Offloading computing tasks beyond the edge: A data-driven analysis

Sadia Khizar*†, Marcelo Dias de Amorim*, and Vania Conan†
*Sorbonne Université, CNRS, LIP6, Paris, France – {sadia.khizar,marcelo.amorim}@lip6.fr
†Thales SIX GTS, Gennevilliers, France – {vania.conan}@thalesgroup.com

***Abstract*—Mobile devices have always been left out of the network infrastructure due to their limited capacities. However, it turns out that they are becoming more and more sophisticated and, above all, increasingly numerous. Given their ubiquity, they offer untapped resources to extend the computing capacity of the MEC (*multi-access edge computing*). Nevertheless, such resources vary over time due to the dynamics of the network. In this paper, we investigate how the mobility of nodes impacts the task offloading process. To this end, we use traces of actual user equipment (UE) mobility from a cellular operator. We also quantify the impact of task duration and completion delay. Our results show that mobile nodes' offloading potential beyond the edge is significant, even for short delays.**

***Index Terms*—Beyond-the-edge computing, mobility, data analysis, task offloading.**

I. Introduction

Operators seek to reduce latency while providing a high quality of service to end-users. As a result, several technologies have emerged, including mobile computing task offloading. It already stands out as a promising and low-cost solution to reduce the cellular network burden. Most researchers focus on offloading tasks from a mobile device to a local server or the network infrastructure on a remote cloud [1]–[3]. More recently, the trend is to consider multi-access edge computing (MEC) as a compelling computing facility [4]–[7].

Our approach is unconventional. It consists of offloading computing tasks from the MEC servers to the mobile devices, thus performing computation *beyond the edge* (see Fig. 1). In that way, mobile devices assist in relieving the load from the network infrastructure. They can also improve cell bandwidth and MEC server availability. As a result, it may increase overall MEC performance. Such an approach is all the more promising as the number of mobile devices is expected to reach 13.1 billion by 2023 [8]. However, mobile devices are not always used to their total capacity and remain inactive most of the time [9], [10]. Furthermore, they are becoming increasingly sophisticated (high-end processors, high storage resources, multiple sensors). Thus, MEC could benefit from the use of mobile devices as potential mobile workstations. Therefore, assigning computing tasks to mobile nodes looks promising, but many challenges still need to be overcome to confirm its viability.

In this paper, we specifically address the impact of device mobility on task execution success. We seek to determine to what extent the untapped resources of end-user devices could contribute to offloading computing tasks from the MEC servers. In other words, we evaluate the capacity that mobile nodes can provide when the MEC server is about to offload a computing task.

ISBN 978-3-903176-42-3

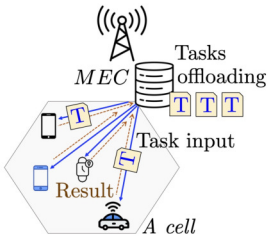


Fig. 1: Computing offloading tasks to mobile devices.

We characterize a task as a *duration*, i.e., the time it takes for a mobile node to complete the task's execution. We further consider that a mobile device must complete the task within some given time frame, which we call *completion delay*. To assess the success of task execution, we compare the task duration with the time the node is present in the cell, allowing us to obtain the *potential* number of nodes that would be able to execute the task. We extend our research to analyze cell dynamics according to node presence frequency in a cell. In our study, we use real mobility traces to assess available offloading resources.

Our analyses provide an assessment of node mobility's impact on the task offloading potential of a cell. On the one hand, the completion delay impact shows that the more significant the completion delay, the more possibilities to execute a task. On the other hand, continuously increasing the completion delay is unnecessary because the gains become negligible after a certain point. Thus, depending on how fast the MEC needs to execute tasks, it can identify the maximum number of tasks that mobile nodes can perform. Our other results indicate that node mobility is highly heterogeneous. We find sustainable nodes and intermittent nodes. These different types of nodes impact the task offloading strategy. Even the nodes that repeatedly enter and exit a cell can contribute to the MEC task offloading.

The paper is structured as follows. In Section II, we introduce our model. We present the cumulative presence time of nodes in a cell as a metric for estimating a cell's offloading potential. We also introduce the completion delay and explain its impacts on task execution through an example.

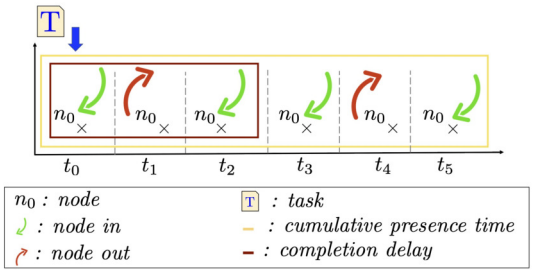


Fig. 2: Cumulative presence time determination.

In Section IV, we present the datasets. In Section III, we investigate the number of nodes that may carry out a given task successfully by using a real-world cellular dataset. We vary the completion delay to observe its impact on the offloading potential. Then, we analyze cell dynamics according to the frequency of the presence of nodes. In this work, we focus on nodes that come and go from the cell to understand how much MEC can rely on them. We end with a conclusion and perspectives of future work in Section V.

II. Offloading strategy and assumptions

We consider a scenario where the MEC server offloads tasks to mobile nodes. We model the allocation process of computing tasks to mobile devices as follows. Firstly, we assume that a MEC server can only offload a task to mobile devices in its cell. Indeed, this strategy excludes the communication cost and management overheads among base stations. Secondly, mobile nodes process the task in the offloader's cell. When the node leaves this cell, the mobile node suspends the task to make its computing resources available to the other MEC servers. When the node returns to the cell that initially assigned the task, it resumes execution to give the result back to the original MEC server. At this point, mobile nodes' continuous presence in the cell is unnecessary to complete the assigned task's execution as long as the node returns to the original cell in the allotted time. We define a task *completion delay* to specify the maximal period during which a node must deliver back the task result to the MEC server. Indeed, the node's cumulative presence time in the cell must be higher or equal to the task's computation time to consider a successful execution.

We note T the time it takes to execute a task and t the moment the task is assigned to a mobile node n. The node completes the process only if its cumulative time of presence in the cell is at least T. We consider a discrete time and define $\hat{n}(t)$ =1 if the node is present in the cell at the time t and $\hat{n}(t)$ =0 otherwise. We thus define the cumulative time of presence of node n over period $[t_a, t_b]$ as:

$$\Delta_{t_a \to t_b}(n) = \sum_{i=t_a}^{t_b} \hat{n}(i). \quad (1)$$

We note the completion delay as d_c. Thus, we consider that node n successfully completes the task if:

$$T \leq \Delta_{t \to t+d_c}(n). \quad (2)$$

TABLE I: Total number of users between 11 a.m to 2 p.m.

Cell 1	Cell 2	Cell 3	Cell 4
659	13,571	10,076	4,890

In Fig. 2, we represent node n_0 coming in and going out of a cell during a given period $[t_0, t_5]$. For example, assume that the MEC server offloads n_0 a task requiring a computing interval time of $T = 1$ and a completion delay fixed to $d_c = 3$. During the period $[t_0, t_5]$, n_0 has a cumulative presence time of $\Delta_{t_0 \to t_5}(n_0) = 4$; whereas the cumulative time of presence of the node in the specified period is equal to $\Delta_{t_0 \to t_3}(n_0) = 2$. Hence, we obtain $1 \leq 2 \leq 3$. In this example, we conclude that it would be a potential success if the MEC server had assigned the task. Using the following example, if the task computation time requires $T = 4$ and $d_c = 5$, the cumulative presence time would be $\Delta_{t_0 \to t_4}(n_0) = 3$. We deduce that this node could not perform such a task. As mentioned previously, when the MEC server sends computational tasks to a node at a time t, it requires that the node must execute the task within the completion delay and in the MEC server cell that assigned the task. Thus, nodes receiving tasks from the MEC server must return to their cell by the time allowed for task completion; otherwise, it is considered lost. In that case, a remote cloud or the MEC itself will perform the task.

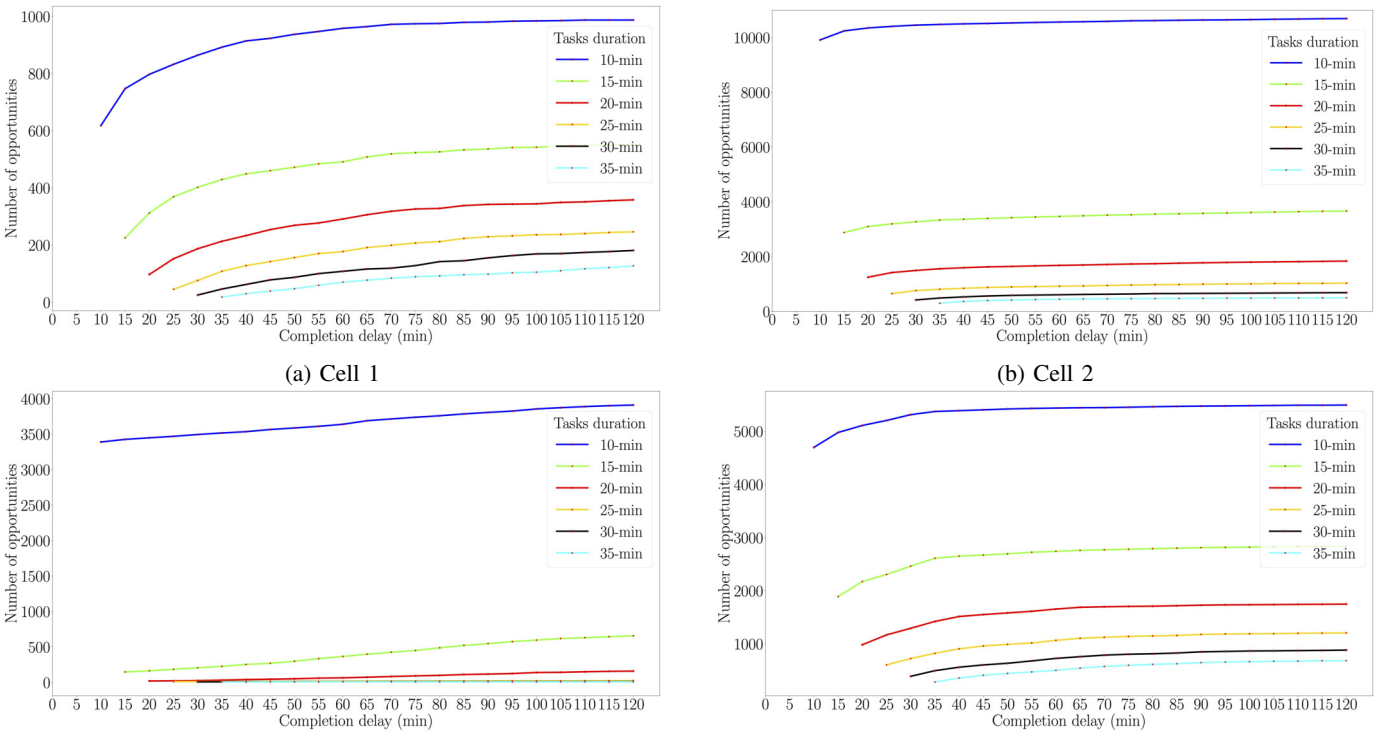
III. Dataset

We use a cellular dataset collected in a major European mobile operator's core network to analyze pedestrian scenarios in our analyses. The dataset describes the mobile traffic generated by all subscribers in Paris [11]. We conduct the study of cumulative presence time and mobility dynamics for four cells between 11 a.m and 2 p.m (i.e., 3 hours) on a weekday. We discretize the time into 5-minute intervals. To allow for a more accurate understanding of our results, we present in Table I the number of nodes in each cell during the given period. The cells are:

- Cell 1 is located in a residential neighborhood in the downtown district of Parmentier. Cell 1 has 659 nodes.
- Cell 2 is located near Gare du Nord, the largest railway station in Paris and a major transportation hub for both commuters and distant travelers. The cell has 13,571 nodes.
- Cell 3 is located in the Havre-Caumartin district. The cell contains an underground metro station leading to the Galleries Lafayette store. The number of nodes is 10,076.
- Cell 4 is located in a commercial district near the Saint Lazare train station. It is also a busy district with employees having their lunch break and visitors doing their shopping. The cell contains 4,890 nodes during the observation period.

IV. Data-driven analyses

This section investigates the impact of the completion delay on task offloading opportunities and the dynamics of nodes' mobility. Firstly, we take a look at the maximum computational capacity that each of the cells can offer. For that, we



(a) Cell 1

(b) Cell 2

(c) Cell 3

(d) Cell 4

Fig. 3: Impact of varying task completion delay. The plots show the number of task offloading opportunities, for the four cells, for tasks of duration ranging from 10 to 35 min, and with increasing completion delay (up to 120 min).

TABLE II: Number of possibilities for $T = 5$.

Cell 1	Cell 2	Cell 3	Cell 4
2,412	28,863	14,784	14,201

choose the shortest task that a node can execute, in our case $T = 5$ minutes. We thus obtain the number of opportunities to offload 5-minute tasks that the MEC server can offload (Table II). For $T = 5$, the MEC server has 2,412, 28,863, 14,784, 14,201 opportunities to offload this task in cell 1, cell 2, cell 3 cell 4, respectively. We observe a significant number of possibilities for the MEC server to offload a 5-minute task. Indeed, there are around twice to four times more offloading opportunities than nodes in cells. We strongly believe that there is available resource capacity to be exploited beyond the edge. Let us further examine how this capacity evolves with increasing duration tasks and the tasks that accept increasing completion delay.

A. *Impact of the completion delay*

In Fig. 3, we plot the number of tasks completed successfully when increasing the completion delay[1]. We compute each node's cumulative time presence in the cell and deduce, for a task of a given duration T, whether it can perform the task in the given period and how many such tasks it would be able to perform the task in the given period to accomplish. We consider tasks with duration T ranging from 10 to 35 minutes. We also consider completion delays from 10 to 120 minutes. We make the following three main observations.

Firstly, for the shortest task duration (10 minutes) and most extended completion delay (120 minutes), there are between 26% and 41% task offloading opportunities compared to a 5-minute task. More precisely, in cell 1, for $T = 10$ minutes, the MEC server can offload 41% of tasks corresponding to 987 possibilities instead of 2,412 possibilities for a 5-minute task. The task offloading opportunities for cell 2 correspond to 63% (10,693 tasks) less than a 5-minute task (28,863). We have fewer opportunities in this cell when the MEC server chooses to offload a 10-minute task than a 5-minute. Finally, in cell 3 and cell 4, relative to a 5-minute task, the MEC server can offload 26% and 38% of tasks corresponding to 3,910 and 5,493 possibilities. Overall, by increasing the task's duration from 5 to 10 minutes, the MEC server can still offload 41% of 10-minute tasks to the nodes. Even in a scenario using the shortest task duration (10 minutes) and completion delay (10 minutes), we obtain 22% to 34% of task offloading opportunities compared to a 5-minute task. So this confirms the approach showing there are indeed many opportunities for task offloading.

Secondly, when the task's further duration increases, there is a sharp drop in task offloading opportunities for the four cells. For the most extended completion delay (120 minutes) and tasks between 15 and 35 minutes, we have between 22,8%

[1]Note that the y-axis graduations are not identical to ensure clear readability of the curves.

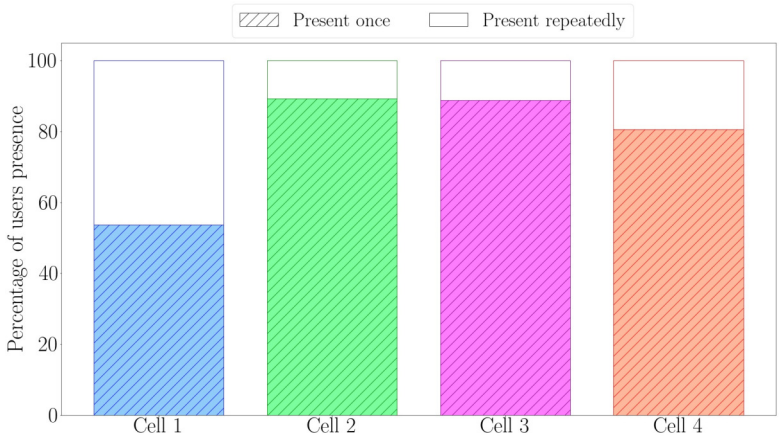


Fig. 4: Dynamics of the presence of nodes, those present once in the cell and those returning to the cell (present repeatedly).

to 5% opportunities to offload tasks in cell 1 compared to a 5-minute task. For cell 2, the MEC server can offload between 12,6% and 1,7% for tasks of 15 to 30 minutes. Thus, we notice that cell 2 has rare options for tasks up to 35 minutes. In cell 3, the MEC server has 4,4% for $T = 15$ minutes and almost zero opportunities for $T = 35$ minutes compared to $T = 5$ minutes. There are between 19,96% to 4,8% of task offloading opportunities in cell 4 compared to a 5-minute task. On the whole, we observe a decrease in the task offloading opportunities for the MEC server for a 15 to 30 minutes task. Indeed, we have between 77% to 95,6% task offloading opportunities compared to the initial offloading capacity. However, it is still possible for cell 1 and cell 4 to offload tasks of very long durations (up to 35 minutes).

Thirdly, increasing the completion delay continually improves the number of task offloading opportunities. However, there are nevertheless some discrepancies depending on the cells. Let us observe these discrepancies for $T = 10$ minutes. In cell 1, we observe that it is possible to run between 615 (25%) to 987 (41%) by increasing the completion delay value from 10 minutes to 120 minutes. Thus, we get additional 16% opportunities to perform the task. Moreover, we reach a certain time plateau at a given time where the completion time has a minor impact. This time plateau is visible after 105 minutes for cell 1. For cell 2, a significant number of nodes can handle a 10-minute task. There are between 9,914 (34%) and 10,693 (37%) task offloading opportunities—nevertheless, completion delay has a minor impact of only 3% additional possibilities. Thus, it is clear that the increase in completion time has a minor effect and, for others, a more significant impact, such as cell 1 and cell 4. We also notice that there is the appearance of a ”plateau” at a certain point in time. It corresponds to when the completion delay’s impact becomes negligible, and the number of opportunities becomes almost constant.

B. Understanding the dynamics of cells

To understand the differences in task offloading opportunities that we have observed among the four cells, we take a deeper look at the mobility of nodes in the cells. In Fig. 4 we

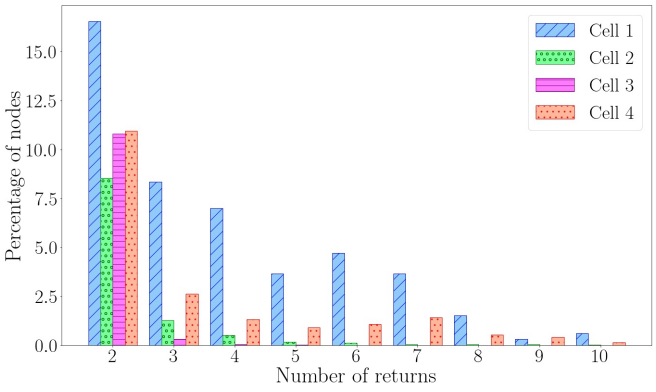


Fig. 5: Number of returns of a node to a cell.

categorize, for each of the four cells, the mobility of nodes in 2 different categories, depending on whether a node:

- Is present once. In this case, it corresponds to the nodes that appear only once during the considered period in a consecutive way. So they are present for at least one or more consecutive intervals of time. We call this kind of node *sustainable nodes*. We illustrate them in Fig. 4 by hatched lines. These nodes are the ones to which the MEC server can offload tasks by choosing the moments of their continuous presence in the cell.
- is present repeatedly. We refer to the nodes that return several times to the cell as *intermittent nodes*. When they come in and go out of they can stay for one or several time intervals. They are also present at least for two intervals. Given the volatile nature of intermittent nodes, it is difficult for the MEC server to assign a task directly. Therefore, we will also characterize these nodes’ behavior to estimate how much the MEC server can rely on them to accomplish tasks.

We observe that most often, nodes only appear once. It is pronounced for cells 2, 3, and 4. Indeed, for cell 1, we have 53.71% of nodes appearing only once. On the other hand, there are 46.29% nodes that make multiple journeys in the cell 1. In cell 2, we have a significantly higher percentage of

nodes present once 89.30%. Almost the same configuration is visible for cell 3. Regarding cell 4, it also contains more nodes that appear once (80,61%). The time of day and behavior of users can explain that there is a majority of nodes that appear only once. In fact, between 11 a.m and 2 p.m, users tend to go to their workplace to start the day at noon or have lunch. Indeed, cell 2 and cell 3 contain the most number of nodes appearing only once since these cells correspond to transport access locations. The length of the observation windows can also explain the result. Indeed, regular working hours start between 7 a.m to 9 a.m and end between 5 p.m to 7 p.m. or for some people between 6 a.m to 2 p.m and 12 p.m to 8 p.m. However, we observe the mobility from 11 a.m to 2 p.m. Thus we may see nodes only once, either because of their working hours - the behavior of mobility and the selected time window does not cover the whole day.

We observe that few nodes do not return to the cell. Even though few nodes are returning to the cells, we observe a different distribution between the cells. There are more nodes in cell 1 and cell 4. However, they are and almost identically distributed for cells 3 and 4. We have 46.29% intermittent in cell 1. For cell 2, there is 10.70% that come and go several times. Nodes percentage that makes multiple journeys in the cell 3 are almost the same of the cell 2 11.15%. Regarding cell 4, there are more intermittent nodes (19.40%) than the two last cells. We notice that cell 1 contains many nodes returning, followed by cell 4. It helps explain why task offloading opportunities grow significantly with increased completion delay in that case 3.

Fig. 5 represents the distribution of node returns to the cell. It enables us to know how many times a node returns to the cell to contribute to task offloading possibilities. We emphasize that some nodes come back to the cell up to 10 times in the 3 hours. It implies that they are at least capable of running a 50 minutes task. As expected, cell 1 has the highest percentage of incoming nodes, followed by cell 4. From 9% to 17% of nodes return in the cells at least twice, thus offering the possibility of performing a 10 minutes task. According to Fig. 4, cell 2 and cell 3 have almost the same percentage of intermittent nodes (about 11%). However, the pattern is different. For cell 3, nearly all nodes are making round trips twice, offering a possibility to offload a task of at least 10 minutes. Concerning cell 2, the frequency of presence of the nodes goes up to 9 round trips. Thus they can offer at least 45 minutes of calculations. Moreover, looking at the distribution of node returns to the cell, we can see the impact on performance with increased task duration. It is particularly true for cell 1 and cell 4, although at a smaller scale, as we have already assumed.

V. Conclusion and perspectives

Our approach consists of using mobile nodes on behalf of the MEC to multiply its computing possibilities. We adopt a data-driven approach to evaluate the potential of offloading computing tasks on mobile nodes. For this purpose, we analyzed the impact of the number of possibilities to offload computational tasks by varying the completion delay. The greater the MEC's tolerance, the more opportunities we have until a specific moment where the impact is minor. We analyze the nodes' dynamics in cells, considering using mobile nodes as much as possible, whether sustainable or intermittent. We continue the study of intermittent nodes by looking their the distribution of returning in cells. MEC intends to take more profits, the risk of assigning the task to a non-returning node and thus losing the task. We will extend our work by analyzing intermittent nodes and sustainable nodes' presence time in future works. We plan to evaluate whether they form a pool of resources persisting over time and regularly. Obtaining a pattern with the number of nodes available over time would allow us to acquire reliable resources to offload tasks. In our future work, we will also support the idea of investigating the battery capacity of the devices and their computing and storage capacity to obtain general modeling of the resources available beyond the edge.

VI. Acknowledgements

This work was partially supported by the ANR CANCAN project (ANR-18-CE25-0011). We thank Chi-Dung Phung from Cnam, Cezary Ziemlicki, and Zbigniew Smoreda from Orange Labs for their data collection support.

References

[1] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 1285–1293.

[2] Y. Liu, M. J. Lee, and Y. Zheng, "Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2398–2410, 2016.

[3] H. Peng, W.-S. Wen, M.-L. Tseng, and L.-L. Li, "Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment," *Applied Soft Computing*, vol. 80, pp. 534–545, 2019.

[4] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[5] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, October 2016.

[6] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, and R. Wang, "User mobility aware task assignment for mobile edge computing," *Future Gener. Comput. Syst.*, vol. 85, pp. 1–8, 2018.

[7] B. Yang, O. Fagbohungbe, X. Cao, C. Yuen, L. Qian, D. Niyato, and Y. Zhang, "A joint energy and latency framework for transfer learning over 5g industrial edge networks," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2021.

[8] Cisco, "Cisco annual internet report," https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, 2018–2023.

[9] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*. USA: USENIX Association, 2010.

[10] J. Haj-Yahya, Y. Sazeides, M. Alser, E. Rotem, and O. Mutlu, "Techniques for reducing the connected-standby energy consumption of mobile devices," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 623–636.

[11] "The cancan projet - content and context based adaptation in mobile networks." [Online]. Available: https://cancan.roc.cnam.fr/