Computational Offloading for the Industrial Internet of Things: A Performance Analysis

Sirine Bouhoula*, Marios Avgeris†, Aris Leivadeas*, Ioannis Lambadaris†
*Department of Software and IT Engineering, École de technologie supérieure (ÉTS), Montréal, Canada
†Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
sirinebouhoula@gmail.com, MariosAvgeris@cunet.carleton.ca, aris.leivadeas@etsmtl.ca, ioannis@sce.carleton.ca

***Abstract*—The challenge of minimizing mission response times and the energy consumption of computationally-constrained IoT devices, is becoming increasingly important in Industry 4.0 environments, where mission-critical tasks require real-time and latency-sensitive processing. In this context, the computational offloading paradigm becomes an attractive alternative for utilising remote, computationally capable resources placed at the network Edge, to execute demanding applications. In this paper, an analysis of the main options for task execution (on-device, Edge and Cloud) is performed, under different sensing and actuation scenarios. These results outline the need of an intelligent decision making mechanism to dynamically alternate between the available execution environments, as a response to the varying networking conditions.**

***Index Terms*—Industrial Internet of Things (IIoT), Industry 4.0, Edge Computing, Computational Offloading, Latency minimization, Energy/Power consumption minimization.**

I. Introduction

The revolution of Industry 4.0 has conceptualized a shift in the operations related to the production line and the distribution of products. In this context, emerging technologies like the Industrial Internet of Things (IIoT) and concepts pertaining the fields of Data Analytics and Artificial Intelligence (AI), are integrated with modern networking paradigms in the form of Edge and Cloud Computing frameworks, in order to smartly automate industrial processes [1]. Inevitably, this introduces a plethora of low-energy and low-capability collaborating devices to a smart factory ecosystem, which are regularly orchestrated to perform computationally intensive and energy-voracious tasks (e.g., AI-assisted sensing and actuation). Consequently, this increased interconnectivity combined with the frequent demand in real-time complex algorithm execution, calls for advanced latency and energy consumption minimization on the field [2].

This is where computational offloading, i.e., the act of transferring resource-intensive computational tasks to separate external devices in the network proximity, comes in handy. In the 5G and beyond era, the increased availability of Edge and Cloud networking enables the low-latency access to computationally capable resources, paving the way to the wide adoption and realization of computational offloading frameworks [3]. In detail, Edge Computing establishes a hierarchy of intelligent processing elements between devices and gateways throughout the Edge-Cloud continuum, to tackle the IIoT shortcomings in a scalable, context-aware and interoperable manner.

Although Edge Computing constitutes a particularly prominent way of dealing with latency-sensitive and energy consuming industrial tasks, solely utilising remote computational resources is not enough; potentially, various disturbing phenomena can occur during the transmission and processing of the tasks, including unexpected increases in propagation and processing delay, as well as downtime [4]. Naturally, this probability of service quality degradation increases as the factory environment becomes more dynamic (i.e., mobile IIoT devices/robots) and that is why IIoT devices are still involved in the task processing loop; to account for periods where network access is unavailable or unreliable. Consequently, a major challenge from a network and resource optimization point of view, is to combine the local and remote resources in an efficient way and guarantee service availability at all times.

A. Related Work & Motivation

In this section, we provide a comprehensive study of some prominent works focused on addressing the Computation Offloading problem in similar settings. In [3], the authors present a comprehensive study on the computational offloading problem, emphasizing on the mathematical, AI and control theory-related solutions in Edge and Cloud Computing environments. The open challenges and future directions list the development of fault tolerant solutions which account for network disturbances, among others.

Trying to improve the Edge-Cloud continuum processing efficiency of the tasks from devices with limited computation and communication capabilities, the authors in [5] propose a hierarchical computing scheme with an offloading pipeline strategy. This framework jointly decides on the offloading platform and the power allocation, using successive convex approximation to transform the emerging non-convex problem. In a similar manner, Hao et al. [6] propose a scheme called intelligent task offloading (iTaskOffloading) for an Edge-Cloud collaborative system. Here, a cognitive engine determines the personalized offloading strategy, based on the users' various requirements on the latency.

This work was supported in part by the CHIST-ERA-2018-DRUID-NET project "Edge Computing Resource Allocation for Dynamic Networks"

Many works have also tackled the lack of end device capabilities in industrial environments through the use of computational offloading. The authors in [1] propose a computation offloading mechanism for robotic applications. In particular, they realize an IoT-enabled localization and path planning framework and verify the expected gains of computation offloading by utilizing a real edge computing setting. Bebortta et al. [7] develop a computation offloading mechanism for time-critical IIoT tasks. Their framework efficiently handles industrial workloads by modeling them as stochastic processes to observe the number of data packets denied service, due to the finite number of busy MEC server, and act accordingly. Finally, in [8], Peng et al. formulate the joint computation offloading and resource allocation problem as a multi-objective optimization problem and develop an Edge-Cloud collaborative intelligent optimization method. They mainly focus on minimizing energy and time consumption.

B. Contributions & Outline

In order to enable the implementation of effective, intelligent, computational offloading decision making mechanisms, as the aforementioned ones, a comprehensive study and analysis on the specific problem at hand has to be performed. In a smart factory environment, this translates to i) defining the Key Performance Indicators (KPIs) that the computational offloading framework needs to achieve (e.g., specific Quality of Service - QoS measurements and device energy consumption), ii) designing a use case that accurately reflects the real-world industrial environment that the framework will have to operate in, iii) assembling an architecture from hardware and software components that are already present or can easily be implemented in a smart factory setting and iv) stressing the framework under various dynamic scenarios in order to detect the bottlenecks in its operation. In this work, our aim is to exactly conduct such a structured analysis, which eventually outlines the need of an intelligent industrial computational offloading mechanism. The key contribution of this paper is threefold:

- An Edge-Cloud infrastructure is considered where the IIoT and Edge Computing devices are placed on site (i.e. in the smart factory premises) and the Cloud datacenter is placed in a remote location. To perform a realistic computational offloading analysis, a hardware/software architecture is assembled, consisting of mobile IIoT devices, sensors and generic compute nodes easily found in an industrial setting.
- A use case is designed to accurately capture the range of conditions under which a computational offloading framework will have to operate in. First, two generic industrial services-applications are developed for the sole purpose of this use case: i) one based on data-analytics and ii) a machine learning image recognition one. Then, the KPIs of their execution are decided, i.e., total service execution time and device energy consumption.
- Finally, the setting is implemented in the premises of the ETS university and a thorough real-world evaluation is performed. To this purpose, three characteristic scenarios are examined, which cover the possible execution platforms: i) on-device local execution, ii) offloading at the Edge devices in close proximity and iii) offloading at the remote Cloud through the public internet. The results pinpoint the benefits as well as the shortcomings of each execution mode and indicate the need for a smart offloading decision making mechanism, able to respond to the dynamic conditions of a factory floor setting.

The rest of the paper is organized as follows. In Section II the architecture overview and the individual components are presented, together with the developed applications' specifications. Section III discusses the use case design. Section IV presents the performance evaluation under the three distinct execution scenarios and Section V concludes the paper.

II. Architecture Overview

The main scenario addressed in this work involves a mobile IIoT device (i.e., a robot) traversing the factory floor, sensing the environment (i.e., monitoring the operating conditions, using temperature, humidity, gas and optical stimulation) and actuating accordingly (i.e., raising alerts). This functionality is a key component to realizing autonomous monitoring in Industry 4.0 use cases which automate part of the industrial workplace safety protocols enforcement. However, a common problem in such a scenario is that the dynamic nature of the device's mobility and the uncertainty in the network, computational and energy resources conditions, can potentially hamper the timely completion of such a critical mission. Thus, the importance of having multiple alternative platforms for executing the individual parts of this process is evident. Following, we provide a comprehensive description of the hardware as well as the software components that comprise the proposed architecture.

A. Hardware Components

1) SunFounder PiCar-X[1]*:* The PiCar-X is a development kit for smart robots on which a Raspberry Pi works as the control center. The PiCar-X is energy efficient, easy to control and assemble, as well as can accommodate multiple sensors that can also be expanded manually. It is also equipped with a camera that is a requirement for this project and wheels so the robot is able to move around the factory floor.

2) Raspberry Pi 4 Model B[2]*:* This credit card-sized single board computer is a fully programmable PC that runs the open-source Robot Operating System (ROS). Its board consists of Video Core IV graphics processing unit (GPU), ARM v8-compatible quad-core cortex-A72, 4GB of RAM, which allow the adequate execution of machine learning-based image recognition and simple data analysis tasks. One powerful feature of the Raspberry Pi is the row of General Purpose Input/Output (GPIO) pins along the edge of the board. The Raspberry Pi standard 40-pin GPIO header is leveraged to install the required sensors.

[1]https://www.sunfounder.com/products/picar-x

[2]https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

3) Sensors:

- Raspberry Pi Camera: The Raspberry Pi Camera Rev 1.3 Board plugs directly into the CSI connector on the Raspberry Pi and is used to collect still shots and videos of the devices surroundings for further processing.
- DHT11 sensor: The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It consists of a humidity sensor and a thermistor to measure the surrounding air and convert them to digital signals on the data pin.
- MQ-X sensor: The MQ-X series sensors are used in gas leakage detecting equipment in the industry. They are suitable for rapidly detecting liquefied petroleum gas (LPG), iso-butane and propane particles.

4) Generic Edge Compute Node: To emulate the Edge Computing infrastructure of a smart factory, a generic compute node with a 4-core Intel Core i5 processor, Intel Integrated Graphics GPU and 8GB of RAM is connected to a generic 802.11ac access point.

5) AWS-based Cloud Compute Node: Acting as the Cloud Compute Node, a c4.2xlarge instance is reserved on Amazon Elastic Cloud Compute (EC2)[3]. This instance comes equipped with 8 vCPUs and 16GB of RAM and it is reachable through the public Internet.

B. Software Components

1) Image Recognition Service (IRS): An image recognition component was implemented from scratch for the purposes of this work. This service receives optical stimulation from the Raspberry Pi Camera (i.e., still shots and/or video streams) and processes it in order to detect anomalies in the robot's environment. In an offline phase, an image recognition model is trained accordingly, by using the OpenCV library and the MobileNetSSD object detection model on a large series of data. In its online phase, this service i) processes the frames received from the camera, ii) decides on the potential severity of an incident and iii) raises alerts. The combination of these processes is a source of highly compute-intensive tasks.

2) Data Analysis Service (DAS): Another software component implemented from scratch for the purposes of this work. This data analysis-based service receives environmental measurements from the temperature, humidity and gas sensors in order to assist in the detection of anomalies in the factory. Here, in the offline phase, a forecasting model that can detect the onset of disastrous incidents (e.g., fire outbreaks) is trained, using the Pandas and the NeuralProphet libraries and a large series of data. In its online phase, this service again processes the measurements and raises alerts based on the outcome of the model. This component produces mildly intensive computational tasks. A basic decision making procedure follows the execution of the IRS and DAS components to decide on the severity of the situation.

[3]https://aws.amazon.com/ec2/

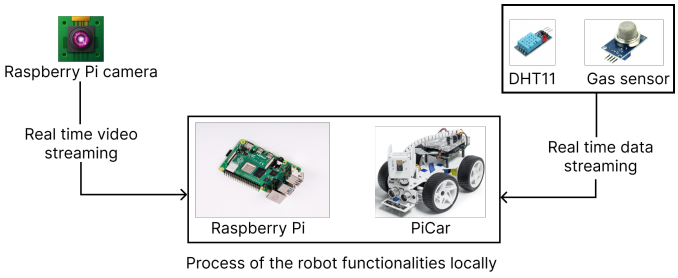

Fig. 1. Data Flow in the On-device Execution scenario.

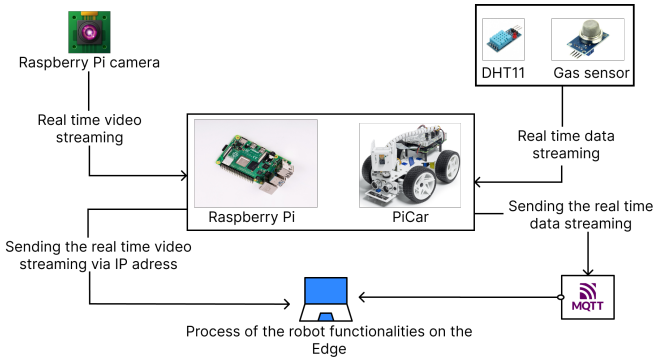

Fig. 2. Data Flow in the Edge Execution scenario.

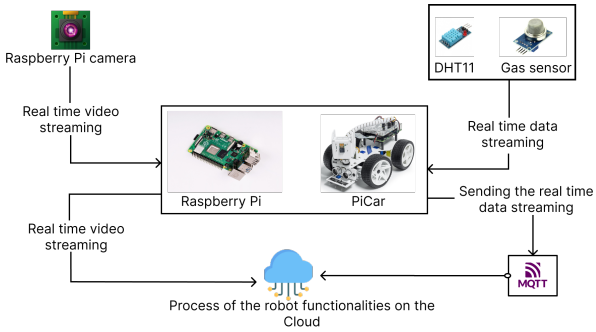

Fig. 3. Data Flow in the Cloud Execution scenario.

III. Use Case

The designed use case aims to employ varying numbers of IIoT-enabled mobile robots throughout a smart factory floor, with the mission to monitor the environment and detect potential emergencies as early as possible. Subsequently, the robots raise alerts, supporting the initiation of a series of actions towards the handling of the risk, in an effort to limit the engagement of human resources. In this approach, the following KPIs are considered: rapid identification of an emergency situation (i.e., total mission execution time) and prolonged mission duration (i.e., device energy/power consumption).

The goal of this work is to conduct a performance analysis and reveal the offloading opportunities during similar industrial critical missions, based on the proposed KPIs. To this purpose, three scenarios are described and examined,

capturing the main strategies that can be employed regarding the platform level of the computational tasks execution:

A. On-device Execution

In this first task execution scenario, the computation is performed locally, on the robot. As illustrated in Fig. 1, real-time measurements of environmental temperature and humidity values, as well as values indicating the presence of smoke and flammable gases, are streamed to the Raspberry Pi from the on-boarded sensors. Additionally, video captured from the camera is streamed to the Pi and the local processing of both streams by their respective services is used to decide whether an emergency situation is detected in the factory floor.

B. Offloading Execution at the Edge

For the second task execution scenario, the two services are modified accordingly and deployed at the Edge compute node; their modifications concern the way the services acquire data: a REST API is attached to the IRS and an MQTT-based solution is used for the DAS. In detail, as shown in Fig. 2, the measurements and the video frames are offloaded from the Raspberry Pi to the Edge computing node through a LAN 802.11ac connection.

C. Offloading Execution at the Cloud

In the final execution scenario (Fig. 3), the two services are placed on the AWS Cloud compute node, using the same REST API and MQTT communication methods respectively. The only difference is that the measurements and the video frames are now offloaded from the Raspberry Pi to the Cloud computing node, through the public Internet.

IV. Numerical Results

In this final section, the setup of the experiments and the findings of the experiment execution are presented, along with the respective evaluation and comparison of the obtained results. As already mentioned, this evaluation acts as a valuable performance analysis which will pinpoint the offloading opportunities and discuss the trade-offs between the available computational execution platforms for an industrial setting.

A. Experimental Setup

The basic hardware setup used in all three experiment scenarios introduced in Section III, consists of the IIoT robot, a wireless access point, one Edge compute node and one Cloud compute node, as described in Section II. The Edge node is placed in the ETS University's premises, while AWS at US-East (Ohio), is selected to host the Cloud compute node in order to achieve an unbiased simulation of the real-world Cloud communication conditions. To precisely monitor and evaluate the real-time energy consumption of the robot, the MakerHawk UM25C USB Tester[4] is utilized, which provides voltage (V), current (A) and power consumption (W) measurements. Regarding the simulation configuration, each experiment scenario spans for 1 minute. Printed images of indoor fires where used to trigger the IRS, while a small handheld lighter was used to trigger the DAS. For raising the alerts after processing the video streams and data measurements, a simple e-mailing SMTP-based solution was developed in Python.

The results depicted in Fig. 4, 5, 6 and 7 are used to evaluate the efficiency of the three scenarios. Specifically, Fig. 4 depicts the mean power consumption on the robot for the different computational execution mode. For comparison purposes, the idle consumption of the robot is illustrated as well. Fig. 5 contains the resource utilization in terms of CPU and RAM, for each platform and scenario, while Fig. 6 and 7 show the total mission execution time achieved (until the alert is raised), in each situation and the impact that the mobility of the robot has on the offloading procedure (in frame rate achieved). It is noted here that the measurements presented are averaged over 10 executions for each experiment.

B. Robot Power Consumption

Measuring the power consumption is only meaningful for the battery-powered mobile robot, as we assume that the Edge and Cloud compute nodes have constant DC power supply. The optimization of the robot's energy and power consumption has an immediate effect on prolonging the mission duration, a critical KPI for industrial processes.

In Fig. 4, the average power consumption is presented for the three different execution modes: idle (as a reference point), offloading and executing locally on the device. Two things should be noted here; first, that only the IRS was examined in this family of experiments, as it is the most compute-intensive of the two services. Second, that power consumption-wise, we assumed no distinction between offloading to the Edge and the Cloud, as both procedures initiate from the same Access Point. For obtaining the power consumption, we constantly acquired voltage and current measurements from the robot's power supply and applied them on Ohm's formula. As expected, executing the IRS locally is proven to be a strenuous task for the robot, as we observe up to 75% greater power consumption than the idle state. This potentially leads to rapid battery

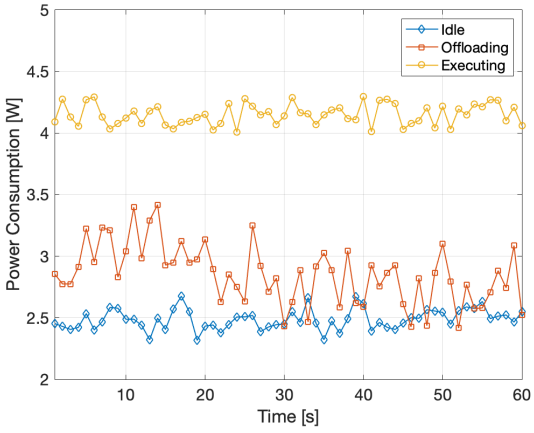

Fig. 4. Average Robot Power Consumption.

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

depletion. On the other hand, offloading to either the Edge or the Cloud, increases the power consumption only by 25% when compared to the idle state, prolonging in this way the monitoring mission for up to 2 hours more, compared to the on-device execution (when the robot operates on a $5000mAh$ battery).

C. Platform Resource Utilization

Regarding the resource utilization, the IRS and the DAS components where simultaneously executed in each platform. Once again, as expected, the robot struggles at the execution of such compute-intensive tasks and the results come as a confirmation; as shown in Fig. 5, the Edge and the Cloud are both way more efficient, with the Edge performing up to 58% better CPU-wise and 87% better memory-wise. Executing the services on the Cloud yields an even smaller resource footprint, with 80% less CPU and 93% less memory resources utilized on average, respectively.

These results highlight the need for a hierarchical offloading framework, which will allow the vital software components for the robot's mission (e.g., navigation and motor operation) to be unobstructively executed on-device, while the service components are selectively placed on the Edge and Cloud nodes. Another benefit from executing the industrial services remotely, is that multiple IIoT agents that are required to execute the same services can share the Edge and Cloud deployments, leading to increased resource utilization throughout the smart factory.

D. Mission Execution Time & Accuracy

To measure the execution time for the whole mission, both the IRS and the DAS components where deployed to each platform and triggered at the same time. Fig. 6 depicts the results of the first part of this family of experiments; the total execution time is broken down to two procedures: i) the streaming analysis procedure, where both the video and the data streams are analysed for outlier events and ii) the decision making procedure, where the results of the analysis

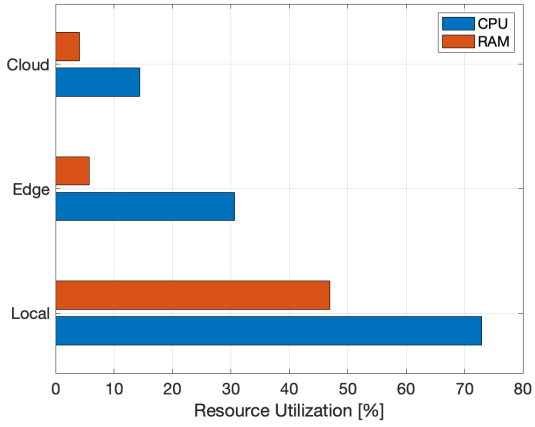

Fig. 5. Average Platform CPU and RAM Utilization per Execution Scenario.

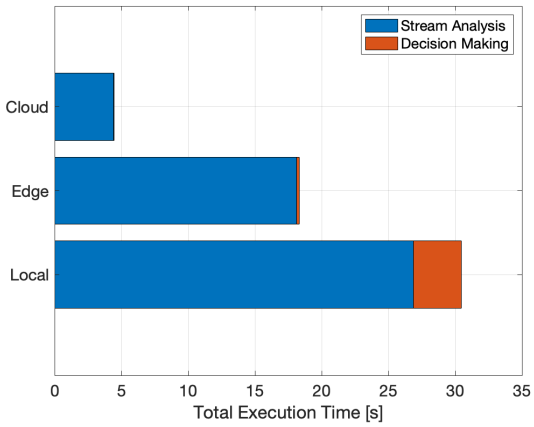

Fig. 6. Average Mission Execution Time per Execution Scenario.

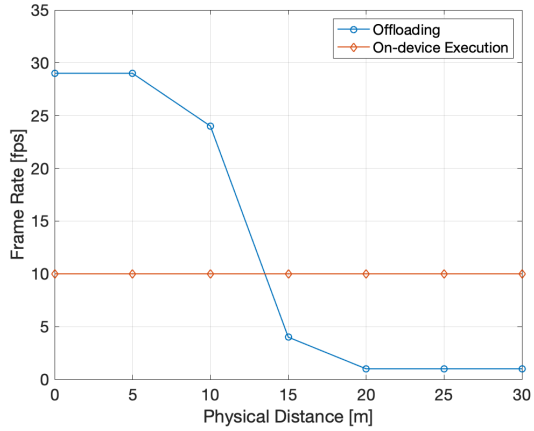

Fig. 7. Average Frame Rate vs. Physical Distance from the Access Point.

are fed to the Decision Making component and an alert is e-mailed if an emergency is detected. Offloading the executions excels in this area as well, with the Edge and the Cloud nodes responding 45% and 87% faster, respectively, to the emergency than the robot. This decrease in emergency response times is of paramount importance, especially when considering the amount of damage a fire can make to an indoor environment in just a few seconds.

In the second part of this family of experiments, the mobility effect on the mission accuracy is studied. To this purpose, the IRS component was examined, as video streaming has a direct way of measuring service degradation through frame rate drop. Fig. 7 shows that, in our setting, moving beyond a $14m$ radius from the Access Point, has devastating results on the number of frames that are actually offloaded to the Edge/Cloud ($< 4fps$) and subsequently on the emergency detection accuracy, making the on-device execution a preferable alternative. This comes to validate that exclusively utilising remote computational resources is not a panacea.

Overall, offloading the compute-intensive tasks at the Edge or the Cloud, balances the resource utilization in the infras-

tructure, prolongs mission duration and decreases the mission response times while increasing the accuracy, under normal network conditions. Quantity-wise, the Cloud seems to overpower the Edge; however, some of the Edge's benefits could not be easily plotted in figures and assessed. The average propagation delay in the Cloud was measured 4 to 5 times bigger compared to the Edge, making the Edge a preferable option for missions that require ultra low latency communication. Additionally, the issue of data privacy is a critical and difficult to visualise one and this is an area where an Edge computing infrastructure excels. Contrary to the Cloud, sensitive data like images of private premises travel solely in "in-house" networks and nodes without the risk of being exposed. Furthermore, Edge infrastructures could be owned by and deployed at the factory, making them a much more cost efficient solution compared to third-party Clouds.

V. Conclusion

This paper presented a comprehensive study and performance analysis on the smart factory-based computational offloading problem. The goal of this analysis is to provide useful insight and motivate the development and wide adoption of intelligent computational offloading frameworks. To this end, first an industrial framework's objectives where defined, then a representative use case was designed and finally a Edge-Cloud architecture was implemented. The evaluation showed that when the network conditions are suitable, offloading the computational tasks to either the Edge or the Cloud is preferable than on-device execution. However, the importance of a framework that orchestrates the offloading of the tasks is evident, when network disturbances, mobility, privacy and cost minimization are considered. Future plans include the development of such a framework that will be able to dynamically distribute the computational execution between the device, the Edge and the Cloud infrastructure.

References

[1] D. Spatharakis, M. Avgeris, N. Athanasopoulos, D. Dechouniotis, and S. Papavassiliou, "A switching offloading mechanism for path planning and localization in robotic applications," in *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. IEEE, 2020, pp. 77–84.

[2] M. Avgeris, D. Spatharakis, D. Dechouniotis, A. Leivadeas, V. Karyotis, and S. Papavassiliou, "ENERDGE: Distributed energy-aware resource allocation at the edge," *Sensors*, vol. 22, no. 2, p. 660, 2022.

[3] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, A. Leivadeas, N. Athanasopoulos, N. Mitton, and S. Papavassiliou, "Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions," *Computer Networks*, vol. 195, p. 108177, 2021.

[4] D. Dechouniotis, N. Athanasopoulos, A. Leivadeas, N. Mitton, R. Jungers, and S. Papavassiliou, "Edge computing resource allocation for dynamic networks: The druid-net vision and perspective," *Sensors*, vol. 20, no. 8, 2020.

[5] C. Kai, H. Zhou, Y. Yi, and W. Huang, "Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 2, pp. 624–634, 2020.

[6] Y. Hao, Y. Jiang, T. Chen, D. Cao, and M. Chen, "iTaskOffloading: intelligent task offloading for a cloud-edge collaborative system," *IEEE Network*, vol. 33, no. 5, pp. 82–88, 2019.

[7] S. Bebortta, D. Senapati, C. R. Panigrahi, and B. Pati, "An adaptive performance modeling framework for QoS-aware offloading in MEC-based IIoT systems," *IEEE Internet of Things Journal*, 2021.

[8] K. Peng, H. Huang, B. Zhao, A. Jolfaei, X. Xu, and M. Bilal, "Intelligent Computation Offloading and Resource Allocation in IIoT with End-Edge-Cloud Computing Using NSGA-III," *IEEE Transactions on Network Science and Engineering*, 2022.