Scheduling the Execution of Tasks at the Edge

Kostas Kolomvatsos
Dept. of Informatics and Telecommunications, University of Athens, Athens 15784, Greece &
Dept. of Computer Science, University of Thessaly
Lamia 35100, Greece
kolomvatsos@cs.uth.gr

Thanasis Loukopoulos
Dept. of Biomedical Informatics
University of Thessaly
Lamia 35100, Greece
luke@dib.uth.gr

***Abstract*— The Internet of Things provides a huge infrastructure where numerous devices produce, collect and process data. These data are the basis for offering analytics to support novel applications. The processing of huge volumes of data is a demanding process, thus, the power of Cloud is already utilized. However, latency, privacy and the drawbacks of this centralized approach became the motivation for the emerge of edge computing. In edge computing, data could be processed at the edge of the network; at the IoT nodes to deliver immediate results. Due to the limited resources of IoT nodes, it is not possible to have a high number of demanding tasks locally executed to support applications. In this paper, we propose a scheme for selecting the most significant tasks to be executed at the edge while the remaining are transferred into the Cloud. Our distributed scheme focuses on mobile IoT nodes and provides a decision making mechanism and an optimization module for delivering the tasks that will be executed locally. We take into consideration multiple characteristics of tasks and optimize the final decision. With our mechanism, IoT nodes can be adapted to, possibly, unknown environments evolving their decision making. We evaluate the proposed scheme through a high number of simulations and give numerical results.**

Keywords—IoT, Edge Computing, Energy Requirements, Task Execution

I. INTRODUCTION

Internet of Things (IoT) offers a vast infrastructure where numerous devices produce and share data. The dynamic nature of IoT built on top of the autonomous nature of the IoT nodes while the heterogeneity of data requires novel processing models for knowledge production. In legacy systems, data are transferred to Cloud for further processing to derive knowledge and support intelligent analytics. In any case, powerful processing mechanisms and efficient data management techniques are necessary due to the huge volumes of data. Large scale data are difficult to be processed no matter the power of Cloud. Processing data in Cloud may involve a distant datacenter, thus, increased communication costs with negative effects in the support of applications (e.g., increased latency). In addition, Cloud applications do not take into consideration possible mobility of IoT nodes that impose the management of the spatio-temporal aspect of the collected data. Full centralization may disregard significant issues like privacy, heterogeneity, lack of control in the data processing, increased latency and so on. Researchers have concluded that processing the data at the edge of the network offers multiple advantages avoiding the aforementioned problems [4]. With the advent of edge and fog computing, data can processed at the edge of the network limiting the latency for delivering analytics.

In edge computing, numerous human-controlled devices form the edge like PCs, tablets, smartphones, sensors or nano datacenters [8]. They are employed to collect and, in parallel, process the data. Edge-centric computing aims at a distributed model that interconnects heterogenenous resources controlled by various entities (i.e., nodes) to the fog and, accordingly, to Cloud. Fog computing is the next step, one hop away for the data production and the aforementioned pre-processing model. In fog computing, nodes communicate with cloudlets and cloudlets communicate with Cloud to realize the data processing. Both edge and fog computing aim to keep multiple processing tasks close to the nodes. This way, we can easily handle the heterogeneity of the nodes when supporting applications. In addition, mobile edge computing [10] aims to offer an architecture that enables Cloud computing and an IT service environment at the edge of the cellular network. Mobile edge computing offers key emerging technologies of cellular networks together with virtualization capabilities. *Network Function Virtualisation* (NFV) and *Software Defined Networking* (SDN) capabilities offer many advantages in the management of resources and services.

For supporting applications, IoT nodes should conclude a set of tasks. Actually, tasks are generated, possibly at high rates, at the nodes and should be concluded immediately. Tasks are related to processing needs defined by IoT nodes' environment. Each task could be separated into a set of sub-tasks. Without loss of generality, when we refer to the term 'task', we focus on the execution either of a generic task or a sub-task. In the relevant literature, task allocation/scheduling originates in the management of a group of nodes. The allocation of tasks is adopted to result the assignment of each task to a node while task scheduling aims to schedule the location and the sequence of execution for each task. The aim is to maximize the performance and minimize the energy consumption, thus, maximizing the lifetime of the network. Multiple research efforts deal with centralized approaches, thus, the allocation/scheduling models suffers from the aforementioned drawbacks reported for Cloud computing. In this paper, we build on the autonomous nature of nodes focusing on mobile devices and propose a distributed scheme for the allocation/scheduling of the incoming tasks. However, in contrast to other research efforts, we consider that each task will be executed either by the node itself or in the Cloud. In the

Cloud, a set of virtualized resources/services could be present that fulfill tasks requirements. Our scheme could be easily connected with a cloudlet/Cloud infrastructure where tasks could be assigned to. In the literature, one can observe various efforts for providing virtualized functionalities. A framework that offers experimentation capabilities on top of virtualized resources is SoftFIRE (https://www.softfire.eu). Our research subject is to separate the incoming tasks in two sets: those that will be executed locally and those that will be transferred to the Cloud.

We consider that nodes do not share any tasks with their counterparts in the network, thus, coordination and communication costs are eliminated. Each node adopts a decision making scheme that derives if a task should be executed locally or not. For instance, if a node sees that a task requires increased computational resources and energy could be transferred to Cloud. However, if this task is significant for supporting applications or requires limited time for getting the response, nodes may decide to execute it even if it will deplete its resources. Due to energy constraints, each node could execute a limited number of tasks, thus, it selects those that maximize the performance. The performance is maximized when nodes efficiently support IoT applications and save energy at the same time. We propose a scheme that consists of two modules: (a) a pre-processing scheme (a decision making model) for delivering the significance of each task based on the known *Analytical Hierarchy Process* (AHP) and an aggregation function; (b) a selection scheme for delivering the tasks that will be executed locally or not. The latter module is based on the solution of the known *Knapsack problem* [5] and aims to select the appropriate tasks under the fulfillment of energy constraints while, in parallel, incorporating the significance of each task. Our scheme transforms the IoT nodes to be evolved in, possibly, unknown and dynamic environments where various processing tasks demand for execution. This way, nodes can be adapted in their environment (through the required tasks) as they decide the execution of tasks based on multiple criteria.

The paper is organized as follows. Section II reports on the related work while Section III presents our scheme. In Section IV, we describe the proposed modules and argue for their adoption in tasks selection. In Section V, we provide the experimental evaluation of the proposed scheme and in Section VI, we conclude our paper discussing our future research plans.

II. Related Work

Edge computing aims to provide an intermediate level of processing to alleviate the burden of the Cloud infrastructure. Edge computing offers (near) real time processing of the collected data just before they are sent into Cloud for further processing. The efficient processing at the edge is secured through the adoption of smart gateways [1] and micro datacenters [9]. Unfortunately, due to the limited computational resources of IoT nodes, it is not possible for them to execute any task defined by users or applications. There are multiple tasks that demand for increased computational resources, thus, they can only be executed in Cloud.

Tasks allocation / scheduling is a subject widely studied in the research community. Task scheduling is studied for the allocation of various tasks in *Wireless Sensors Networks* (WSNs). In [16], the authors present an algorithm for task mapping and scheduling taking into consideration the energy constraints. The proposed solution adopts channels modelling, concurrent task mapping, communication and computation scheduling. This effort manages the nodes as a group. The focus of [2] is on collaborative processing among multiple nodes. The proposed algorithm adopts a linear task clustering and a node assignment mechanism based on task duplication and migration schemes. The algorithm concurrently allocates task and communication events for reducing the communication cost. In [7], a model for minimizing the execution time is described. The proposed optimal task scheduling is applied in a clustered network. Two phases are adopted: the intra-cluster and the inter-cluster scheduling. The focus of the algorithm is on the elimination of the transmission collisions. In [3], the authors propose a model for allocating the incoming tasks in multiple sensors according the energy requirements. The focus is on a task pre-processor and a scheduler that allocates the tasks in the available nodes. The pre-processor tries to identify the energy requirements of the incoming tasks and based on energy monitoring activities, it decides the final scheduling. The scheduler allocates the best available node to the incoming tasks while meeting its *Quality of Service* (QoS) requirements.

A task may consist of a set of sub-tasks. The efficient combination and execution of the sub-tasks will lead to the efficient completion of the initial, more generic, task. In [17], a scheduling algorithm for the available sub-tasks is proposed. The aim is to assign each sub-task into the available nodes for maximizing the performance. For each assignment, the algorithm takes into consideration energy constraints, the compatibility of sub-tasks to nodes and the topology of the network. The provided solution maximizes network lifetime and satisfies the constraints of the concluded assignments. Zenith [18] proposes a methodology for resource allocation in a set of small-scale micro-datacenters that represent ad-hoc and distributed collection of a computing infrastructure. Zenith allows service providers to establish resource sharing contracts with edge infrastructure. Service providers enjoy latency-aware scheduling provisions securing that the assigned task will meet the latency requirements.

Task scheduling heavily depends on the application domain. For instance, IoT and Cloud define different requirements in task scheduling due to the different nature of the application domain. Cloud mainly offers the available services on demand while IoT may involve applications where IoT nodes push their data and knowledge. A review on scheduling algorithms that fit on both the Cloud and IoT are discussed in [15]. Task scheduling is more important in IoT and edge computing as IoT nodes are characterized by limited computational capabilities. In addition, energy constraints define more requirements for scheduling. Cloud could serve as the intermediate between the IoT devices and applications exhibiting a vast infrastructure where increased computational (possibly virtualized) capabilities are available for processing. In [13], the authors propose a model for data distribution on a

set of IoT devices. The model aims to eliminate bandwidth and storage constraints. The assigned tasks are executed on top of a specific amount of data while only one task is executed in each machine. However, this could not be the usual case when we consider real applications. Simulated annealing to solve the problem of the adaptation of scheduling optimization techniques in the load of computing tasks in Cloud applications [12]. The model tries to build on the parallelization of allocating various tasks in a multi-cloud system. Another tasks allocation scheme for Cloud is presented in [20]. The provided model takes into consideration both task and nodes diversity. Workloads are clustered into different classes with the same characteristics through the adoption of the K-means algorithm. On top of the K-means results, the proposed algorithm sets the number of the necessary nodes for executing the tasks, thus, it saves energy in the network. In [11], the authors propose a QoS aware resource scheduling algorithm. In this algorithm, *Particle Swarm Optimization* (PSO) is adopted to derive the final scheduling. The aim is to reduce time and ensure the load balancing in order to maximize the performance.

III. Problem Description and Preliminaries

Without loss of generality, we consider that a set of nodes $N=\{n_1, n_2, \ldots\}$ are available. Each node n_i is capable of executing a set of tasks $E=\{\varepsilon_1, \varepsilon_2, ...\}$. Tasks are characterized by a set of characteristics $C=\{c_1, c_2, \ldots, c_{|C|}\}$. Example characteristics could be: the size, the type, energy requirements, the deadline (if present) and so on. Energy requirements could be based not only on the size but also on other parameters like the complexity of the task that imposes the number of computations to conclude it.

We consider an *Execution Era* (EE) that is the time interval for allocating and executing a set of tasks. Each node should separate this set into two disjoint sub-sets i.e., tasks that will be executed in the node (locally) and tasks that will be transferred for execution in Cloud. This decision is based on multiple criteria, i.e., the aforementioned characteristics and the requirement for real time results. Based on these characteristics, each node n_i should realize the significance S_j of each task. S_j depicts if a task should be executed immediately at the edge to deliver real time results. In an EE, tasks are seen as a group, thus, n_i manage them in parallel.

In unknown environments, tasks and their requirements are dynamically generated. The proposed scheme aims to support mobile IoT nodes with an adaptive scheme to be fully aligned with their internal status and tasks requirements. The proposed model optimizes the selection of tasks according to the status of each node in order to maximize the performance. Nodes are based on an adaptive and evolving mechanism that optimizes any decision during their functioning. The knowledge acquired at each EE can be easily incorporated into the knowledge base of the IoT nodes, thus, to support the evolvement of the decision making. In Fig. 1, we present the envisioned architecture for supporting each node's behavior. We focus on the allocation scheme that consists of two parts: (a) The pre-processing part responsible to define the S_j for tasks present in the current EE. The aim is to 'sort' the tasks according to their significance and their time constraints and give priority for execution to tasks that should be concluded immediately or their characteristics indicate that their execution will be efficient if it will be realized at the edge; (b) The selection part that splits the set of tasks, in the current EE, into two disjoint sub-sets, i.e., tasks that will be executed locally (at the edge) and tasks that will be transferred to Cloud for further processing.

For pre-processing, we produce two 'judgments', i.e., the tasks ranked by (a) their characteristics; (b) their deadline, i.e., the time constraint to deliver the final result. Both 'judgments' affect the S_j value, thus, we adopt an aggregation model for deriving the final S_j. For the first 'judgment', we take into consideration the entire set C and apply the widely known AHP. We aim to assign a weight in each task that will represent S_j. AHP acts on top of a set of alternative options among which the best decision is to be made. It is important to notice that, since some of the parameters / characteristics could be contrasting, it is not true that the best option is the one which optimizes each single criterion, rather the one which achieves the most suitable trade-off among the different criteria (i.e., size, type, energy requirements, etc). AHP will realize our 'view' on the significance of each characteristic that, finally, will affect S_j. The advantage is that our scheme does not require any modeling of the domain under consideration while being efficient in practice. The second 'judgment' is delivered on top of the time constraints of each task. Actually, in this 'judgment', the S_j is equal to the inverse of the deadline. The shorter the deadline is, the more significant the task becomes. However, nodes do not want to exhaust their energy reserves to service a single task that demands immediate responses if this task is not important for their functioning. We select to separate the management of the time constraints from the remaining characteristics to pay more attention on the temporal aspect of the problem and avoid to 'aggregate' it with other characteristics. This way, the proposed model will be more efficient in the decision making. It should be noted that the final aggregated S_j will be delivered by the pre-processing part that will be the basis for the final selection of the tasks that will be executed locally.

The first part of the proposed scheme will deliver an ordered set of tasks according to S_j. Afterwards, the second part (i.e., the selection process) will build on top of this set. We focus on the energy constraints in n_i aiming to derive the set of tasks that will be executed locally. A solution to the typical Knapsack problem [5] fits well in this setting. The Knapsack problem belongs to combinatorial optimization problems. Our ordered set of tasks, each of them has been assigned with a weight (energy requirements) and a value (S_j), should be processed and identify the sub-set of tasks that could be executed by n_i. Actually, we adopt the 0-1 version of the problem. The total weight (energy requirements) of the locally executed tasks should be below the capacity of the knapsack (the remaining energy resources of n_i). It should be noted that we consider nodes equipped with batteries, i.e., mobile nodes. If the battery is exhausted, no task could be executed or data could be collected. If the battery is replaced or charged, energy resources are updated and adopted in the upcoming EE.

Through the above described setting, we envision an intelligent node capable of autonomously deciding in real time, in consecutive eras, which tasks will execute at the edge to

reduce the latency and efficiently support applications. Our focus in on multiple parameters before the final decision is made. We target to optimized decisions in each EE according to the current needs. A decision could involve the absence of tasks selected for local execution to the entire set of tasks present in the specific EE. This forms a complex behavior for each node. The novelty is that we propose a sequential processing of the incoming tasks before we conclude the final decision. This sequential approach deals with an iterative processing of tasks with different techniques (a decision making and an optimization technique) to maximize the performance.

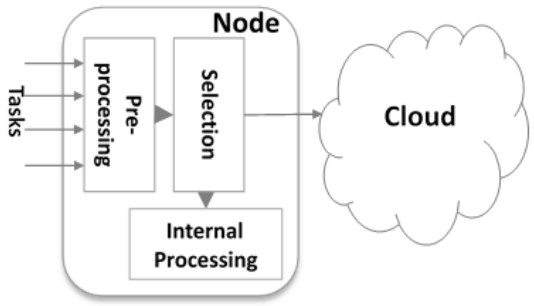

Fig. 1. The architecture of the proposed scheme.

IV. The Proposed Approach

A. The Pre-processing Module

The selection of the appropriate tasks involves strategic decision making to identify the tasks that maximize the performance at the current EE. AHP [6], [14] offers a method for decomposing the allocation problem into a hierarchy of sub-problems. Each sub-problem can be easily evaluated, thus, solved. A numerical scale is adopted to support the ranking of each alternative, i.e., the focus on various task characteristics. AHP allows users to assess the relative weight of multiple criteria or multiple options against given criteria in an intuitive manner. In case quantitative ratings are not available, policy makers or assessors can still recognize whether one criterion is more important than another. The basic process to carry out the AHP consists of the following steps:

1. The first step is to decompose the problem into a hierarchy or goal, criteria, sub-criteria and alternatives. The decision problem is separated into its constituent parts. The resulted structure comprises a goal or focus at the topmost level, criteria (and sub-criteria) at the intermediate levels, while the lowest level contains the options.
2. For each pair of criteria, we rate the relative 'priority' of every criterion against the others. An assignment of a weight between 1 (equal importance) and 9 (extreme importance) to the more important criterion is included in this step.
3. A square matrix is generated for the pair comparisons. The elements in the diagonal are set to 1. When in the cell (i,j) of the matrix it is a value over 1 means that the i^{th} criterion is better than the j^{th} criterion. The opposite stands when the cell (i,j) contains a value lower than 1.
4. We calculate the principal eigenvalue and the normalized eigenvector of the aforementioned matrix to get a relative importance of the criteria under consideration.
5. In a final step, each element's score is combined with the criterion weights to produce an overall score for each option.

In the pre-processing module, we take into consideration the entire set C providing a ranking of each c_i and create the |C|X|C| matrix. Table I presents an example matrix. In this matrix, we denote that the size of a task is five times more important than the type and so on.

Based on the aforementioned matrix, we can calculate the 'significance' S_j for the j^{th} task. S_j is calculated as follows:

$$S_j = \sum w_i \frac{A_{ij}}{\sum A_{ij}}, \begin{cases} 1 \le i \le |C| \\ 1 \le j \le |E| \end{cases} \quad (1)$$

where w_i is the weight of i^{th} characteristic derived by AHP and A_{ij} is the frequency or the value of the specific i^{th} characteristic in the tasks participating in the current EE. Frequencies are adopted in the case of nominal/ordinal characteristics while real values are considered for numeric characteristics. The pre-processing module derives the ordered set of task according to S_j as calculated by Eq(1). For each of the adopted characteristics, we define the utility that the node gains in the form of a fraction as depicted in Eq(1). The fraction indicates the amount of utility that the node gains for a specific task compared to the remaining tasks in the current EE. For instance, the higher the demand for a task is, the higher the utility becomes (as a percentage of the sum of demands for all tasks). The utility for the demand is multiplied by the weight of the demand as defined by AHP, resulting the final utility for the specific characteristic. The same rationale holds true for the remaining characteristics. The importance of each characteristic consists of a strategic decision towards paying attention on a sub-set of them. This will be depicted when the decision for executing locally the specific tasks is realized.

TABLE I. EXAMPLE OF PAIR WISE COMPARISONS.

	Size	Type	Demand
Size	1/1	5/1	1/5
Type		1/1	1/7
Demand			1/1

→

Size	Type	Demand
1/1	5/1	1/5
1/5	1/1	1/7
5/1	7/1	1/1

Let T_j be the time constraint of a task for returning the final response (i.e., deadline). T_j could have any value, e.g., it may be 'strict' meaning that the task immediately demands the final response/result or it could be more 'relaxed' meaning that the result may be delivered at any time (thus, the latency of the processing in Cloud could be acceptable). However, a task ε_j that may demand immediate response may be characterized by increased size/demand. The proposed mechanism considers the tradeoff between all these characteristics when delivering the final decision. In the pre-processing mechanism, apart from the adoption of the AHP technique, we consider a second 'judgment' of the S_j. This 'judgment' is made on top of T_j. Actually, we get $S_j=1/T_j$. The lower the T_j is, the higher the S_j becomes. This means that a

task with a limited deadline is considered as more significant compared to the remaining tasks. Let S'_j be the significance of the j^{th} task as derived by the AHP process and S''_j be the significance derived by the inverse of the time constraints. We propose the use of an aggregation function $f(S'_j, S''_j) \rightarrow \mathbb{R}^+$ to derive the final S_j. $f(\cdot)$ aggregates two different views of the system, i.e., one retrieved by the AHP process and one affected by the time required for delivering the final response. We adopt a simple, however, efficient aggregation function, the linear opinion pool. For each j, we provide the final S_j as follows: $S_j = f(S'_j, S''_j) = \alpha S'_j + (1 - \alpha) S''_j$, with $\alpha \in [0,1]$. α is adopted to enhance the effect of the AHP result or the time constraints. For instance, when $\alpha \rightarrow 0$, the proposed model pays more attention on the time constraints while the scenario where $\alpha \rightarrow 1$ pays attention on the remaining characteristics as depicted by the AHP result. The parameter α could be the result of a learning process to be adapted in the observed data. This way, the proposed model will align the effect of the AHP and time requirements as defined by each task. The learning model for α's realization is left for future work.

B. The Final Selection of Tasks

Let K tasks 'participating' in the current EE, i.e., $E=\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_K\}$. Each ε_j has a weight (energy requirements) r_j and a value (significance) S_j. r_j is derived by the size and the complexity of each task and its calculation is not a subject of this paper. The node also has a constraint related to the upper amount of energy W that could be used to support the tasks. The 0-1 Knapsack problem indicates that given a set of K tasks numbered from ε_1, τo ε_K, we should solve the following optimization problem:

$$\text{Maximize} \sum_{j=1}^{K} x_j S_j$$

$$\text{Subject to} \sum_{j=1}^{K} r_j x_j \leq W, \; x_j \in \{0,1\}$$

In the above equations x_j represents the cardinality of instances for the j^{th} task to be included in the knapsack. A solution to the problem indicates the tasks that should be executed locally. The solution could be broken into a set of true/false decisions according to whether the current task will be included in the knapsack or not. A task is included in the knapsack by checking the energy that is left over. Hence, the current decision is to check the available energy capacity or equivalently the energy requirements of the tasks already included in the knapsack. The important is that we build on top of the AHP results that indicate the significance of each task. Our optimization problem after the definition of the significance of a task, it checks which tasks could be executed based on the current energy savings. The knapsack oriented solution, it is executed in each EE to depict any updates in the energy levels. As the battery is exhausted, as natural, the tasks that will be executed locally (at the edge) will be limited.

V. Experimental Evaluation

A. Evaluation Setup and Performance Metrics

We report on the performance of the proposed scheme through a high number of simulations. For this, we define a simulator written in Java involving several classes for the implementation of, among others, the AHP and the knapsack problem modules. We emulate a real environment by setting up the involved parameters to values that depict real settings. We adopt a set of characteristics, i.e., C={complexity, type, supported application, size, energy requirements, demand}. C could be easily expanded to involve more characteristics. Without loss of generality, we do not focus on hidden correlations between the characteristics. For instance, the energy requirements of a task may be depended on the complexity and the size. Our decision has to make with the importance of each separate characteristic in nodes functioning. For each nominal/ordinal characteristic we define specific values, e.g., for the complexity, we adopt the most known complexities like the logarithmic, the linear and so on. The same rationale holds true for the remaining nominal/ordinal characteristics. For numeric characteristics, we adopt an upper value as follows. For the size of tasks, we consider that the upper size limit is equal to s_{max}=1,000KB, thus, the size of a task is realized in the interval $[1, s_{max}]$. For energy requirements, we consider that the upper limit is defined through the parameter e_{max}. In our simulations, we adopt $e_{max} \in \{10, 100\}$. The energy requirements of a task are realized in the interval $[1, e_{max}]$. When $e_{max} = 10$, we adopt tasks with low energy requirements while the opposite stands when $e_{max} = 100$. In addition, we assume that nodes are equipped with batteries. When AA batteries are used, the upper energy capacity of a node is equal to 330 mAh while lithium batteries lead to an upper capacity of 3,000 mAh. In our simulations, we consider that the capacity of nodes batteries is equal to 1,500 mAh (an 'average' value). We also consider that when the energy of the node is exhausted, an *energy depletion event* takes place. This means that the battery should be replaced or charged. When such events are realized, al the tasks will be transferred to the Cloud. The event is identified and the energy capacity of nodes is initiated again to 1,500 mAh. Additionally, with probability of 1%, we simulate replacement/recharging events at random intervals. Finally, for the demand, we define an upper value $d_{max} = 100$. The demand takes values in the interval $[0,d_{max}]$. The demand shows the rate of requests for the specific task.

We simulate 10,000 EEs. We adopt simulation scenarios for various values of |E|. Actually, we consider that $|E| \in \{10, 50, 100, 500\}$. When $|E| = 500$, we simulate the management of 5,000,000 tasks, in total. At each EE, we create |E| tasks with random values as explained above. For the selection of values, we consider the Uniform. Finally, in the AHP weights definition, we consider that energy requirements and the demand for a task are the most important characteristics compared to the remaining. Characteristics that are not important for the strategic decision making are the type and the supported applications. We also adopt $\alpha = 0.5$.

The performance of the proposed scheme is evaluated by the following metrics. We measure the throughput of each node

τ as the percentage of tasks that are executed locally (at the edge of the network). The higher the τ is, the higher the throughput becomes. We also focus on the energy spent for the execution of the selected tasks. We provide the average energy requirements γ that depicts the energy requested for each selected task. The metric γ is defined as follows:

$$\gamma = \frac{1}{|ST|} \sum_{j=1}^{|ST|} e_j \quad (2)$$

where ST is the set of the selected tasks and e_j is the required energy to conclude the task. | | depicts the cardinality of the set. The average demand δ represents the demand for a task as concluded in the performed experiments. δ is defined as follows:

$$\delta = \frac{1}{|ST|} \sum_{j=1}^{|ST|} d_j \quad (3)$$

where d_j is the demand for every selected task. Finally, we adopt ω as the number of energy depletion events in the entire set of our experiments. ω shows how many times the battery of the nodes should be replaced or charged. Actually, ω depicts the upper number of tasks that a node, exhibiting a specific energy capacity, could support. The following equation holds true:

$$\omega = | e_c < e_T | \quad (4)$$

where e_c is the current energy capacity and e_T is the energy threshold below which the battery should be replaced or charged.

B. Performance Assessment

We report on the performance of the proposed scheme through the provision of a set of tables where the realizations of the adopted metrics are presented. Initially, we report on the complexity of the proposed scheme.

The computational complexity of our scheme is affected by the complexity of the two parts, i.e., the pre-processing and the tasks selection part. In the first part, the process that contributes more in the final complexity is the calculation of the AHP weights. As the AHP involves matrix multiplication, the worst case performance is $O(|C|^3)$. The second part is mainly affected by the complexity of the optimization process. The proposed model depends on the number of tasks K, thus, the complexity is O(K). If we consider that tasks' characteristics are constant, the total complexity depends on the number of tasks in each EE.

We evaluate the proposed scheme concerning the time required to conclude the final set of tasks that will be executed locally. We run a set of 10,000 EEs and get the time for executing the AHP and the finalization of S_j for reach task (we denote this time with T_{AHP}) as well as the time required to conclude the final list of tasks concluded by the Knapsack method (we denote this time with $T_{Knapsack}$). Table II reports on our results. Our measurements are in milliseconds. We observe that the proposed mechanism is very efficient concerning the temporal aspect of the problem. Our model requires 6.8 milliseconds (approx.) to execute the Knapsack calculation (for |E| = 10,000) while the AHP and the aggregation results require at most 0.08 milliseconds (approx. - for |E| = 10,000). It should be noted that these results are the mean of the required time as observed in each EE. In total, every EE requires a conclusion time in the level of milliseconds to final the final decision.

TABLE II. RESULTS FOR THE TIME (MILLISECONDS) REQUIRED TO CONCLUDE THE FINAL SUB-SETS OF TASKS.

|E|	T_{AHP}	$T_{Knapsack}$
10	0.0035	0.0206
100	0.0088	0.1677
500	0.0373	2.4496
1000	0.0446	3.3534
10000	0.0796	6.7635

In Fig. 2, we present our results for the τ metric. We observe that the proposed scheme manages to locally execute the majority of the incoming tasks. This is more intense when e_{max} = 10. In this scenario, as the AHP pays attention on the energy requirements and the demand, the low energy requirements make the nodes capable of executing the tasks at the edge. The high energy requirements (e_{max} = 100) lead to a limited throughput especially when |E| ∈ {100, 500}. In this setting, only a few tasks are selected to be executed at the edge. This is natural, as the high energy requirements make the battery of nodes immediately exhausted.

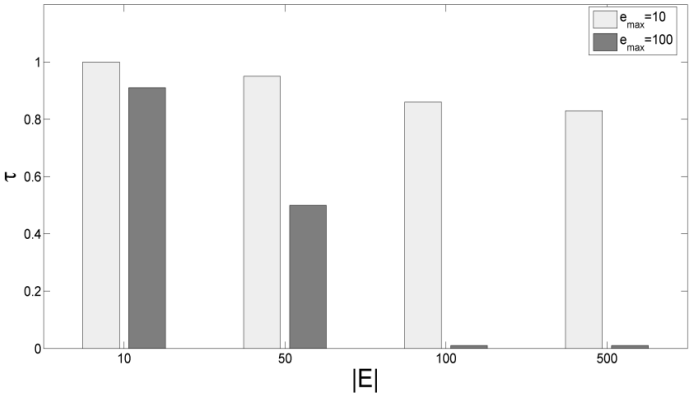

Fig. 2. Results for the τ metric.

Fig. 3 reports on our results related to the γ metric. In general, the energy requirements of the selected tasks are kept at low levels compared to the upper energy limit. This means that multiple tasks could be adopted and executed at the edge. The interesting is that as |E| increases, the energy requirements of the selected tasks decreases. As natural, the energy requirements are high when e_{max} = 100, however, they are below the median of the interval.

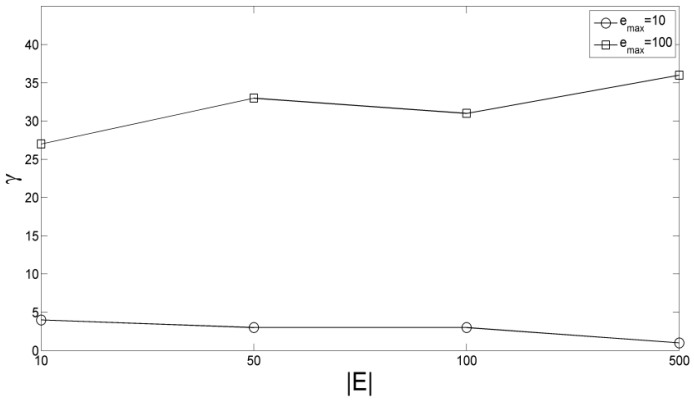

Fig. 3. Results for the γ metric.

In Fig. 4, we present our results related to the δ metric. In these results, we depict the effect of the multi criteria decision making mechanism. We observe that as e_{max} increases, the demand of the selected tasks decreases. The reason is that the proposed scheme pays attention on multiple characteristics at the same time, thus, the high energy requirements of the entire set of the incoming tasks negatively affect the remaining characteristics. The selection of tasks is mainly based on the energy requirements. Recall that the strategic decision is to consider the energy requirements together with the demand as the most important characteristics in the AHP model. However, pair comparisons between energy requirements and demand with the remaining characteristics make the energy to be the most important issue in the selection process.

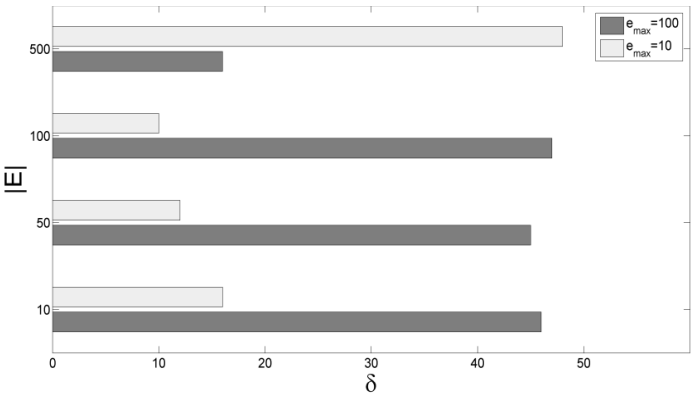

Fig. 4. Results for the τ metric.

Finally, Table III presents the number of the energy depletion events in the aforementioned evaluation scenarios. The high energy requirements negatively affect the performance as multiple energy depletion events are observed. |E| together with e_{max} negatively affect ω exhibiting an increased number of events. It is important to notice that the high energy requirements when e_{max} = 100 make the execution of tasks at the edge probably impossible due to the large amounts of energy required for their conclusion. Such a setting could not be the appropriate one to be executed by nodes with limited resources, e.g., mobile nodes, especially when a large number of tasks should be executed. Hence, the best solution when nodes are facing numerous tasks with increased energy requirements is to transfer them to Cloud.

TABLE III. RESULTS FOR THE Ω METRIC.

|E|	ω	
	e_{max}=10	e_{max}=100
10	121	833
50	454	4999
100	1428	9999
500	1666	9999

VI. CONCLUSIONS AND FUTURE DIRECTIONS

Numerous IoT nodes interact with their environment and undertake the responsibility of executing tasks for supporting applications. Nodes exhibit limited computational resources, however, they could perform simple processing tasks. They could execute tasks at the edge of the network limiting the latency in delivering analytics. A strategic decision is to select the most significant tasks to be locally executed while the remaining could be transferred to Cloud. We propose a scheme for the optimal selection of tasks based on two modules: a pre-processing module that derives the significance of each task and an optimization / selection process that delivers the final set of tasks that will be locally executed. The final decision is based on multiple characteristics of tasks to provide a more intelligent approach in the decision making. Our simulations show that when the energy requirements of the incoming tasks are low, the proposed scheme manages to execute locally the majority of them. Future extensions of our work deal with the definition of a more complex decision making mechanism that will adopt in the pre-processing module a classification model. The classification model could assign the incoming tasks to specific execution templates building on top of the experience gained in past interactions.

ACKNOWLEDGMENT

This effort is supported by the European Commission (Horizon 2020) that aims to provide research and technological development under the project ENFORCE which is part of the grant agreement no 687860 (SoftFIRE).

REFERENCES

[1] Aazam, M., Hung, P. P., Huh, E. N., ' Smart Gateway based Communication for Cloud of Things', in 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, 2014.

[2] Awadalla, M. H. A., 'Task Mapping and Scheduling in Wireless Sensor Networks', IAENG International Journal of Computer Science, vol. 440(4), 2013.

[3] Bharti, S., Pattanaik, K., 'Task Requirement Aware Pre-processing and Scheduling for IoT Sensory Environments', Ad Hoc Networks, vol. 50, 2016, pp. 102-114.

[4] Bittencourt, L., Diaz-Montes, J., Buyya, R., Rana, O., Parashar, M., 'Mobility-aware Application Scheduling in Fog Computing', IEEE Cloud Computing, March/April, 2017, pp. 34-43.

[5] Cormen, T., Leiserson, C. E., Rivest, R. L., Stein, C., 'Introduction to Algorithms', 3rd Edition, MIT Press, 2009.

[6] Dagdeviren, M., Yüksel, 'Developing a fuzzy analytic hierarchy process (AHP) model for behavior-based safety management', Information Sciences, 178(6), 2008, 1717–1733.

[7] Dai, L., Chang, Y., Shen, Z., 'An Optimal Task Scheduling Algorithm in Wireless Sensor Networks', International Journal of Computers, Communications and Control, vol. 1, 2011, pp. 101-112.

[8] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., Riviere, E., 'Edge-centric Computing: Vision and Challenges', ACM SIGCOMM Computer Communication Review, vol. 45(5), 2015.

[9] Greenberg, A., Hamilton, J., Maltz, D., Patel, 'The Cost of a Cloud: Research Problems in Data Center Networks', in ACM SIGCOMM, vol. 39(1), 2008, pp. 68-73.

[10] Huang, W., 'A Survey on Mobile Edge Computing', in 10th International Conference on Intelligent Systems and Control, 2016.

[11] Krishnapriya, S., Joby, P. P., 'QoS Aware Resource Scheduling in Internet of Things-Cloud Environment', International Journal of Scientific & Engineering Research, vol. 6(4), 2015.

[12] Moschakis, I., Karatza, H., 'Towards Scheduling for Internet-of-Things Applications on Clouds: A Simulated Annealing Approach', Concurrency and Computation, Combined Special Issues on the Internet of Things: shaping the new Internet space and advances on Cloud services and Cloud computing, vol. 27(8), 2015, pp. 1886-1899.

[13] Pasteris, S., Wang, S., Makya, C., Chan, K., Herbster, M., 'Data Distribution and Scheduling for Distributed Analytics Tasks', IEEE SWC Conference, 2017.

[14] Saaty, T.L., 'How to make a decision: The Analytic Hierarchy Process', European Journal of Operational Research, vol. 48, 1990, pp. 9-26.

[15] Shanthan, BJ., Kumar, A. D. V., Govindrajan, E., Arockian, L., 'Scheduling for Internet of Things Applications on Cloud: A Review', Imperial Journal of Interdisciplinary Research, vol. 3(1), 2017.

[16] Tian, Y., Ekici, E., Ozguner, F., 'Energy-Constrained Task Mapping and Scheduling in Wireless Sensor Networks', in IEEE International Conference on Mobile Adhoc and Sensor Systems, 2005.

[17] Voinescu, A., Tudose, D. S., Tapus, N., 'Task Scheduling in Wireless Sensor Networks', in 6th International Conference on Networking and Services, 2010.

[18] Xu, J., Palanisamy, B., Ludwig, H., Wang, Q., 'Zenith: Utility-Aware Resource Allocation for Edge Computing', in IEEE Edge, 2017.

[19] Zhang, Q., Zhani, M. F., Boutaba, R., Hellerstein, J. L., 'Dynamic Heterogeneity-Aware Resource Provision- ing in the Cloud', IEEE Transactions on Industrial Informatics, vol. 2(1), 2014.

[20] Zhang, Q., Zhani, M. F., Boutaba, R., Hellerstein, J. L., 'Dynamic Heterogeneity-Aware Resource Provision- ing in the Cloud', IEEE Transactions on Industrial Informatics, vol. 2(1), 2014.