Analyzing Per-Application Energy Consumption in a Multi-Application Computing Continuum

Saeedeh Baneshi, Anuj Pathania, Benny Akesson, Andy Pimentel
University of Amsterdam
Amsterdam, The Netherlands
{s.baneshi,a.pathania,a.d.pimentel,k.b.akesson}@uva.nl

Ana-Lucia Varbanescu
University of Twente
Enschede, The Netherlands
a.l.varbanescu@utwente.nl

***Abstract*—In today's digital society, diverse computing devices—from edge devices to data centers—support various applications, each with specific performance and energy characteristics. Analyzing application energy consumption is crucial for improving energy efficiency, optimizing resources, and reducing environmental impact.**

However, while comprehensive energy measurements are feasible for specific configurations, they are impractical for assessing diverse application mappings. Still, stakeholders such as cloud providers, developers, users, and researchers need insights into application-level energy behavior for informed decision-making.

In this work, we propose a fine-grained simulation approach for analyzing application energy behavior in edge/cloud environments. We implemented our approach as an enhanced version of the iFogSim framework. We demonstrate its effectiveness by evaluating different multi-application scenarios and configurations for a video surveillance application. Our approach facilitates the fast evaluation of different scenarios and deployment strategies, providing insights that can contribute to more energy-efficient edge/cloud computing systems and digital services.

***Index Terms*—Computing Continuum, Application Energy Consumption, Application Mapping, Scenario analysis, Edge Computing, Simulation**

I. Introduction

Most IoT applications require high Quality of Service (QoS) to ensure timely data processing. For example, intelligent surveillance cameras require quick analysis for object detection and tracking with minimal latency. Cloud computing handles these applications well, but the growing number of connected cameras and the necessary high-throughput data traffic raises concerns about network congestion and high energy consumption [1]–[3]. Edge computing has emerged as a viable alternative, enabling the use of processing resources closer to the data source to reduce data traffic and congestion.

The allocation of application modules across various devices, spanning the entire continuum from edge nodes to the cloud, significantly impacts energy consumption and latency [4], [5]. To optimize device selection and placement policies, a detailed analysis of application energy consumption behavior across various mappings is essential. Therefore, *fine-grained* characterizing the end-to-end energy consumption of applications is critical. This requires integrating both *computation and networking energy consumption* and addressing the complexities of multi-application environments to achieve realistic end-to-end energy estimations.

To facilitate such end-to-end analysis, we present an enhanced version of the iFogSim framework, improved to combine networking and computing energy consumption and to support energy accounting for multi-application simulations. To demonstrate our improvements, we use a surveillance application as a basic case-study, and simulate various multi-application mapping scenarios by running different instances and/or different workloads of this application simultaneously. Our simulation results illustrate the non-negligible impact of communication energy consumption, as well as the significant differences between mappings, especially in the context of multi-applications. Such information is essential for stakeholders, such as application developers, system operators, and integrators to make informed decisions regarding their systems architecture, resource mapping, and scheduling policies.

The three main contributions of this work are as follows:

1) We extend iFogSim's energy model to also consider the energy consumption of communication. To this end, we develop: (a) a time-based model to estimate the energy consumption of NICs (Network Interface Cards) in end-user devices, enabling support for space-shared communication in the network, and (b) a flow-based model for shared devices to report the energy consumption of sending tasks.
2) We improve iFogSim's reporting to collect finer-grain data, thus enabling collecting virtual machine energy consumption, an essential improvement for analysis of multi-application scenarios.
3) We illustrate how to characterize and compare the energy consumption - including both computing and communication - of two co-located applications using various mapping scenarios. We also showcase the impact of changing application workload.

Open Source Contributions: The code for our enhanced iFogSim is publicly available, under MIT license, at https://github.com/saeedehbaneshi/IFogSim.

The remainder of this article is organized as follows. Section III discusses related work. Section II introduces the iFogSim framework and the surveillance case study. Section IV outlines our enhanced version of iFogSim, detailing the modeling of network energy consumption and the granularity improvements per virtual machine and tasks. Next, in

Section V, we describe the experimental scenarios and analyse the results. Finally, in Section VI, we conclude the paper and highlight future research directions.

II. Background

In this section, we introduce iFogSim, a simulation framework for IoT and Fog computing, and present the modeling of our case-study surveillance application - within iFogSim.

A. iFogSim

Today's computing continuum infrastructures have diverse resources that support distributed applications. The energy consumption and Quality of Service (QoS) of these applications vary with placement strategies and device characteristics. Measuring energy consumption in such environments is challenging and costly [6], [7]. Therefore, simulation is a practical alternative for estimating energy consumption in the computing continuum.

There are several simulators for modeling computing continuum and edge-cloud architectures. One of these well-known simulators is iFogSim [8], a framework built on top of CloudSim [9], a popular framework for modeling cloud computing environments. iFogSim simulates components across the computing continuum, from edge devices to cloud data centers [8], allowing researchers to evaluate the effectiveness of various resource management techniques. It is designed to measure performance metrics like latency, network usage, energy consumption, and cost.

iFogSim supports IoT sensors, actuators, fog devices, and cloud data centers interconnected via network links. Using the Sense-Process-Actuate model, sensors send data to IoT networks, fog devices process it, and insights are translated into actions for actuators [8].

iFogSim models applications as independent modules with specific computational requirements. It simulates physical components like sensors, fog devices, and actuators with attributes defining their capabilities and interactions. Applications are represented as directed graphs with vertices as processing modules and edges as data dependencies. iFogSim uses virtual machines (VMs) for module execution and includes management components, such as ModuleMapping and Controller, to facilitate resource allocation and deployment within the fog environment, enabling detailed simulation of various fog computing scenarios [7], [10].

B. Surveillance Case Study

For our experiments, we consider a surveillance application within the iFogSim framework. The application processes data captured by the smart cameras. Initially, the cameras analyze the data, which is then processed by "continuum" equipment based on placement strategy. Each module handles and forwards tasks, known as *tuples*, for further processing [4], [8], [11].

The functions of the application modules are as follows:
Motion Detector: embedded in smart cameras, it continuously analyzes video to detect motion and forwards relevant data.

TABLE I: Description of inter-module edges of application case studies [8]

Tuple type	CPU load [MI]	Data Size [B]
RAW_VIDEO_STREAM	1000	20000
MOTION_VIDEO_STREAM	2000	2000
DETECTED_OBJECT	500	2000
OBJECT_LOCATION	1000	100
PTZ_PARAMS	100	28

Object Detector: it receives data from the motion detector, extracts moving objects, calculates their coordinates, and initiates tracking.
Object Tracker: it receives object coordinates, computes the optimal configuration for Pan-Tilt-Zoom (PTZ) cameras, and periodically transmits this information to PTZ control.
PTZ Control: It is embedded within smart cameras and adjusts the camera based on PTZ parameters from the object tracker, serving as the system's actuator.
User Interface: The application presents a user interface by streaming selected video feeds containing tracked objects to the user's device.

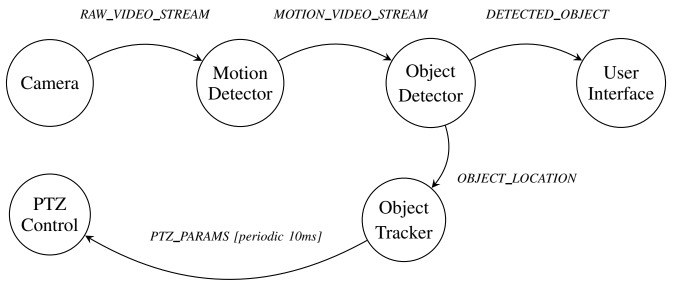

Fig. 1: Surveillance Case-study Application [8].

Figure 1 shows the graph of the surveillance application containing application modules and edges and Table I shows the processing requirement and data size of tuples considered in this application.

III. Related work

Energy consumption across modern computing environments is increasingly recognized as a critical issue, driven by the proliferation of diverse computing components and applications [12]. Despite growing interest in understanding and mitigating this energy use, a lot of existing research focuses on field-specific analyses: IoT, cloud computing, and networking provide specific methods and tools for the analysis (and reduction) of energy consumption of their infrastructure in isolation. Our work, by contrast, focuses on building an end-to-end energy model, as there is limited research integrating different components into a single, unified model or framework.

Li et al. [1] investigate an end-to-end energy consumption model for edge-cloud-based IoT platforms, using a camera-based traffic monitoring application as a case study. They considered three main infrastructure components for IoT service deployment: IoT devices, wired communication networks, and cloud infrastructure. Energy consumption for each component is modeled through a combination of simulation and measurement. However, their model focuses on energy consumption from the device perspective and does not account for application-level energy consumption on each device. Additionally, their work addresses a single-application scenario rather than multi-application scenarios. Another broad study, by Ahvar et al. [4], explores the energy impact of IoT wifi devices on Cloud and telecommunication infrastructure through simulations and real measurements. They develop an end-to-end energy model, focusing on low-bandwidth IoT applications like smart meters and sensors. This work underscores the significance of accurately assessing the energy footprint of IoT deployments and emphasizes the need for energy optimization strategies. Neither of these end-to-end studies combines networking and computing energy by focusing on modeling the energy consumption of applications, nor do they address multi-application scenarios. To the best of our knowledge, our work is the first to target a fine-grained simulation-based approach for studying per-application energy behavior in multi-application edge-cloud environments. To facilitate this novel approach, we provide new, reusable enhancements to the iFogSim simulation framework, thus enabling comprehensive end-to-end energy-consumption analysis of single applications deployed in a multi-application computing continuum.

The original iFogSim has been often used to assess energy efficiency of edge computing. Research using iFogSim can be typically classified into two categories: studies that model applications to compare energy consumption between cloud-only and edge-computing scenarios, often showing the greater efficiency of edge computing [10], [11], and those that propose application placement strategies, demonstrating their efficiency through case studies [13]. Unlike these studies, our research adapts the simulator to support a multi-application environment, focuses on the energy consumption of application modules rather than just device-level consumption, and combines both communication and processing energy consumption.

IV. Proposed Fine-Grained Simulation Approach

Our goal is to use iFogSim to study and analyze the end-to-end energy consumption of applications. An existing study revealed that the original iFogSim only reflects devices' energy consumption and does not account for application energy consumption [14]. Although the study proposed improvements to the framework also to report the energy consumption of applications, we identify two further iFogSim limitations that hinder a realistic estimation of application energy in multi-application scenarios: 1) Lack of communication energy consumption analysis, and 2) Coarse-grain modeling and reporting. The following sections present our contributions in tackling these limitations, and thus improve iFogSim to support more realistic scenarios.

A. Networking Energy Model

iFogSim focuses on the *computation* energy consumption, i.e., the energy spent by the processing application tasks, and neglects the energy consumption related to networking and communication. We have extended the simulator with two additional carefully selected energy models, chosen from a range of options, thus enabling it to also account for communication energy consumption.

Modeling networking and communication energy consumption is a well-explored area. Much of this research has primarily focused on modeling the energy consumption of devices for various technologies. For such modeling, researchers often rely on platforms, such as NS-3, which is widely recognized and utilized for implementing different networking models and technologies [4], [15], [16].

While NS-3 is a powerful tool for studying and analyzing networking energy at the device level, it is not suitable for application-level energy estimation research. NS-3's low-level models are too detailed for this purpose.A more abstract, high-level simulator is needed to consider networking infrastructure concerning applications.

To address this need, we explored various models for networking energy consumption, including the ECOFEN model, flow-based models, and combinations of wired and wireless communication networks [16]–[18]. Our objective is to develop a general framework capable of estimating the end-to-end energy consumption of applications. Thus, we need networking models that do not require details about the number of hops between devices and their specific technologies, as such information is often obscured from users.

For this reason, models like ECOFEN, which rely on detailed, device-centric information, are not well-suited to our purpose. Instead, we prefer application-centric models that estimate the energy consumption of application services, support multi-application scenarios, and attribute consumption to specific applications. Consequently, we consider a flow-based model to estimate the communication energy consumption of applications [17]. According to this model, the continuum includes two types of devices: shared devices and end-user equipment, each with specific energy models [17].

1) Networking Energy Model for Shared Devices: Shared devices, such as fog devices and cloud data centers, are capable of simultaneously serving multiple applications and supporting multitenancy. Among these devices, the network interface card (NIC) is the responsible unit for transferring data for applications while consuming energy. Monitoring the energy consumption of this unit provides a representation of networking energy consumption [19].

Given the shared nature of these devices, it becomes important to track the active time of the NIC unit for each application. This is because the NIC may concurrently transfer data for multiple applications on different ports. In our chosen flow-based model for these devices, we monitor the energy consumption of the NIC unit across distinct active intervals for each application. Additionally, we account for intervals during which the NIC transfers data for multiple applications on different ports. The model considers the allocation of this energy to the active applications based on their respective bandwidth usage ratios [17].

Equation 1 illustrates the active energy consumption model utilized for a shared device. According to this model, applications should pay for active energy based on the size of the transferred data. Equation 2, on the other hand, shows the idle energy consumption for a shared device, using the flow-based model. Based on this equation, applications should pay for idle energy during their active intervals, proportionate to their bandwidth ratio.

$$E^{sh}_{active} = \sum_{\text{intervals}} \left(BW_i \times T_{\text{interval}} \times \frac{P_{max} - P_{idle}}{BW_{aggregated}} \right) \quad (1)$$

$$E^{sh}_{idle} = \sum_{\text{intervals}} \left(BW_i \times T_{\text{interval}} \times \frac{P_{idle}}{BW_{used}} \right) \quad (2)$$

The parameters for both models are:

- P_{max} is the maximum power consumption,
- P_{idle} is the idle power consumption,
- T_{interval} is the duration of each active interval,
- BW_i is the bandwidth of application i,
- $BW_{aggregated}$ is the aggregate bandwidth of both the uplink and downlink of NIC, and
- BW_{used} is the used bandwidth of links - uplink, downlink, or both, depending on which links are active.

2) Networking Energy Model for End-User Devices: End-user devices are specific to individual users within their premises. Depending on the type of these devices, users can access various services and applications, such as email or social media. We consider a time-based energy model for end-user devices [17]. According to this model, as the device is exclusively dedicated to users and their applications, applications should pay for the total idle energy consumption of the device, proportionate to their active time ratio [17]. However, this model addresses time-shared scenarios and does not account for space-shared scenarios, as seen in iFogSim, where devices have dedicated bandwidth for uplink and downlink communication, allowing data transmission in a space-shared approach. Consequently, we modified the time-based model to support space-shared scenarios and consider transferring data for multiple applications at the same time by considering the active time of the NIC.

Equation 3 illustrates the time-based active energy consumption model utilized for end-user devices. According to this model, applications should pay for active energy during their active intervals, proportionate to their bandwidth ratio.

Equation 4 shows the modified versions of time-based idle energy consumption, innovatively adapted and extended to address multi-application scenarios. Based on this equation, applications should pay for the total idle energy consumption of the device based on their bandwidth ratio.

$$E^{user}_{active} = \sum_{\text{intervals}} \left(BW_i \times T_{\text{interval}} \times \frac{P_{max} - P_{idle}}{BW_{aggregated}} \right) \quad (3)$$

$$E^{user}_{idle} = P_{idle} \times \left(\frac{\text{Simulation time}}{\text{NIC active time}} \right) \times \sum_{\text{intervals}} T_{\text{interval}} \times \left(\frac{BW_i}{BW_{used}} \right) \quad (4)$$

Finally, all these models are symbolic, and therefore generic. However, to use them to report the energy of communication in user-relevant applications, and combine them with models for computation energy consumption, we need to use calibrated values for the parameters. In this work we use data from literature [17]; the values are presented in Table II.

TABLE II: Power Parameters of Network Equipment [17].

Device Type	Device Name	Power (Watt)	
		Maximum	Idle
Shared	Cloud	12300	1070
	Proxy Server	4550	4095
	Router	4550	4095
End-user	Smart Camera	4.6	2.8

B. Fine-Grained Energy Reporting

A second improvement we made in iFogSim concerns the reporting of energy consumption. To this end, we add fine-grained, per-VM energy consumption calculation and reporting for both computing and networking energy consumption.

1) Computation Energy: Originally, iFogSim only reports the computation energy consumption of devices and does not consider the energy consumption of applications. A previous study [14] showed how iFogSim can be enhanced to report the energy consumption of applications in single-application scenarios. However, in real-life situations, multi-tenancy and multi-application scenarios are common. Thus, additional changes are needed to accurately account for and report per-application energy consumption for such scenarios [20].

Moreover, in iFogSim, applications are divided into multiple application modules, each with different requirements. Therefore, having the energy consumption data of these application modules could help users identify energy-intensive components of applications for optimization purposes.

To address these needs, we not only improved iFogSim to support the reporting of application energy consumption in multi-application scenarios, but also enhanced it to provide insight into the energy consumption of virtual machines hosting application modules. This fine-grained energy accounting considers the MIPS allocation ratio of VMs, allowing for the characterization of the computation energy consumption at the application module level. Specifically, iFogSim divides and assigns its computation capacity, measured in MIPS, among its running VMs. In accounting for the energy consumption of VMs, we take into account their MIPS share ratio to accurately assess the energy consumption of each VM.

2) Networking energy: In iFogSim, tuples represent the requirements of incoming tasks for each application module. Whenever an application module processes its incoming tuple, it generates a new one to send to another application module. Thus, tuples transmission between modules contributes to the networking energy consumption [8].

To enhance reporting of energy consumption in iFogSim, we implemented our networking energy models (see Section IV-A) for each device NIC, based on their type (shared or end-user) in iFogSim with fine granularity. Thus, iFogSim can report the energy consumption of each tuple, providing users with detailed information about the networking and

TABLE III: Fog devices for our continuum architecture [8].

Device type	Computational Capacity [MIPS]	RAM [GB]	Power [W] Max	Power [W] Idle
Cloud	44800	40	1648.0	1332.0
Proxy server	2800	4	107.3	83.4
Router	2800	4	107.3	83.4
Smart camera	500	1	87.5	82.4

communication energy consumption of tasks. Consequently, users can conduct experiments to analyze placement policies, to adjust data sizes of sent tuples, or to improve the energy characteristics of NICs in devices.

V. Evaluation

To validate our fine-grained per-application approach to energy consumption analysis and reporting, we propose several experiments featuring a representative computing continuum architecture, one application with multiple configurations, and different multi-application scenarios. This section describes our experimental setup, the experiments, and provides an analysis of the results.

A. Experimental Setup

Figure 2 depicts the architecture considered for the continuum in our experiments. Table III shows the configuration of the fog devices in our case study. Each component is defined by its computational capacity in million instructions per second (MIPS), reflecting its frequency, maximum power consumption (i.e., when fully utilized), and idle power consumption, as well as the size of RAM.

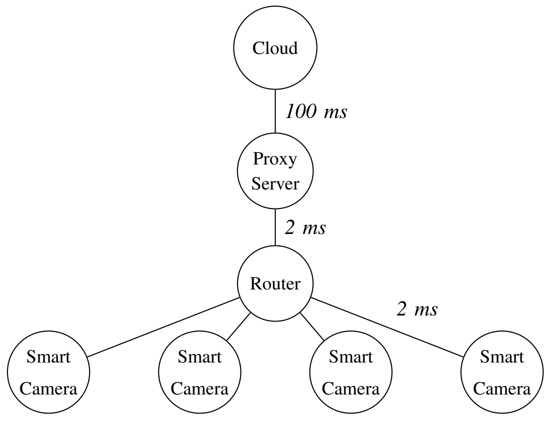

Fig. 2: Continuum architecture including links latencies.

To illustrate our fine-grained simulator's capabilities to analyze per-application energy consumption in a multi-application context, we propose two classes of experiments: we investigate the impact of different mapping scenarios and application workloads on energy consumption. For these experiments, we configure two instances of the surveillance case-study (see Section II-B) , and simulate both applications concurrently, with cameras initiating frame transmission for processing every 5ms [8]. Simulations are carried out for a duration of 2000ms (thus, about 400 frames).

B. Mapping Scenario Analysis

We explore different mapping scenarios for the two applications. We assume the edge device (the smart camera in our case study) can do some processing, and therefore, considering the surveillance application requirements, we assign the motion detector modules to the smart cameras in all scenarios. We also assign the user interface module to the cloud in all scenarios.

TABLE IV: Mapping scenarios of surveillance case study for execution models.

Scenario	Application	Mapping
Router_only	App_1	Object_detector, Router Object_tracker, Router
	App_2	Object_detector, Router Object_tracker, Router
Router_Proxy	App_1	Object_detector, Router Object_tracker, Router
	App_2	Object_detector, Proxy Object_tracker, Proxy
Router_Cloud	App_1	Object_detector, Router Object_tracker, Router
	App_2	Object_detector, Cloud Object_tracker, Cloud
Proxy_only	App_1	Object_detector, Proxy Object_tracker, Proxy
	App_2	Object_detector, Proxy Object_tracker, Proxy
Proxy_Cloud	App_1	Object_detector, Proxy Object_tracker, Proxy
	App_2	Object_detector, Cloud Object_tracker, Cloud
Cloud_only	App_1	Object_detector, Cloud Object_tracker, Cloud
	App_2	Object_detector, Cloud Object_tracker, Cloud
Module_Router_Proxy	App_1	Object_detector, Router Object_tracker, Proxy
	App_2	Object_detector, Router Object_tracker, Proxy

We consider different devices as targets for the other two modules (Object detector and Object tracker). Table IV illustrates all mapping scenarios with the target devices of the modules in each scenario.

Since the capacity of fog nodes is limited, the ordering of processes affects the latency and energy consumption of applications when their modules run on the same target device. Therefore, we consider two types of scenarios:

- Application-Wise Ordering: assigning modules of one application first, and then proceed with the modules of the next application. For example, we decide on the placement of the Object detector and Object tracker modules in the first application and then determine the placement of these two modules in the second application.
- Module-Wise Ordering: sequentially assigning the modules of both applications. For example, we start with the Object detector module for both applications before moving on to the Object tracker.

We run separate simulations for all these scenarios and analyze the energy consumption of computing and communication for each scenario and each application.

Both the computation and networking energy, as depicted in Figures 3 and 4, respectively, show better results in the Router_Only scenario where modules of both applications are assigned to the router. However, when considering the high latency of this scenario caused by the limited capacity of the

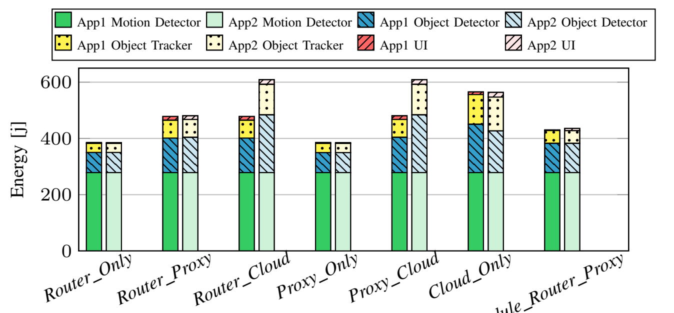

Fig. 3: Computational energy consumption of VMs across different mapping scenarios when running two applications concurrently.

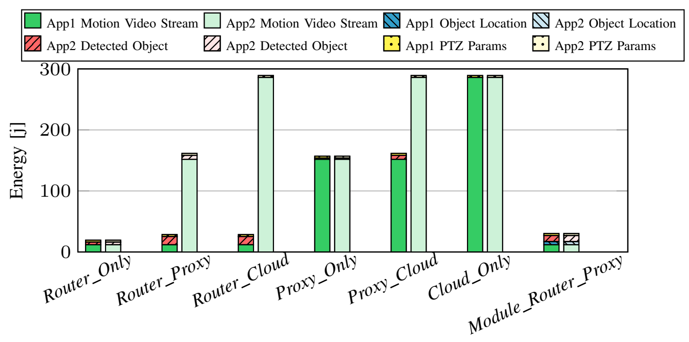

Fig. 4: Networking energy consumption of Tuples across different mapping scenarios when running two applications concurrently.

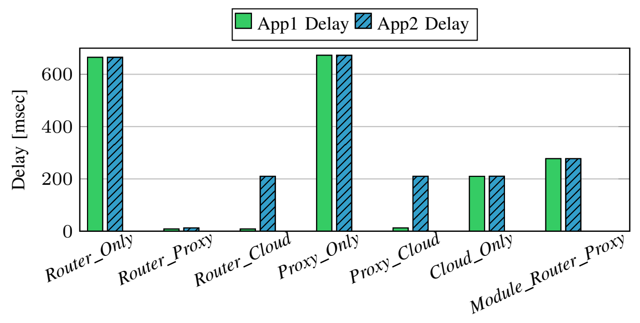

Fig. 5: Observed delay across different mapping scenarios when running two applications concurrently.

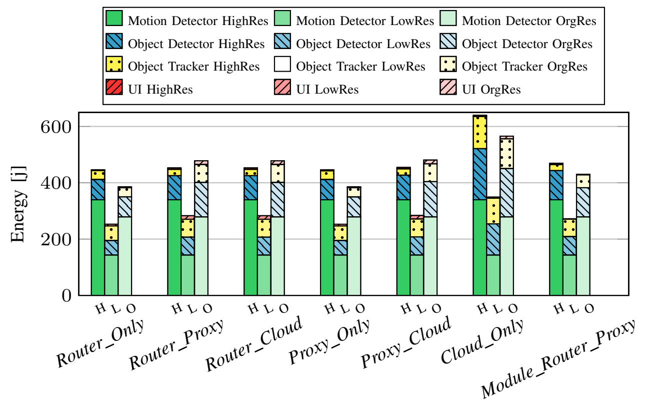

Fig. 6: Energy consumption of VMs of first application with different resolution across different mapping scenarios.

device and the resulting *overflow*, the importance of offloading to the cloud becomes evident, as illustrated in Figure 5. "*Overflow*" occurs when the number of tasks exceeds the device's capacity within a time interval, leading to unprocessed tasks queued for subsequent intervals. This backlog can grow over time, indicating overloaded processing resources.

Additionally, by comparing energy and latency across different scenarios, we find that Module-Wise mapping generally yields better results than Application-Wise mapping, and this makes Module-Wise mapping the better overall choice. For instance, in the Router_Cloud scenario, where two modules of the first application are assigned to the router and those of the second application are assigned to the cloud, the mapping is ApplicationWise.

Additionally, Figures 3 and 4 show that, in some scenarios – such as those where parts of the application are offloaded to high-level devices like the cloud, which are distant from the user – networking energy consumption becomes significant. This underscores the importance of considering this networking energy consumption as part of the simulation results.

C. Analysing different workloads

Higher-resolution videos improve object detection accuracy, but demand more computational resources and bandwidth for data transfer. Thus, we expect that lowering video resolution conserves computing resources, reduces network usage, and may reduce energy consumption. To test this hypothesis, we conducted experiments with various resolutions, examining how these workload changes affect the energy consumption of various application modules.

TABLE V: Application requirements with different resolutions

Tuple type	Resolution	CPU load [MI]	Data Size [B]
RAW_VIDEO_STREAM	Original	1000	20000
	High Res.	2000	40000
	Low Res.	500	10000
MOTION_VIDEO_STREAM	Original	2000	2000
	High Res.	4000	4000
	Low Res.	1000	1000
DETECTED_OBJECT	Original	500	2000
	High Res.	1000	4000
	Low Res.	250	1000
OBJECT_LOCATION	Original	1000	100
	High Res.	1000	100
	Low Res.	1000	100
PTZ_PARAMS	Original	100	28
	High Res.	100	28
	Low Res.	100	28

Table V displays the computation requirements and data sizes of sending tuples for each application resolution. To model high-resolution applications, we doubled these values, while for low-resolution applications, we halved the values.

Figure 6 illustrates VM energy consumption for one application across different scenarios and resolutions. Unexpectedly, the energy consumption of VMs hosting higher-resolution applications occasionally showed comparable or lower energy consumption than VMs running the original resolution. Similarly, we observed unexpected energy consumption for transferring tuples, where, in some scenarios, the original resolution consumed more energy than the high-resolution version.

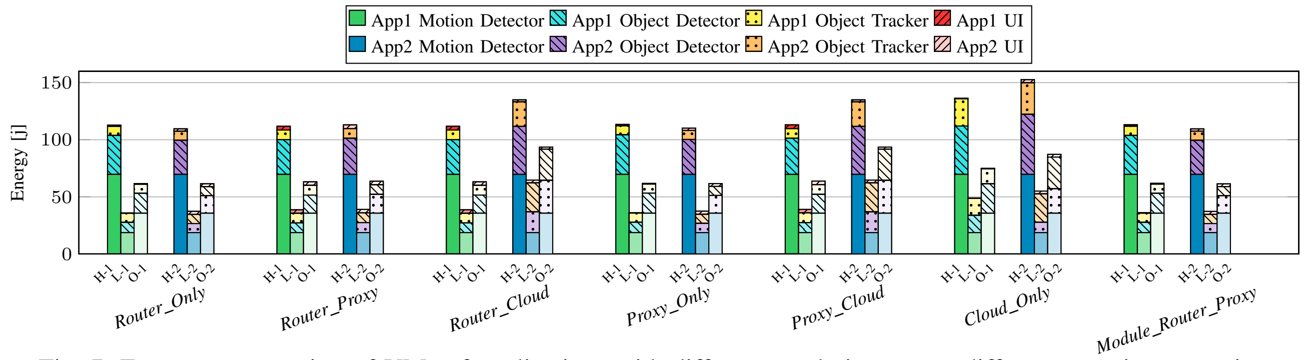

Fig. 7: Energy consumption of VMs of applications with different resolution across different mapping scenarios.

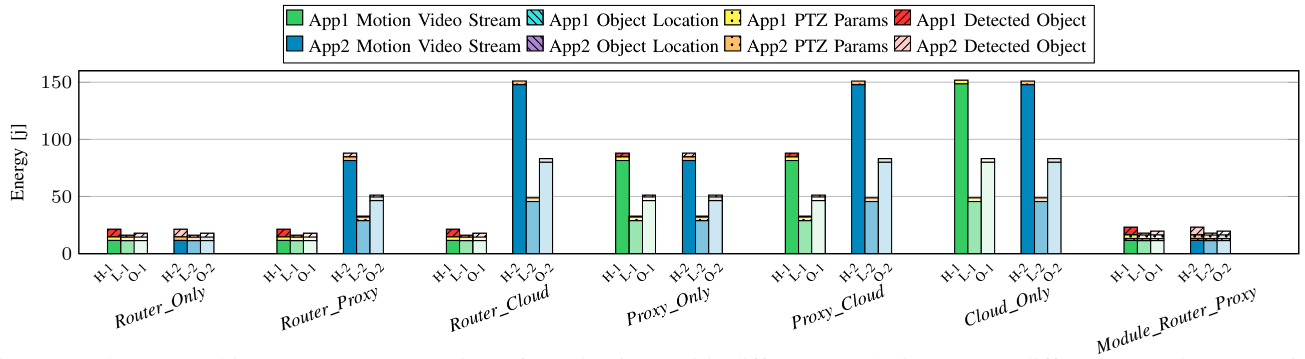

Fig. 8: Tuples networking energy consumption of applications with different resolution across different mapping scenarios.

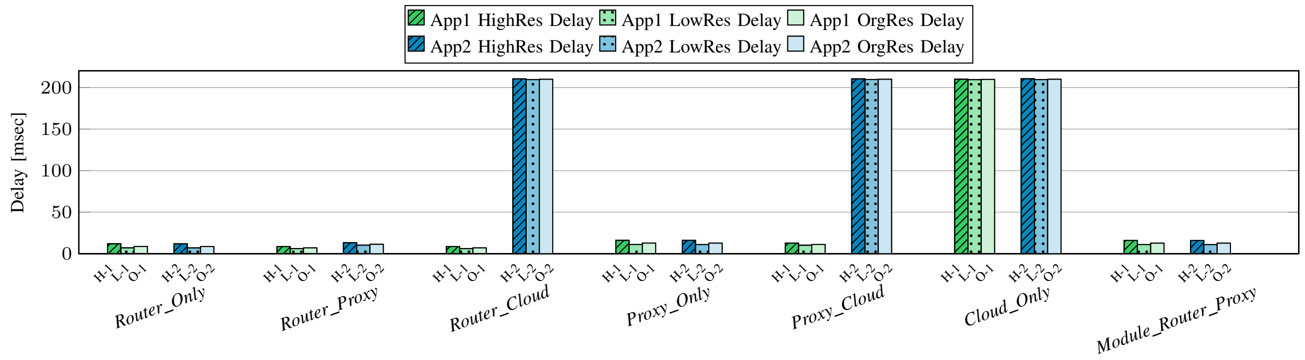

Fig. 9: Average delay of applications with different resolution across different mapping scenarios.

Furthermore, the average delay for the original resolution was higher compared to the delay of the high resolution.

We analyzed the application's behavior under different resolution settings, from 0.2x to 5x of the original, as depicted in Figure 10. Initially, increasing the resolution caused router overflow, and increased the delay. Beyond a certain resolution, however, the camera's processing capacity became the bottleneck, reducing the emission rate and, consequently, lowering the delay. To mitigate overflow and maintain reliability, we doubled the device capacities and extended the camera emission interval to 20ms.

We further observed unexpectedly high energy consumption for VMs when one or both applications were running on the cloud. We investigated the simulator code to address this issue. In iFogSim, each event (e.g., tuple arrival or completion) triggers functions that adjust VM states, update energy consumption, and verify tuple completion status. We found that the simulator only allocated MIPS to VMs with running tuples, neglecting the target VM of arriving tuples. This led to infinite finish times estimates, and higher energy consumption for some VMs. We therefore improved the simulator to allocate MIPS to both running VMs and target VMs, thus enabling accurate finish time estimation and observing the correct/expected energy consumption.

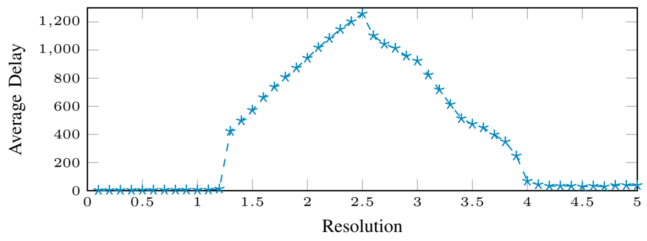

Fig. 10: Average delay of a single application across different resolution scales in Router_Only mapping scenario.

Figures 7, 8, and 9 illustrate computation energy consumption of VMs, networking energy consumption of sending tuples, and average delay of applications with different resolution across various scenarios, respectively. As depicted in these figures, in scenarios where both apps are assigned to the same device, such as 'Router_Only', or scenarios where device characteristics are identical, such as 'Router_Proxy', the energy consumption of the applications is similar. In scenarios where one of the applications is assigned to the cloud, like 'Router_Cloud', the energy consumption of the application assigned to the cloud is higher than the other one, highlighting the better energy efficiency of assigning application modules to devices closer to the user, as expected.

In the 'Cloud_Only' scenario, we observed slight differences in application energy consumption. This variation stems from iFogSim's consideration of a minimum event time of 0.1, affecting tuple finish time estimations. As a result of small tuple sizes and ample cloud capacity, tuples finish processing earlier than estimated, influencing energy updates based on the arrival time of other tuples. This, in turn, leads to varied energy consumption.

Based on these results, it becomes evident that the energy consumption and average delay of applications improved in module-wise scenarios. Furthermore, this experiment demonstrates that reducing the resolution of applications and offloading them to edge nodes, closer to the users, can achieve better energy efficiency and latency.

VI. Conclusion and Discussion

With the increasing use of digital services, concerns are raised about the growing energy footprint and environmental impact of ICT. A better understanding of the energy consumption of digital services is needed to enable stakeholders to take action and limit the environmental impact of the field.

In this study, we contribute to this call for action by enhancing state-of-the-art continuum simulators to improve energy consumption estimates by introducing an enhanced version of the iFogSim framework for multi-application simulations. Our approach enables a comprehensive analysis of application energy behavior across diverse mapping scenarios by implementing fine-grained models for both computation and networking energy consumption.

Through experiments focused on a surveillance application case study, provided by iFogSim, we demonstrate the power of this simulation approach in identifying energy-intensive modules and guiding informed decisions on mapping strategies, architectural configurations, and workload resolutions.

Our study has made contributions, including developing and implementing energy models for Network Interface Cards (NICs) of different network devices in iFogSim, along with the detailed characterization and comparison of energy consumption across diverse scenarios.

However, the accuracy of iFogSim's results is inherently limited to the system architecture, requiring careful parameter calibration. Key challenges include modeling and validating different communication technologies and addressing application-specific data variability, which impacts energy consumption and deployment strategies. Conducting real-world validation and refining energy models for diverse ICT architectures are crucial steps for improving model accuracy and reliability. Future research should also develop best practices for energy-efficient digital services. Tackling these challenges will enhance the accuracy and utility of energy consumption models, leading to more sustainable ICT practices.

References

[1] Y. Li *et al.*, "End-to-end energy models for edge cloud-based iot platforms: Application to data stream analysis in iot," *Future Generation Computer Systems*, vol. 87, pp. 667–678, 2018.

[2] R. Oma *et al.*, "A tree-based model of energy-efficient fog computing systems in iot," in *CISIS*. Springer, 2019, pp. 991–1001.

[3] W. Tang *et al.*, "An offloading approach in fog computing environment," in *2018 IEEE SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*. IEEE, 2018, pp. 857–864.

[4] E. Ahvar *et al.*, "Estimating energy consumption of cloud, fog, and edge computing infrastructures," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 2, pp. 277–288, 2019.

[5] M. Jansen *et al.*, "Continuum: Automate infrastructure deployment and benchmarking in the compute continuum," in *FastContinuum'23, in conjunction with ICPE, Portugal, April*, 2023.

[6] A. Brogi *et al.*, "How to place your apps in the fog: State of the art and open challenges," *Software: Practice and Experience*, vol. 50, no. 5, pp. 719–740, 2020.

[7] R. Mahmud *et al.*, "Modelling and simulation of fog and edge computing environments using ifogsim toolkit," *Fog and edge computing: Principles and paradigms*, pp. 1–35, 2019.

[8] H. Gupta *et al.*, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[9] R. N. Calheiros *et al.*, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[10] K. S. Awaisi *et al.*, "Towards a fog enabled efficient car parking architecture," *IEEE Access*, vol. 7, pp. 159 100–159 111, 2019.

[11] I. Sarkar *et al.*, "Fog computing based intelligent security surveillance using ptz controller camera," in *10th International Conference on ICCCNT*. IEEE, 2019, pp. 1–5.

[12] C. Preist *et al.*, "Analyzing end-to-end energy consumption for digital services," *Computer*, vol. 47, no. 5, pp. 92–95, 2014.

[13] M. Mahmoud *et al.*, "Towards energy-aware fog-enabled cloud of things for healthcare," *Computers & Electrical Engineering*, vol. 67, pp. 58–69, 2018.

[14] S. Baneshi *et al.*, "Estimating the energy consumption of applications in the computing continuum with ifogsim," in *International Conference on High Performance Computing*. Springer, 2023, pp. 234–249.

[15] P. Serrano *et al.*, "Per-frame energy consumption in 802.11 devices and its implication on modeling and design," *IEEE/ACM Transactions on networking*, vol. 23, no. 4, pp. 1243–1256, 2014.

[16] A.-C. Orgerie *et al.*, "Ecofen: An end-to-end energy cost model and simulator for evaluating power consumption in large-scale networks," in *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE, 2011, pp. 1–6.

[17] F. Jalali *et al.*, "Fog computing may help to save energy in cloud computing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728–1739, 2016.

[18] H. Casanova, "Simgrid: A toolkit for the simulation of application scheduling," in *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE, 2001, pp. 430–437.

[19] P. Mahadevan *et al.*, "A power benchmarking framework for network devices," in *NETWORKING 2009: 8th International IFIP-TC 6 Networking Conference, Aachen, Germany, May 11-15, 2009. Proceedings 8*. Springer, 2009, pp. 795–808.

[20] I. Odun-Ayo *et al.*, "Cloud multi-tenancy: Issues and developments," in *Companion Proceedings of the10th International Conference on utility and cloud computing*, 2017, pp. 209–214.