Services Orchestration at the Edge and in the Cloud on Energy-Aware Precision Beekeeping Systems

Hugo Hadjur, Laurent Lefèvre, Doreid Ammar

Services Orchestration at the Edge and in the Cloud on Energy-Aware Precision Beekeeping Systems

Hugo Hadjur
aivancity School for Technology, Business & Society
Paris-Cachan, France
Inria, Univ Lyon, EnsL, UCBL, CNRS, LIP
Lyon, France
hadjur@aivancity.ai

Doreid Ammar
aivancity School for Technology, Business & Society
Paris-Cachan, France
Inria, Univ Lyon, EnsL, UCBL, CNRS, LIP
Lyon, France
ammar@aivancity.ai

Laurent Lefèvre
Inria, Univ Lyon, EnsL, UCBL, CNRS, LIP
Lyon, France
laurent.lefevre@inria.fr

***Abstract*—Precision Beekeeping is a subfield of Precision Agriculture. It collects data on colonies of bees using sensors to assist beekeepers and preserve bees. The existing solutions include collecting voluminous data, such as images and sound, from up close or inside a beehive. The collection, transfer, and processing of such data are energy-intensive processes. However, most Precision Beekeeping systems work under a limited energy budget. Thus, such systems must take this constraint into account. This article focuses on the placement of energy-aware Precision Beekeeping services at the edge and in the cloud. We deploy a set of smart beehives in order to conduct experiments and collect real data. Then, we simulate large-scale systems with embedded services to add more realism to the experimental study. Our results show that parameters like the number of smart beehives, the frequency of data collection, and the characteristics of the cloud server impact the placement of these services to enhance energy efficiency.**

***Index Terms*—Precision Beekeeping, Internet of Things, Energy Efficiency, Edge-Cloud Computing, Artificial Intelligence**

I. Introduction

Pollination is essential for the balance of flora and human agriculture. It relies on several actors, mainly insects like butterflies, wasps, flies, and bees. The latter includes different species. Among them, the *apis mellifera* (western honey bee) is the most common domesticated species.

Precision Beekeeping (PB) [1] aims at optimizing the production of honey and preserving bees with data mainly collected thanks to sensors placed around or inside the beehive, making it a **connected beehive**. In this paper, we will also refer to the term **smart beehive** when some intelligent service is included in a connected beehive in addition to the collection of data.

In PB, Internet of Things (IoT) is used for gathering, processing and transmitting data and for sending alerts to beekeepers. Developing such systems next to beehives and bees results in several constraints like placing sensors in such ways that intrusion toward bees is minimized. The most important constraint, tackled in this article, is energy consumption and efficiency of PB systems as connected beehive must be autonomous without the availability of a power outlet.

This paper aims at exploring the optimal placement of services inside a smart beehive between the edge devices (the smart beehives) and the cloud servers. It also optimizes the framework of large-scale networks of smart beehives thanks to the simulation of the model of our deployed system.

The paper is organized as follows: Section II focuses on the state of the art of the different tackled subjects. Section III describes the smart beehive system and the collected data. Section IV analyzes the energy of the system without intelligent services. Section V focuses on one intelligent service and the choice between placing it at the edge or in the cloud. Section VI shows a simulation of the architecture of the presented system at large-scale. Finally, Section VII concludes the paper and describes future works.

II. Related Works

A connected beehive relies on its microcontroller (Raspberry Pi, Arduino), sensors, network, and some energy to power the system. The literature has many approaches and there is not a universal solution in terms of architecture and choice of hardware [2].

There are different types of data collected like weight [3], temperature [4], bee count [5], and image data [6]. In this paper, we focus on services relying on sound data in an energy-aware PB system.

Sound & Vibrations: The sound or vibrations is often collected thanks to sensors directly placed inside the hive, either on the inner wooden structure for microphones [7] or on frames for accelerometers [8]. In rare cases, such sensors are placed right at the entrance to record takeoffs and landings [9].

Energy: The addition of a sensor is one among many sources of energy consumption. The energy of a smart beehive depends on its microcontroller, whether it has a sleep state, the duration of it being switched on, and the intensity at which the code uses processors. In PB, it is rare that researchers focus on the consumption of energy of their system, but there are occurrences of analyses. The work described in [10]

mentions the daily consumption of the system (2 watt hours). [11] and [12] share the current consumption divided by nodes in their system. The authors of [13] develop the previously cited analysis and present the daily energy budget calculations for each node and for phase (sense, send, sleep). [14] opts for a solar-powered battery charge analysis and checks for differences in terms of sampling rates, solar panel orientation, and sampling power. The abnormal case of a power outage is mentioned in [15], and the autonomy of the system only powered by an unplugged battery is measured (75 hours). The same runtime duration is calculated in [3], and this temperature, humidity, and the weight-measuring system can last around 12 days autonomously. A data reduction method is presented in [16], and raw data transfer versus reduced data is energetically compared. The current drawn over one iteration of measures is displayed in [17] to spot the consumption of each subtask. The same approach is presented in [18], and different variations of parameters within each task are then measured and compared.

III. System Architecture

This section aims to describe our autonomously running solution for collecting bee-related data capable of monitoring its energy consumption. Energy data is used as a basis for the orchestration of energy-aware services.

Hardware: The system, as shown in Figure 1, relies on two Raspberry Pi, which are single-board mini-computers.

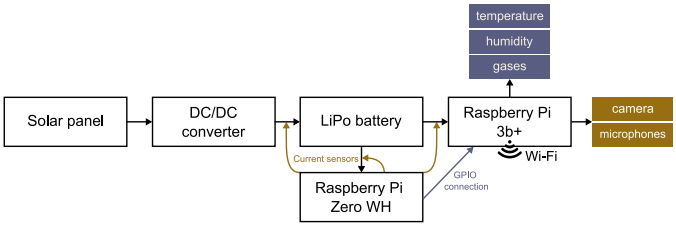


Fig. 1: Architecture of the deployed system showing essential hardware, energy node, and network

First, the Raspberry Pi 3b+ is the most complete of the two as it includes four USB 2.0 ports, a port for a Raspberry Pi camera, 2.4 gigahertz and 5 gigahertz IEEE 802.11 wireless LAN and Bluetooth connectivity, as well as a quad-core 1.4 gigahertz 64-bit processor and 1 gigabytes of RAM. In comparison, the second one, a Raspberry Pi Zero WH, is a lightweight version of its sister model with one micro USB port, a camera connector, the same Wi-Fi and Bluetooth connectivity, a 1 gigahertz single-core CPU, and 512 megabytes of RAM.

On top of each Raspberry Pi, a Grove Base Hat is installed to wire sensors. For the Raspberry Pi Zero WH, its hat sensors are three identical ± 5 amperes DC/AC current sensors. Each of these sensors is linked to an electric wire: the two Raspberry Pi's power supplies and the wire that goes from the solar panel to the battery. For the Raspberry Pi 3b+, a temperature and humidity sensor (SHT31) are used. In addition to hats' sensors, the Raspberry Pi 3b+ collects sound through three USB microphones (range: 20 hertz to 16 kilohertz) and images from a Raspberry Pi camera module 2 The temperature and humidity sensor, the gas sensor, and the microphones are placed inside the beehive on the queen excluder. The camera is placed at one end of the entrance of the beehive and faces the other end to take pictures of the whole bees' takeoff and landing area.

Location: Five smart beehives are currently deployed. Two are located to the South of Paris in Cachan, France, and the others are in Lyon, France. All of them are positioned on a university building's roof and are in an urban area, but there are parks bordering both campuses.

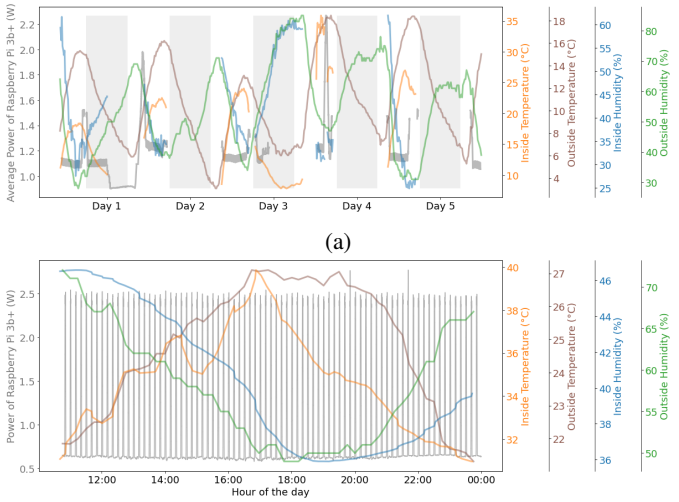
Energy: Each system relies on a solar panel, current converter, and battery for its energy. The solar panel is a monocrystalline 30 watts panel. The battery is a 20 000 mAh power bank. Between the two, the current is converted to 5 volts (the adequate voltage for the battery) thanks to a DC/DC Step-Down 5 volts/3 amperes converter.

IV. Energy consumption of the deployed system

In this section, we aim to calibrate the frequency at which the Raspberry Pi 3b+ wakes up and performs a set of tasks at regular intervals.

Data collection routines: The Raspberry Pi Zero, which records the current, is always switched on. At regular intervals, it sends a signal through GPIO to wake up the other Raspberry Pi, the beehive data recorder. Once the Raspberry Pi 3b+ wakes up, it collects data from all its sensors, including three 10-second audio samples collected at the same time and five 800×600 pixels images spread over five seconds, and it transfers the data through Wi-Fi to a remote data storage cloud server. In addition, at regular intervals, the Raspberry Pi Zero WH transfers the latest consumption data. Sizes of samples and arrangement of steps for the data collection and transfer are chosen thanks to an analysis presented in [18]. To complete the data, the weather data from both locations is collected at regular intervals.

Figure 2a displays the activity of the Raspberry Pi 3b+ of one beehive together with the in-hive temperature and humidity, and the meteorological data at the same moment. At this moment, the colony of bees was yet to be introduced inside the beehive, hence the abnormally low inside temperature. The data is shown over one full week to emphasize energy consumption dynamics. In this graph, the gray background shows time periods after sunset, whereas the white background shows daytime periods. This graph also shows moments when the system is not running due to the lack of light at night. We suppose that the low luminosity takes the solar panel's output voltage to uncontrolled values, thus affecting the batteries and the electronics. Figure 2b highlights the intervals at which the Raspberry Pi 3b+ wakes up. Its energy consumption is shown alongside the in-hive temperature and humidity, and the meteorological data. The spikes of consumption correspond to the moments between turning on and shutting down the Raspberry Pi 3b+. In this graph, the frequency of the wake-up GPIO signal is set to 10 minutes.



(a)

(b)

Fig. 2: (a) The in-hive temperature and humidity, the external temperature and humidity, and the averaged Raspberry Pi 3b+ consumption during one week of Spring 2022 in Cachan (France). (b) The in-hive temperature and humidity, the external temperature and humidity, and the Raspberry Pi 3b+ consumption showing consecutive wake-ups during 13 hours of one day in Summer 2022 in Cachan (France).

Comparing wake-up frequencies: We focus on the frequency at which the Raspberry Pi 3b+'s data collection routine is performed. We choose to monitor the energy of the Raspberry Pi 3b+ at different wake-up frequencies: every 5, 10, 15, 30, 60, and 120 minutes. For example, Figure 2b shows the frequency of 10 minutes, where two consecutive wake-up calls from the Raspberry Pi Zero are spaced by 10 minutes. The frequency parameter's values are realistic and could be implemented in the system, depending on the chosen beehive-related services. For example, for a service tracking the temperature of the beehive, collecting data every 60 or 120 minutes suffices. On the other hand, in a period of collection of large datasets, collecting data every 5 minutes becomes reasonable. The experiment is designed so that each setting is active for at least nine hours in a row in order to collect substantial energy data. This duration is chosen because it corresponds to daylight time, where our energy node is more stable and able to provide constant power to the electronics. Doing so, the collected energy is a sum of the active power (when the Raspberry Pi boots, performs tasks, and shuts down) and the sleep power of the sleep state, which consumes non-null resources: the cost of being able to receive wake-up calls.

The data collected during this experiment covers 319 routines of the system. All routines are identical. They follow the steps presented at the beginning of this section. It is then possible to compute average costs for one routine. On average, the Raspberry Pi 3b+ is turned on, performs its tasks, and shuts down in 1 minute and 29 seconds, with an average power of 2.14 watts. This gives an average energy cost of 190.1 joules from the boot to the shutdown. The standard deviation for the lengths of routines is 3.5 seconds. This relatively high variability can be explained by the variance of the duration of the data transfer correlated to the unstable network throughput. The standard deviation for the average power of routines is 0.009 watts, which shows stability. The average consumed power for on/off cycles at different wake-up frequencies is presented in Figure 3. At the highest frequency, when the system wakes up every 5 minutes, the average power values reach their maximum: 1.19 watts on average. When the duration between two consecutive wake-ups increases, the average power decreases and converges toward a value close to 0.62 watts, which is the consumption of the Raspberry Pi 3b+ in a sleep state. This shows that the sleep state is significant when considering long-term consumption.

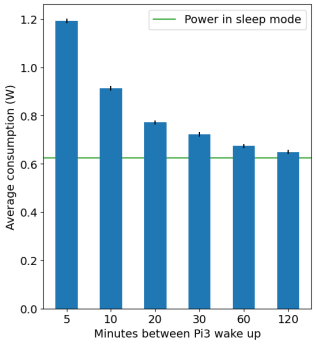


Fig. 3: Average power used for several durations between two consecutive wake-ups of the smart beehive's system

V. Orchestration of Smart Services

As observed in 2a, there are moments where the edge is not able to fully capture and process the data because of limited available energy. Since collecting data has to be performed at the edge, gaining flexibility in terms of costs can be achieved during the processing step of the data thanks to a cloud server.

In this section, we consider adding smart services on top of data collection and transfer systems. We investigate which option is more energy efficient between performing the services' tasks at the edge or in the cloud. We consider two main scenarios:

- Edge scenario: the edge system wakes up, collects data, executes the service's tasks at the edge, optionally stores the data locally, sends results directly to the user (the beekeeper's phone, for example), and shuts down. This scenario does not require a cloud server.
- Edge+cloud scenario: the edge system wakes up, collects data, sends it to a remote cloud server, and shuts down. Meanwhile, a cloud server receives the data and performs the service's tasks. The unfolding of these steps can be found in Figure 4. In the previous section, we introduced a routine that collects and transfers the collected data to a remote cloud server, so the energy consumption data collected in that section will compute the costs of the edge device in this scenario.

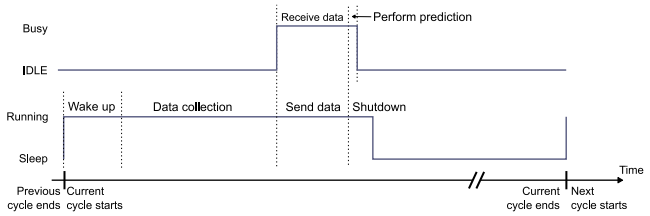


Fig. 4: Task diagram for one cycle of data collection, processing, and transfer for edge+cloud scenario

Our selected cloud server has an Intel i7-8700K processor and an Nvidia RTX2070 GPU. This allows to train the deep learning models considered in this paper in few minutes. Compared to the average bare-metal cloud server, this is a less energy-intensive option. Our selected edge device is the previously introduced Raspberry Pi 3b+, capable of performing numerous tasks: pollen detection, counting bees, and swarm prediction, among others. Most of these tasks include AI models. In this paper, we focus on the energy consumed by classification models for the presence of the queen inside a beehive, thanks to sound data. In practice, in addition to the previously introduced data collection routine, an audio sample is used to predict the presence of the queen bee. For this paper's connected beehive architecture, the prediction is performed thanks to the sample from the microphone placed in the middle part of the queen excluder grid. The selected task of queen detection has already been studied with deep learning (convolutional neural network, CNN) and classical machine learning (support vector machine, SVM) methods [19]. The latter is an adapted choice because it allows acceptable performances for reasonably small datasets (less than tens of thousands of samples). We choose to follow the same methodology by selecting one classical machine learning option and one deep learning option.

For the training of both models, 1647 audio samples labeled with the presence of the queen are used. The selected training features are mel-scaled spectrogram features computed from 10-second audio recordings of bees sampled at 22050 hertz. For the spectrogram function, the length of the fast-Fourier transform window is 2048, the number of audio samples between adjacent short-time Fourier transform columns is 512, and the number of generated Mel-bands is 128. Vector features are passed as it is for the training phase of the SVM model, whereas they are converted into images for the CNN model. The chosen architecture for the deep learning model is ResNet18. It is trained on a dedicated cloud server, through 4 epochs, with a learning rate of 0.001. Although pre-trained, thanks to regular real-world pictures, the model is able to achieve state-of-the-art performances when trained with spectrogram images. For the classical machine learning alternative, the SVM classifier is set with a radial basis function kernel, a regularization parameter of 20, and a kernel coefficient of 10^{-5}. The training phase of CNN models has a significant energy cost, but it is a less frequent task than the use of the trained models. Therefore, in this paper, we only focus on the energy consumption of the usage of the trained models.

These models and parameters reproduced state-of-the-art performances for the queen detection task. Figure 5 shows the measured energy for the Raspberry Pi 3b+ to perform predictions and the accuracy of the trained models as functions of images' length. The 100 by 100 pixels resolution is the smallest that shows converged results, with a classification accuracy of 99%. Larger images as input of the CNN model show similar performances, but the energy cost when executing the model on the Raspberry Pi is greater than 100 by 100 pixels images. This cost increases as a quadratic function of the number of pixels in the images, which is the time complexity of the use of the trained AI model. Therefore, using 100 by 100 pixels images for the CNN model is the optimal choice when considering the accuracy and the energy cost of the model's execution at the edge.

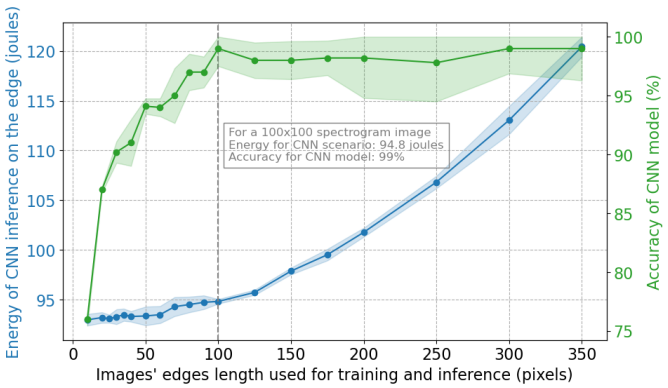


Fig. 5: Energy of CNN model execution and its accuracy on the test set as a function of training set's images' size.

Thanks to this focus on the energy consumption of one service, the upcoming paragraphs will generalize the results on complete edge and edge+cloud scenarios that include this service. We observe the edge+cloud scenario in real conditions. First, the Raspberry Pi 3b+ and the cloud server are in sleep and idle mode, respectively. The edge device is then turned on and starts collecting data. Then, the edge transfers the collected data to the cloud, and the edge starts shutting down as the server executes the service's tasks. Finally, the server switches to idle mode. The energy consumption graph of the Raspberry Pi during this routine shows a spike during the data transfer step. This step is more energy-intensive than the data collection because the network components have a larger energy cost than the sensors.

We then focus on energy consumption measures performed during both scenarios and for both queen detection AI models. Each of the two models (SVM and CNN) is tested in the edge and edge+cloud scenarios. Thanks to the measures of the energy consumption of the edge and the cloud for the four options, we aim to select the best scenario for each service. Table I and Table II show the duration and the energy consumption for all cases when 5-minute cycles are

considered. For each scenario, we show the collected time and energy consumption step by step, in chronological order. Since the queen detection model's execution in the cloud takes less time than the shutdown of the Raspberry Pi 3b+, the shutdown is divided into two lines in the edge+cloud scenario, and their sum adds up to the 9.9 seconds shown in edge scenarios. The choice of the model does not generate a significant difference: only 1.2 joules of difference (0.3% of the edge's total cost) for the energy cost of the Raspberry Pi 3b+ in the edge scenarios and 61.7 joules (0.4% of the server's total cost) for the energy cost of the cloud server in the edge+cloud scenarios. In the edge+cloud scenarios, the service is not executed at the edge, so the energy cost of the edge is not influenced by the choice of the service. Then, the global energy cost of the IoT system is more significant for a edge+cloud scenario than an edge scenario because of the presence of a server that must be turned on and available at all times to receive data. However, this cost allows the edge to perform fewer tasks, hence a smaller energy cost for the edge+cloud scenario at the edge: there is a reduction of 12.1% and 12.4% of consumed energy for the SVM and CNN model, respectively. This can be useful in cases where the edge's energy must be saved. For example, when the energy available in the cloud is not a limiting factor and the energy produced by the solar panel at the edge is limited. A large-scale simulation with many smart beehives matched with one server could demonstrate the viability of the edge+cloud scenario.

Edge Task	Energy of Edge (joules)	Time (seconds)
Scenario: Edge (SVM)		
Sleep	111.6	178.5
Wake up & Data collection	131.8	64.0
Queen detection model (SVM)	98.9	46.1
Send results	3.0	1.5
Shutdown	21.0	9.9
Total	**366.3 joules**	**300 seconds**
Scenario: Edge (CNN)		
Sleep	116.9	187.0
Wake up & Data collection	131.8	64.0
Queen detection model (CNN)	94.8	37.6
Send results	3.0	1.5
Shutdown	21.0	9.9
Total	**367.5 joules**	**300 seconds**

TABLE I: Tasks of edge with time and energy consumed per 5-minute cycle for queen detection edge scenarios

VI. Simulation at large scale

Currently, five smart beehives are deployed and equipped with the system described in this paper. It is probable that, most likely, PB researchers and even beekeepers who want to equip their hives require embedding such a system at a larger scale or in a whole apiary. We can also suppose an organization of several beekeepers putting their hardware in one unique network of edge and cloud computing.

This section aims at putting the currently deployed systems and their cloud servers on a large scale. A simulation method is proposed to determine whether a large network of smart beehives is energetically viable.

A. Description of the simulation model

The simulation model has three main components: the client, the server, and the allocator. In the practical case of this paper, one client refers to one smart beehive and one server refers to one cloud server. However, this method can encapsulate any IoT device linked to a server.

- Client: its tasks are to acquire and optionally process and transfer data. It is initialized thanks to the power consumption in the sleep state, a series of actions (active state) and their respective time and power consumption, and the time between two consecutive wake-ups.
- Server: its tasks are to receive data from clients and process them. It also has a series of actions, time, and power consumption. It supports a maximum amount of clients allowed in parallel. Each server allows their clients to start communication at specific times so that every client within a group has to start their communication with the server at the same time as the other clients of the group, all synchronized in time thanks to a specific hardware (GPS, for example). We will refer to these specific time windows as time slots. One server can allow multiple time slots. The shorter the time window for the server's tasks, the greater the number of time slots. For example, in a 5-minute cycle, given a data transfer and a model execution's duration of 1 minute, a server can allow 5-time slots.
- Allocator: its main task is to allocate several clients to servers. It takes a list of clients, creates servers based on their features and the features of the server type, allocates every client to one server, and links them to a wake-up time slot. Currently, it has one filling policy: filling a server with clients by filling one slot up to its maximum after another.

Simulation will be performed in theoretical perfect situation as well as realistic cases involving losses. The results will be described in the following paragraphs.

B. Results in an ideal theoretical situation

Figure 6 shows the simulation results for the number of servers required and the energy consumed per client and in total, respectively. This simulation recreates the previously introduced edge+cloud scenario and edge scenario for a network of clients and servers with clients ranging from 10 to 400. The tasks' duration and power are initialized thanks to the measures described in Section IV and Section V, considering that one client is a smart beehive. The simulation also supposes a client's wake-up frequency of 5 minutes and a server that can allow a maximum of 10 clients per time slot. Since the cost per client increases exponentially when considering less than 10 clients, the lowest number of clients is set at this value for readability reasons. Figure 6 shows that the energy consumption per client of edge devices (shown in red) is equal to one client's energy consumption, 322 joules. It is due to the

Edge Task	Energy of Edge (joules)	Cloud Server Task	Energy of Cloud Server (joules)	Time (seconds)
		Scenario: Edge+Cloud (SVM)		
Sleep	131.9	Idle	9415	211.1
Wake up & Data collection	131.8	Idle	2854	64.0
Send audio	37.3	Receive audio	1032	15.0
Shutdown	0.2	Queen detection model (SVM)	6.3	0.1
Shutdown	20.8	Idle	437	9.8
Total	**322.0 joules**		**13744.3 joules**	**300 seconds**
		Scenario: Edge+Cloud (CNN)		
Sleep	131.9	Idle	9415	211.1
Wake up & Data collection	131.8	Idle	2854	64.0
Send audio	37.3	Receive audio	1032	15.0
Shutdown	2.1	Queen detection model (CNN)	108	1.0
Shutdown	18.9	Idle	397	8.9
Total	**322.0 joules**		**13806 joules**	**300 seconds**

TABLE II: Tasks of edge and cloud with time and energy consumed per 5-minute cycle for queen detection edge+cloud scenarios

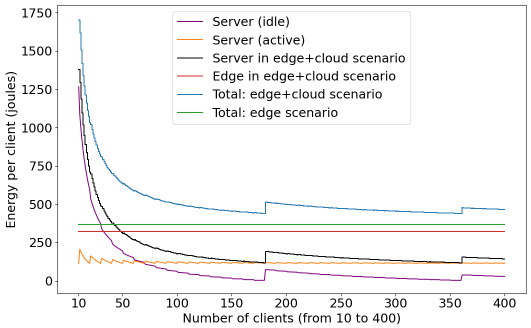
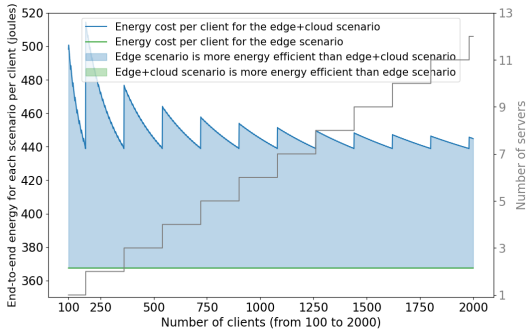
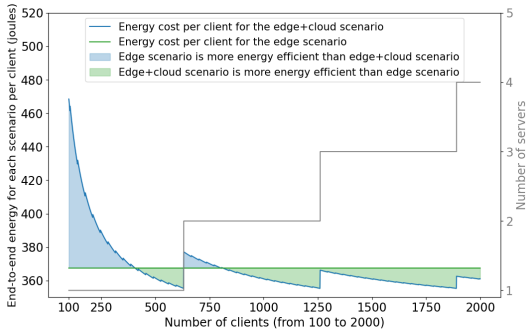


Fig. 6: Simulation of energy consumption for all entities in a client-server scenario with no loss. Number of clients allowed in parallel in time slots: 10.



(a) Number of clients allowed in parallel in time slots: 10.



(b) Number of clients allowed in parallel in time slots: 35.

Fig. 7: Comparison of end-to-end energy per client for the two scenarios with different server settings

fact that the tasks performed at one edge do not depend on the server's parameters or the number of edge devices in the network. The server's overall energy consumption per client (shown in black) converges towards 116 joules, the energy for which all the server's time slots are full. The smart beehive case brings the overall best cost per beehive (shown in blue) to 116 + 322 = 438 joules. This can be described as the theoretical best cost per client. Also, it is 16% greater than the overall cost in the edge scenario. However, it is fair to compare the worth of the energy produced at the edge by the solar panel and the energy available in the cloud. It is more likely that cloud servers are powered by nationally produced electricity, so this part is not (or at least less) under pressure. On the other hand, the energy produced at the edge, thanks to the solar panel and stored inside the battery, is less reliable, and the amount of energy available is much lower than in the cloud. Therefore, one joule of energy used at the edge is not equivalent to one joule of energy used on the cloud in terms of availability and resilience. One could argue that the 16% increase of energy in the edge+cloud scenario does not make this scenario less efficient than the edge scenario.

This difference between the two scenarios decreases as the number of clients allowed in parallel per time slot increases. Figure 7 focuses on the simulation data for the end-to-end total energy for both scenarios between 100 and 2000 clients and highlights the differences when the maximum clients allowed in parallel changes. This range for the number of clients is different from the simulation results shown previously because increasing the number of clients allowed in parallel increases the total number of clients allowed per server. Figure 7a shows the same setting as Figure 6: 10 clients at maximum in parallel. Figure 7b shows the simulation results when this parameter takes 35 as values. In those three figures, the blue area corresponds to settings where the average cost of one

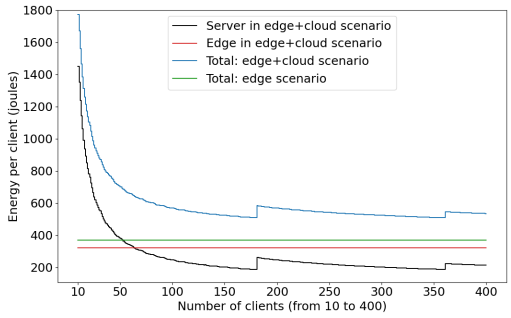
client is lower in the edge scenario than in the edge+cloud scenario. The green area corresponds to parameters making the edge+cloud scenario more energy-efficient. We have observed that 26 clients are the tipping point when the edge+cloud scenario can become more energy efficient when used efficiently. Figure 7b shows that when 35 clients are allowed in parallel, 406 clients are needed to make the edge+cloud scenario more energy-efficient than the edge scenario. With these settings, the maximum difference in favor of the edge+cloud scenario is 12.5 joules at 630 clients, just before the need for an additional server when the first server is fully used. This graph also shows that from 803 clients, the edge+cloud scenario is more energy-efficient than the edge scenario and remains this way for any higher number of clients considered. This is because the more servers needed, the fuller they are on average because they are all full but one.

C. Results with losses for clients and cloud servers

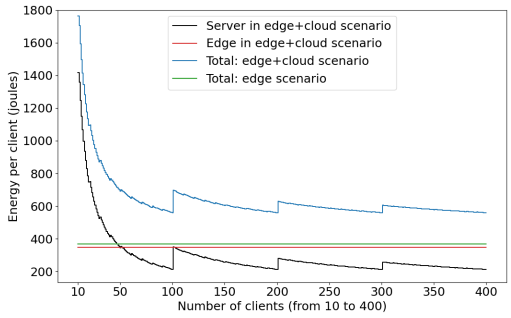
In this paragraph, the following parameters are added to depict real-life losses and costs:

- Loss scenario A: A penalty when a server's time slot starts saturating with its number of clients. The limit at which the penalty starts is set at 5 clients below the maximum allowed per slot. Each additional client penalizes the whole energy slots by 10%.
- Loss scenario B: A time penalty of 1.5 extra second per client for clients' data transfer time. Since the model considers that clients of the one-time slot are synchronized, they send their data simultaneously.
- Loss scenario C: A loss of clients at every wake-up time: we use a random Gaussian distribution (mean: 10% of the total number of clients; standard deviation: 2) to draw the number of lost clients.

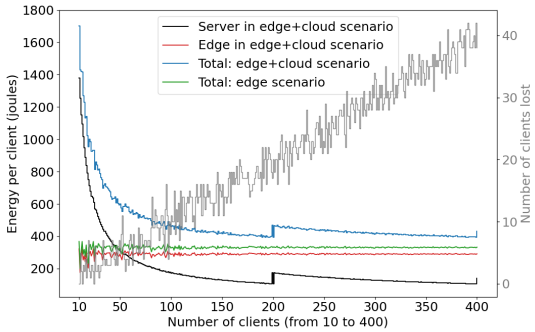
The above-mentioned values for the loss parameters are chosen thanks to the understanding gained thanks to the data collection period. Figure 8 shows the simulation results of the energy consumed per client when losses are added to the models. Figure 8a shows that the first considered loss increases the energy consumed by the server. In this setting, the cost of the server converges towards 186 joules (to be compared with the 116 joules, which represent the lowest server cost for no loss model). Figure 8b highlights that extra transfer duration causes the number of time slots per 5 minutes to decrease, thus decreasing the maximum number of clients per server. More servers are needed for an equivalent number of clients compared to the no-loss scenario (for 350 clients: 4 servers when duration penalty is applied versus 2 servers in the no-loss case). Since there are fewer clients per server, the cost of servers per client increases. In this case, its minimal value is 212 joules. In Figure 8c, we display the random Gaussian number of clients lost alongside the energy consumption metrics. The x-axis displays the initial number of clients, so the energy consumed appears to be less than the default case without any loss. However, this is a case where lost clients cause other losses: all the services and the data would be missing. Figure 8d generalized the three



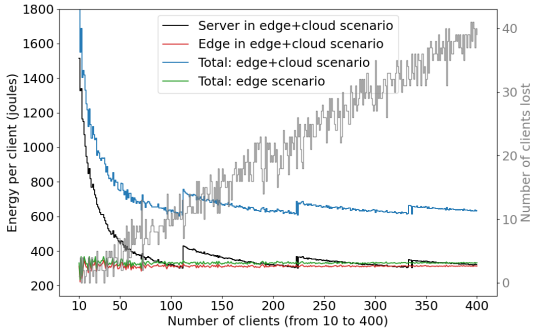
(a) Loss scenario A



(b) Loss scenario B



(c) Loss scenario C



(d) All three types of losses combined.

Fig. 8: Simulation of energy consumption in a client-server scenario with several types of loss. The number of clients allowed in parallel in time slots: 10.

previous cases. On this figure, there are some abnormal rises around 225 clients and 340 clients. Those spikes are due to the varying number of randomly lost clients making the required number of servers decrease, although the initial number of clients increases.

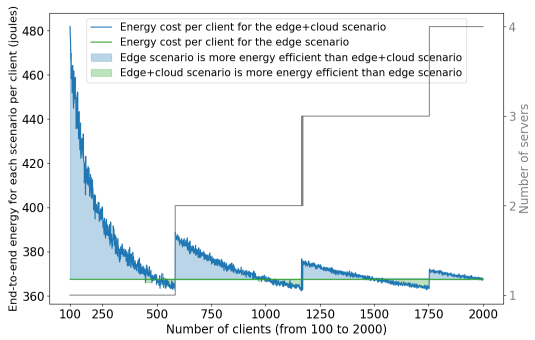


Fig. 9: Comparison of end-to-end energy per client for the two scenarios with different server settings and with loss. Number of clients allowed in parallel in time slots: 35.

Finally, Figure 9 shows the comparison of the two scenarios with all three types of loss considered, and diplays more realistic simulated energy consumption values. This set of parameters shows that the limit of clients per time slot of 35 becomes of little bit worse than its equivalent without loss, but still has some intervals where the edge+cloud scenario is more energy-efficient than the edge scenario. Thanks to this simulation, it is possible to scale a network of clients and servers, given costs and penalties of energy consumption or time, which depend on the number of clients. For example, it is safe to assign three servers when the number of clients is between 1600 and 1750, and the edge+cloud scenario will be more energy-efficient than the edge scenario.

VII. Conclusion

We present in this article an advancement in the orchestration of services in IoT systems at the edge and in the cloud with the use of direct continuous measures and large-scale simulation. We develop an end-to-end energy-aware Precision Beekeeping edge system and introduce a client-server energy simulation model. We analyze the consumption costs of our system and their variation when the frequency of the main routine varies. The system and the model show that the design of systems using one edge should aim towards removing the use of a cloud server which requires much more energy than the edge. In a case with many clients, adding a cloud server is more energy-efficient than the edge-only counterpart when the cloud server is used close to its maximal capacity. Future work will refine our simulation model, add new scenarios and refine the numerical estimations of losses. We will also investigate the use of machine learning and deep learning models to improve the simulation model and build connected beehives' intelligence to tune its parameters and choose between a set of scenarios.

References

[1] A. Zacepins, E. Stalidzans, and J. Meitalovs, "Application of information technologies in precision apiculture," in *Proceedings of the 13th International Conference on Precision Agriculture (ICPA 2012)*, Indianapolis, IN, USA, 07 2012.

[2] H. Hadjur, D. Ammar, and L. Lefèvre, "Toward an intelligent and efficient beehive: A survey of precision beekeeping systems and services," *Computers and Electronics in Agriculture*, vol. 192, p. 106604, 2022.

[3] N. Anuar, M. A. Md Yunus, M. Baharuddin, S. Sahlan, A. Abid, M. Ramli, M. Amin, and Z. Lotpi, "Iot platform for precision stingless bee farming," in *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 06 2019, pp. 225–229.

[4] M. Giammarini, E. Concettoni, C. Zazzarini, N. Orlandini, M. Albanesi, and C. Cristalli, "Beehive lab project - sensorized hive for bee colonies life study," in *2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 10 2015.

[5] W. Chen, C. Wang, J. Jiang, and E. Yang, "Development of a monitoring system for honeybee activities," in *2015 9th International Conference on Sensing Technology (ICST)*, 2015, pp. 745–750.

[6] K. Bjerge, C. E. Frigaard, P. H. Mikkelsen, T. H. Nielsen, M. Misbih, and P. Kryger, "A computer vision system to monitor the infestation level of varroa destructor in a honeybee colony," *Computers and Electronics in Agriculture*, vol. 164, p. 104898, 2019.

[7] S. Cecchi, A. Terenzi, S. Orcioni, P. Riolo, S. Ruschioni, and N. Isidoro, "A preliminary study of sounds emitted by honey bees in a beehive," in *Audio Engineering Society Convention 144*, May 2018.

[8] M.-T. Ramsey, M. Bencsik, M. Newton, M. Reyes, M. Pioz, D. Crauser, N. Simon-Delso, and Y. Le Conte, "The prediction of swarming in honeybee colonies using vibrational spectra," *Scientific Reports*, vol. 10, 2020.

[9] V. Kulyukin, S. Mukherjee, and P. Amlathe, "Toward audio beehive monitoring: Deep learning vs. standard machine learning in classifying beehive audio samples," *Applied Sciences*, vol. 8, p. 1573, 09 2018.

[10] A. Cunha, J. Rose, J. Prior, H. Aumann, N. Emanetoglu, and F. Drummond, "A novel non-invasive radar to monitor honey bee colony health," *Computers and Electronics in Agriculture*, vol. 170, p. 105241, 03 2020.

[11] F. E. Murphy, M. Magno, P. M. Whelan, J. O'Halloran, and E. M. Popovici, "b+wsn: Smart beehive with preliminary decision tree analysis for agriculture and honey bee health monitoring," *Computers and Electronics in Agriculture*, vol. 124, pp. 211–219, 2016.

[12] W. Hong, B. Xu, X. Chi, X. Cui, Y. Yan, and T. Li, "Long-term and extensive monitoring for bee colonies based on internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7148–7155, 2020.

[13] F. Edwards-Murphy, M. Magno, L. O'Leary, K. Troy, P. Whelan, and E. Popovici, "Big brother for bees (3b) - energy neutral platform for remote monitoring of beehive imagery and sound," in *6th International Workshop on Advances in Sensors and Interfaces (IWASI)*, 2015.

[14] F. Edwards-Murphy, E. Popovici, P. Whelan, and M. Magno, "Development of an heterogeneous wireless sensor network for instrumentation and analysis of beehives," *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, vol. 2015, 2015.

[15] S. Gil-Lebrero, F. Quiles-Latorre, M. Ortiz-López, V. Sánchez-Ruiz, V. Gàmiz-López, and J. Luna-Rodr'iguez, "Honey bee colonies remote monitoring system," *Sensors*, vol. 17, no. 1, 2017.

[16] D. Kridi, C. Carvalho, and D. Gomes, "Application of wireless sensor networks for beehive monitoring and in-hive thermal patterns detection," *Computers and Electronics in Agriculture*, vol. 127, 2016.

[17] A. Zacepins, A. Kviesis, V. Komasilovs, and F. Muhammad, "Monitoring system for remote bee colony state detection," *Baltic Journal of Modern Computing*, vol. 8, 09 2020.

[18] H. Hadjur, D. Ammar, and L. Lefèvre, "Analysis of energy consumption in a precision beekeeping system," in *Proceedings of the 10th International Conference on the Internet of Things*, ser. IoT '20. New York, NY, USA: Association for Computing Machinery, 2020.

[19] I. Nolasco and E. Benetos, "To bee or not to bee: Investigating machine learning approaches for beehive sound recognition," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 133–137.