Journal of Universal Computer Science, vol. 26, no. 11 (2020), 1455-1474
submitted: 2/9/20, accepted: 2/10/20, appeared: 28/11/20

Toward a Fuzzy-based Approach for Computational Load Offloading of IoT Devices

Lelio Campanile
(Università degli Studi della Campania "L. Vanvitelli", Caserta, Italy
lelio.campanile@unicampania.it)

Mauro Iacono
(Università degli Studi della Campania "L. Vanvitelli", Caserta, Italy
mauro.iacono@unicampania.it)

Fiammetta Marulli
(Università degli Studi della Campania "L. Vanvitelli", Caserta, Italy
fiammetta.marulli@unicampania.it)

Michele Mastroianni
(Università degli Studi della Campania "L. Vanvitelli", Caserta, Italy
michele.mastroianni@unicampania.it)

Nicola Mazzocca
(Università degli Studi di Napoli "Federico II", Napoli, Italy
nicola.mazzocca@unina.it)

Abstract: Technological development and market expansion offer an increased availability of resources and computing power on IoT nodes at affordable cost. The edge computing paradigm allows keeping locally on the edge of the network a part of computing, while keeping all advantages of the cloud and adding support for privacy, real-time and network resilience. This can be further improved in IoT applications by flexibly harvesting resources on IoT nodes, by moving part of the computing tasks related to data from the edge server to the nodes, raising the abstraction level of the data aspects of the architecture and potentially enabling larger IoT networks to be efficiently deployed and managed, in a stand-alone logic or as a component of edge architecture. Anyway, an efficient energy management mechanism is needed for battery powered IoT networks, the most flexible implementations, that dynamically balances task allocation and execution in order to In this paper we present a fuzzy logic based power management strategy for IoT subsystem that aims at maximizing the duration of the network by locally migrating part of the computing tasks between nodes. As our goal is to enable the deployment of semi-autonomic large IoT networks, our proposal does not rely on external resources for migration control and operates on a local basis to ensure scalability: at the best of our knowledge, this differentiates our proposal with respect to similar solutions available in literature.

Key Words: IoT, energy management, fuzzy logic, edge computing, IoT scalability, WSN

Category: C.2.1, C.2.4, C.3, C.4

1 Introduction

Internet of Things (IoT) is a popular technology that allows pervasive sensing by low-cost, interconnected devices. On one hand, cost of devices makes them largely available for many different applications and makes deployments that use disposable sensors viable, and, on the other hand, enable the implementation of solutions with significant on-board computing power and resources, so that both distributed sensing and processing can be performed and distributed resource control and negotiation can be directly available at the nodes. In edge computing based systems, part of the computing resources are pushed to the edge of the network, so that part of the computational logic can be executed closer to the users or the peripherals that interact with the cloud by means of edge nodes: similarly, it is possible to exploit the capabilities of IoT nodes to push further some of the computing and control logic towards the very edge of the overall infrastructure.

When IoT is used to implement sensor networks, the possibility of pushing computing and control logic to the sensors allows better flexibility. Both load balancing and energy management can be performed locally, so that larger sensor networks can be implemented that are resilient with respect to weariness, energy exhaustion, routing reconfiguration, workload variations, unexpected loss of connection to the edge server, temporary fragmentation of the sensor network. A good energy management policy and task migration features, implemented locally in the sensor network without the need for support or coordination from the edge server, can provide such resilience and robustness in case of dependability issues and maximum scalability with minimal additional network traffic.

1.1 Application context

The context in which this solution has to be considered is that of large scale monitoring systems, to implement applications that benefit of the deployment of a large number of battery-powered IoT sensors. A first example of such setups may be provided by environmental monitoring systems, based on the deployment of heterogeneous sensors in very large wild areas, such as forests, sea zones, dangerous locations (e.g. contaminated areas), where positioning and replacing defective nodes is expensive or complex and battery usage must be maximized to keep the network as much efficient as possible (see [D'Arienzo et al., 2013]). Another example is provided by emergency management systems, in which the IoT network is meant to support operations and safety of operators and other subjects may depend on the lifetime of the network, that allows a real-time situation about the target locations or physical infrastructures where the emergency situation arises, such as the case of an extended fire to be managed in a large industrial complex or in a skyscraper. This paper applies to IoT and edge computing based emergency management systems as presented in previous works [Campanile et al., 2020a], [De Arcangelis et al., 2020], [Campanile et al., 2020b] and [Cavalieri d'Oro et al., 2019].

Our reference architecture consists of two main subsystems: a field subsystem and a high-level services subsystem. The high-level services subsystem is a common cloud-based application support, running software modules that do not require real-time, such as data analysis and data mining tools, integration with other data sources, decision support, alert management, that may benefit of on-demand resources, elasticity and the other advantages provided by the cloud. The field subsystem is meant to execute data gathering and real-time or critical computing, and is in turn composed of an edge server and an IoT network: this ensures high responsiveness for essential tasks and low-level activities, but poses a management problem if IoT network resources are scheduled by the edge server, limiting its extension, and because nodes may disconnect for battery exhaustion and continuity of the network may be lost due to arising range gaps. Our proposal aims at mitigating such limitations by delegating and distributing workload management to balance energy consumption and node effort.

1.2 Our proposal in brief

In this paper we propose a fuzzy logic based solution for energy management in IoT-based sensor networks with task migration. The proposed approach acts locally on close sensors to balance energy usage by task migration and keeping migrated tasks close to the sensors on which they are originally meant to run, to minimize additional network traffic and maximize network survivability on the long term. Sensors operate on a peer-to-peer basis and compensate possible different on-board resources or available energy by sharing their resources with close nodes that are in need by making them available, on a voluntary basis, for task offloading. The approach is transparent with respect to higher level workload reconfigurations and adaptive, as the used variables for decisions are battery charge status, CPU workload, memory usage and resource requirements of the task to offload, and a fifth implicit variable, that is the distance between the sensors that are negotiating the offload operation. The implementation is kept simple, with a minimal number of member functions per variable and a low-cost defuzzification method, so that the impact on node resources is affordable. The effectiveness of the proposed solution is shown by a simulation of a proof-of-concept logical implementation that provides the energy levels evolution over time, given a network setup and a task distribution.

At the best of our knowledge, this is the first proposal of an approach of this kind, as other proposals in literature about fuzzy logic based approaches for energy management in IoT involve the edge server. We believe that this is a promising theme, worth of investigation, with potentially relevant development both on the methodological level and the application level, as it allows improving the resiliency of IoT and edge systems with a natively distrubuted and scalable approach.

This paper is organized as follows: next Section 2 presents related work; Section 3 presents the energy model for the system; Section 4 describes the design of our fuzzy approach; in Section 5 the results of a proof-of-concept simulation-based testing campaign are provided and analyzed, together with the description of the reference algorithm for offloading management; conclusions follow.

2 Related work

The amount of data consumed by mobile devices has grown exponentially, partially due to the rise of IoT. For this reason, many efforts have been directed towards methods for mitigating the congestion and strain on the network, generally by introducing various strategies of offloading, or by bringing the data and computations closer to the devices themselves through edge and fog computing.

From the infrastructural point of view, [Shu et al., 2010] proposed an adaptive connection scheme for WSN to dynamically adjust the transmission radius and data generation rate, considering the interaction among physical, network and transport layers. [Pal, 2015] adopts a resource-constraint mobile device to efficiently access cloud-based applications to extend mobile cloud platforms by opportunistic networks.

Energy efficiency is a classical limit: in [Marin and Dobre, 2013] a solution based on a contextual search among mobile devices is proposed to select an an opportunistic on-the-fly offloading execution of mobile applications in a hybrid cloud. [Yang et al., 2014] presents an experimental study on performances and energy trade-offs in mobile systems, focusing on mobile database applications, showing that the energy consumption has a dynamic connection with performances. In [Ciobanu and Dobre, 2018] the Drop Computing paradigm, a novel offloading technique, is proposed to increase processing speed, reduce deployment costs and lower mobile device battery consumption: it exploits the crowd of mobile nodes belonging to users and edge devices as opportunities to offload data and computations. Energy saving in mobile environments has been addressed also by implementing energy harvesting [Wang et al., 2017] and mobile edge computing [Feng et al., 2017] techniques, where IoT devices offload the computation tasks to edge devices such as the base stations, access points or smart devices within their reach. Exploiting edge device resources, offloading can reduce computing latency, save battery and enhance security for computing intensive IoT systems. Selection of candidate edge devices for offloading is coped in [Dinh et al., 2017], [Bi and Zhang, 2018] and [Sun et al., 2019].

Fading and interference problems in offloading connections are considered in [Wang et al., 2017]. [Mao et al., 2016] proposes a mobile offloading scheme based on Lyapunov optimization to minimize the worst-case expected computation cost. In [Xu et al., 2017] an online learning based selection method is proposed, considering energy harvesting and the power consumption model: anyway, these assumptions may not be practically applied for IoT devices, especially with multiple edge candidates. In [Min et al., 2019] a computation offloading framework based on reinforcement learning [Sutton and Barto, 2018] is proposed for IoT devices, considering harvesting models with no knowledge of the mobile edge model, computing latency and energy consumption model to select edge devices, to compute the quota of tasks to be offloaded, exploiting the idea that each offloading decision is made in accordance to the radio transmission rate of each IoT-edge link in the previous time slot, the predicted renewable energy harvesting model and the current battery level of the IoT device.

In [Xu et al., 2017] an efficient reinforcement learning based resource management algorithm is proposed to optimize the on-the-fly workload offloading rate toward cloud and edge servers to minimize service latency and operational costs. The computation offloading strategies include the application of Q-learning techniques, to derive the optimal offloading rates and to reduce the attack rate of smart attackers at IoT devices [Xiao et al., 2016], and to achieve optimal offloading and reduce response time in edge-based offloading [Zhang et al., 2016]. Q-learning, Dyna-Q and post decision state techniques are applied to malware detection offloading in order to improve the detection performance without being aware of trace generation and channel models in dynamic radio environments.

In [Ko et al., 2017] the conditions of computation offloading are explored; a workload measurement model based on DMIPS (Dhrystone Millions of Instructions per Second) is used and the energy related behaviors with or without offloading are compared. The main aim of this study consists in proposing a double computation offloading technique that offloads from wearable devices to smartphones and further offloads to cloud servers. Three experimental scenarios are proposed, to measure the energy consumption for each wearable or smart device and to analyze energy efficiency of this decision model.

As an alternative to the use of deep learning models, several fuzzy-based approaches have been introduced in recent years to improve energy management in WSN and IoT. Fuzzy logic is capable of making real-time decisions, even with incomplete information, and since fuzzy logic based systems can handle the linguistic rules in a natural way, they are particularly suitable in several contexts, such as WSNs and IoTs applications. Moreover, fuzzy systems can be used by combining different parameters and rules, that may produce an optimal result. The use of rule-based fuzzy logic controllers (FLC) enables the implementation of multicriteria control strategies. For instance, the use of smart techniques for setting and tuning FLC can improve the energy savings in a WSN. For this reason, the FLC, based on linguistic rules instead of inflexible reasoning, can be the right choice to describe a mechanism for energy saving to prolong the lifetime of network nodes [Collotta et al., 2017]. In [Li et al., 2017] an approach integrating an analytic hierarchical process with a fuzzy comprehensive evaluation method is applied to an IoT-based industrial energy management systems (EMS), aiming to fully evaluate the operational level of energy intensive equipment. The main contribution provided by this work consists in introducing a functional framework for indexing and evaluating the operational level of industrial energy-intensive equipment (e.g., manufacturing processes) in an IoT-based EMS architecture. In [Basic et al., 2019] some side effects of introducing computation offloading by using edge computing are analyzed, in particular for latency-sensitive applications and compute-intensive tasks. To ensure non-intermittent services in case of user mobility, most approaches accelerate the handoff transfer time but do not reduce its frequency. Moreover, the handoff mechanisms used in cellular networks do not consider computing workload, so cannot be applied to edge offloading. Further, considering dense edge deployment, optimal offloading edge node selection is vital: the authors proposed a fuzzy logic based selection algorithm considering bandwidth, CPU speed and latency and consider perceived response time with respect to the closest or highest bandwidth node. They also perform an evaluation of their proposed approach by offloading directed acyclic graph models of real-world mobile applications. The results they obtained seem to show that a significant reduction in the application response time and monetary cost of execution can be obtained, by controlling the number of handoffs among edge nodes.

3 Energy consumption model and offloading costs

In the field of research regarding IoT low-power devices, facing off the issues related to the energy consumption model is of paramount importance. Following [Enokido et al., 2014], in this paper the Simple Power Consumption model (SPC) is used. In the SPC model, a computing node consumes the maximum electric power $maxE$ if at least one application process is performed; otherwise, if it is in idle state, it consumes the minimum electric power $minE$ (Fig.1). Despite its simplicity, this model, due to the low number of processes running on low-power nodes, suits well to the purpose of analysis [Oma et al., 2018]. However, the total execution time is affected by the number of processes running on the same node; the consequence is that energy consumption depends on the execution time of all processes.

The Offloading costs must also be taken in account, because of the limited

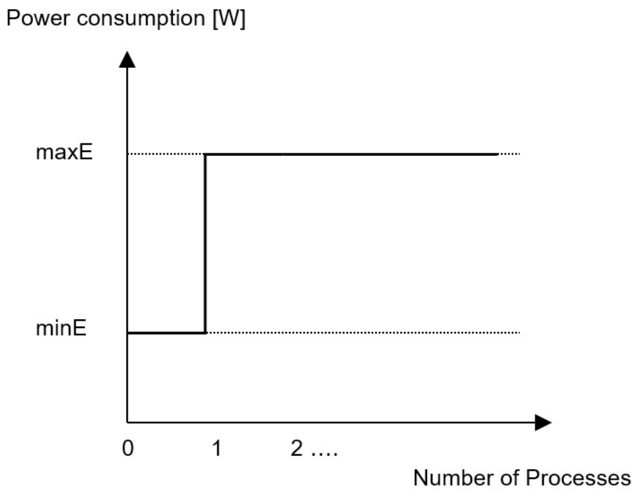

Figure 1: Simple Power Consumption (SPC) Model

battery capacity of nodes. There are two types of costs in offloading operations: Time Cost (TC) and Energy Cost (EC).

The TC relates to the time necessary to transmit the whole task from a node to another. According to [Xu et al., 2020], TC may be computed using the following formula:

$$TC = D/Rate * ZD^2 \tag{1}$$

where D is the data size of the task to migrate, $Rate$ is the data rate of the network connection, and ZD is the shortest path between transmitter and receiver nodes (ZD=1 if nodes communicate directly each other). As evident in Formula 1, TC is strongly influenced by the number of nodes to cross for the communication between the sender and receiver nodes; thus, the routing algorithm plays a primary role in order to maintain TC low.

EC represents the energy consumed by a single node because of the task offloading, and is composed by two different contributes: the energy consumed in the communication (E_k) and the surplus of energy consumed by the node due to the task which perform migration (E_t).

$$EC = E_k + E_t \tag{2}$$

E_k is computed for a single node with the following Formula [Xu et al., 2020]:

$$E_k = e_k + D/Rate * B \tag{3}$$

where e_k is the baseline energy consumption of the data transmitter and B is the power needed by the transmitting section of the network interface of the node; it is to be noted that B depends on the distance between the communication partners.

The energy associated with the task which performs migration (E_t) is:

$$E_t = maxE * TC \tag{4}$$

in which TC is described in Formula 1. Substituting as in 2, EC is computed as follows:

$$EC = e_k + D/Rate * (B + maxE * ZD^2) \tag{5}$$

Is to be also noticeable that the same amount of energy EC is consumed by: i) the transmitting node; ii) the receiving node; iii) all intermediate nodes involved in communication.

The study summarized in this section reached a result, the evaluation of the energy consumed in migration (EC, Formula 5), that will be used, when our research will come in a more mature, as a benchmark to perform the cost-benefit analysis of task migration, and also to optimize the weight used in the fuzzy logic part of the algorithm.

Formula 5 puts into evidence that the energy consumed by a single node depends on three parameters that need to be taken into account to minimize the migration EC: i) the size D of the data to be exchanged to provide migration; ii) the shortest path ZD (Formula 1) between transmitter and receiver nodes, that depends on routing algorithm; iii) the power B needed for data transmission (Formula 3), that depends on the distance between the partners of communication.

With regard to the minimization of data size, the commonly used technique for this purpose is data compression, but this technique is well-known to be very time-consuming (i.e. energy consuming, in our model) and at first glance it does not seem to be effective in terms of cost-benefit for the energy saving environment that is investigated in this paper.

The second parameter depends on the chosen routing algorithm; routing is outside the scope of this introductory paper, and will be considered in the next steps of our research.

The third parameter depends on the distance between sender and receiver node; this parameter is taken into account in the algorithm proposed in this paper.

4 Design of the Fuzzy Inference System (FIS)

A Fuzzy Inference System (FIS) is based on 4 steps, which are shown in Figure 2. The *fuzzifier* transforms crisp inputs from real world into fuzzy inputs to the *inference engine*. The inference engine interacts with inputs via *rules* in the form of *IF.... THEN...* clauses. The results of computation are transformed in crisp outputs by *defuzzifier*.

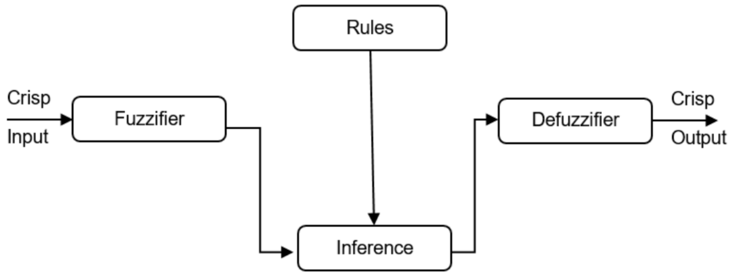

Figure 2: Structure of a Fuzzy Inference System

There are many degrees of freedom in the implementation of a FIS: the number and the shape of member functions in the fuzzification and defuzzification steps, and the type of inference model (Mamdani or Sugeno). Since our algorithm must run directly on IoT nodes, the main criteria we use to design the FIS are based on minimal battery consumption for nodes, i.e. minimal computational cost and minimal memory space occupied by the fuzzy-based algorithm. For these reasons, the member functions for the fuzzifier are all implemented using linear-shape functions (trapezoids), due to the lower computational cost than Gaussian shape functions; this might not be the most accurate solution for modeling some of the variables (e.g. battery level), but is essential to keep the computation cost as low as possible. The number of member functions for the variable representing the battery level is 4: $Idle$, Low, $Medium$, $High$. All other member functions are in number of 3 for each variable (Low, $Medium$, $High$).

Regarding the inference process, there are two main types of FIS: the Mamdani [Mamdani and Assilian, 1975] and the Sugeno [Takagi and Sugeno, 1985] types. Many authors in the past decade ([Jassbi et al., 2006], [Jassbi et al., 2007], [Hamam and Georganas, 2008] and more recent works like [Dhimish et al., 2018], [Chaudhary, 2019], [Amri et al., 2019]) had faced off the comparison of the two different types of inference systems to be used in various application domains. The differences between Mamdani ad Sugeno inference systems may be summarized in Table 1.

	Mamdani	**Sugeno**
Strenghts	Can be used with Multiple Output systems Well suited to human input Widely used in Decision Support Systems	Works well with optimization and adaptive techniques Low computational load Widely used in Control Systems Less prone to data noise
Weaknesses	High computational load More prone to data noise	Only used with Single Output systems

Table 1: Strenghts and weaknesses of Mamdani and Sugeno type of inference systems

The choice for our inference system has fallen on Sugeno type. This decision depends on its low computational load and for its reduced sensitivity to data noise. Due to this choice, there is no member function in the defuzzifier step. In case of Sugeno FIS, crisp output is obtained using a weighted average of the consequent rules [Hamam and Georganas, 2008].

Generally speaking, the weight to be used in the FIS can be optimized by using various algorithms, including artificial neural networks. Since this is an initial step in finding a full-featured effective and simple offloading algorithm, the choice is to refrain from using weights in the output computation (i.e., all rules have weight equal to 1) at this stage of our research.

4.1 Variables

The approach is based on 4 fuzzy variables plus an additional one. The 4 considered fuzzy variables are:

- battery level (in terms of percentage of residual charge of the IoT node);
- CPU workload (in terms of current average percentage of IoT node CPU load over the last epoch of the algorithm, as sensed by the hardware meters);
- memory usage (in terms of percentage of total memory currently used on the IoT node);
- resource requirements of the task to be offloaded (in terms of size in MB, from 0 to 1000 MB).

This choice is motivated on the basis of a health vs collaborativeness model: each IoT node has to balance selfishness (keeping as much as possible its own battery level high and its CPU and memory free to survive as long as possible) and collaborativeness (offering help to neighbours by trading part of its resources, to avoid becoming isolated from the rest of the sensor network because its neighbours die for battery exhaustion, or asking for help when battery is going to be exhausted soon or too fast).

Tasks belong to two sets: low level tasks, that cannot be offloaded because they manage the node or the sensing hardware or integration with the sensor network, and high level tasks, that deal with data integration, preprocessing, computing, network management or other business logic operations, that can be freely offloaded to other nodes.

The battery level variable is modeled by means of 4 member functions, namely Idle, Low, Medium and High, that are represented in Fig. 3. The CPU workload is modeled by means of 3 member functions, namely Low, Medium and High, that are represented in Fig. 4. The memory usage is modeled as well by means of 3 member functions, namely Low, Medium and High, that are represented in Fig. 5. Finally, task requirements are modeled by means of 3 member functions too, namely Low, Medium and High, that are represented in Fig. 6.

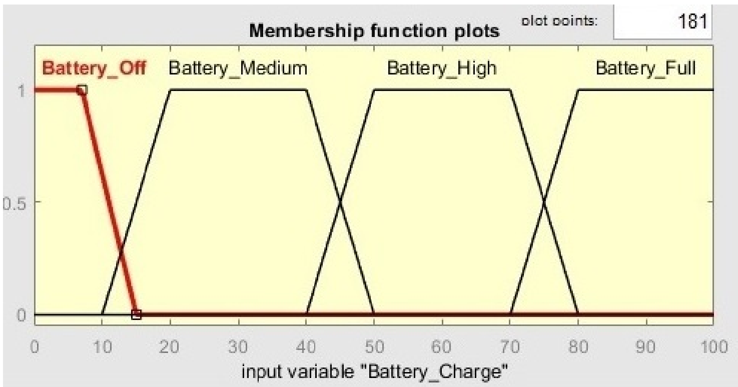

Figure 3: Member functions for battery level

4.2 System behavior

In the following, we consider all IoT nodes to be identical, with no loss of generality, for sake of clarity. We consider as reference IoT node an Arduino based unit, in order to make the scale of available resources clear.

All decisions about scheduling and offloading are taken on-line and independently by each node, according to a negotiation cycle. The offloading negotiation

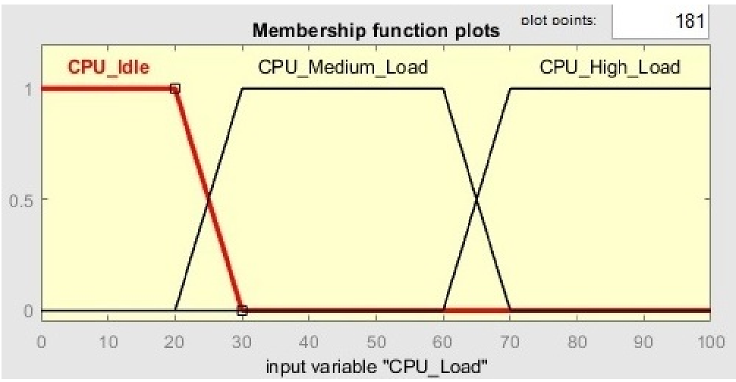

Figure 4: Member functions for CPU workload

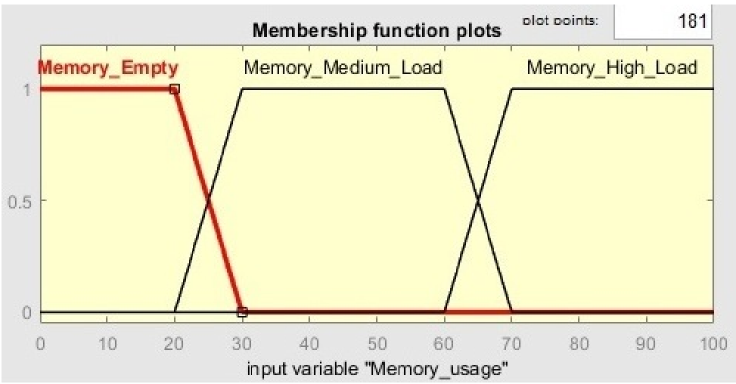

Figure 5: Member functions for memory usage

cycle is run periodically by each node, with a period that depends on the needs of the application (e.g. considering the frequency of sensing operation or the volume of produced information or the variability of the high level application running by the task spanning on the sensor network, or the frequency of high level operations ordered by the edge node to the sensor network). Each run is an epoch for the network energy management subsystem.

At the beginning of each epoch, each node evaluates:

1. if it can volunteer for offloading according to its on-board available resources and its workload;

2. if it needs to ask other nodes for offloading;

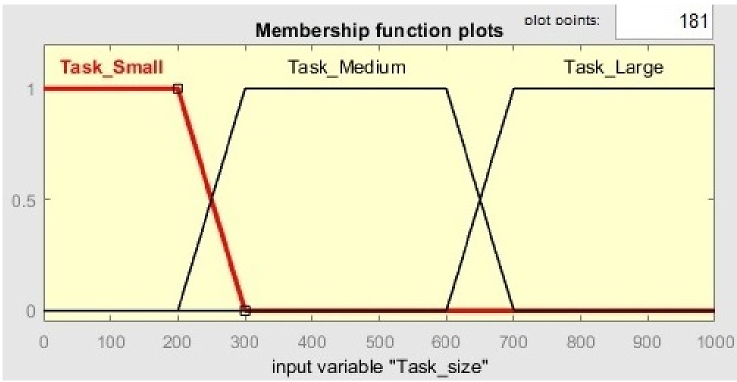

Figure 6: Member functions for task requirements

3. if in need, which of the neighbor nodes is the best candidate for offloading a task, according to the result of the cycle that they announced to the sensor network in the first step and to the additional variable, that is the distance between it and the best choice offload target neighbor node, that affects the energy needed for transmitting the task and perform further communications.

The result of the evaluation is another variable, with three possible values, namely No_Offloading, Maybe_O, OK_Offloading, that respectively stand for not available for offloading, volunteering for offloading if no other best choice, volunteering for offloading.

The resulting inference system presents 36 possible selection rules, that can be evaluated with a low computing cost with the chosen member functions (a trapezoidal shape).

5 Case study

The simulated architecture is in Fig. 7. The reference configuration of the IoT network that has been assumed for our tests is structured, for the sake of simplicity, on a grid topology, characterized by an equal distance between adjacent nodes in each row and column. Each node can directly communicate with all neighbors in each direction, including every possible direction and distance (reachable by the connection), in order to negotiate task offloads: in the figure, each arrow depicts one of the possible partners for offloading for a node chosen as example.

Each IoT node is composed of several layers: a hardware layer, including all sensors as well (*HW/Sensing*), a operating system layer, also managing networking (*OS/Networking*), an *Energy Management* layer implementing the strategy

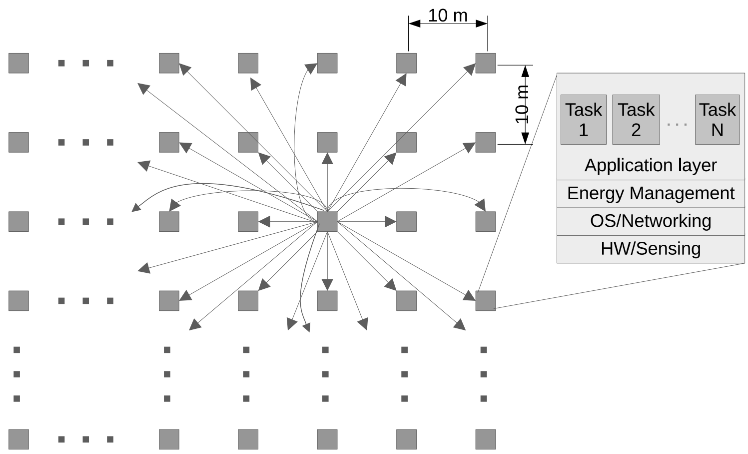

Figure 7: General network and node configuration

proposed in this paper and an *Application Layer* that basically consists of all the tasks that implement computing that may be offloaded. This organization has been designed to minimize the overhead of offloaded tasks that use local sensors, as the related communications happen between OS/Networking layers of the two involved IoT nodes, at the lowest layer possible.

In order to perform preliminary tests, a simulation campaign has been conducted in an instance of this simplified environment. The campaign must be intended as a proof-of-concept test, rather than as a test on real data, and is meant to test and tune the algorithm and to verify the rules of the FIS.

The used simulator is MATLAB, due to the simplicity of the related Fuzzy Logic Toolbox. The simulation environment is detailed in Table 2.

Simulator	MATLAB ver. 9.8 Fuzzy Logic Toolbox ver. 2.7
Hardware	Sony Vaio Ultrabook SVP123A1CM
Operating System	MS Windows 10 Enterprise 2015 LTSB Version 10.0, 64 bit
CPU	Intel Core i5 4200 CPU @ 1.60 GHz
Memory	4 GB

Table 2: Simulation environment

The test campaign for this preliminary study is implemented as follows:

– all nodes are arranged in a square shape;

– each node is located at the same distance (10m) from each row and column neighbor;

– all the tasks to be migrated are *LARGE* (1MB);

– in the startup phase, the battery level is set to the maximum charge, and the computational load and memory occupation are set to a random value to account for variability in local behavior and minor configuration and setup differences;

– at the end of each epoch, the values of the battery level is decremented by an estimated value, while computational load and memory occupation are incremented or decremented depending on the migration operations.

The test algorithm is described in pseudo-code in Listing 1.

```
Battery_Level = FULL
Computational_Load = rand()
Memory_Occupation = rand()
Task_Dimension = LARGE

While(true)

    Offloading = FIS(Battery_Level, Computational_Load,
   Memory_Occupation, Task_Dimension)

    for i in NUM_NODES
        Send_Offloading_Status(MY_NUMBER, i, Offloading)
        Offloading_Status[i] = Get_Offloading_Status(i)
    end for

    // First step: determining the need of overloading for
     a node
    if Battery_Level <= MEDIUM_BATTERY_LEVEL then
        if Computational_Load >= HIGH_LOAD .OR.
   Memory_Occupation >= HIGH then
            set Need_Offloading = TRUE
            Send_Node = Search_Node()
            // set the variables for next epoch
            Memory_Occupation = Memory_Occupation - LARGE
            Computational_Load = Computational_Load - q *
   LARGE
            Battery_Level = Battery_Level - maxE * 300
        endif
    endif

    // wait 5 minutes
    wait(300)

end while
```

Listing 1: Migration management algorithm

The test was conducted with different setups characterized by an increasing number of nodes (4, 8, 16, 32, 64, respectively), and using the described algorithm with and without the task migration feature: in the first run tasks are not migrated, while in the second run tasks are migrated. Each run ends when all battery are exhausted. Multiple runs have been executed for each setup. Figure 8 presents the maximum discharge time in each case, comparing the runs with or without migrations enabled. The showed values concern the average discharge time of the last surviving IoT node over multiple runs of each case.

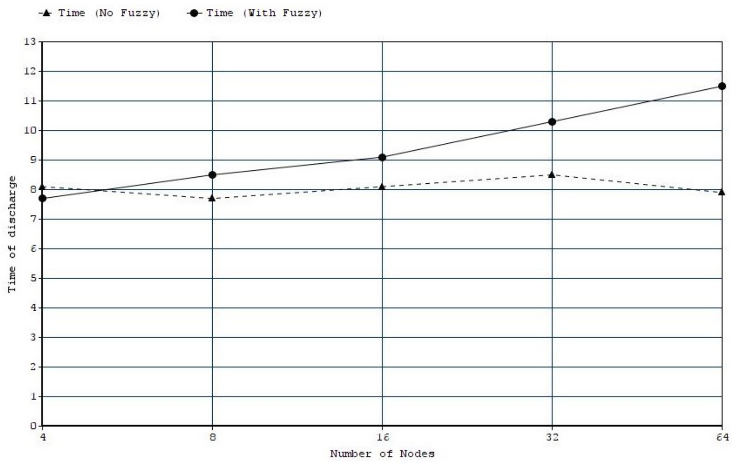

Figure 8: Average maximum discharge time for the simulated cases

Not surprisingly, the impact of task migration is as much appreciable as the number of nodes grows. In fact, the discharge time of all batteries grows when the number of nodes grows.

The evolution of migrations requests versus time is also a relevant aspect to be examined. In Figure 9 the number of migration requests, the number of successful migrations and the number of rejected migrations are shown for the 32 nodes case. In the beginning of the simulation, batteries are all fully charged and there is no migration request. When batteries are all almost exhausted, all requests are rejected.

As can be seen in Figure 9, the algorithm proposed works quite well in successfully migrating tasks between nodes, except when the battery level is very low in all nodes, as could be expected. Moreover, the algorithm is not in operation for much of the time (when the battery level is high). This means that, in the algorithm proposed, the integration of features such as task balancing or

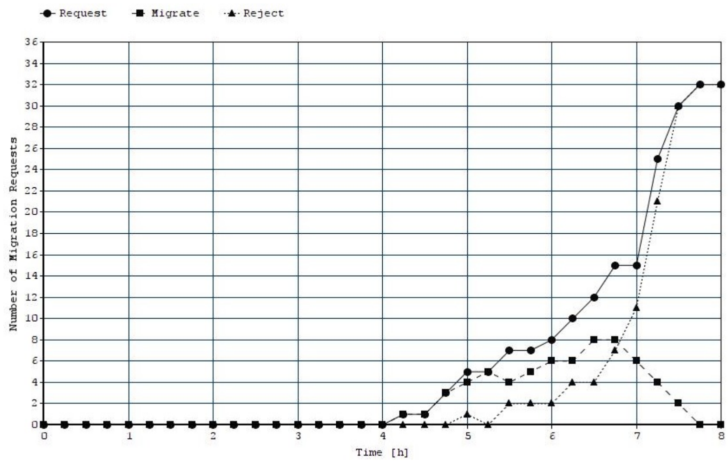

Figure 9: Number of Migration requests (Nodes=32)

load forecasting may result in a positive impact in order to reach the goal of energy consumption minimization.

In Figure 10 the spatial distribution of migrated tasks is showed. When a task may be successfully migrated, it is also interesting to see if the destination node is close or far. Successfully migrated tasks are divided into three categories according to the distance at which the node they are migrated to (”Distance=1” labels the tasks that have been migrated to an adjacent node, ”Distance=2” labels the tasks that have been migrated to a node that is adjacent to an adjacent node, ”Other” labels tasks that have been migrated on all other nodes).

Results show that the algorithm achieves the objective of maintain low the distance between sender and receiver. In fact, the number of tasks migrated in a node with Distance ¿ 2 is very low in all cases.

6 Conclusions and future work

In this paper we presented a preliminary analysis of a fuzzy-based algorithm to implement an energy optimization management technique for battery-powered IoT nodes based on peer-to peer task migrations in edge computing IoT systems. Results show that our solution, that is easily implemented on IoT nodes with low requirements, provides improvements with no need of support by an edge server that are worth further investigations with more detailed models. We believe this direction can achieve significant results in terms of increased resilience of large scale IoT and edge systems, due to the inherently distributed and autonomic approach. Future work, that is already ongoing, is planned along three

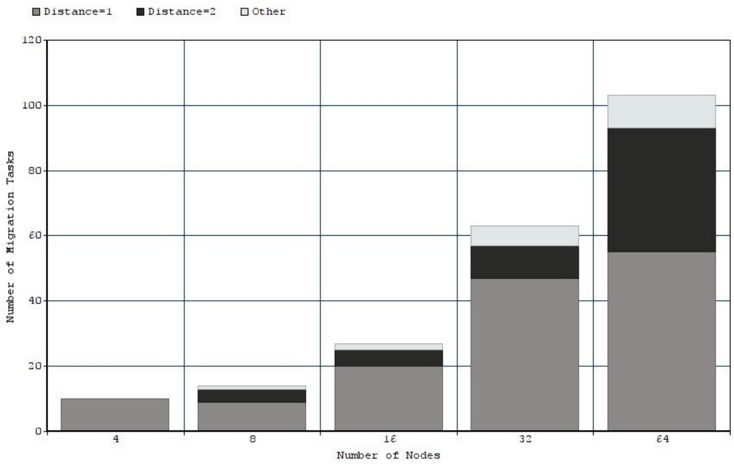

Figure 10: Destination of Migrated Tasks

directions: the first is based on removing simplification hypotheses, e.g. including the effects of routing strategies and of data compression algorithms according to target application scenarios; the second is focused on the adoption of a more detailed energy model, with reference to literature and to actual experimental hardware that is being acquired; the third is oriented to the implementation of the overall strategies in the application fields that have been explored in our previous publications.

Acknowledgements

This work has been partially funded by the internal competitive funding program "VALERE: VAnviteLli pEr la RicErca" of Università degli Studi della Campania "Luigi Vanvitelli" and by project "Attrazione e Mobilità dei Ricercatori" Italian PON Programme (PON_AIM 2018 num. AIM1878214-2).

References

[Amri et al., 2019] Amri, S., Khelifi, F., Bradai, A., Rachedi, A., Kaddachi, M. L., and Atri, M. (2019). A new fuzzy logic based node localization mechanism for Wireless Sensor Networks. *Future Generation Computer Systems*, 93:799 – 813.

[Basic et al., 2019] Basic, F., Aral, A., and Brandic, I. (2019). Fuzzy handoff control in edge offloading. In *2019 IEEE International Conference on Fog Computing (ICFC)*, pages 87–96. IEEE.

[Bi and Zhang, 2018] Bi, S. and Zhang, Y. J. (2018). Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Transactions on Wireless Communications*, 17(6):4177–4190.

[Campanile et al., 2020a] Campanile, L., Gribaudo, M., Iacono, M., and Mastroianni, M. (2020a). Performance evaluation of a fog WSN infrastructure for emergency management. *Simulation Modelling Practice and Theory*, 104:102120.

[Campanile et al., 2020b] Campanile, L., Iacono, M., Marulli, F., and Mastroianni, M. (2020b). A simulation study on a WSN for emergency management. *Communications of the ECMS*, 34(1):1–7.

[Cavalieri d'Oro et al., 2019] Cavalieri d'Oro, E., Colombo, S., Gribaudo, M., Iacono, M., Manca, D., and Piazzolla, P. (2019). Modeling and evaluating a complex edge computing based systems: An emergency management support system case study. *Internet of Things*, 6:100054.

[Chaudhary, 2019] Chaudhary, A. (2019). Mamdani and Sugeno Fuzzy Inference Systems' Comparison for Detection of Packet Dropping Attack in Mobile Ad Hoc Networks. In Abraham, A., Dutta, P., Mandal, J. K., Bhattacharya, A., and Dutta, S., editors, *Emerging Technologies in Data Mining and Information Security*, pages 805–811, Singapore. Springer Singapore.

[Ciobanu and Dobre, 2018] Ciobanu, R.-I. and Dobre, C. (2018). Mobile interactions and computation offloading in drop computing. In *International Conference on Network-Based Information Systems*, pages 361–373. Springer.

[Collotta et al., 2017] Collotta, M., Pau, G., and Maniscalco, V. (2017). A fuzzy logic approach by using particle swarm optimization for effective energy management in IWSNs. *IEEE Transactions on Industrial Electronics*, 64(12):9496–9506.

[D'Arienzo et al., 2013] D'Arienzo, M., Iacono, M., Marrone, S., and Nardone, R. (2013). Petri net based evaluation of energy consumption in wireless sensor nodes. *J. High Speed Networks*, 19(4):339–358.

[De Arcangelis et al., 2020] De Arcangelis, L., Iacono, M., and Lippiello, E. (2020). Towards a multiparadigm approach to model energy management in WSN for IoT based edge computing applications. *Communications of the ECMS*, 34(1):1–7.

[Dhimish et al., 2018] Dhimish, M., Holmes, V., Mehrdadi, B., and Dales, M. (2018). Comparing Mamdani Sugeno fuzzy logic and RBF ANN network for PV fault detection. *Renewable Energy*, 117:257 – 274.

[Dinh et al., 2017] Dinh, T. Q., Tang, J., La, Q. D., and Quek, T. Q. (2017). Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Transactions on Communications*, 65(8):3571–3584.

[Enokido et al., 2014] Enokido, T., Aikebaier, A., and Takizawa, M. (2014). An extended simple power consumption model for selecting a server to perform computation type processes in digital ecosystems. *IEEE Transactions on Industrial Informatics*, 10(2):1627–1636.

[Feng et al., 2017] Feng, J., Liu, Z., Wu, C., and Ji, Y. (2017). AVE: Autonomous vehicular edge computing framework with ACO-based scheduling. *IEEE Transactions on Vehicular Technology*, 66(12):10660–10675.

[Hamam and Georganas, 2008] Hamam, A. and Georganas, N. D. (2008). A comparison of Mamdani and Sugeno fuzzy inference systems for evaluating the quality of experience of Hapto-Audio-Visual applications. In *2008 IEEE International Workshop on Haptic Audio visual Environments and Games*, pages 87–92.

[Jassbi et al., 2007] Jassbi, J., Alavi, S. H., Serra, P. J. A., and Ribeiro, R. A. (2007). Transformation of a Mamdani FIS to First Order Sugeno FIS. In *2007 IEEE International Fuzzy Systems Conference*, pages 1–6.

[Jassbi et al., 2006] Jassbi, J. J., Serra, P. J. A., Ribeiro, R. A., and Donati, A. (2006). A Comparison of Mandani and Sugeno Inference Systems for a Space Fault Detection Application. In *2006 World Automation Congress*, pages 1–8.

[Ko et al., 2017] Ko, J., Lee, J., and Choi, Y.-J. (2017). Computation offloading for energy efficiency of smart devices. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, UbiComp '17, page 109–112, New York, NY, USA. Association for Computing Machinery.

[Li et al., 2017] Li, Y., Sun, Z., Han, L., and Mei, N. (2017). Fuzzy comprehensive evaluation method for energy management systems based on an Internet of Things. *IEEE Access*, 5:21312–21322.

[Mamdani and Assilian, 1975] Mamdani, E. and Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1 – 13.

[Mao et al., 2016] Mao, Y., Zhang, J., and Letaief, K. B. (2016). Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605.

[Marin and Dobre, 2013] Marin, R.-C. and Dobre, C. (2013). Reaching for the clouds: Contextually enhancing smartphones for energy efficiency. In *Proceedings of the 2nd ACM Workshop on High Performance Mobile Opportunistic Systems*, HP-MOSys '13, page 31–38, New York, NY, USA. Association for Computing Machinery.

[Min et al., 2019] Min, M., Xiao, L., Chen, Y., Cheng, P., Wu, D., and Zhuang, W. (2019). Learning-based computation offloading for IoT devices with energy harvesting. *IEEE Transactions on Vehicular Technology*, 68(2):1930–1941.

[Oma et al., 2018] Oma, R., Nakamura, S., Duolikun, D., Enokido, T., and Takizawa, M. (2018). An energy-efficient model for fog computing in the Internet of Things (IoT). *Internet of Things*, 1-2:14 – 26.

[Pal, 2015] Pal, S. (2015). Extending mobile cloud platforms using opportunistic networks: Survey, classification and open issues. *Journal of Universal Computer Science*, 21(12):1594–1634.

[Shu et al., 2010] Shu, L., Hauswirth, M., Zhang, Y., Ma, J., Min, G., and Wang, Y. (2010). Cross layer optimization for data gathering in wireless multimedia sensor networks within expected network lifetime. *Journal of Universal Computer Science*, 16(10):1343–1367.

[Sun et al., 2019] Sun, H., Zhou, F., and Hu, R. Q. (2019). Joint offloading and computation energy efficiency maximization in a mobile edge computing system. *IEEE Transactions on Vehicular Technology*, 68(3):3052–3056.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[Takagi and Sugeno, 1985] Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(1):116–132.

[Wang et al., 2017] Wang, F., Xu, J., Wang, X., and Cui, S. (2017). Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *IEEE Transactions on Wireless Communications*, 17(3):1784–1797.

[Xiao et al., 2016] Xiao, L., Xie, C., Chen, T., Dai, H., and Poor, H. V. (2016). A mobile offloading game against smart attacks. *IEEE Access*, 4:2281–2291.

[Xu et al., 2017] Xu, J., Chen, L., and Ren, S. (2017). Online learning for offloading and autoscaling in energy harvesting mobile edge computing. *IEEE Transactions on Cognitive Communications and Networking*, 3(3):361–373.

[Xu et al., 2020] Xu, X., Zhang, X., Gao, H., Xue, Y., Qi, L., and Dou, W. (2020). BeCome: Blockchain-Enabled Computation Offloading for IoT in Mobile Edge Computing. *IEEE Transactions on Industrial Informatics*, 16(6):4187–4195.

[Yang et al., 2014] Yang, P., Jin, P., and Yue, L. (2014). Exploiting the performance-energy tradeoffs for mobile database applications. *Journal of Universal Computer Science*, 20(10):1488–1498.

[Zhang et al., 2016] Zhang, W., Hu, Y., Zhang, Y., and Raychaudhuri, D. (2016). Segue: Quality of service aware edge cloud service migration. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 344–351. IEEE.