to achieve satisfactory energy efficiency figures are discussed.

As far as energy profiling is concerned, an exhaustive review of the techniques explored so far in the ecosystem of mobile

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

applications running on devices powered by batteries is provided in [9]. By exploiting a thematic taxonomy, reference [9] describes the open research issues on this topic and underlines the need of a lightweight but accurate energy profiling model, yet to come according to the authors. Such a model should provide a simple way to predict energy consumption when the application behavior is known or can be predicted.

The analysis of energy consumption in the specific case of WebRTC-based applications is reported in [10], where Long-Term Evolution (LTE) devices are taken as references. Moving to the case of resource-constrained devices, the works in [11], [12], [13] deserve some attention as they are the very few ones that consider Raspberry PI platforms. In more words, reference [11] describes the use of WebRTC over Raspberry PI 2 B for large scale disasters scenarios, enabling real-time video streaming from an Unmanned Aerial Vehicle (UAV) powered by on-board battery to first responders (e.g., rescue teams) operating on the ground. The authors investigate the maximum number of users that can be simultaneously connected to the on-board server and the frames per second (fps) that provide an acceptable trade-off between quality and computational load. On the other hand, the authors in [12] considered Raspberry PI 3 as on-board platform for an UAV, as in our case, to implement a WebRTC-based gaming system. The main limitation in such a scenario is represented by the available power on-board the UAV, stored in lightweight and small capacity batteries that Commercial Off-the-Shelf (COTS) UAVs typically carry. A comparison on power consumption among different platforms, both non-constrained and constrained ones, is provided in [13], also suggesting techniques to further reduce the power consumption of Raspberry boards, which we implemented in our testbed.

Battery capacity consumption in resource-constrained devices is also analyzed in [14], which aims at measuring the power required to implement video encoding and streaming operations on a Imote2 wireless sensor node, equipped with a PXA271 XScale processor. In this case, an H.264 video codec is used and the received video quality is estimated through the ROPE (Recursive Optimal per Pixel Estimate) method. The most interesting point of this work is the design of a framework able to minimise the battery consumption, consisting in the proper configuration of coding and transmission parameters aiming at matching the distortion threshold, without the need to continuously estimate the relevant parameters for each video frame. Finally, reference [15] analyses the performance level provided by WebRTC on mobile devices in terms of several metrics, aiming at providing a comprehensive characterization. Nevertheless, energy issues are not taken into account.

The work contained in this paper investigates the power consumption and its impact on subjective video quality on a Raspberry PI, providing real-time video streaming services. In order to enhance overall performance with respect to power consumption, power absorption minimization is carried out to prolong the battery duration, as also proposed in [11]. Moreover, it provides a solution alternative to what proposed

Fig. 1: Raspberry PI 3 model B: a credit-card-sized computer that features an ARM CPU (quad core 1.2GHz Broadcom BCM2837 64bit), a Wi-Fi chipset (Broadcom BCM43438), 1GB RAM.

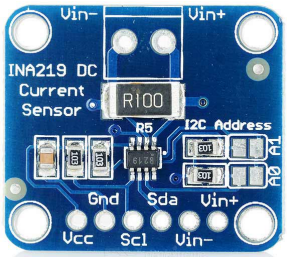


Fig. 2: The Adafruit INA219 DC current sensor in use to estimate the power absorbed by the Raspberry during its functioning.

in [16], which considered the GStreamer framework to deliver real-time video streams from moving vehicles (such as UAVs) to fixed receivers. In particular, [16] applied multipath techniques to increase the reliability when delivering video streams, neglecting however energy consumption implications. By contrast, this work provides insights on the latter by relying on the use of the WebRTC framework, focusing on power consumption and achievable Quality of Experience (QoE), in order to provide a set of different solutions to be used in scenarios with different requirements.

III. Testbed

This section describes the testbed we designed. The aim is in experimentally exploring the possibility of reducing the power consumption of WebRTC on resource-constrained devices while maintaining an overall video quality and fluidity comparable with that achievable at default settings.

The WebRTC-based sender application runs on a Raspberry PI 3 Model B, shown in Figure 1. The power absorbed by the Raspberry is estimated by using an Adafruit INA219 DC current sensor, shown in Figure 2. The current sensor is connected to the white cable in Figure 3, according to the schematics in Figure 4. In addition to the Raspberry and the

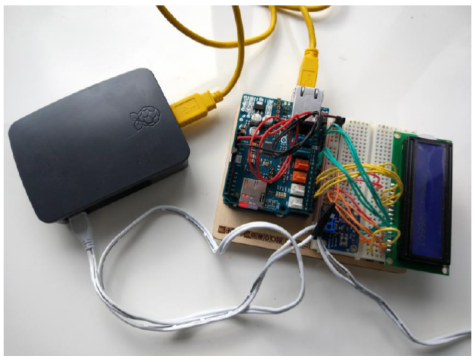


Fig. 3: Measurement system: the Arduino board is installed on a wood panel, along with the breadboard (on the right), according to the schematics in Figure 4. The Raspberry is on the left (inside the dark grey case).

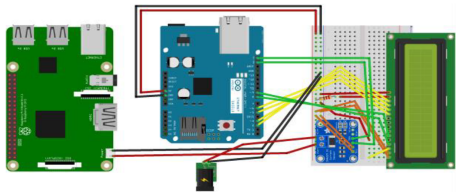


Fig. 4: Schematics: the Raspberry is on the left (green board), the Arduino in the middle (blue board), and the sensor current (along with a display for debugging purposes) is installed on the breadboard on the right (white colored).

current sensor, an Arduino board has been used to control the current sensor and to store the readings: in order to achieve that, a pre-loaded script is launched on the Arduino board at the bootstrapping. In fact, an USB cable (in yellow) is visible in Figure 3, connecting the Raspberry to the Arduino board, the latter used to perform monitoring tasks on different hardware so that energy measurements are not biased by such a task.

A. Platform in use

Our intent is in establishing if WebRTC can be used to stream a video when running on top of a resource-constrained platform, such as a Raspberry board. The requirements under consideration are: (*i*) reduced form factor, in order to have a small and lightweight platform ready for deploying; (*ii*) low power consumption, in order to have a platform that can run on batteries; (*iii*) the use of a COTS board for contained operating costs. For instance, consider an UAV streaming towards a fixed ground station by means of a Wi-Fi connection in a precision agriculture scenario [17], [18] or similar. Because of the limited payload that a small UAV can carry and the limited available power, a lightweight and energy-efficient platform is the preferred choice. A Raspberry is a resource-constrained platform with low overall power consumption (see Section III-B), yet capable of video encoding at a reasonable quality: because of this, it has been chosen as reference platform for this testbed.

TABLE I: Power consumption of the Raspberry PI 3 Model B in idle conditions (baseline).

CPU frequency	Average value	5th percentile	95th percentile
600 MHz	1,231.9 mW	1,185.7 mW	1,362.6 mW
1,200 MHz	1,435.7 mW	1,376.7 mW	1,637.5 mW

B. Power measurement

As anticipated before, the current sensor in Figure 2 has been used to monitor the power consumption. In details, current and voltage are sampled at a rate of 10Hz by the script running on the Arduino board. Before the testbed, which is described in Section IV, we estimated the power consumption of the Raspberry board in idle conditions, in order to assess a reference baseline. The HDMI connector has been turned off because not in use in our testbed, in order to minimize unnecessary power consumption. The results are reported in Table I, showing the power consumption [mW] when the CPU is forced to work at 600 or at 1,200 [MHz]. Along with the average instantaneous value, the 5th and the 95th percentile are shown with the intent of fully characterizing the power absorption in idle conditions.

C. Parameters under consideration

In this section, the parameters used as inputs are described, in order to characterize the different configurations we considered. Each tuple (set of parameter values) generates a different output, so that we collected a large amount of data: according to the metrics described in Section III-D, the most relevant results are then shown in Section IV.

Table II summarizes the considered parameters. We varied the video source rate in the range 170-1,700 [Kbps] with steps of 170. The maximum rate is imposed by the framework when using a fixed resolution of 640×480 pixels, as in our tests. Then, we considered the impact of using 1 or 2 concurrent video encoding threads, which affects the video quality on the one hand and the power consumption on the other hand. In addition, two CPU frequency values have been taken into account (as provided by the dynamic voltage frequency scaling driver of the Raspberry): 600 and 1,200 MHz, set in userspace during the manual tests, and set by the on-demand governor of the operating system (i.e., Raspbian) during the automatic tests. CPU frequency impacts on the video quality on the one hand and on the power consumption on the other hand. Eventually, we tested the impact of the so-called *cpu_used* parameter[1] of the VP8 video codec in use. The parameter provides a trade-off between encode quality and encode speed by varying the CPU utilization according to the formula $target\_cpu\_utilization = (100\,(16 - cpu\_used)/16)\%$. The

[1]Further details on it can be found in the WebM documentation, available at https://www.webmproject.org/docs/encoder-parameters/, in Section 2.

TABLE II: Parameters used in our testbed.

Name	Value / range of values
Source rate	170 - 1,700 Kbps
Number of threads	1 - 2
CPU frequency	600 - 1,200 MHz
cpu_used (WebM VP8)	10 - 15

lower the value of the parameter *cpu_used*, the larger the use of the CPU, thus the larger the available set of resources allocated to the encoding process; vice versa, the impact on the CPU is reduced. It is worth underlining here that this setting also depends on other running tasks: in order to remove any side effects due to CPU time allocated to unwanted tasks in our testbed, only WebRTC-generated threads are running in user-space during the tests.

D. Performance metrics

In this section, the performance metrics under consideration are described. Since our interest lies in the trade-off between video quality and power consumption, we selected the following five metrics of interest: (*i*) fps to characterize the fluidity of the video; (*ii*) the Peak signal-to-noise ratio (PSNR) to evaluate the video quality; (*iii*) the Structural SIMilarity (SSIM) index, as complement to PSNR, to enrich the evaluation of the video quality; (*iv*) the average instantaneous power consumption; (*v*) subjective video quality, evaluated by ten different users[2]. Furthermore, the so-called Quantization Parameters (QP) are logged: they are automatically set by the framework according to the other input parameters. QP range into $1-56$, with decreasing video quality as QP increase.

IV. Performance Evaluation

In this section, the results provided by our extensive testbed are reported, with the objective of characterizing the video quality and fluidity, and the power consumption of a resource-constrained platform, such as a Raspberry PI 3 Model B, when WebRTC runs on top of it. In our tests, we used the *crowd_run* video[3], which we converted to a resolution of 640x480 pixels and to 30 fps. The aspect ratio is 4:3 and the pixel format is YUV420. The video codec in use is VP8, the default video codec in WebRTC. The aforementioned video has been chosen because of the high dynamicity of the scene, captured from above, in order to mimic a video stream from an UAV, which is our reference test-case.

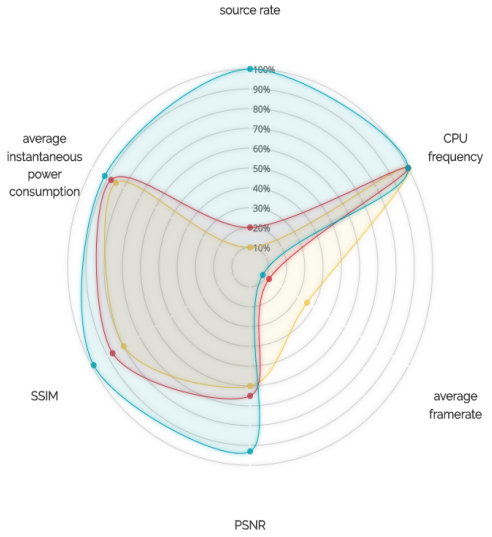
We now present the results of our testbed, in a graphical manner (see Figure 5) for an easier visual assessment, and in a more precise numerical manner (see Table III). Table III is divided into four subsections referring to the four radar charts in Figure 5: the first subsection is referred to the configuration with default settings (see Figure 5a); the other three subsections are related to manual settings, emphasizing alternatively *high quality*, *high framerate*, or *minimum power consumption* (see Figures 5b, 5c, and 5d). The four radar charts in Figure 5 contain multiple plots each: each plot refers to a given set of parameters. Thus, each chart shows how different outputs per configuration can be achieved by varying the value of the parameters. The numerical values of the parameters and of the related results in the radar charts can be found in Table III. In fact, each row of Table III refers to a plot in the radar charts, and the first column helps in matching numerical and graphical results. To ease the reader, Table III reports the settings in use on the left, and the results on the right. The second column of Table III reports the value of the parameter *cpu_used*, and the third column reports the average source rate[4] [kbps]. The fourth column reports the (forced) CPU frequency value, then QP and the number of used threads can be read in the fifth and sixth columns. The seventh, eighth, and ninth columns are related to the aforementioned quality metrics (fps, PSNR, and SSIM), while the tenth column reports the subjective evaluation, according to the following ranking: $(bad, poor, fair, good, excellent)$. Then, the last column show the absolute value of power consumption, and the so-called *additional power consumption (%)* that is calculated as the additional power spent to run the test w.r.t. the baseline values in Table I. Figure 5 shows normalized values in the scale $0 - 100\%$, where 100% means that that value is the highest one found during the whole testbed.

Figure 5a shows the setup provided by WebRTC in automatic mode (default settings): the main goal is the video quality, at the cost of power consumption and average framerate. It must be noted that no acceleration in hardware is available for WebRTC default video-codecs as we write: it is likely that the average framerate may benefit from such a feature. Empirically, we conclude that the default settings of WebRTC ignore power consumption as an optimization objective: such a choice can have a sensible impact in scenarios where energy efficiency cannot be neglected. Figure 5b shows the *cost* of obtaining the highest video quality in our testbed: PSNR and SSIM reach peak values, but at the cost of a very low framerate (1 fps). It means that such a setup can be useful only in scenarios in which such limitations can be accepted in favor of a higher resolution video stream. Figure 5b also shows that increasing source rates have a negligible effect on power consumption, so that the highest value can be selected with no additional cost. Figure 5c shows the configurations that provide a fluid framerate: in order to achieve such an objective, video quality must be partially sacrificed. In fact, very low source rates can be chosen and low/medium values of the quality metrics are visible. CPU must be set at the higher frequency value, thus causing medium to high power consumption. This setup should be used only when a fluid video is to be preferred to the overall video quality. Figure 5d shows the setup that provides the minimum power consumption: video quality can be privileged over framerate, or

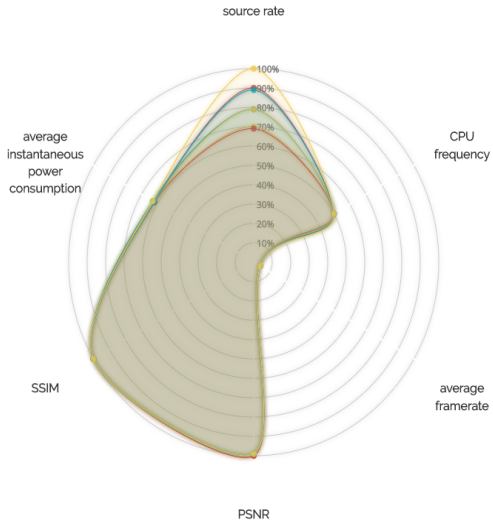
[2]Two example tests are available at: http://wnlab.isti.cnr.it/webrtc.

[3]The crowd_run video is available at https://media.xiph.org/video/derf.

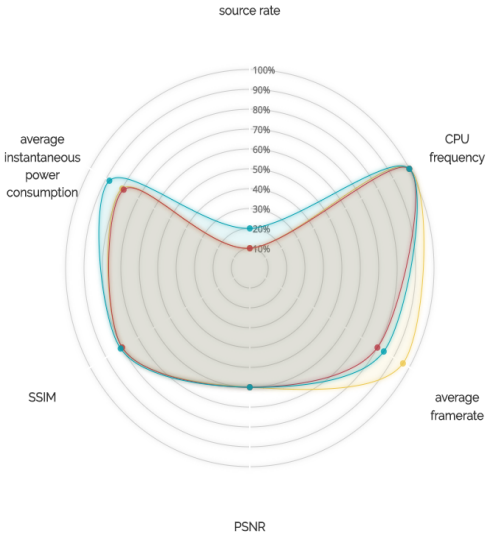
[4]Such an average value may be different from the chosen ones in Section III-C because the video encoder may slightly deviate from the input value because of performance reasons.



(a) default settings (automatic configuration)



(b) high quality (manual configuration)



(c) high framerate (manual configuration)

(d) minimum power consumption (manual configuration)

Fig. 5: Comparison between achievable performance level when using manual or automatic (default) configurations (refer to Table III for numerical results).

vice versa. Minimum power consumption implies that the CPU is set to the low frequency value (600 MHz), and a low source rate must be set. It is worth underlining here that no automatic configuration can provide such a low consumption level, which can prove of interest in very low-power scenarios.

To conclude, it is worth noting in Table III that PSNR and SSIM metrics of the *high quality* manual configuration (see Figure 5b) have higher values than those obtained by the default settings (see Figure 5a); furthermore, lower energy consumption is visible. Thus, manual settings can provide higher video quality and lower power consumption, but at the cost of a slightly lower framerate. In extreme cases, manual settings can reduce the energy consumption up to 30% w.r.t. default settings while providing comparable video quality and fluidity: for instance, compare lines 3 and 10 of Table III.

In this experimental work, we aimed at assessing if a reasonable power saving would be possible in low-power scenarios: this is confirmed by our extensive testbed here presented and discussed, showing that the power absorption can be reduced from 3.2-3.4 [W] to approximately 2.3 [W] with no appreciable differences in the delivered video quality and fluidity.

V. Conclusions

In this work, we tested the use of the WebRTC platform for real-time video streaming running on a resource-constrained platform, such as a Raspberry PI board. Size, weight and power requirements have been taken into account having in mind a reference use case that involves video-streaming from an UAV towards a fixed ground receiver node. The main

TABLE III: Settings and results of the considered configurations.

SETTINGS						RESULTS				
Reference scenario (#, ref., setting)	*cpu used*	*Source rate [Kbps]*	*CPU freq. [Mhz]*	*QP*	*# of threads*	*Avg fps*	*PSNR [dB]*	*SSIM*	*Subjective video quality*	*Avg power consumption [mW]*
1. Fig. 5a - auto	10	170	1,200	56	1	9	27.9	0.798	good	3,207 (+123%)
2. Fig. 5a - auto	10	340	1,200	48	1	3	30.3	0.864	good	3,320 (+131%)
3. Fig. 5a - auto	10	1,698	1,200	6	1	2	43.5	0.986	excellent	3,450 (+140%)
4. Fig. 5b - manual	13	1,178	600	3	1	1	46.2	0.991	excellent	2,325 (+89%)
5. Fig. 5b - manual	10	1,187	600	2	1	1	46.5	0.992	excellent	2,314 (+88%)
6. Fig. 5b - manual	12	1,342	600	2	1	1	46.1	0.991	excellent	2,357 (+91%)
7. Fig. 5b - manual	13	1,350	600	2	1	1	46.1	0.991	excellent	2,346 (+90%)
8. Fig. 5b - manual	12	1,513	600	2	1	1	46.2	0.991	excellent	2,337 (+90%)
9. Fig. 5b - manual	10	1,529	600	2	1	1	46.6	0.992	excellent	2,331 (+89%)
10. Fig. 5b - manual	14	1,700	600	2	1	1	46.2	0.991	excellent	2,375 (+93%)
11. Fig. 5c - manual	13	170	1,200	56	2	24	27.8	0.794	good	3,021 (+110%)
12. Fig. 5c - manual	15	170	1,200	56	1	20	28	0.797	fair	2,960 (+106%)
13. Fig. 5c - manual	14	340	1,200	56	2	21	28	0.799	good	3,303 (+130%)
14. Fig. 5d - manual	15	170	600	56	1	8	28	0.797	poor	2,263 (+84%)
15. Fig. 5d - manual	15	170	600	56	2	19	28	0.798	good	2,260 (+83%)
16. Fig. 5d - manual	12	340	600	44	1	2	31.5	0.889	fair	2,272 (+84%)

objective is the energy characterization of such a setup, in order to clarify the energetic impact of the WebRTC platform and the achievable video quality. To obtain that, we conducted an extensive testbed comparing results coming from default (automatic) and manual settings, investigating the relationship among available input parameters and showing that the default settings of the WebRTC framework privileges video quality over energy efficiency. Then, we have shown how comparable video quality can be delivered at a lower energetic cost (reduced up to 30% w.r.t. default settings), thus opening to scenario with stricter consumption constraints. In the near future, we plan to test the proposed setup on-board of an UAV to also characterize the impact of a realistic communication channel.

ACKNOWLEDGMENTS

This work has been partially supported by the Tuscany region in the framework of SCIADRO and MOSCARDO research projects (FAR-FAS 2014), and by the BIGDATA-GRAPES project (grant agreement N. 780751) funded by the EU Horizon 2020 research and innovation programme under Information and Communication Technologies.

REFERENCES

[1] 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.

[2] C. Holmberg, S. Hakansson, and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements," RFC 7478, Mar. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7478.txt

[3] M. Kritzler, M. Murr, and F. Michahelles, "Remotebob: Support of on-site workers via a telepresence remote expert system," in *International Conference on the Internet of Things*. ACM, 2016, pp. 7–14.

[4] L. Van Ma, J. Kim, S. Park, J. Kim, and J. Jang, "An efficient Session_Weight load balancing and scheduling methodology for high-quality telehealth care service based on WebRTC," *The Journal of Supercomputing*, vol. 72, no. 10, pp. 3909–3926, 2016.

[5] J. Gillis, P. Calyam, A. Bartels, M. Popescu, S. Barnes, J. Doty, D. Higbee, and S. Ahmad, "Panacea's glass: Mobile cloud framework for communication in mass casualty disaster triage," in *Mobile Cloud Computing, Services, and Engineering*. IEEE, 2015, pp. 128–134.

[6] Z. Li, "COAST: A Connected Open plAtform for Smart objecTs," in *Information and Communication Technologies for Disaster Management*. IEEE, 2015, pp. 166–172.

[7] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "Energy efficient multimedia streaming to mobile devices - a survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 579–597, 2014.

[8] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 179–198, 2013.

[9] R. W. Ahmad, A. Gani, S. H. A. Hamid, F. Xia, and M. Shiraz, "A Review on mobile application energy profiling: Taxonomy, state-of-the-art, and open research issues," *Journal of Network and Computer Applications*, vol. 58, pp. 42–59, 2015.

[10] G. Carullo, M. Tambasco, M. D. Mauro, and M. Longo, "A performance evaluation of WebRTC over LTE," in *Wireless On-demand Network Systems and Services*, Jan 2016, pp. 170–175.

[11] T. Kobayashi, H. Matsuoka, and S. Betsumiya, "Flying Communication Server in case of a Largescale Disaster," in *Computer Software and Applications Conference*, vol. 2. IEEE, 2016, pp. 571–576.

[12] D. Safadinho, J. Ramos, R. Ribeiro, R. Caetano, and A. Pereira, "UAV Multiplayer Platform for Real-Time Online Gaming," in *World Conference on Information Systems and Technologies*. Springer, 2017, pp. 577–585.

[13] G. Bekaroo and A. Santokhee, "Power consumption of the Raspberry Pi: A comparative analysis," in *Emerging Technologies and Innovative Business Practices for the Transformation of Societies*. IEEE, 2016, pp. 361–366.

[14] J. Sun, D. Wu, and S. Ci, "Battery capacity footprinting and optimization analysis for wireless multimedia communication," in *Global Telecommunications Conference*. IEEE, 2011, pp. 1–5.

[15] A. Heikkinen, T. Koskela, and M. Ylianttila, "Performance evaluation of distributed data delivery on mobile devices using WebRTC," in *Wireless Communications and Mobile Computing Conference*. IEEE, 2015, pp. 1036–1042.

[16] 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.

[17] M. Bacco, A. Berton, E. Ferro, C. Gennaro, A. Gotta, S. Matteoli, F. Paonessa, M. Ruggeri, G. Virone, and A. Zanella, "Smart Farming: Opportunities, Challenges and Technology Enablers," in *IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany)*, 8-9 May 2018, pp. 1–6.

[18] M. Bacco, A. Berton, A. Gotta, and L. Caviglione, "IEEE 802.15.4 Air-Ground UAV Communications in Smart Farming Scenarios," *IEEE Communications Letters*, pp. 1–4, July 2018.