Ann. Telecommun. (2017) 72:415–429
DOI 10.1007/s12243-017-0583-6

Consensus-based resource allocation among objects in the internet of things

Virginia Pilloni[1] · **Luigi Atzori**[1]

Received: 15 September 2016 / Accepted: 16 May 2017 / Published online: 30 May 2017

Abstract The pervasive use of smart objects is encouraging the development of the Internet of Things (IoT) vision, where even the most common and simple object is expected to acquire information from the surrounding ambient and to cooperate with other objects to achieve a common goal . In such a heterogeneous and complex scenario, optimal allocation of resources to application tasks (e.g., available energy, computing speed, storage capacity) is paramount to fairly distribute them and not overload some objects. In this paper, we focus on finding the optimal assignment to the physical devices that can perform the same task needed by the running applications. To this, we rely on the technologies that have been already developed around the notion of Virtual Object (VO), which is the digital counterpart of the physical object and is used to augment its functionalities with the use of virtualization technologies. Our contribution is twofold. Firstly, we extend the current functionalities of VOs to make them capable of implementing a distributed strategy for the allocation of tasks among objects: the information model is enhanced to include the Quality of Information (QoI) notion and the possible different architectural solutions are presented. Secondly, we propose a distributed algorithm where VOs negotiate to reach a consensus on resources allocation, in order to distribute the workload among the objects that can cooperate to the same task and to ensure that the QoI requirements are fulfilled. Simulation results show that, compared to a static frequency allocation, the algorithm enhances the performance of the system with an average improvement of 27% in network lifetime and confirms the compliance to QoI requirements.

Keywords Resource allocation · Internet of things · Virtual objects

1 Introduction

The Internet of Things (IoT) [2] is characterized by a huge number of objects that dynamically cooperate and make their resources available, with the aim of achieving a common objective. Thanks to the pervasive spread of smart objects, the IoT is expected to offer amazing improvements in collecting, processing and distributing information [6]. Context awareness provided by IoT elements is going to improve users' knowledge, their relationship with nature and their lifestyle. Furthermore, not only will the IoT technology enable users to communicate with objects: the objects themselves, including the most common and simple, will have the ability to communicate with each other and gain the intelligence to provide information on their status or acquire data from other objects. This ability will be also fostered by the widespread adoption of cloud computing technologies [8], along with the introduction of the Virtual Object (VO) concept [16]. The VO represents the virtual counterpart of one or more IoT-related physical entities, called Real World Objects (RWOs). The VO virtualizes the RWOs it refers to, by semantically describing their resources, capabilities, functionalities, and collected data. Major functionalities implemented by the VO are caching and provisioning of relevant data also when the

✉ Virginia Pilloni
virginia.pilloni@diee.unica.it

Luigi Atzori
l.atzori@diee.unica.it

[1] DIEE, University of Cagliari, Cagliari, Italy

physical device is not reachable, implementing different interfaces and languages to extend the range of scenarios the objects can take part to, implementing complex security mechanisms, optimizing the management of the object battery.

One of the major issues in IoT scenarios is that of resource allocation management. Indeed, it may happen that more than one RWO is capable of performing the same task (e.g., temperature sensing of a given geographical area). Therefore, tasks need to be efficiently assigned to RWOs, so that their resource usage is optimized (e.g., energy, processing capabilities, communication bandwidth, storage), provided that task's required quality conditions are still satisfied.

This issue is addressed in this work, which proposes a resource allocation mechanism that takes advantage of the features offered by the VOs. Accordingly, herein, we provide two major contributions, as depicted in Fig. 1, (i) we extend the current functionalities of VOs to make them capable of implementing a distributed strategy for the allocation of tasks among objects and (ii) we propose a new distributed algorithm where VOs negotiate to reach a consensus on resources allocation. Specifically, we extend the VO information model to include the features that are needed in a distributed task allocation scenario, including the Quality of Information (QoI) that measures the characterization of the information provided by the objects. Such model is defined so that the VOs can act as the core components in the distributed algorithm following different possible architectural scenarios. As to the proposed distributed algorithm, we have defined it so as to distribute the workload among the objects that can cooperate to the same task and to ensure the QoI requirements are fulfilled. The proposed solution is based on the consensus algorithm, which has the advantage of making a group of distributed peers to reach a common target even in the case they may not be connected during the whole convergence process. This is indeed the case of IoT objects that may opportunistically take part to the deployment of IoT applications without assuring a fixed level of participation. In our specific case, the objects agree on a common lifetime and accordingly modify the level of participation to the application till a convergence is reached. Simulation results show that, compared to a static frequency allocation, the algorithm enhances the performance of the system with an average improvement of 27% in network lifetime and confirms the compliance to QoI requirements.

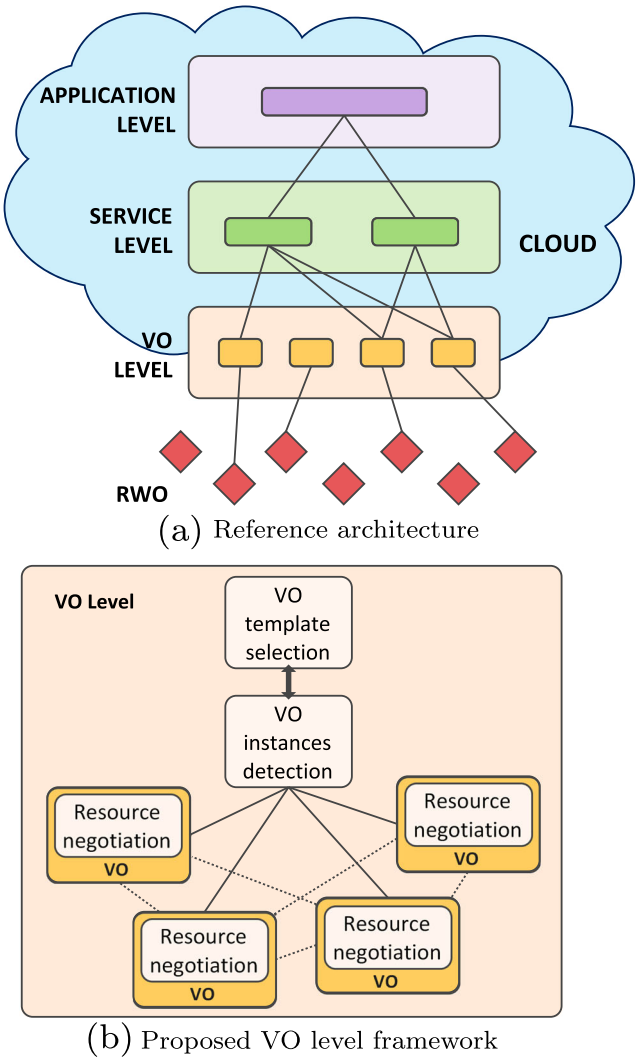


Fig. 1 The considered cloud-based IoT architectural model and proposed framework. RWOs are Real World Objects. VOs are Virtual Obects. **a** Reference architecture. **b** Proposed VO level framework

To contextualize the work and the ideas developed, in Section 2, some previous studies on the concept of virtualization in IoT are presented. Section 3 provides a functional analysis of the reference architecture and the problems related to the allocation of tasks focused on QoI achievement. Section 4 tackles the description of the resource allocation model developed. The implemented solutions have been tested using real devices on an application scenario specifically modeled. Simulations and experimental results will be presented in Section 5. Finally, conclusions and future works are presented in Section 6.

2 Preliminaries

2.1 Reference architectural model

Most of the cloud-based IoT platform implementations rely on the use of a virtualization layer, which is used to implement some functions that augment the capabilities of the physical devices. In this work, we specifically rely to the concept of Virtual Object (VO), which consists in the virtual

counterpart of the physical devices (i.e., the Real World Objects, RWOs) [16, 22, 28]. This is a software module running typically in the cloud, which speaks for the physical counterpart and introduces some functionalities that could not be taken by the real world objects, such as deciding when and how to take part to the IoT applications, caching of data already generated by the physical devices, supporting its discovery from external systems, as well as making the inter-objects communications possible by translating the used dissimilar languages. Any other functionality that may make the physical device smarter can be implemented here, leaving to the physical counterpart just the sensing and actuating activities. Clearly, the VOs are implemented in the virtual layer, as depicted in Fig. 1a. These communicate directly with the physical devices implementing all the types of communications protocols and APIs that the later are capable to understand. On top of the virtualization layer, an additional layer is implemented where services that rely on the composition of different VOs are implemented. For instance, the combination of sensed data from different devices to provide the crowd view about a given magnitude is implanted at this layer. On top of this layer is where the applications are deployed and executed.

Typically, there is one process that implements all the functionalities of a single VO, which is associated to a single physical device. Accordingly, the VO is the view provided about a physical device to the external world. However, there are cases where one VO could be associated to different physical counterparts of which then it combines the different sensing and actuating capabilities. Vice versa, it may happen that different VOs are associated to the same physical device, as each one of these VOs provides different access rights and different views. In the following of the paper, without losing of generality, we refer to the 1-to-1 case to make things simpler to explain.

Following this architecture, RWOs are virtualized and represented as VOs by the VO level, i.e., they are semantically described in terms of their functional characteristics, status, location, and potential uses, in accordance with a template that should match the type of RWO it is associated to (e.g., embedded device type, smartphone model). Furthermore, VO supports discovery and mash up of services, improving the objects' energy management efficiency, as well as addressing heterogeneity and scalability issues. Virtualization not only allows collecting a set of information but also provides RWOs with additional intelligence given by the cooperation with other RWOs and the awareness of the context in which they operate, so that they are transformed into entities that can be used to supply services and applications.

An implementation of a cloud-based IoT platform that follows this architectural model has been carried out at the University of Cagliari [7].[1] This has been used for the definition of the proposed solution.

2.2 Past works

2.2.1 Virtual objects

With a thorough comparative analysis between Cyber Physical Cloud (CPC), Cloud of Sensors (CoS), and IoT, the authors in [24] show how these three technologies exploit Cloud Computing potentialities, and how much they are related in the objective of linking digital and real worlds. They base on the concept of object *virtualization*, according to which the physical components of an object can be abstracted and made available as virtual resources. Virtualization allows the higher layers of the IoT architecture to (i) interface with devices; (ii) provide device with the required commands, adapted to their native communication protocol; and (iii) monitor their activities and connection capabilities. A VO is the virtual counterpart of one or more real objects, and as such, it inherits all their functionalities, characteristics, and acquired information [18]. Since virtualization is such a fundamental component of the IoT, many well-known middlewares, such as SENSEI [25], IoT-A [11] and iCore [22], are based on it.

Combining virtualization with context-awareness, the IoT system is able to achieve a clear knowledge of the resources and functionalities made available by its objects. Since the IoT is characterized by scarce resources, they need to be managed and orchestrated in an efficient way. The process of detecting the most appropriate IoT objects' resources that are able to fulfill the applications' requirements needs to be accomplished in a distributed and automatic way, in order to cope with the dynamic nature of the IoT.

In the literature related to the VO technologies, the problem of dynamic assignment of tasks to the physical devices is only superficially treated. Specifically, the following two elements are missing: an information model that allows for representing the changing context in which the physical devices are operating so as to better manage their involvement in the deployment of the different IoT applications; implementation of the dynamic allocation of tasks to the physical devices through the VOs as the components that can better take the burden of implementing the logic so as not to deplete the physical devices batteries.

2.2.2 Resource allocation

Resource allocation has been extensively studied in Wireless Sensor Networks (WSNs), particularly with reference to network lifetime. In [20], a distributed task allocation that

[1] http://www.lysis-iot.com/

focuses on the reduction of the overall energy consumption and task execution time into a heterogeneous WSN is proposed, with attention to nodes' residual energy. In [21], a dynamic distributed allocation mechanism based on gossip has been proposed, where the required sensing frequency is entirely assigned to the nodes that correspond to the highest network lifetime. A similar approach is studied in [26], where a distributed algorithm based on particle swarm optimization is proposed. In [15], the issue of energy saving in Wireless Cooperative Networks is addressed. The algorithm proposed in this paper aims to find a trade-off between efficiency and fairness, by using a game-theoretic approach. Since the main criticality of wireless networks is their lifetime, all these algorithms mainly focus on maximizing this resource.

The last cited papers are some representative works for task allocation in WSNs. However, the focus of our work is different as we consider the more complex IoT scenario. The main difference when moving from WSNs to IoT is that objects can be grouped opportunistically because they are found to be able to provide collaborative services and then they have to find the way to act in a coordinated way. This introduces much more heterogeneous scenarios with respect to the case of WSN networks where objects are managed by the same system and have similar characteristics. Additionally, different objects able to perform the same task can be found available (e.g., measurement of the traffic in the same street, the measurement of the humidity and/or the temperature in a room, the detection of moving objects/persons in a given environment, the monitoring of the luminosity in a public square) and then, it is necessary to decide to which one allocate the needed tasks [29]. This can be done exclusively or assigning part of the task to each of the available peers depending on a defined objective. To this, the use of VOs for the description of the objects and their potentialities become necessary. This should represent the context of the RWO that dynamically changes and should be used to find the candidates for task assignments.

Accordingly, as far as IoT networks are concerned, resource allocation is an open issue. Most of the existing studies on resource allocation for IoT are focused on IoT service provisioning, such as in [9] and [27]. In these studies, the aim is to allocate the resources that enable service execution, which consists in finding the available nodes that can perform the needed task. However, they do not focus on finding the best configuration that corresponds to an optimal resource allocation. None of the works found in the literature tries to find the optimal resource allocation associated to the lowest impact of the application assigned to the network. However, as highlighted in [14], when resource allocation is considered, a central broker is used, which is aimed at deciding which objects should be involved in the required task execution, which brings to a centralized solution and that cannot completely benefit from the opportunistic participation of the objects to the IoT platforms. For instance in [13], the authors propose the allocation of tasks among objects taking into account the available resources with particular attention to an urban-scale IoT environment. In this case, the proposed solution is a service resource allocation approach which minimizes data transmissions between users' mobile devices, which has been transformed into a variant of the degree-constrained minimum spanning tree problem and applied a genetic algorithm to reduce the time needed to produce a near-optimal solution in a centralized node.

As to the QoI, it has only been partially considered in this context. This is defined as the characterization, in terms of some salient attributes represented in the form of metadata, of the goodness of the data collected, processed and flowing through a network. In [3], the authors highlight and extend upon past work in the areas of QoI and then refine a taxonomy of pertinent QoI and VoI (Value of Information) attributes anchored around a simple ontological relationship between the two. They also introduce a framework for scoring and ranking information products based on the basis of QoI/VoI. Some examples of QoI requirements are data sampling rate, precision, and provenance [30]. In [5], the authors have investigated the QoI maximization problem by jointly optimizing the data rate and transmit power again only for the case of lifetime-constrained wireless sensor networks. In this specific case, the QoI at the sink node is characterized by the virtue of the network utility, which quantifies the aggregated value of the data gathered from different sensor nodes. [12] is one of the few works where the authors extend the problem to the IoT scenario and consider the challenge of maximizing the quality of information collected to meet decision needs of real-time IoT applications. A novel scheduling model is proposed, where applications need multiple data items to make decisions, and where individual data items can be captured at different levels of quality. The optimization is then performed in a centralized way at the service layer. Differently, we assign the job of allocating the tasks to the VOs that consider the current RWOs context and address the problem with a distributed approach. We present a first attempt in resource optimization [4], where, however, QoI was not taken into account.

3 Proposed strategy

Following the reference architectural model previously described, when a user requests an application, this application is subdivided by the Service Level into services or tasks, which are then dynamically mapped to the appropriate VOs, which take in charge their execution by involving the relevant RWOs. To this, we propose a strategy to be adopted

when more VOs are found that can trigger physical devices that can perform the same task. Indeed, at this point, there is the need to decide which device really to select or how to distribute the burden to more than one. The proposed strategy starts by considering the Quality of Information (QoI) constraints, necessary to correctly execute the application, which comes with the service execution request.

For each service execution request, the resource allocation approach proposed in this paper finds the VO template instances that best suit the required functionalities and QoI requirements. Once these services are mapped to the appropriate VOs, a consensus algorithm is run by them in order to negotiate the most appropriate workload distribution, with the aim of extending RWOs lifetime, based on their residual energy and already assigned services. The major blocks of the proposed strategy are shown in Fig. 1b, where though the information stored in the templates the appropriate VO instances are selected and the relevant VOs start the negotiation following a distributed consensus-based algorithm.

In the rest of this section, we describe how the existing information model for VOs have been extended, how the selection process works and which are the possible scenarios where this process can be implemented. The algorithm for the resource allocation is then presented in the following section.

3.1 The virtual object template and information model

The VO performs the fundamental task of collecting the varying and changing information of the real world, supported by mechanisms of learning and self-management, and exposes it to the digital world. To this end, a model that ensures the interaction among VOs in a generic way is needed. Indeed, VOs need to expose an interface to their characteristics and functionalities that are easily and straightforwardly accessible by the VO level. In addition to providing access to real-world data, it is necessary that VOs perform a coordination function, allowing multiple requests from the higher level to be addressed to the same object in a synchronized way. Furthermore, since the IoT objects ecosystem is made by fixed and mobile objects, the context where objects operate may change. For this reason, the model needs to be dynamically updated after its creation. All these features are provided by the VO template.

The VO template enables the creation, search and selection of VOs in an automated way. It is closely associated with a particular type of object that becomes part of the IoT system and thus must be mapped into the virtual world.

In order to implement templates that will be instanced to describe VOs, an information model that encodes all the information used for the appropriate involvement of VOs in the IoT application deployment and delivery is used. The information model that inspired the one proposed in this work is based on the iCore FP7 Project [22].

However, to make the iCore model more effective for our target, we extended it taking into account the mobility of objects, their temporal features and their QoI-related characteristics. This enhancement is meant to improve the VO search, discovery, and selection processes that enable the tasks assignment to the most appropriate VOs, with a QoI-oriented perspective. Figure 2 shows the new elements in dashed border boxes. Note that in this figure there is a distinction between ICT (Information and Communications Technologies) and non-ICT objects. This is to highlight that among the RWOs, there are objects that are realized with ICT technologies and then can directly take part to the IoT without additional components; this is the

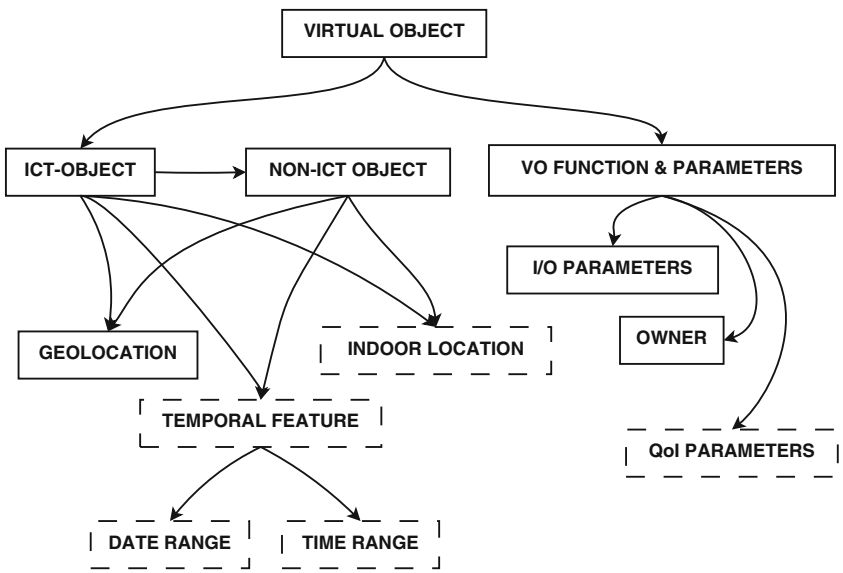


Fig. 2 VO Information Model used. *Solid border boxes* correspond to elements included in the iCore VO Information Model. *Dashed border boxes* are new elements introduced by the proposed architecture

case of, e.g., smartphones, laptops, sensors, and TV set-top-boxes. Others are referred to as non-ICT objects as they don't use natively these technologies and then need to be associated with ICT objects to take part to the IoT. These can include buildings, rooms, streets, bicycles, and people, among others. The following are new introduced elements in the model:

a) **Indoor location:** It is particularly useful in cases of indoor environments. It is used only if the VO is located indoors, and describes the place where it is located in terms of type of location (e.g., room, hall, house) and identification number or string. This could be an element that enhances the scalability of the system. It enables the model to be used not only in large-scale distributed environments (metropolitan areas or neighborhoods), but also in small size environments and internal locations (such as buildings or structures in which a geo-localization of the nodes is not enough).
b) **Temporal features:** The use of the temporal feature, both in terms of date and time range, allows knowing the activity phases of a device associated with its VO. Knowing the date and time in which a mobile device is located in a given place, helps the association process among ICT and non-ICT object. It also ensures the ability to know in advance when a particular resource is available, when it is possible to refer to it, and how long it has not been updated.
c) **QoI parameters:** The information model, on which the selection processes are based, includes a field dedicated solely to the QoI parameters. The values in this field are named uniquely based on RWOs characteristics. In addition, it introduces their descriptive aspects, that enable their identification. The parameters stored in this field will therefore be examined in the selection phase and allow an optimized choice of the resources to use. QoI parameters take into account issues such as network characteristics (e.g., latency, bandwidth), device characteristics (e.g. energy consumed, waste energy, memory, processor), and applications requirements (e.g. precision, reliability, latency).

According to this information model, templates are created, assuming that the VOs that they represent will be part of the system in the future. Templates reside in the Cloud, in dedicated repositories, and are used at the moment of instancing a new VO. When a new VO needs to be instanced, typically because a new RWO has joined the system, the selection of the more suitable template is made. The deployment of the VO occurs by loading the desired template from the Cloud. The RWO associated to the VO must contain the drivers necessary to abstract the hardware and interface with the rest of the system. The template fields are then filled in with the information associated to the new VO.

3.2 The virtual object selection process

The goal of this work is to achieve a resource allocation optimization mechanism that takes into account QoI policies, according to the applications required by the upper levels of the architecture. It is necessary that requests from the service level are related to a series of QoI parameters. In this way, the processes of search, selection, and activation of the VO instances can be performed. Requests from the service level must contain, in addition to the VO functional, spatial, and temporal search parameters, those related to QoI.

The process of resources allocation starts from the VO level, where service requests are processed. The request contains all the parameters that ensure a research refinement and allows the execution of optimization processes for resource allocation. Based on it, the VO level, among the available VOs, looks for the VO template instances that fulfill all the service requirements, both from the functional and the QoI point of view. The RWOs associated with the selected VOs are chosen to cooperate for the execution of the service. However, before the service execution is started, the available resources are evaluated by the resource optimization process, so that the workload is fairly distributed among the RWOs.

In order to explain the process more clearly, we introduce an explanatory example, to which we refer in the following as the *HVAC management* example. We suppose that the service level receives a request to evaluate, every 10 min, if the heating, ventilation, and air conditioning (HVAC) system of office A inside the building located at coordinates $\{x,y\}$ has to be turned on, based on the mean temperature computed every 10 min of the temperature values collected every minute, and on the mean presence of someone in the office, computed every 5 min on the presence monitoring values gathered every 30 s. The error has to be no higher than 80%. The service level analyses the application and subdivides it into tasks. The list of tasks is summarized in Table 1. The service level then sends the requested tasks to the VO level, which analyses them and determines the VO templates that can respond. Hence, the VO level starts searching for VO instances characterized by a template with the appropriate parameters. Taking, for example, task 1, the selected VO instances have a template with parameters equal to those

Table 1 List of tasks for the HVAC management example

Task 1	Temperature sensing collected every minute
Task 2	Mean temperature computed every 10 min
Task 3	User presence sensing collected every 30 s
Task 4	Mean user presence computed every 5 min
Task 5	Evaluation to change HVAC status computed every 10 min

described by Fig. 3. Suppose that the VO level finds three VOs that match the queried one, which correspond to the following RWOs, each equipped with a temperature sensor: a smartphone, a smart watch and a digital thermometer. The fact that the same service can be provided by such heterogeneous devices is completely transparent to the VO level, which can manage all of them simply by managing their VOs and related attributes. At this stage, the VO level sends a request for temperature measurement to one of the VOs, including also the required frequency $F^{ref} = 1/60$ Hz. The VOs can then start reaching consensus using the approach described in Section 4.2, regardless of their localization with respect to their related RWO. After the first task has been assigned, the same process is used to allocate the following tasks.

3.3 Possible scenarios

As previously described, in the IoT scenario, we refer to the VO level that deploys the execution of services to RWOs. Typically, the VO consists in software modules that run in the cloud or edge of the network infrastructure. However, the fact that RWOs may be intelligent objects capable of performing complex operations suggests that some VO level's functionalities can be moved from the cloud to the objects themselves, provided that they have the skills to perform them. Accordingly, there are different scenarios that are possible and depend on the specific RWO characteristics. The closer is the VO to the physical device the lower the delay in taking some actions due to the interaction between the virtual and the physical counterparts.

Starting from this remark, we detected three possible scenarios, as depicted in Fig. 4. Note that here Lysis is the name of the cloud-IoT platform used for the proposed solution [7]. The first scenario (see Fig. 4a) relates to the case in which the resource selection process concerns a limited number of RWOs that are able to communicate with each other using short-range technologies and that are characterized by sufficient computational power and energy. The optimization process can be distributed on RWOs, in such a way that the management of resources is as near as possible to the point where they are used. The optimization mechanism does not pass through the Internet network, i.e., the RWOs do not have to send information about their changes on the context to the cloud as the VOs are running locally. Then, the amount of communications to the Internet is reduced, limiting the relevant energy consumption. Additionally, the algorithm is faster as once the decision is taken it is directly implemented by the devices.

The second case (Fig. 4b) refers to situations where the lack of resources by RWOs do not guarantee the possibility of a distributed optimization. In this case, if the devices are located in a limited area, it is possible to exploit the gateways of local networks. Thanks to the fact that devices are connected to local networks, gateways can take charge of VOs' functionalities and run the optimization process, involving all the RWOs interested in the connection. Clearly, this is possible if the gateway implements the services to run processes when required by the cloud.

The third scenario is depicted in Fig. 4c. This is the one related to the case in which the devices selected to perform the task are not in the same area, but they are located in different places and at a great distance so that their communication can take place only through the Internet. In this case, the optimization process is carried out in the cloud and then it is centralized. This scenario is also the one related to the case where the objects don't have sufficient computational capacity to perform the optimization process, whether they are close and can communicate using short-range technologies or are far each other.

It is important to highlight that the overall procedure is always controlled by the management functionalities of the IoT platform that is running in the cloud, which should control the deployment of VOs in the cloud, gateway or in the RWO depending on the changing conditions and determined case-by-case.

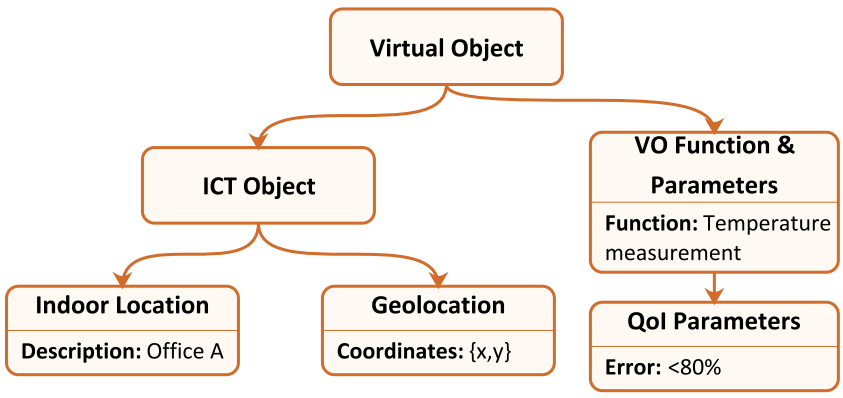
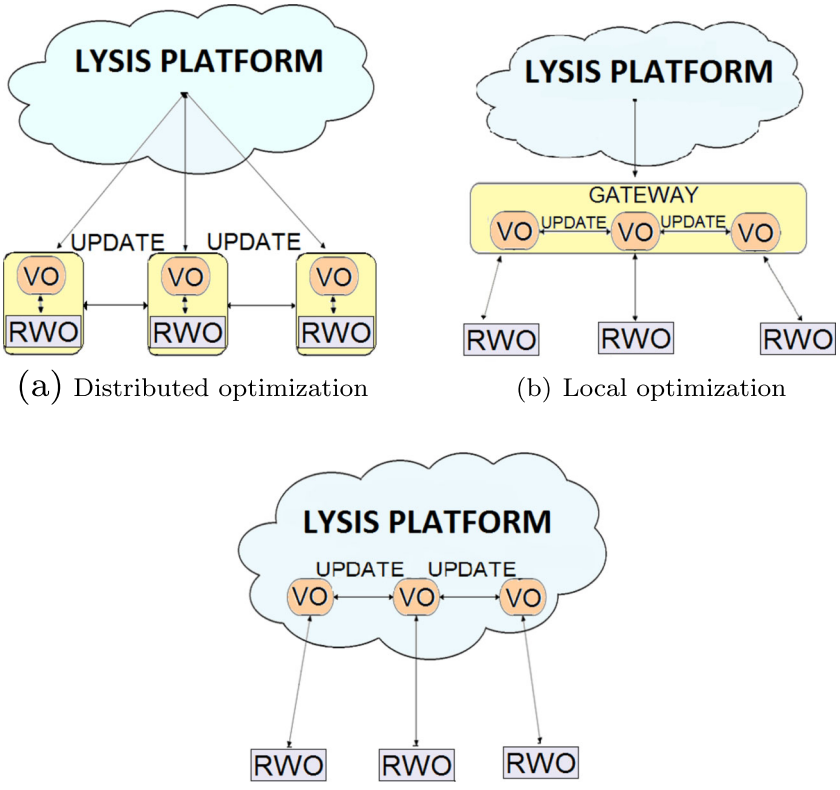


Fig. 3 VO Informaton Model required to respond to the query for task 1 coming from the Service Level, in the HVAC management example



(a) Distributed optimization (b) Local optimization

(c) Centralised optimization

Fig. 4 Location of the proposed algorithm into three typical IoT scenarios with reference to objects' resource allocation. **a** Distributed optimization. **b** Local optimization. **c** Centralised optimization

4 The resource allocation model

In this section, the resource allocation algorithm is presented. More specifically, the algorithm proposed in the following focuses on lifetime optimization, but it can be easily extended to focus on other objects' resources different from the residual energy, such as storage capacity or processing speed.

In Section 3.3, three possible different scenarios have been described. Since the more complex to be treated is the first one, i.e., the one that is characterized by a distributed optimization, in the following the discussion will only focus on this particular scenario. Nevertheless, note that the same optimization can be performed by gateways (see Fig. 4b) or by the cloud (see Fig. 4c) using the same equations, but in a centralized fashion.

4.1 Consensus-based resource allocation optimization

The resource optimization strategy proposed in this paper relies on a consensus-based algorithm where VOs decide the amount of resources to allocate to a task, in order for the workload to be fairly distributed, so that their lifetime is optimized.

As defined in [31], the lifetime of a node is the time until it depletes its battery. The lifetime of the RWO associated to VO i at time t is expressed as

$$\tau_i(t) = \frac{E_i^{res}(t)}{\sum_k P_{ik}^c(t) + P_i^o(t)} = \frac{E_i^{res}(t)}{\sum_k E_{ik}^c \cdot f_{ik}(t) + P_i^o(t)} \quad (1)$$

where $E_i^{res}(t)$ is its residual energy, $P_{ik}^c(t)$ and E_{ik}^c are the power and energy consumed by the RWO associated to VO i to perform task k, $f_{ik}(t)$ is the frequency at which VO i performs task k, and $P_i^o(t)$ is the offset power consumed by the other activities of the node (e.g., tasks that are assigned directly by the user).

We base on the assumption that optimizing the network lifetime is equivalent to adjusting the VOs' power consumption so that their associated nodes reach the same lifetime. This means that, if we consider two VOs i and j that received an activation request for task k, at time t_c when the algorithm converges, $\tau_i(t_c) = \tau_j(t_c)$. Therefore,

$$\sum_k \alpha_{ik}(t_c) f_{ik}(t_c) + \frac{P_i^o(t_c)}{E_i^{res}(t_c)} = \sum_k \alpha_{jk}(t_c) f_{jk}(t_c) + \frac{P_j^o(t_c)}{E_j^{res}(t_c)} \quad (2)$$

where $\alpha_{ik}(t) = E^c_{ik}/E^{res}_i(t)$. Defining the total amount of power consumption contributions with the exception of task k as $\delta_{ik}(t) = \sum_{l\neq k} \alpha_{il}(t) \cdot f_{il}(t) + \frac{P^o_i(t)}{E^{res}_i(t)}$, from Eq. 2 follows that

$$f_{jk}(t_c) = \frac{\alpha_{ik}(t_c)}{\alpha_{jk}(t_c)} \cdot f_{ik}(t_c) + \frac{\delta_{ik}(t_c) - \delta_{jk}(t_c)}{\alpha_{jk}(t_c)} \tag{3}$$

According to accuracy constraints provided by the higher layers, the collaborative completion of a task is required to be performed at a reference frequency $F^{ref}_k = \sum_j f_{jk}(t_c)$. Using Eq. 3 in this identity, after some simple computations and multiplying and dividing by the number N_k of VOs involved in task k, we obtain

$$\alpha_{ik}(t_c) \cdot f_{ik}(t_c) = \frac{\bar{\varphi}_k}{\bar{\beta}_k(t_c)} + \frac{\bar{\gamma}_k(t_c)}{\bar{\beta}_k(t_c)} - \delta_{ik}(t_c) \tag{4}$$

with

$$\begin{aligned} \bar{\varphi}_k &= \frac{F^{ref}_k}{N_k} \\ \bar{\beta}_k(t_c) &= \frac{1}{N_k} \cdot \sum_j \frac{1}{\alpha_{jk}(t_c)} \\ \bar{\gamma}_k(t_c) &= \frac{1}{N_k} \cdot \sum_j \frac{\delta_{jk}(t_c)}{\alpha_{jk}(t_c)} \end{aligned}$$

It is easy to notice that they represent mean values evaluated over all the VOs that are able to perform task k. This fact, along with the consideration that, in the scenario described by Fig. 4a, nodes that are assigned to the same task are located close to each other, and thus they can communicate directly without passing through the cloud, leads to the conclusion that their value can be estimated in a distributed way using an average consensus algorithm.[2] We assume to have a system where nodes may not be connected during the whole convergence process. For this reason, in this paper, the consensus algorithm proposed in [19], which is robust against topology changes, is used.

Since variations of α and δ are negligible over the time needed by the algorithm to converge (as it will be clarified in the experiments), in the following we consider them constant and omit their dependence from time. Nevertheless, if substantial variations of them are experienced, the algorithm needs to start again.

4.2 Lifetime optimization algorithm

As soon as VO i receives an activation request for task k from the VO layer, it verifies if it is able to satisfy the minimum level of QoI required by the higher levels. If it is not, it sets f_{ik} to 0 and informs the VO layer about it. Otherwise, it initializes its local values $\varphi_{ik} = \varphi^0_{ik}$, $\beta_{ik} = \beta^0_{ik}$ and $\gamma_{ik} = \gamma^0_{ik}$. As far as φ_{ik} is concerned, only one VO receives the reference frequency F^{ref}_k from the VO layer and sets φ^0_{ik} to it. The other VOs set it to 0. The initial local values are set as follows:

$$\begin{aligned} \varphi^0_{ik} &= \begin{cases} F^{ref}_k & \text{if } F^{ref}_k \text{ is given} \\ 0 & \text{otherwise} \end{cases} \\ \beta^0_{ik} &= \frac{1}{\alpha_{ik}} \\ \gamma^0_{ik} &= \frac{\delta}{\alpha_{ik}} \end{aligned} \tag{5}$$

and starts the consensus with its neighbors. Whenever VO i receives an update from one of its neighbors j, it computes the following updates:

$$\varphi^+_{ik} = \varphi_{ik} - \lambda^\varphi_1 \sum_j (\varphi_{ik} - \varphi_{jk}) - \lambda^\varphi_2 \sum_j \text{sgn}(\varphi_{ik} - \varphi_{jk}) \tag{6a}$$

$$\beta^+_{ik} = \beta_{ik} - \lambda^\beta_1 \sum_j (\beta_{ik} - \beta_{jk}) - \lambda^\beta_2 \sum_j \text{sgn}(\beta_{ik} - \beta_{jk}) \tag{6b}$$

$$\gamma^+_{ik} = \gamma_{ik} - \lambda^\gamma_1 \sum_j (\gamma_{ik} - \gamma_{jk}) - \lambda^\gamma_2 \sum_j \text{sgn}(\gamma_{ik} - \gamma_{jk}) \tag{6c}$$

$$\tau^+_i = \frac{\beta^+_{ik}}{\varphi^+_{ik} + \gamma^+_{ik}} \tag{6d}$$

$$f^+_{ik} = \frac{1}{\alpha_{ik}} \cdot \left(\frac{1}{\tau^+_i} - \delta_{ik} \right) \tag{6e}$$

where $\lambda^\varphi_1, \lambda^\beta_1, \lambda^\gamma_1, \lambda^\varphi_2, \lambda^\beta_2$, and λ^γ_2 are tuning parameters that affect the convergence time and steady-state accuracy [19], and that will be better explained in the following subsection. If $f^+_{ik} > 0$ and if its value has changed after the update, the VO sends the updated value of φ^+_{ik}, β^+_{ik} and γ^+_{ik} to its neighbors. It may happen that $f^+_{ik} \leq 0$. In this case, the VO cannot participate into executing task k. Therefore, it sets f_{ik} to 0 and informs its neighbors, which restart the consensus process. The algorithm can be considered converged when f_{ik} does not change consistently after the updates.

4.3 Convergence time and steady-state accuracy

As it is specified in [19], the tuning parameters of the update functions need to be set to

$$\begin{aligned} 0 \leq \lambda^\varphi_1, \lambda^\beta_1, \lambda^\gamma_1 &\leq \frac{1}{N_k} \\ \lambda^\varphi_2, \lambda^\beta_2, \lambda^\gamma_2 &\geq \frac{2T \cdot \Pi}{\varepsilon} + \mu^2, \qquad \mu \neq 0 \end{aligned} \tag{7}$$

where N_k is the number of VOs involved in the consensus, ε and T are positive constants, and T is a horizon time interval such that the involved VOs are connected at least for an

[2] As defined in the introduction of this section, note that Eq. 4 can as well be used to evaluate the optimal frequency values in a centralized fashion

ε amount of time ($\varepsilon \leq T$); Π and μ are weight parameters. Appropriately choosing the tuning parameters affects the accuracy of the solution of the algorithm, as well as the convergence time, as follows:

$$\begin{aligned} \text{Accuracy} &= [2 \cdot (T - \varepsilon) + \xi] \cdot \Pi \\ \text{Convergence time} &\leq \left(\frac{T}{\varepsilon \mu^2}\right) \cdot \max_{i,j} |x_i^0 - x_j^0| \end{aligned} \tag{8}$$

where $\xi > 0$ is an arbitrary infinitesimally small parameter, and $x_i(0), x_j(0)$ are the initial values for VOs i and j of the generic consensus variables, which in our case are those specified by Eq. 6. These conditions ensure that the algorithm converges to a solution in a finite time, and with an accuracy that depends on the tuning parameters.

Supposing that $T = \varepsilon$, i.e., the VOs are always connected during the consensus process:

$$\begin{aligned} \text{Accuracy} &= \xi \cdot \Pi \\ \text{Convergence time} &\leq \left(\frac{1}{\mu^2}\right) \cdot \max_{i,j} |x_i^0 - x_j^0| \end{aligned} \tag{9}$$

5 Performance analysis

The proposed resource allocation mechanism has been implemented and tested on a realistic scenario that focuses on the home healthcare/assistance of a patient with minor health problems. These are the application functionalities that need to be provided: the patient vital signs need to be constantly monitored; the patient needs to occasionally visit a primary care physician for regular checks, not necessarily for severe problems; after the visit, the patient needs to go to the pharmacy to buy the medicines that he takes daily.

The main devices involved in this scenario are (i) sensors, e.g., temperature, humidity, brightness, presence, gas, and smoke; (ii) wearable devices for vital sign monitoring; and (iii) intelligent devices, e.g., smartphones, tablets, and laptops.

The complexity of the scenario is due to three factors:

- the objects' heterogeneity: efficient cooperation needs to be ensured among sensors, controllers, actuators and smart objects, which have different capabilities and likely adopt different communication standards and are produced by different manufacturers;
- the need to obtain homogeneous data from heterogeneous sources (different data formats and precision);
- the same object is used to provide different services at the same time, and thus coordination is needed.

The implementation of the algorithm has been performed on Arduino Mega 2560 [1] boards, which microcontroller is a ATmega 2560. The local network was created through XBee S1 802.15.4 modules, by Digi International [10]. These modules use the IEEE 802.15.4 networking protocol for fast point-to-multipoint or peer-to-peer networking. The XBee modules are ideal for low-power and low-cost applications. The XBee modules have been connected to Arduino via serial port, using Xbee USB serial adapters by DF Robot [23].

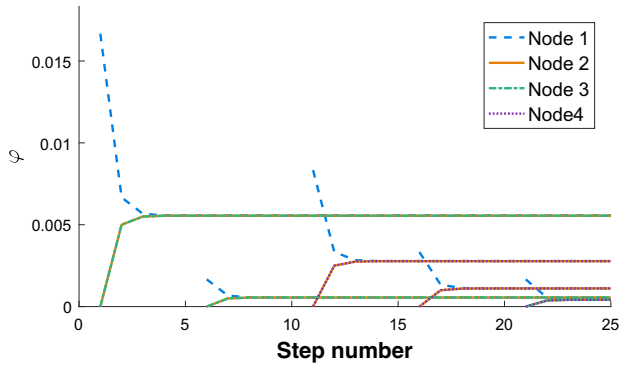
Several tests were run to validate the proposed framework and the presented reference scenario. Each test involves the activation of a variable number of services, ranging from 1 to 10, and different configurations of the devices that could simultaneously perform the required services. For each configuration, we simulated the behavior of the heterogeneous RWOs described above, with different values of residual energy, consumed energy for single service, provided QoI levels, and different reference frequencies for the service execution were used. We compared the results obtained with the Optimal Resource Allocation algorithm proposed in this paper, to which we refer as ORA in the following, with three approaches:

- **EqF:** static allocation mechanism where the reference frequency assigned to the VOs is equally divided by the number of VOs that can perform the task;
- **TAN:** dynamic distributed allocation mechanism based on game theory proposed in [20], where the whole reference frequency is assigned to the node with the highest utility, which is proportional to the ratio between energy consumption and residual energy;
- **DLMA:** dynamic distributed allocation mechanism based on gossip proposed in [21], where the reference frequencies are entirely assigned to the nodes that correspond to the highest network lifetime.

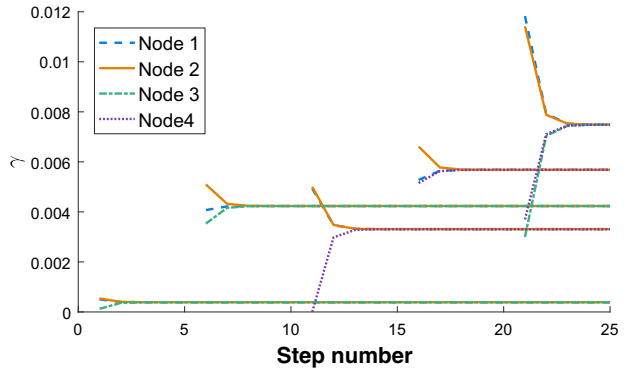
In order to describe the algorithm's behavior, the HVAC management example has been tested. We assumed to have 4 RWOs: a smartphone, a smart watch, a digital thermometer, and an infrared presence sensor. Not all the RWOs can perform all the tasks that have been described in Section 3.2 (see Table 1). The correspondence between tasks and RWOs is reported in Table 2. Figure 5 shows how the four RWOs reach consensus for the five different tasks. A different line style is associated to each device. The plots show the behavior of the consensus local update values φ_{ik}, β_{ik} and γ_{ik} described in Section 4 (Fig. 5a, b, c) and the lifetime τ_i

Table 2 Correspondence between RWOs and tasks for the HVAC management example

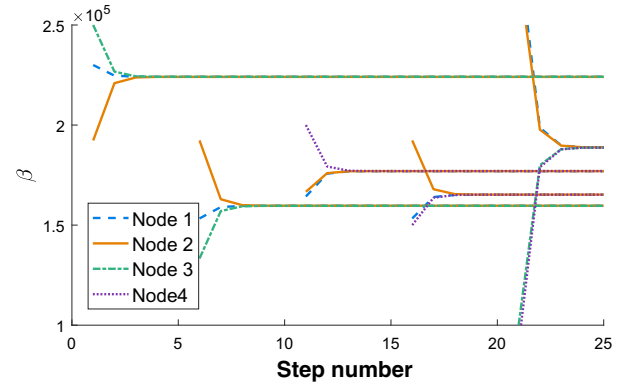
	Task 1	Task 2	Task 3	Task 4	Task 5
Smartphone	X	X	X	X	X
Smart watch	X	X	X	X	X
Digital thermometer	X	X			X
Presence sensor			X	X	X



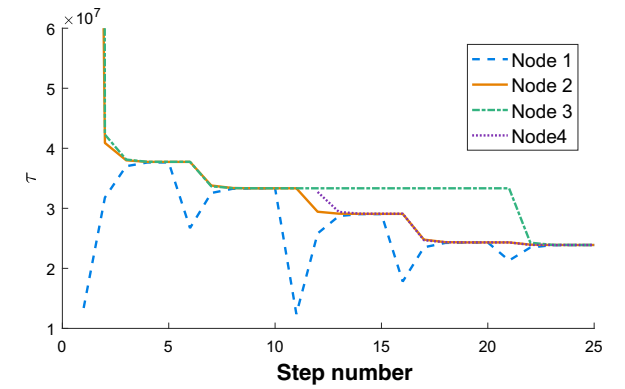
(a)



(b)

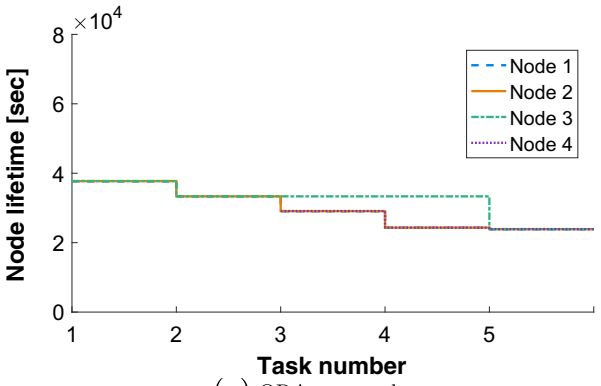


(c)

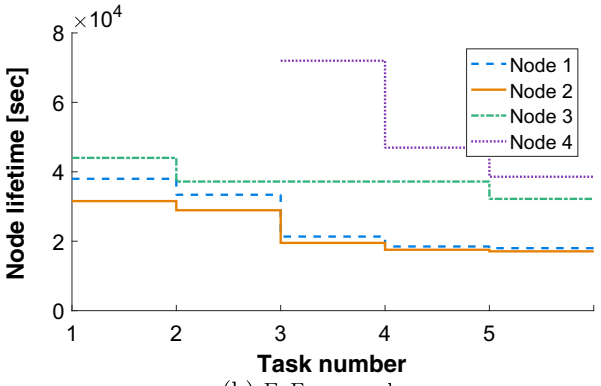


(d)

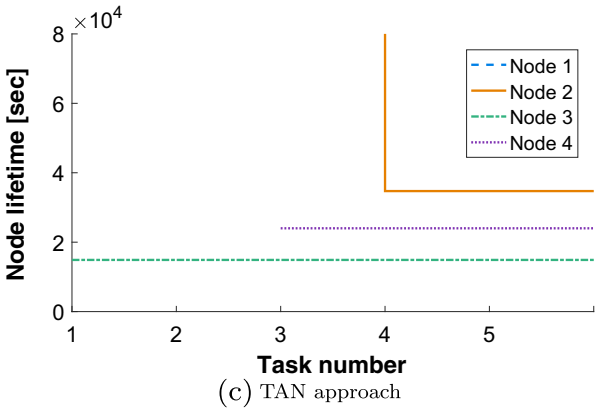
Fig. 5 HVAC management example plot for algorithm convergence, **a** φ parameter. **b** γ parameter. **c** β parameter. **d** RWO's lifetime τ



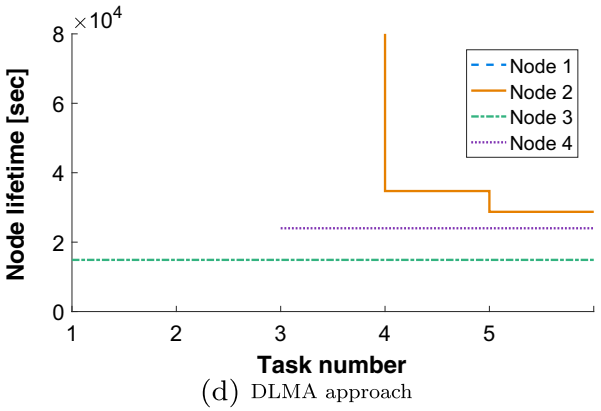
(a) ORA approach



(b) EqF approach



(c) TAN approach



(d) DLMA approach

Fig. 6 Node lifetime changes in HVAC management example using different approaches. **a** ORA approach. **b** EqF approach. **c** TAN approach. **d** DLMA approach

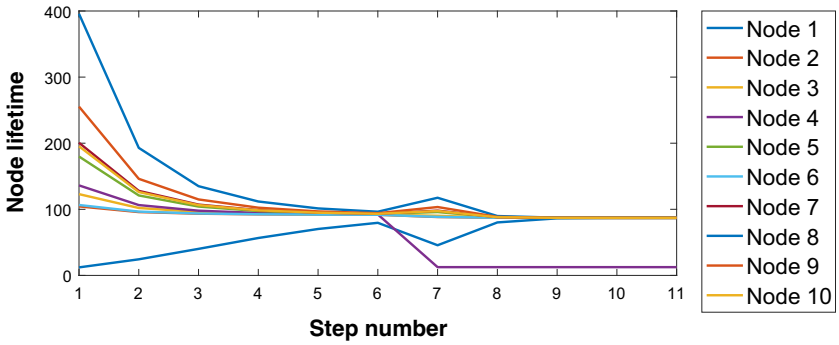


Fig. 7 Example of the ORA algorithm's behavior when the δ value increases for node 4 while reaching consensus over a task

(Fig. 5d) as a function of the algorithm's step number. It is possible to see how, when a task is activated, the values of the local updates related to each device converge in a few steps. Every time a new task has to be allocated, the consensus algorithm starts with the initialization of the local update values, corresponding to the peaks in the figures (at time steps 2, 6, 11, 16, and 21). After each initialization, the local update variables start to converge, as the reference frequency is distributed in a fair manner, and the device lifetime converge to the same value. For each task, the steady state is obtained at step numbers 5, 10, 15, 20, and 25. Since not all the nodes are able to perform the same tasks, there are some nodes that do not take part to every convergence process. This is the case, for example, of node 3, which cannot perform tasks 3 and 4. This can be also proved by Fig. 5d, where the lifetime of node 3 remains unchanged, while the other three nodes converge to a lower lifetime because they have to take charge of a greater workload. However, when the fifth task, which all the nodes can perform, is assigned, node 3 reach consensus with the other nodes, and their lifetime converge again to the same value. It is reasonable to suppose that, with reference to task 5, node 3 takes charge of a higher workload than the other nodes, since it did not take part to the previous two tasks, and thus its lifetime is higher. Indeed, it is possible to see from Fig. 5d that the lifetime value for nodes 1, 2, and 4 does not change much after the assignment of task 5.

The HVAC management example has been also tested using EqF, TAN, and DLMA approaches. Results are shown in Fig. 6. It is possible to note that, although the other approaches may results in higher lifetime values for some nodes, the ORA approach is the only one that corresponds to a higher network lifetime, i.e., it is less likely that one node depletes its battery before the others, as all the nodes experience a fair resource allocation. Indeed, the lowest lifetime observed is higher than that of the other approaches.

The proposed algorithm's behavior has been also tested with reference to substantial changes in its modeling parameters' value. Figure 7 shows an example of 10 nodes that started consensus for a task. In this example, we assume that the δ value, which accounts for any possible power consumption coming from other activities, suddenly increases for node 4. In this case, the lifetime of node 4 decreases so much that it cannot participate to the task anymore, and thus the consensus algorithm is started again by all the other nodes excluding node 4.

To analyze the benefits of the ORA algorithm, we compared lifetime values using the four different approaches: ORA, EqF, TAN, and DLMA. Results are shown for different numbers of tasks and involved devices. Figure 8 shows the results for five assigned tasks, when the number of involved devices changes, with a 95% confidence interval. The data analysis shows that ORA brings in all cases to an improvement of the network lifetime. The graphs show that the best results are obtained for a lower number of assigned tasks. Indeed, the average improvement of network lifetime decreases as we assign more tasks.

We also tested the lifetime improvement experienced for a variable number of devices, when they reach a consensus on five tasks. From results shown in Fig. 9, it is possible to

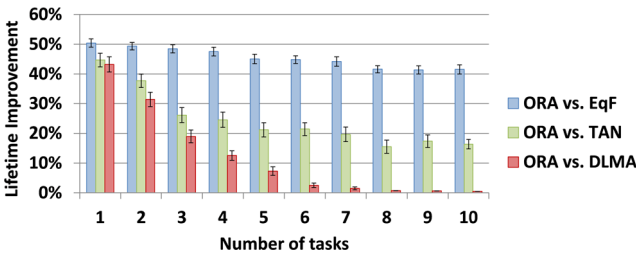


Fig. 8 Average values of percentage improvements in network lifetime for a number of devices equal to 5

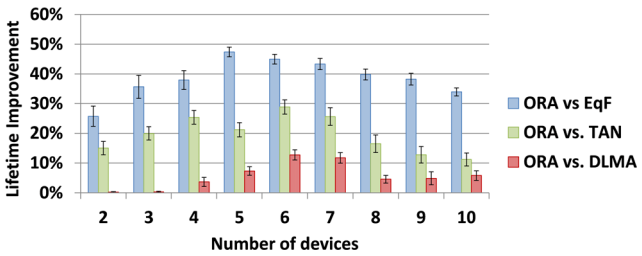


Fig. 9 Average values of percentage improvements in network lifetime for a number of tasks equal to 5

Table 3 Average values of percentage improvements in network lifetime with respect to the EqF approach, for a number of tasks equal to 5, for different values of reference frequency per task F^{ref} and different number of devices

Lifetime improvement [%]		Number of devices								
		2	3	4	5	6	7	8	9	10
F^{ref}	1	24	29	30	25	24	21	17	16	15
	2	28	35	40	36	35	33	32	30	28
	3	26	36	38	47	45	43	40	38	34
	4	25	37	42	45	46	45	44	42	41
	5	25	36	43	43	49	49	48	48	47

see a constant increase in lifetime improvement for a number of devices from 2 to 5 or 6, followed by a low decrease for higher numbers of tasks. This is consistent with the fact that when the number of devices increases, the workload per single devices decreases, and the ORA algorithm gets more efficient into appropriately allocating the workload to the single nodes, with respect to the other approaches. Nevertheless, at a certain point, the workload per single device gets so low that the efficiency of the ORA algorithm starts to decrease. This is also proved by Table 3, where lifetime improvement results for different values of reference frequency per task are reported for the EqF comparison. Similar behavior is reported for the TAN and DLMA comparisons. From the table results, it is possible to infer that, as the workload increases (i.e. the reference frequency increases), the number of devices for which the efficiency of the ORA algorithm starts to decrease gets higher. It has to be noted that, although the ORA algorithm's efficiency decreases a bit when the number of devices increases, it is always more efficient than the other considered approaches.

The behavior of the algorithm was also evaluated from the time performance point of view. The convergence times measured during the testing phase have been analyzed and represented in Fig. 10, as a function of the number of assigned services. It goes from 533 msec when only one task is assigned to 2.03 sec when 10 tasks are assigned.

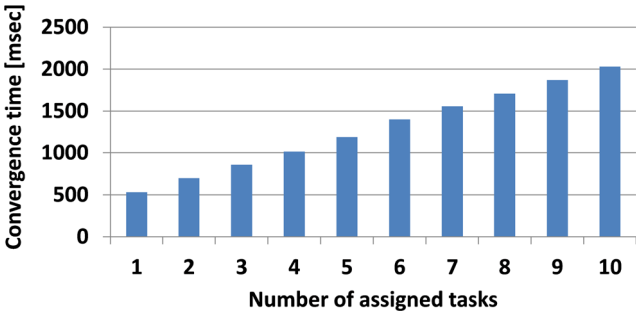


Fig. 10 Average values of convergence time

Table 4 Convergence time for different numbers of tasks and devices

Convergence time [sec]		Number of tasks				
		2	4	6	8	10
Number of devices	2	0.60	0.87	1.21	1.52	1.84
	4	0.63	0.92	1.24	1.62	1.95
	6	0.74	1.21	1.35	1.85	2.04
	8	0.80	1.22	1.61	1.83	2.18
	10	0.84	1.43	1.76	2.25	2.54

Furthermore, Table 4 summarizes the convergence time values obtained for 2 to 10 tasks and for 2 to 10 devices. As it is possible to notice, convergence time increases when the number of tasks and the number of involved devices increases. Nevertheless, from 2 to 10 devices it increases no more than 27%, while from 2 to 10 tasks its highest increase is 67%.

We further tested the algorithm for different values of λ_1 and λ_2 parameters (see Section 4.3), to understand how their changes affect the obtained results. Figures 11 and 12 show how convergence time and steady-state accuracy change when assigning 5 tasks to 5 nodes, comparing results when the well-known consensus algorithm in [17] is used, i.e., for different values of λ_1 when $\lambda_2 = 0$

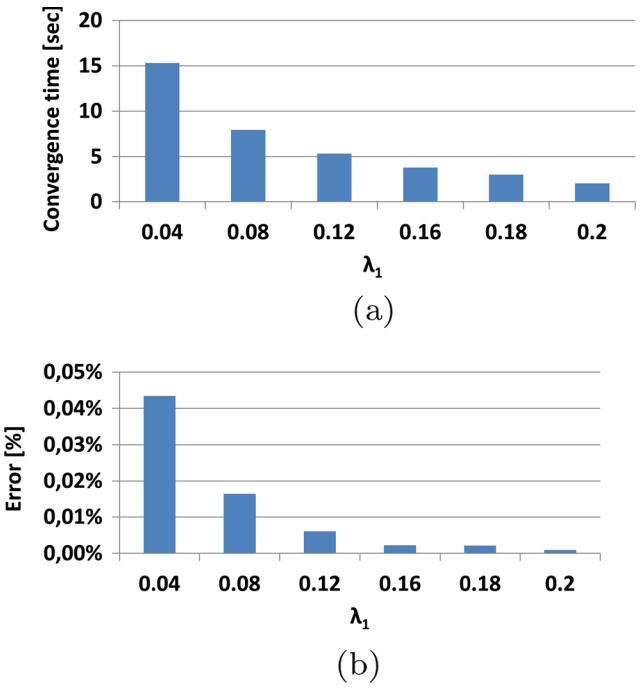


Fig. 11 Convergence time and error and error on steady-state accuracy for different values of λ_1 when the algorithm in [17] is used ($\lambda_2 = 0$), **a** convergence time. **b** error

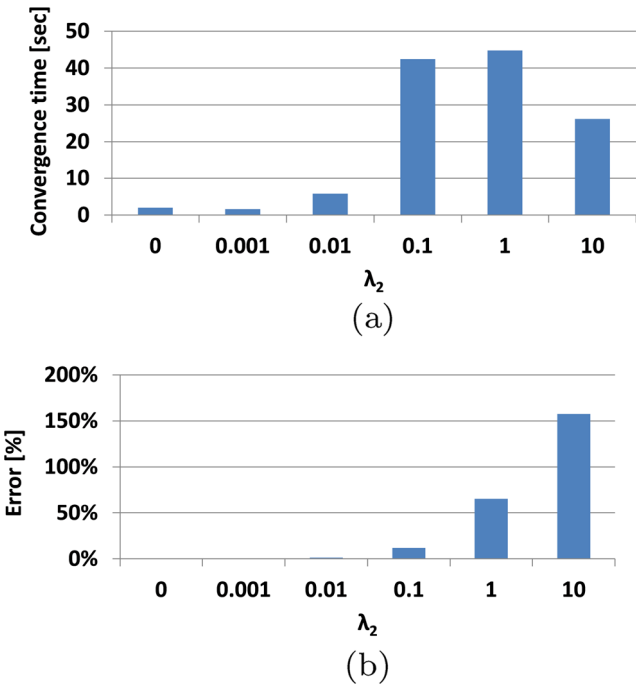


Fig. 12 Convergence time and error on steady-state accuracy for different values of λ_2 when the algorithm in [19] is used, for a fixed value of $\lambda_1 = 0.2$, **a** convergence time. **b** error

(Fig. 11), and when the algorithm in [19] is used, for different values of λ_2 when $\lambda_1 = 0.2$. Note that the values $\lambda_2 = \{0, 0.001, 0.01, 0.1, 1, 10\}$ are able to overcome noise percentages of $\{0, 0.5, 2, 2.5, 8, 50\%\}$ with respect to the correct data values. When $\lambda_2 = 0$, the consensus algorithm in [19] corresponds to the one proposed in [17]. We observed that, when [17] is used, it is always preferable that λ_1 is equal to its highest limit, which in this case is 0.2, both in terms of convergence time and accuracy. On the other hand, when [19] is used, it would be preferable that λ_2 is set to its lowest limit, but this is not always possible, as in noisy conditions the algorithm might not be able to reach convergence. In such situations, λ_2 needs to be set to a value that is higher than 0, so that convergence is surely reached, as discussed in Section 4.3. Nevertheless, its value should be set to the lowest possible, in order to prevent convergence time and error on steady-state accuracy from increasing too much. Note that for $\lambda_2 = 10$ the convergence time decreases with respect to the case where $\lambda_2 = 1$. This is due to the fact that the error is so high that the result is unreliable, and thus it can be reached in a shorter amount of time.

6 Conclusions

The analysis of the issues related to the identification and selection of resources through the use of VOs has allowed to implement a process of optimization of the allocation of tasks, which improves the QoI offered by the object resources in an IoT scenario. The consensus-based algorithm on which the process is based uses the parameters measured on the physical resources and shares among the VOs the frequency of tasks' execution required by the application, so as to provide the best possible QoI. The modeled scenario ensures the validation of the proposed framework and the improvement of its performance. In all the tests performed, the simulation results have demonstrated an average improvement of 27% in network lifetime.

The optimization process implemented has the goal to select VO instances that would guarantee the minimum QoI level and improve the lifetime of objects. Future developments will focus on the study of a multi-objective algorithm that will also take into account other resources, such as storage capacity and processing speed.

Acknowledgements This work was supported in part by Italian Ministry of University and Research (MIUR), within the Smart Cities framework (Project: CagliariPort2020, ID: SCN_00281).

References

1. Arduino (2015) Arduino mega 2560. https://www.arduino.cc/en/Main/ArduinoBoardMega2560
2. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805
3. Bisdikian C, Kaplan L, Srivastava M (2013) On the quality and value of information in sensor networks. ACM Trans Sens Netw (TOSN), 9(48)
4. Colistra G, Pilloni V, Atzori L (2014) Task allocation in group of nodes in the IoT: a consensus approach. In: 2014 IEEE International Conference on Communications (ICC), IEEE, pp 3848-3853
5. Du P, Yang Q, Shen Z, Kwak KS (2016) Quality of information maximization in lifetime-constrained wireless sensor networks. IEEE Sensors J 16(19)
6. Evans D (2011) The internet of things how the next evolution of the Internet is changing everything, Whitepaper, Cisco Internet Business Solutions Group (IBSG)
7. Girau R, Martis S, Atzori L (2016) Lysis: A platform for iot distributed applications over socially connected objects. IEEE Internet of Things Journal
8. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (IoT): a vision, architectural elements, and future directions. Futur Gener Comput Syst 29(7):1645–1660
9. Guinard D, Trifa V, Mattern F, Wilde E, 2011 From the internet of things to the web of things: resource-oriented architecture and best practices. In: Architecting the Internet of Things, Springer, pp 97–129
10. Inc DI (2015) Xbee s1. http://www.digi.com/products/xbee-rf-solutions/modules/xbee-series1-module
11. IOT-A (2015) Internet of things architecture
12. Kim JE, Abdelzaher T, Sha L, Bar-Noy A, Hobbs R, Dron W (2016) On maximizing quality of information for the internet of things: a real-time scheduling perspective. In: 2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), IEEE, pp 2325–1301

13. Kim M, Ko IY (2015) An efficient resource allocation approach based on a genetic algorithm for composite services in iot environments. In: IEEE International Conference on Web Services, IEEE, pp 543–550
14. Luong NC, Hoang DT, Wang P, Niyato D, Kim DI, Han Z (2016) Data collection and wireless communication in internet of things (iot) using economic analysis and pricing models: a survey. IEEE Commun Surv Tutorials 18(4)
15. Militano L, Araniti G (2014) Introducing fairness-efficiency tradeoff for energy savings in wireless cooperative networks. Wirel Pers Commun 76(1):3–21
16. Nitti M, Pilloni V, Colistra G, Atzori L (2015) The virtual object as a major element of the internet of things: a survey. IEEE Commun Surv Tutorials 18(2):1228–1240
17. Olfati-Saber R, Fax JA, Murray RM (2007) Consensus and cooperation in networked multi-agent systems. Proc IEEE 95(1):215–233
18. Pacual-Espada J, Sanjuan-Martinez O, G-Bustelo BCP, Lovelle JMC (2011) Virtual objects on the internet of things. Int J Interact Multimedia Artif Intell 1(4)
19. Pilloni A, Franceschelli M, Pisano A, Usai E (2014) Recent advances in sliding-mode based consensus strategies. In: Variable Structure Systems (VSS), 2014 13th International Workshop on, IEEE, pp 1–6
20. Pilloni V, Navaratnam P, Vural S, Atzori L, Tafazolli R (2013) Cooperative task assignment for distributed deployment of applications in wsns. In: 2014 13th International Workshop on Variable Structure Systems (VSS), IEEE, pp 1–6, pp 2229–2234
21. Pilloni V, Franceschelli M, Atzori L, Giua A (2016) Deployment of applications in wireless sensor networks: a gossip-based lifetime maximization approach. IEEE Trans Control Syst Technol 24(5):1828–1836
22. iCore Project (2015) iCore: empowering IoT through cognitive tecnologies
23. Robot D (2015) Xbee s1. http://www.dfrobot.com/index.php?route=product/product&product_id=72#.Vd7g1fntlHx
24. Sehgal VK, Patrick A, Rajpoot L (2014) A comparative study of cyber physical cloud, cloud of sensors and internet of things: their ideology, similarities and differences. In: 2014 IEEE International, IEEE Advance Computing Conference (IACC), pp 708–716
25. SENSEI-ICT (2008) Integrating the physical with the digital world of the network of the future
26. Shen Y, Ju H (2011) Energy-efficient task assignment based on entropy theory and particle swarm optimization algorithm for wireless sensor networks. In: 2011 IEEE/ACM International Conference on Green Computing and Communications (GreenCom), IEEE, pp 120–123
27. Silverajan B, Harju J (2009) Developing network software and communications protocols towards the internet of things. In: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE, ACM, p 9
28. Vlacheas P, Giaffreda R, Stavroulaki V, Kelaidonis D, Foteinos V, Poulios G, Demestichas P, Somov A, Biswas AR, Moessner K (2013) Enabling smart cities through a cognitive management framework for the internet of things. Commun Mag IEEE 51(6):102–111
29. Vogler M, Schleicher J, Inzinger c, Dustdar S (2016) Optimizing elastic iot application deployments. IEEE Transactions on Services Computing
30. Wang W, De S, Toenjes R, Reetz E A comprehensive ontology for knowledge representation in the internet of things. In: Trust, Security and Privacy in Computing and Communications, IEEE, pp 1793-1798
31. Yun Y, Xia Y, Behdani B, Smith JC (2013) Distributed algorithm for lifetime maximization in a delay-tolerant wireless sensor network with a mobile sink. IEEE Trans Mobile Comput 12(10):1920–1930