Estimating the Energy Consumption of Applications in the Computing Continuum with *iFogSim*

Saeedeh Baneshi[1](✉), Ana-Lucia Varbanescu[1], Anuj Pathania[1], Benny Akesson[1,2], and Andy Pimentel[1]

[1] University of Amsterdam, Amsterdam, The Netherlands
s.baneshi@uva.nl
[2] TNO-ESI, Eindhoven, The Netherlands

Abstract. Digital services - applications that often span the entire computing continuum - have become an essential part of our daily lives, but they can have a significant energy cost, raising sustainability concerns. The computing continuum features multiple distributed layers (edge, fog, and cloud) with specific computing infrastructure and scheduling decisions at each layer, which impact the overall quality of service and energy consumption of digital services. Measuring the energy consumption of such applications is challenging due to the distributed nature of the system and the application. As such, simulation techniques are promising solutions to estimate energy consumption, and several simulators are available for modeling the cloud and fog computing environment.

In this paper, we investigate *iFogSim*'s effectiveness in analyzing the end-to-end energy consumption of applications in the computing continuum through two case studies. We design different scenarios for each case study to map application modules to devices along the continuum, including the Edge-Cloud collaboration architecture, and compare them with the two placement policies native to *iFogSim*: *Cloud-only* and *Edge-ward* policies. We observe *iFogSim*'s limitations in reporting energy consumption, and improve its ability to report energy consumption from an application's perspective; this enables additional insight into an application's energy consumption, thus enhancing the usability of *iFogSim* in evaluating the end-to-end energy consumption of digital services.

Keywords: Digital services · End-to-end energy consumption · Edge computing · Simulation

1 Introduction

Scientific discovery, product development, data science and artificial intelligence, online shopping, and entertainment rely increasingly on digital services. As such, digital services have become a crucial part of our daily life, but they come with a significant, rapidly-increasing energy cost, raising sustainability concerns [20]. Worldwide estimates project the ICT sector to reach 21% of global energy consumption by 2030 [14].

Users access digital services through smartphones, laptops, and tablets, triggering the entire continuum from the device to the *edge*, *fog*, and *cloud* [12]. *Edge* devices are closest to the end-user and improve user experience by reducing latency and providing faster access to data [1]. The *fog*, located closer to the data center, handles data processing, storage, and communication [24]. The *cloud* is a remote location for (large-scale) data storage, management, and processing [18,19]. Each layer has its specific computing infrastructure, and scheduling decisions at each layer impact the overall quality of service (QoS) and energy consumption of digital services [2].

Measuring the energy consumption of digital services spanning the entire computing continuum - from edge to cloud - is a challenging task due to the distributed nature of the system and the application. Simulation techniques can estimate the energy consumption to overcome this challenge [16].

Various simulators are available for modeling the cloud and fog computing environment and estimating the energy consumption and performance of different components of the architectures. For example, *CloudSim* [8] and *GreenCloud* [15] focus on modeling cloud computing, while *EdgeCloudSim* [23] and *iFogSim* [10] model fog computing. We focus on *iFogSim*, a toolkit for modeling and simulating resource management techniques in the IoT (Internet of Things), edge, and fog computing environments.

We explore the effectiveness of *iFogSim* in analyzing the end-to-end energy consumption of applications, aiming to identify its strengths and weaknesses for such an analysis. We find there is a lack of reporting actual energy consumed by the application on the device: in fact, *iFogSim* reports the total energy *consumed by the device*, which is misleading for applications that do not fully utilize all resources available to them (for example, they only use the cloud sporadically). Therefore, we improve iFogSim's energy reporting function to also include energy consumption and performance from the application perspective. We demonstrate the usefulness of such reports using two case studies: the Surveillance application and the Latency-sensitive online VR game.

Specifically, we conduct our investigation using six Edge-Cloud continuum scenarios for the Surveillance application, and four scenarios for the VR game application, and compare the results. We find a significant difference between the energy consumption results of *iFogSim* from the device- and application-perspectives. The results also confirm that running applications on fog devices close to the user can reduce energy consumption and improve performance.

In summary, the main contributions of this paper are as follows:

- We show how to design different Edge-Cloud collaborative scenarios to map application modules to devices along the computing continuum for two applications. These complement the standard Edge-ward and Cloud-only scenarios from *iFogSim*.
- We compare six scenarios for mapping application modules to devices along the continuum in a surveillance application and four mapping scenarios in a VR game application. We demonstrate how to analyze and contrast their energy consumption and performance using *iFogSim*.

The remainder of this article is organized as follows. In Sect. 2, we provide background information on frameworks for modeling cloud and fog computing environments, and we introduce our chosen tool, *iFogSim*. In Sect. 3, we describe the process for modeling the topology architecture and application and configuring them using *iFogSim*. We also present two applications: a surveillance application and a Vr game application used as case-studies in our experiments and explain the scenarios we designed for the mapping of applications modules. Next, in Sect. 4, we discuss our empirical evaluation setup and experiments and analyze our results. Section 5 discusses related work and Sect. 6 concludes the paper and highlights directions for future research.

2 Background

We briefly introduce the setup and technology used in this work: the computing continuum and the *iFogSim* simulation.

2.1 The Computing Continuum

Digital devices can produce massive amounts of (heterogeneous) data that require fast and increasingly complex processing [9,10], using different tools to provide understandable results for users. However, the more complex the analysis is, the more challenging the processing becomes; consequently, resources local to the devices become insufficient, and computation offloading is needed. Offloading *all* data processing to the cloud is not always the most efficient solution due to, for example, the high latency of the centralized computing approach or privacy concerns. Fog computing provides a more effective way for data processing, by offloading the analysis to a combination of different layers of decentralized computing [9,24], and only relying on cloud offloading. In this context, fog and cloud computing form the *computing continuum*. Underlying the compute continuum is a complex infrastructure of devices (i.e., sensors), edge devices (i.e., processing stations between sensors and the cloud), and data centers [12].

Offloading processing to different components of the computing continuum has different impacts on user quality-of-service (QoS) and energy consumption. There is a growing need to quantify these costs and assess the sustainability of digital services [6]. However, measuring the energy consumption of applications across this complex continuum is challenging, because the components are distributed (geographically and in terms of ownership), making uniform measurements virtually impossible [16]. Instead, we rely on *simulators* to help estimate the energy consumption of different applications [7] and different offloading scenarios. The key challenge these simulators face is to provide an accurate system representation.

2.2 Modeling Cloud and Fog Computing

Researchers have developed highly accurate cloud computing simulators to model the complex structure of the data centers, their topology, the resource managers and schedulers, and the challenges of multi-tenancy. As such, simulators such as *GreenCloud* [15], *CloudSim* [8], or *OpenDC* [17] often provide a trade-off between very accurate results and the complexity of models, and the cost of simulation. *GreenCloud* focuses on detailed and expensive energy consumption simulation, while *CloudSim* and *OpenDC* are cheaper and more useful for scenario analysis. However, none of them addresses the full computing continuum.

Expanding towards the computing continuum, there exist several simulators. For example, *EdgeCloudSim* is an open-source simulator designed to evaluate the performance of edge computing systems. This tool is based on *CloudSim* and provides network modeling (WLAN and WAN), a device mobility model, and a tunable load generator. However, it does not support the energy consumption model for mobile devices, edge, and cloud data centers. It also does not support task migration among the Edge or Cloud VMs [23].

iFogSim is a simulator that models IoT and Fog environments, and measures the impact of resource management techniques in terms of latency, network congestion, energy consumption, and cost. It simulates components of the computing continuum, including edge devices, cloud data centers, and network links, to measure performance metrics. *iFogSim* uses the *Sense-Process-Actuate* model, where sensors publish data to IoT networks, and applications running on fog devices subscribe to and process data from sensors. The insights obtained are then translated into actions and forwarded to actuators [10].

To study different fog-cloud interactions, *iFogSim* has emerged as a popular choice for modeling various fog computing scenarios. As *iFogSim* offers a good trade-off between ease-of-use and ease-of-change (being Java-based), and given its proven feasibility for many studies, we selected it as the main tool for our own analysis, where we aim to explore its effectiveness in estimating the energy consumption of digital services using *different mappings* across the entire computing continuum.

2.3 The *iFogSim* Toolkit

The architecture of *iFogSim* consists of IoT Sensors placed at the bottommost layer of the architecture and distributed in different geographical locations, IoT Actuators operating at the same layer, and IoT Data Streams, which are sequences of immutable values emitted by sensors. Fog devices connect the sensors to the internet and host application modules; cloud resources are provisioned on-demand from geographically distributed data centers. Developers model applications developed for fog deployment as a collection of modules comprising data processing elements [10] in a Distributed Data Flow (DDF) model.

Being based on *CloudSim*, *iFogSim* leverages the basic event simulation functionalities of its "ancestor": entities communicate with each other by sending events, and the core *CloudSim* layer handles events between fog computing components in *iFogSim*. It models simulated entities and services, including `FogDevice`, `Sensor`, `Tuple`, `Actuator`, and `Application`, as classes. Each class has its attributes and methods to specify the characteristics and behavior of the entity it represents [10].

Physical Components. The physical components in *iFogSim* are sensors, fog devices, and actuators. Sensors are the sources of generating data and behave like IoT sensors: they periodically send data to other devices for further processing [16]. In *iFogSim*, there is a class specifically designed for simulating sensors, which includes a reference attribute to its parent - the fog device it connects to - and the latency between them.

Using the `FogDevice` class, *iFogSim* simulates computing resources for performing tasks. This class has specific attributes like instruction processing rate, maximum and idle power, uplink and downlink bandwidths, memory size, and level of fog device. The instruction processing rate reflects the computing capacity of the fog device. The power attributes reflect energy efficiency. Bandwidth defines the communication capacity of Fog devices.

Finally, using the `Actuator` class, *iFogSim* simulates end nodes that are data sinks. Similarly to the sensor, this class also includes a reference attribute to its parent - the fog device it connects to - and the latency between them.

Application Components. *iFogSim* models applications as directed graphs, where vertices represent the modules that execute the processing task on incoming data, and edges show data dependencies between the processing modules. *iFogSim* uses the following classes to model the applications.

AppModule: It defines the processing elements of the application. `AppModules` receive tasks from other `AppModules` (incoming tasks), process them, and produce tasks for further processing by another `AppModule`. For each module, the number of output tasks per incoming task can be defined by a fractional selectivity or a bursty model [10]. It also has attributes like memory size, computational capacity, and bandwidth.

AppEdge: It models the data dependencies between application modules and carries specific tuple types. It also exhibits the computational requirements and data size of the transported tuple. The edges can transport periodic or event-based tuples.

Tuple: It is a task generated by a module for further processing. The tuple is the fundamental unit of communication between entities, characterized by its processing requirements, measured as the number of million instructions needed and the size of encapsulated data.

AppLoop: It is a sequence of modules (and the links between them). This class defines the control loop of the application. By defining this loop from the sensor to the actuator, we can measure the end-to-end latency of the application.

Management Components. The management components in *iFogSim* are the *ModuleMapping* and the *Controller*. The *ModuleMapping* object identifies available resources in fog devices and decides where to map application modules based on their requirements. *iFogSim*'s default placement policy is *Edge-ward* placement, which deploys modules on devices close to the user if they are powerful enough. Otherwise, it forwards modules to higher-level devices toward the cloud. Alternatively, the *Cloud-only* placement policy deploys all modules to the cloud.

The *Controller* launches modules on their target fog devices using information from *ModuleMapping* and collects simulation results upon completion.

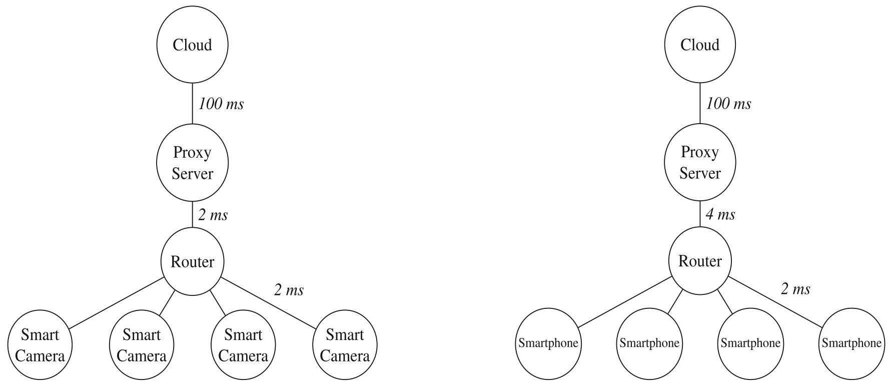
3 Scenario Design with Examples

We present the steps needed to design and execute scenario analysis using *iFogSim*.

3.1 Process

To begin the simulation process, we need to perform the following steps.

Define Topology. We must define the continuum's physical topology by adding sensors, actuators, and fog devices such as routers and proxy servers. We then configure their attributes, such as computational capacity and power consumption. Additionally, we need to define the connections between the fog devices and set the latency and bandwidth of the links. Figure 1 shows the physical topologies, which we used for our experiments for two case studies.



a Surveillance case-study architecture.

b VR Game case-study architecture.

Fig. 1. Case-studies architectures, including link latencies.

Table 1 shows the configuration of the fog devices in our topology for two case studies. Each component is defined by its computational capacity in million instructions per second (MIPS), which reflects its frequency, it's maximum (i.e., when fully utilized) and idle power consumption, and the size of RAM.

Table 1. Configuration of fog devices for two case studies architecture.

(a) Architecture configuration for surveillance application

Device type	Computational Capacity [MIPS]	RAM [GB]	Power [W]	
			Max	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

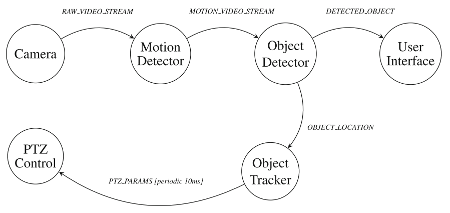
(b) Architecture configuration for VR Game application

Device type	Computational Capacity [MIPS]	RAM [GB]	Power [W]	
			Max	Idle
Cloud	44800	40	1648.0	1332.0
Proxy server	2800	4	107.3	83.4
Router	2800	4	107.3	83.4
Smartphones	1000	1	87.5	82.4

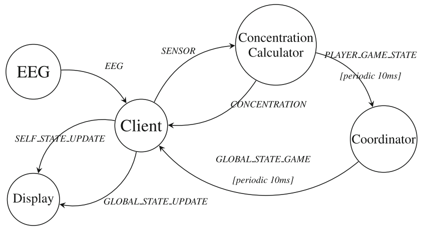
Define Application. The first step for this purpose is to split the application into interdependent `AppModules` and configure the dependency of these modules by defining `AppEdges`. Next, we must define the input and output `Tuple` types for each `AppModule`, and specify the fractional selectivities. Finally, we can define the control loops of the application by defining `AppLoop`, which we can use to monitor the end-to-end performance or delay of specific parts of the application.

Define the Main Function. In this function, we create instances of the sensors, actuators, and fog device classes to model the physical topology. Additionally, we must create instances of application modules, module mapping objects, and controllers. Finally, we must submit our application to the controller and specify the placement policy.

Start Simulation. Finally, the simulation is initiated using the *CloudSim* core of the simulator. During this stage, the simulator generates the entities and queues for storing the produced events. *iFogSim* saves all events, including their start and estimated finish time, source, target, and event tags, which indicate the type of event, in a queue called `FutureQueue`. At each time step, the events that are scheduled for that particular time are moved to another queue, called `DeferredQueue`, for processing. Subsequently, the target entities process the incoming events according to their respective types and trigger new events, which are added to the `FutureQueue` for processing.



a Surveillance Application.



b VR Game Application.

Fig. 2. Case-study Applications [10].

3.2 Case Studies

In our experiments, we considered two case studies available in *iFogSim*: an area-surveillance application and a latency-sensitive online game. In these applications, a large volume of data is produced by Sensors and sent for further processing to the network [10,16]. The processing can be done in the cloud, which is a centralized approach, or by fog devices closer to the user, which is a decentralized approach. However, the requirement for real-time operation makes a decentralized approach more suitable for efficient processing. Figure 2 depicts these applications and their modules.

Surveillance Application. The function of modules in this application are as follows:
Motion Detector: This module is embedded in smart cameras and detects motion in video streams by continuously analyzing raw video and forwarding it for further processing.
Object Detector: This module receives data from the motion detector, extracts moving objects, calculates their coordinates, and activates tracking for new objects.
Object Tracker: This module receives object coordinates, calculates optimal configuration for Pan-Tilt-Zoom (PTZ) cameras, and sends them to PTZ control periodically.
PTZ Control: This module is embedded in smart cameras and adjusts the physical camera based on PTZ parameters received from the object tracker. It serves as the system's actuator.
User Interface: The application provides a user interface by sending a fraction of video streams with tracked objects to the user's device.

Cameras act as sensors, capturing video feeds every 5 millisecond and sending them for further processing. Each module processes incoming tuples and sends new ones for further processing. Table 2a shows the processing requirements and data size per tuple used for modeling this application.

Table 2. Description of inter-module edges of application case studies [10]

(a) Surveillance application Tuples configurations.

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

(b) VR Game application Tuples configurations.

Tuple type	CPU load [MI]	Data Size [B]
EEG	2500	500
SENSOR	3500	500
PLAYER_GAME_STATE	1000	1000
CONCENTRATION	14	500
GLOBAL_GAME_STATE	28	1000
GLOBAL_STATE_UPDATE	1000	500
SELF_STATE_UPDATE	100	500

VR Game Application. The function of modules in this application are as follows:
Client: This module receives EEG signals from the headset and filters them to remove inconsistencies to send for further processing.
Concentration calculator: This module detects the brain state and calculates the concentration level of the user, then updates the player's game state.
Coordinator: This module acts at the global level and updates the state of the game for all connected clients.

In this application, users wear a headset that acts as a sensor for capturing brain signals at a specific rate of 10 ms. Then It sends these signals for further processing in the next modules. Table 2b shows the processing requirements and data size per tuple used for modeling this application.

3.3 Designed Scenarios

In our experiments, we investigate the efficiency of the *iFogSim* simulator in estimating the end-to-end energy consumption of applications. We consider the topologies illustrated in Fig. 1 and the applications illustrated in Fig. 2.

As we assume the device can do some processing, for the surveillance application, we assign the motion detector modules to the smart cameras in all scenarios; we further assign the user interface module to the cloud, as this is the module requiring complex process, without a strict requirement of low latency. For the other processing modules, we designed various mapping scenarios of edge-cloud collaboration. We have listed all six mapping scenarios we considered for the object detector and object tracker modules in Table 3. The Cloud-only and Router-only scenarios are the results of running the application using the two available policies of *iFogSim*: *Cloud-only* and *Edge-ward*.

Table 3. Mapping scenarios of surveillance case study for execution models.

Scenario	Application Module	Target device	Source
Router_only	Object_detector Object_tracker	Router Router	iFogSim
Proxy_only	Object_detector Object_tracker	Proxy server Proxy server	New
Router_Proxy	Object_detector Object_tracker	Router Proxy server	New
Router_Cloud	Object_detector Object_tracker	Router Cloud	New
Proxy_Cloud	Object_detector Object_tracker	Proxy server Cloud	New
Cloud_only	Object_detector Object_tracker	Cloud Cloud	iFogSim

For the VR Game application, by considering the requirements, we assign the client module to smartphones in all scenarios. Additionally, we assign the Coordinator module to the cloud, as this module needs to be globally accessible in order to coordinate users who may not be in the same geographical location.

For the Concentration Calculator module, we designed various mapping scenarios to have insight into the energy consumption on different layers of considered architecture. We have listed all four mapping scenarios we considered for this module in Table 4. The Cloud-only and Edge-only scenarios are the results of running the application using the two available policies of *iFogSim*: *Cloud-only* and *Edge-ward*.

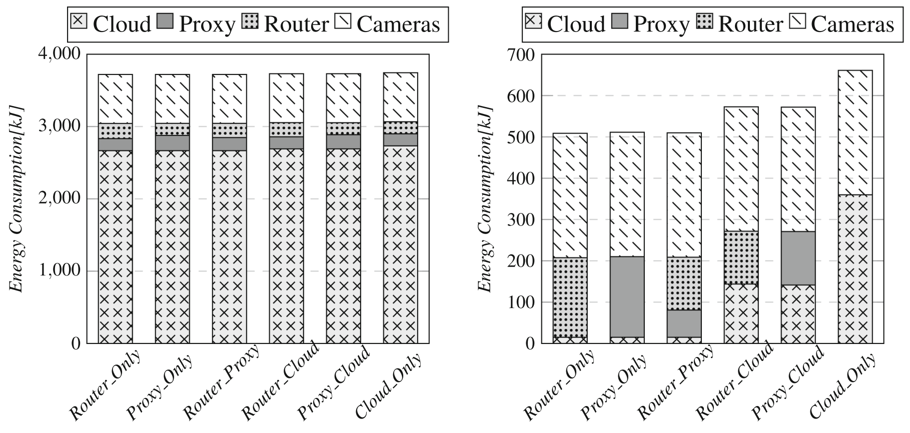
Table 4. Mapping scenarios of VR game case study for execution models.

Scenario	Application Module	Target device	Source
Edge_only	Client Concentration calculator Coordinator	Smartphone Smartphone Cloud	iFogSim
Router_Include	Client Concentration calculator Coordinator	Smartphone Router Cloud	New
Proxy_Include	Client Concentration calculator Coordinator	Smartphone Proxy server Cloud	New
Cloud_Only	Client Concentration calculator Coordinator	Smartphone Cloud Cloud	iFogSim

4 Experimental Analysis and Discussion

Our results have been gathered by simulating the two case studies on their considered architecture for 2000 s seconds per scenario, using *iFogSim*. The simulation output in *iFogSim* includes the execution time, application loop delays, `tuple` execution delays, the energy consumption of each fog device, the cost of execution in the cloud, and the total network usage. We estimate the end-to-end energy consumption by adding the energy of all fog devices in different scenarios.

Figure 3a presents the energy consumption of the surveillance system. We observe no significant difference in the energy consumption of the continuum when running various application scenarios. This lack of difference happens because *iFogSim* reports the energy consumption of devices without considering the application, which is a reasonable approach from the device and overall system energy perspective.



a Device Perspective b Application Perspective

Fig. 3. Surveillance application energy consumption in different scenarios

However, from the application perspective, this is an unreasonable approach: the idle energy consumption of fog devices *that are not used to run the application* should not be included in the estimation of application energy. Therefore, we revised the energy estimation function of the simulator to report not only total energy, but also active energy consumption, a more accurate estimation of the application's energy consumption. Figure 3b shows the application's energy consumption in different scenarios as a sum of the energy spent only by devices actively participating in the computation.

In the VR Game application, similarly, from the device perspective, we observe no significant difference between the total energy of the continuum and the energy consumed by each device across various scenarios. However, when we change the perspective to the application itself and exclude the idle energy of the devices, notable differences in both the total continuum energy and the energy consumption of each device become apparent. Figure 4 depicts the energy consumption of devices across different scenarios with different perspectives.

For both case studies, we observe that by offloading processing from the device closest to the user to Cloud, the energy consumption of the application increases. To understand the impact of this approach on application performance, we also present the so-called delay of processing for both case studies in Fig. 5. We observe that, as expected, offloading the processing to the cloud leads to a higher average delay in the control loop of the applications.

Our results indicate that offloading the application to devices close to the user is a more effective strategy for performance and energy consumption. However, the computing capacity of fog devices, such as routers and proxy servers, is limited for computationally intensive applications.

In this experimental study, we observed that *iFogSim* is a powerful tool for analyzing performance metrics, such as delay, execution cost, network usage, and

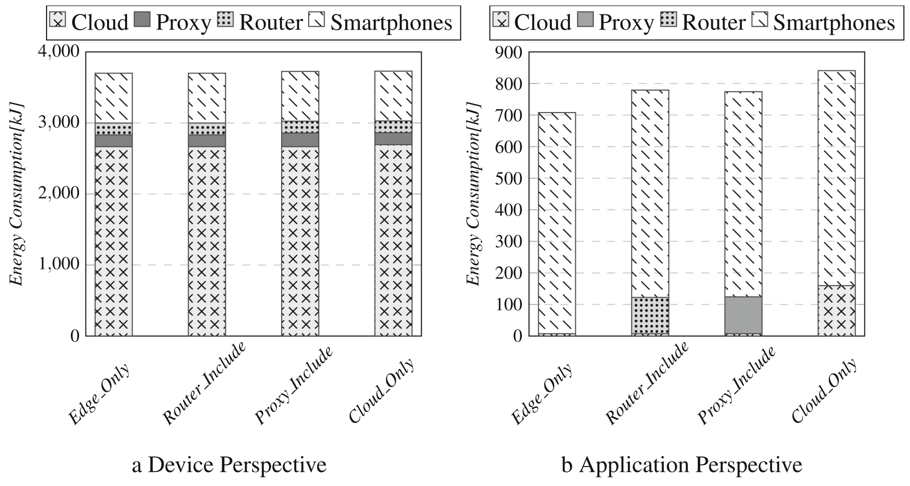


Fig. 4. VR Game application energy consumption in different Scenarios

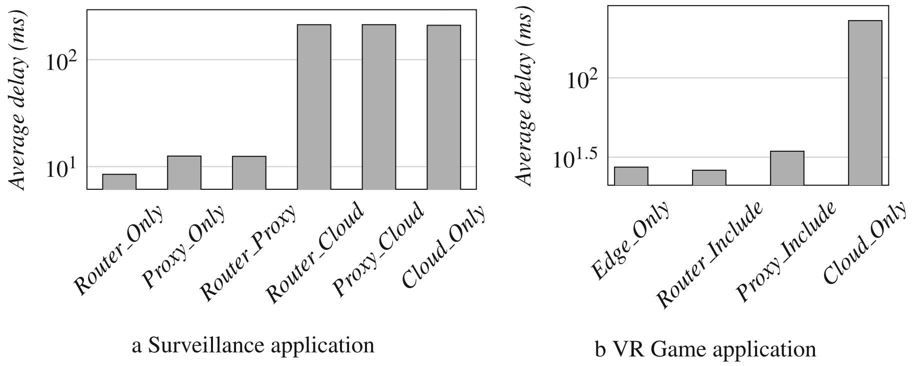


Fig. 5. Average delay of the control loop for different applications.

energy consumption in an IoT, edge, and fog environment. Nevertheless, there exist limitations in its reporting that require further research and improvement. For instance, *iFogSim* reports the energy consumption of individual devices, but it does not provide a realistic estimation of end-to-end energy consumption.

The simple addition of energy from all devices does not accurately represent the end-to-end energy consumption. To obtain a more accurate estimation, we must account for the energy consumed by the links and connections between fog devices. Unfortunately, *iFogSim* does not consider this energy consumption. Additionally, the tool does not account for the application perspective when reporting energy consumption.

5 Related Work

Different studies utilize the *iFogSim* simulator to model fog computing architectures and IoT environments. For example, in [5], the authors employed fog computing architecture for a car parking application. They performed their simulation using *iFogSim* to show the effectiveness of fog computing compared to a traditional, centralized cloud computing architecture in terms of latency and network usage. Similarly, in [11], the authors proposed a fog computing-based architecture for an e-health care application. The work proposed a three-tier structure for remote pain monitoring systems, in which fog nodes reside in the middle tier. They modeled this architecture using *iFogSim* and performed their simulations on different scales to validate that their proposed architecture can reduce execution cost, latency, and network usage, to overcome the inadequacies of cloud computing for this application. The authors of [3] performed a comparative analysis of cloud and fog environments using *iFogSim* to evaluate network usage and cost of execution. Rather than the traditional cloud processing approach, they offloaded the CCTV camera footage analysis to a fog server for processing. The research aimed to decrease execution costs and network usage by introducing a fog setup into the cloud infrastructure. The authors found that offloading computation to a fog server enables multiple users to utilize the same server for different complex tasks. None of these case studies, however, emphasized the process and analysis needed for a detailed, focused comparison of different mappings on the application's energy consumption. In our work, we investigate more scenarios, beyond edge vs. cloud, and showcase a broader spectrum of energy efficiency results.

In two other articles [21,22], the authors focus on security surveillance applications, which are widely used in our daily lives. They present a fog-based approach for such applications because the traditional cloud-based centralized approach for processing data is likely inefficient due to the latency-sensitive nature of the workload, especially as the amount of generated data from cameras increases. To compare the efficiency of fog-based and cloud-based approaches in various scenarios, including varying numbers of areas and cameras, the authors conducted several experiments using the *iFogSim* framework. They compared different parameters, such as execution cost, latency, network usage, and energy cost of these two approaches to prove the effectiveness of the fog-based approach compared to the cloud-based. However, while these studies used the same application as ours, they did not consider different mapping scenarios beyond edge and cloud devices. Instead, they focused on different topologies, with different numbers of areas and cameras for their comparison.

Research has also focused on module or service mapping within fog computing. For example, in [25], the authors proposed an improved version of the *JAYA* approach for the optimal mapping of application modules to minimize the energy consumption of fog devices. They used *iFogSim* to conduct experiments and analyze performance by varying the load of surveillance applications to demonstrate the effectiveness of their approach in reducing energy consumption. In [4], the authors presented the Heterogeneous Shortest Module First

(*HSMF*) Algorithm for mapping application modules on the heterogeneous fog-cloud computing environment. They also considered surveillance applications for their study and conducted experiments using the *iFogSim* simulator to compare the results of their proposed approach to the available cloud-only and edge-ward placement policies of *iFogSim*. They were able to improve the total execution time and total network usage with their proposed method. By comparison, our work does not focus on proposing new placement algorithms - yet; instead, we study the energy impact of different possible mapping scenarios in detail from the application perspective. This information could be used to determine new placement strategies in future work.

6 Conclusion

In this study, we investigated the effectiveness of *iFogSim*, a popular simulator for modeling and simulating fog computing environments, in analyzing the end-to-end energy consumption of applications. Our investigation includes an assessment of the energy consumption and delays across different scenarios when mapping an application on the Edge-Cloud continuum. Our experiments show that *iFogSim* can estimate the energy consumption of devices along the continuum, but its analysis is presented at device-granularity, and not from the application perspective. To model energy consumption from the application perspective, we propose only considering the energy consumption of fog devices *during their active cycle* within the application's lifetime. We added to *iFogSim* the ability to report the application's energy consumption on each device, in addition to the total energy of the device. Our results indicate the energy consumption difference, with this new analysis mode, can be significant. Furthermore, we found that *iFogSim* does not model the energy consumption of connections and links between the devices in the topology, and we currently work on modeling this energy consumption, aiming to have a more realistic estimation of energy consumption for the computing continuum.

By utilizing *iFogSim*, we have successfully evaluated different mapping scenarios, showing the energy consumption and performance trade-offs, and providing insights for optimizing energy efficiency in digital services. We plan to further work on better calibration of the simulation and validating the results in an emulation environment [13] and in a real computing continuum environment.

References

1. Aghapour, E., Sapra, D., Pimentel, A., Pathania, A.: CPU-GPU layer-switched low latency CNN inference. In: 2022 25th Euromicro Conference on Digital System Design (DSD), pp. 324–331. IEEE (2022)
2. Ahvar, E., Orgerie, A.-C., Lebre, A.: Estimating energy consumption of cloud, fog, and edge computing infrastructures. IEEE Trans. Sustain. Comput. **7**(2), 277–288 (2019)

3. Alam, M.S., Jabin, S.J., Alam, A., Hossain, M.I.: Comparative analysis of cloud and fog environment based on network usage and cost of execution using iFogSim. In: DASA 2021, pp. 132–137. IEEE, (2021)
4. Arora, U., Singh, N.: IoT application modules placement in heterogeneous fog-cloud infrastructure. Int. J. Inf. Technol. **13**(5), 1975–1982 (2021)
5. Awaisi, K.S.: Towards a fog enabled efficient car parking architecture. IEEE Access **7**, 159100–159111 (2019)
6. Brogi, A., Forti, S., Guerrero, C., Lera, I.: How to place your apps in the fog: state of the art and open challenges. Softw. Pract. Experience **50**(5), 719–740 (2020)
7. Byrne, J., et al.: A review of cloud computing simulation platforms and related environments. In: CLOSER, pp. 651–663 (2017)
8. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Experience **41**(1), 23–50 (2011)
9. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. Futur. Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
10. Gupta, H., Vahid Dastjerdi, A., Ghosh, S.K., Buyya, R.: iFogSim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Softw. Pract. Experience **47**(9), 1275–1296 (2017)
11. Hassan, S.R., Ahmad, I., Ahmad, S., Alfaify, A., Shafiq, M.: Remote pain monitoring using fog computing for e-healthcare: an efficient architecture. Sensors **20**(22), 6574 (2020)
12. Jansen, M., Al-Dulaimy, A., Papadopoulos, A. V., Trivedi, A., Iosup, A.: The SPEC-RG reference architecture for the compute continuum. In: The 23rd IEEE/ACM CCGRID 2023, India, May 1–4, 2023 (2023)
13. Jansen, M., Wagner, L., Trivedi, A., Iosup, A.: Continuum: automate infrastructure deployment and benchmarking in the compute continuum. In: FastContinuum 2023, in conjuncrtion with ICPE, Portugal (2023)
14. Jones, N., et al.: How to stop data centres from gobbling up the world's electricity. Nature **561**(7722), 163–166 (2018)
15. Kliazovich, D., Bouvry, P., Khan, S.U.: GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. J. Supercomput. **62**, 1263–1283 (2012)
16. Mahmud, R., Buyya, R.: Modelling and simulation of fog and edge computing environments using iFogSim toolkit. In: Fog and Edge Computing: Principles and Paradigms, pp. 1–35 (2019)
17. Mastenbroek, F., et al.: OpenDC 2.0: convenient modeling and simulation of emerging technologies in cloud datacenters. In: 2021 IEEE/ACM CCGrid, pp. 455–464, USA, May 2021. IEEE Computer Society (2021)
18. Oma, R., Nakamura, S., Enokido, T., Takizawa, M.: A tree-based model of energy-efficient fog computing systems in IoT. In: Barolli, L., Javaid, N., Ikeda, M., Takizawa, M. (eds.) CISIS 2018. AISC, vol. 772, pp. 991–1001. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-93659-8_92
19. Pan, J., McElhannon, J.: Future edge cloud and edge computing for internet of things applications. IEEE Internet Things J. **5**(1), 439–449 (2017)
20. Preist, C., Schien, D., Shabajee, P., Wood, S., Hodgson, C.: Analyzing end-to-end energy consumption for digital services. Computer **47**(5), 92–95 (2014)

21. Sarkar, I., Kumar, S.: Fog computing based intelligent security surveillance using PTZ controller camera. In: 10th International Conference on ICCCNT, pp. 1–5. IEEE (2019)
22. Shrestha, S., Shakya, S.: A comparative performance analysis of fog-based smart surveillance system. TCSST J. **2**(02), 78–88 (2020)
23. Sonmez, C., Ozgovde, A., Ersoy, C.: EdgeCloudSim: an environment for performance evaluation of edge computing systems. Trans. Emerg. Telecommun. Technol. **29**(11), e3493 (2018)
24. Tang, W., Li, S., Rafique, W., Dou, W., Yu, S.: An offloading approach in fog computing environment. In: 2018 IEEE SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI, pp. 857–864. IEEE (2018)
25. Vadde, U., Kompalli, V.S.: Energy efficient service placement in fog computing. PeerJ Comput. Sci. **8**, e1035 (2022)