Energy Efficiency (2017) 10:331–357
DOI 10.1007/s12053-016-9458-3

ORIGINAL ARTICLE

Quality of service optimization in solar cells-based energy harvesting wireless sensor networks

Soledad Escolar · Stefano Chessa · Jesús Carretero

Received: 21 July 2014 / Accepted: 1 June 2016 / Published online: 17 June 2016

Abstract In energy harvesting wireless sensor networks, the sensors are able to harvest energy from the environment to recharge their batteries and thus prolong indefinitely their activities. Widely used energy harvesting systems are based on solar cells, which are predictable (i.e., their energy production can be predicted in advance). However, since the energy production of solar cells is not constant during the day, and it is null at night time, these systems require algorithms able to balance the energy consumption and production of the sensors. In this framework, we approach the design of a scheduling algorithm for the sensors that selects among a set of available tasks for the sensors (each assigned with a given quality of service), in order to keeping the sensors energy neutral, i.e., the energy produced during a day exceeds the energy consumed in the same time frame, while improving the overall quality of service. The algorithm solves an optimization problem by using a greedy approach that can be easily implemented on low-power sensors. The simulation results demonstrate that our approach is able to improve the quality of the overall scheduling plan of all networked sensors and that it actually maintains them energy neutral.

Keywords Energy harvesting systems · Wireless sensor networks · Energy efficiency · Quality of service · Solar cells

S. Escolar
Institute of Technology and Information Systems, University of Castilla-La Mancha, Camino de Moledores, s/n. 13051 Ciudad Real, Spain
e-mail: soledad.escolar@uclm.es

S. Chessa
Computer Science Department, University of Pisa and ISTI-CNR, Largo B. Pontecorvo, 3, 56127 Pisa, Italy
e-mail: stefano.chessa@unipi.it

J. Carretero
Computer Science Department, University Carlos III of Madrid, Avda. de la Universidad, 30, 28911 Madrid, Spain
e-mail: jesus.carretero@uc3m.es

Introduction

A wireless sensor network (WSN) is composed by a set of sensor nodes, or simply sensors, with the ability of monitoring its environment through transducers and of transmitting data wirelessly, typically towards a special sensor, or sink, that acts as a bridge between the WSN and the user.

The most important constraint for sensors is doubtless the energy. Generally, they are powered on by limited-capacity batteries that feed the sensor's circuitry and provide the current draw necessary to sustain their operations along the time. This means that if the sensor battery level drops below a minimum level, the sensor simply stops working. One of the approaches for prolonging the sensors' lifetime is the use of energy harvesting systems, which extract the energy found naturally in the environment, for instance from the sun, wind, or vibrations, to power

on the sensors. However, since the energy production from these sources is generally uncontrollable and intermittent, batteries are still required to maintain an adequate energy buffer that can be used in conditions of low production.

In this work, we consider a WSN composed by n sensors, where exactly one sensor acts as the sink of the network, and where exists at least one sensor for monitoring and communication purposes, i.e., $n \geq 2$. In our scenario, each sensor communicates wirelessly with the sink for transmitting and for receiving data; the sink, in turn, communicates with the $n - 1$ sensors. Note that this communication pattern corresponds to a star-connected network topology. Note also that this is just a requirement of the application and not a constraint of the problem (in fact, there exist a wide range of applications that follow this communication pattern, as for instance home/office automation or Internet of Things applications (Barsocchi et al. 2013; Escolar et al. 2014)). Each sensor is equipped with an energy harvesting system based on a solar cell with different opportunities for scavenging. Note that it does not prevent from the case of a sensor or the sink have no restrictions of energy. For example, the solar cells from different manufacturers may have different efficiencies; also, the geographic position of the sensors within the network affect, in general, their energy production. On the other hand, sensors consume energy by executing applications, which we model as a set of tasks. A task performs sensing, processing, and communicating activities with a given rate. Each task has an associated cost and a quality level that expresses the degree in which it fulfills the user requirements; thus, the same activity could be accomplished by alternative tasks each one with a different quality level. Additionally, each sensor has a set of candidate applications for execution with a certain cost and quality; then, the problem becomes the selection of the most adequate tasks to be executed at any time on the sensors, in order to maximize the overall quality level, while all sensors keep a minimum level of battery such that guarantees them to work uninterruptedly. The fact of having a set of cooperating networked sensors complicates the problem, since the sink is affected by any change in the sensors scheduling and, correspondingly, a change in the sink scheduling could imply the adaptation of the scheduling of other sensors, which leads to a continuous reschedule of the applications.

Under these assumptions, we propose a strategy devoted at finding a scheduling of applications within a time frame taken as reference, both for the sensors and for the sink, that maximizes the global quality of service (QoS) of the applications executed on the WSN. In our model, a sensor node executes some activity with a certain degree of quality, i.e., the degree of satisfaction with which the requirements of such activity are fulfilled. For example, a monitoring application could use in alternative two different transducers with different resolutions and different energy costs. If the energy budget is sufficient, the user has a preference for the transducer with high resolution, but the use of the low-resolution transducer (which consumes less energy) is also acceptable. We thus use QoS to model this preference of the user, which, of course, is encoded with the application. Thus, maximizing the global QoS means to get that all sensors and the sink perform their activities with the highest degree of satisfaction for the user.

Four are the main contributions of this paper: (1) the design of an energy harvesting system composed of a real-world sensor node platform and a solar cell; (2) a solar energy prediction model based on the distribution of the daily power output among the hours of a day for each day of the year and for the specific solar cell that we are using; (3) a heuristic that finds a (sub-)optimal assignment of execution plans within the time frame for all sensors in the WSN, that maximizes the quality level achieved by the sensors and avoids their unavailability; and (4) a simulator that generates test cases, with different network configurations and solar cell properties, and applies the optimization algorithms proposed.

The remainder of this paper is organized as follows. After reviewing the related work in "Related work" section, we present in "Solar cells-based sensors" section the design of a real-world sensor node platform connected to a solar cell-based energy harvesting system and provide its energy production model as well as the energy consumption of the applications. Section "System model" describes the system model that supports the scheduling of applications on energy harvesting WSN and formulates the QoS optimization problem. In "Energy management optimization" section, we present the algorithms aimed at finding a (sub-)optimal scheduling of applications for all sensor nodes of the network on the basis of the solar

energy prediction model proposed and in "Evaluation" section we provide the simulation results. Finally, "Discussion and conclusions" section discusses the conclusions and further research.

Related work

Wireless sensors are equipped with a radio transceiver, a set of transducers to sense the surrounding environment (Baronti et al. 2007), a low-power microcontroller for computation purposes as well as a couple of batteries to support their operations. In order to prolong sensors lifetime, applications have to be designed with the objective of reducing their energy consumption, which involves necessarily to reduce the time of use of the physical components. This means that the applications must find a trade-off between a suitable performance and a minimal consumption. While technology continues advancing more and more towards ultra-low components with different states of lower consumptions, techniques as Dynamic Power Management (DPM) (Benini et al. 2000) enable switching off the components when not necessary and waking them up on demand reducing thus their activity. By adjusting the sensing rates (Alippi et al. 2010) and the radio duty cycles (Polastre et al. 2004) of the applications, the activity periods of the components can be adapted to the requirements while maintaining a low energy consumption. Although these strategies prolong the sensor lifetimes by slowing the velocity to which the battery drains, none of them could potentially achieve infinite lifetimes. This is precisely the objective of energy harvesting systems: scavenging energy from an external source and converting it into electricity able to power on constrained embedded systems indefinitely.

Energy harvesting systems have been classified according to the next three categories (Sudevalayam and Kulkarni 2011): (1) the *source*; (2) the *controllability*; and (3) the *predictability*. The first category distinguishes between *ambient* energy sources (that extract energy from the surrounding environment, for example from the sun or the wind) and *human* source (in which case the energy is harvested from active and passive movements of humans, for instance from the blood pressure, breath or from the steps). The controllability is the capacity for extracting energy on-demand: an energy source is controllable if the energy can be extracted when required while it is non-controllable if the energy can be extracted only when it is available. In the latter case, predictability is the ability to forecast the availability of energy according to a prediction model. Solar energy can be classified as $ambient$, $uncontrollable$, and $predictable$ in daily and seasonal cycles (Sudevalayam and Kulkarni 2011; Bergonzini et al. 2009). Solar energy is also one of the most easily accessible, as harvesting systems have been implemented in form of small solar cells, which fit very well the space restrictions of sensor nodes. As a matter of a fact, several sensor platforms employ this type of energy harvesting (e.g., Prometheus (Jiang et al. 2005) and HelioMote (Lin et al. 2005)).

The ability of combining efficiently solar energy harvesting systems connected to the sensors with strategies aimed at balancing the harvested and consumed energy, enables long-lasting applications and the dynamic adaptation of the service level of the application to the available energy along the time. In this sense, QoS refers to the capability to provide assurance that the service requirements of the applications can be satisfied (Xia 2008). QoS does not only concerns the network protocols (e.g., at MAC (Yigitel et al. 2011) and routing (Felemban et al. 2006; Akkaya and Younis 2005; Sohrabi et al. 2000)) but it also concerns the services provided at the application layer for which specific QoS metrics are defined. Examples of such metrics are coverage, exposure, deployment, or reliability (Iyer and Kleinrock 2003). In Chen et al. (2011), it is given a classification of the QoS parameters into two categories: *user-specific*, which includes parameters that the user can usually control (as for instance, priority, periodicity, deadline, and availability), and *low-level* that includes the features that the user cannot usually modify (as, for instance, bandwidth, latency, throughput, and packet loss).

The goal of achieving perpetual operation of a sensor requires exploiting adequately the energy that is harvested from the sun. Due to the fact that the energy is cyclically generated, the problem can be formulated in terms of what application to execute and when. To this aim, algorithms have to be redesigned for scheduling sensor applications along a reference time based on solar energy predictions (Piorno et al. 2009), optimizing the energy level stored in the batteries. Kansal et al. (2007) proposes the use of the harvested energy at an appropriate rate such that the sensor continues

operating perpetually. This could be achieved for a sensor and for a period of reference if the consumption is lower than the production of energy in that period; they call this mode *energy-neutral* operation. They describe an algorithm for dynamically adapting the duty cycle (quality metric) of the sensor such that the constraint of energy neutrality is kept. Moser et al. (2009) proposes to maximize the QoS by adapting the level of service without wasting the harvested energy. The authors explain that, differently to other works, where only continuous parameters for quality maximization are considered (basically, sampling rates and duty cycles), they consider a finite set of discrete levels of service and propose an algorithm to dynamically assign a level of service to a time interval in the future, for which the energy to be harvested has been predicted. The condition for operating uninterruptedly is achieved if, after the assignment, the remaining energy in the battery is larger or equal than a minimal level. Moser et al. (2008) proposes adapting multiple quality parameters to optimize the applications performance in a long-term perspective (days or even weeks) by evaluating different kinds of applications of embedded systems that are modeled as a class of linear programs and by solving them (total or partially) offline.

The quality optimization in scenarios where more than one sensor is involved, and where each sensor is equipped with an energy harvesting system, also called energy harvesting network, has been approached in several works. Differently to traditional WSNs where the focus is on maximizing the network lifetime, the focus in energy harvesting WSNs is on maximizing some performance metric. Kansal et al. (2007) describes a harvesting network where n sensors have a choice for harvesting. The problem becomes how to distribute in space and time the total amount of energy that is harvested among the sensors in such a way that some application-specific performance metric can be maximized while the n sensors remain energy-neutral. The authors provide two particular application examples: a field monitoring application, where data is sampled and routed at constant rate, and an event monitoring application where only special events are transmitted. The first problem is written as a linear program for $n = 4$ nodes in a random topology, where the objective is the maximization of the amount of data to transmit by $n-1$ nodes subject to keep nodes energy-neutral. The second problem was previously described in another work (Kansal et al. 2004) and consists in minimizing the total route delay instead of minimizing the number of hops. For this work, the authors do not provide results but refer to the results of the mentioned paper, which does not consider energy-neutrality. The work described in Fafoutis and Dragoni (2011) aims at maximizing two performance metrics of energy harvesting WSNs: the end-to-end delay and the sensing rate. They propose ODMAC, an on-demand MAC protocol, based on carrier sensing schemes, and that supports multi-hop topologies. In ODMAC, idle listening is minimized by transmitting on-demand, i.e., each transmitter has to wait the receiver to wake up and transmit a beacon packet before the data packet transmission. Note that the frequency of the beacons impacts on the end-to-end delay and, subsequently, on the energy consumption; similarly, the sensing rate impacts on the transmission frequency and, subsequently, on the energy consumption. ODMAC adjusts individually the duty cycle of each node, to the level that the energy consumed is at the same level of the energy harvested, thus maintaining it at state energy neutral. Vithanage et al. (2013) is a first attempt to approach the energy harvesting WSNs in the metering industry by using the harvested thermal energy from radiators to power the nodes of the network (meters). Several are the contributions of this work: (1) to measure the energy harvested from the heat of a radiator, for which they developed a real prototype; (2) to compare analytically the default ALOHA-based MAC protocol typically used in the metering industry (IMR+) against ODMAC, the MAC scheme specifically designed for energy harvesting WSNs. The results of this comparison show the efficiency of ODMAC and its ability to adapt to harvested ambient energy. Lattanzi et al. (2007) studies the problem of optimal routing in energy harvesting WSNs. To this purpose, the authors introduce the metric Maximum Energetically Sustainable Workload (MESW), which is the maximum workload that can be autonomously sustained by the network, where the term "energetically sustainable" means that the power spent by each node due to its workload is lower than the power that it can harvest from the environment. The authors select a set of five routing algorithms for WSNs and compute for each one its MEWS. The optimal MESW ($MESW^{opt}$) is the highest MESW achieved by any routing algorithm, then, the optimality of a routing algorithm is computed as the ratio between its MESW and $MESW^{opt}$. The results show

that routing strategies that do not take into account environmental power provide poor results in terms of workload sustainability. In a later work (Bogliolo et al. 2011), the same authors propose self-adapting maximum flow (SAMF), a routing strategy for energy harvesting WSNs, to route the maximum sustainable workload subject to the capacity constraints, automatically adapting it to time-varying operating conditions. More recently, the authors published a survey (Lattanzi and Bogliolo 2011) that overviews their contributions in this field, where the view of quality is always expressed as the maximum load supported by an energy-harvesting WSN. Vullers et al. (2010) is a tutorial that explores examples of WSNs in different fields, as health, automotive, or maintenance of structures, typically battery-operated. The authors then discuss how the usage of energy harvesting systems may contribute to their autonomy, providing a description of real deployments that use commercial harvesters to achieve unbounded lifetimes. In this article, four technologies for energy harvesting are analyzed and compared (vibrational, thermal, photovoltaic, and RF) in terms of source power and harvested power.

We have addressed the problem of the quality optimization of the applications in several works (Escolar et al. 2012, 2013, 2014a, b), following an approach that increases progressively the number of sensors involved. Our first work (Escolar et al. 2012) proposes a local scheduler aimed at selecting the applications that maximize the quality level in one only sensor (based on sampling rates), adapting its quality level to the energy production of its solar energy harvesting system. The optimization described in Escolar et al. (2014a), which still uses only one sensor, introduces the constraint of energy-neutrality, which states that the sensor' battery budget cannot drop in the period of reference (according to Kansal et al. (2007)). Here, our view of quality refers to any activity of the sensor that can be performed at different satisfaction degrees for the user, for which we provide different scheduling plans, with a cost and a quality associated. We also solve the re-optimization problem, which consists of adapting the initial assignment of scheduling plans according to the excess or deficit of energy production. After that, we approach the problem in harvesting networks first in Escolar et al. (2013), where we propose a solution for just two networked devices, a sensor and a sink, each one using an energy harvesting system, by means of a low-complexity algorithm aimed at finding a (sub)optimal assignment of scheduling plans to the sensor and to the sink, such that the quality is maximized in both nodes while keeping them energy neutral. The work presented in Escolar et al. (2014b) is a further extension of Escolar et al. (2013) where we theoretically discuss the generalization of this problem to consider any number of sensors, i.e., for an energy harvesting network. Differently to Escolar et al. (2013, 2014b), this paper proposes an algorithm for achieving quality optimization in energy harvesting WSNs composed of $n \geq 2$ sensors, including the sink. Each sensor has a different opportunity for scavenging and different scheduling plans with a consumption and a quality associated to its service level. Differently to the works presented by other authors to provide energy-neutral operation in energy harvesting WSNs (Fafoutis and Dragoni 2011; Vithanage et al. 2013; Lattanzi et al. 2007; Lattanzi and Bogliolo 2011; Bogliolo et al. 2011), our focus is to maximize the quality of the applications executed on each node of the network, where quality is a wider concept than the performance network, for which we provide a variety of scheduling plans that can be dynamically selected for execution.

Solar cells-based sensors

We have designed a real energy harvesting system based on solar cells. The purpose of this system is to obtain energy from the solar light and convert it into electricity to recharge the sensor's batteries, with the ultimate goal of achieving potentially infinite lifetimes. Every sensor in the scenario that we are addressing in this paper has attached the energy harvesting system proposed, which is described in detail in this section.

The energy harvesting system consists in a solar module KL-SUN3W (KL 2014) that is connected to a Waspmote (Libelium 2014), an open source wireless sensor platform manufactured by the Spanish company Libelium. The module KL-SUN3W is composed of 28 mono-crystalline silicon solar cells with the property of converting the energy of the photons into electricity by means of the photovoltaic effect. The amount of energy that a solar cell is able to absorb depends on several parameters. First, the *irradiance* (D), which is defined as the density of incident power on the surface of the solar cell under standard

Table 1 KL-SUN3W electrical specifications

Parameter	Value
Peak power (P_{out})	3 W
Maximum power voltage (Vmp)	5.82 V
Maximum power current (Imp)	0.52 A
Open circuit voltage (Voc)	7.38 V
Short circuit current (Isc)	0.55 V
Dimensions (in mm)	225 × 155 × 17
Operating temperature	−40 −85 °C

conditions. In turn, it depends on several factors mainly related to the place where the cell is deployed, as for instance, the solar time, the location, or the inclination with respect to the sun rays. Second, the *efficiency* of the solar cell (η) that represents the percentage of the energy conversion efficiency of the irradiance absorbed at some specific place and time into electrical power. The efficiency is computed as $\frac{P_{out}}{D \times S}$, where P_{out} is the maximum power output provided by the cell (in W), D is the irradiance (in $\frac{W}{m^2}$), and S is the surface of the solar cell (in m^2). According to its manufacturer, under standard conditions (irradiance $= 1000 \frac{W}{m^2}$, temperature = 25 °C, air mass = 1.5 spectrum, angle sun/surface = 41.81°) the solar module KL-SUN3W achieves an efficiency of η =12.8 %. Other electrical specifications are shown in Table 1.

Waspmote is a low power sensor platform composed of an ATmega1281 microcontroller operating at 14.7456 MHz, 8 KB SRAM, 4 KB EPPROM, 128 KB flash memory and the possibility to attach a SD card of 2 GB. Waspmote supports different types of sensorboards (e.g., smart cities, gases, and weather station) and up to 10 different network connections including Zigbee, 802.15.4, WiFi, 868 MHz, 900 MHz, and 3G/GPRS. For its correct working the platform requires a single battery that provides a voltage between 3.3 and 4.2 V. Waspmote counts with four operational modes: On, the normal operation state requires a current draw of $15mA$; Sleep and Deep Sleep states, where the microcontroller passes to a latent state (but it still can be waken up by means of interruptions) requires $55\mu A$; and Hibernate state in which the microcontroller is turned off that requires $0.07\mu A$. According to its manufacturer, the Waspmote operation at Hibernate mode reaches up 1 year without recharging. However, note that the real power demand depends on the application requirements; in particular depends on the components used (e.g., communication module and sensors), on its operational state (e.g., in case of microcontroller On, Sleep, Hibernate), and on its operation time.

The power generated by the module KL-SUN3W is used to recharge the Waspmote's battery, since feeding directly the sensor circuitry could lead to long inactivity periods coinciding with the hours of low or null solar light intensity. Thus, the Waspmote operation is maintained still through the battery, whose level can increase (or decrease) along the time depending on the energy production and on the application consumption. For this reason, the next two subsections present the KL-SUN3W energy production model and the Waspmote energy consumption model. Note, however, that these models may apply to other solar modules and sensor platforms. Figure 1 shows our energy harvesting system composed by the solar module KL-SUN3W (on the left) and the Waspmote platform (on the right).

Energy production model

This subsection presents the energy production model that we propose to estimate the amount of solar energy that is harvested from the solar module KL-SUN3W. Our approach is to elaborate a prediction model based on formulations. However, note that there exist other

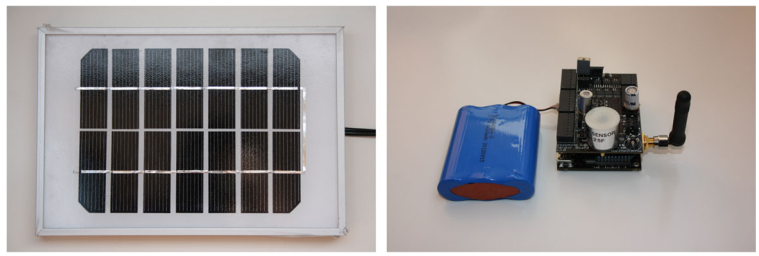


Fig. 1 Our solar energy harvesting system: The solar module KL-SUN3W (on the *left*) and the Waspmote platform (on the *right*)

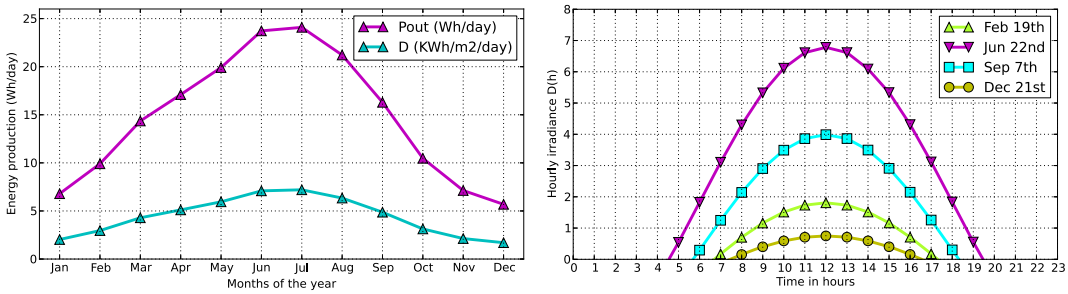


Fig. 2 Daily power output in Madrid (Spain) and its distribution per hour

methods of forecasting that could have been adopted, such as regressive models, intelligent artificial techniques, or numerical weather prediction (Inman et al. 2013).

The maximum power output (ignoring losses) delivered by a photovoltaic system during a period of time (e.g., a month, a day, or a year) can be computed as $P_{out} = D \times \eta \times S$, where D is the value of the irradiance corresponding to the same period of time and it is measured in watts. As an example, if we take the average daily solar irradiance in the city of Madrid (Spain) in the month of February, whose value is 2.96 KWh/m^2 according to the NASA program RETScreen (NASA 2013), the average power output provided by KL-SUN3W in a day of February is 11.22 Wh. We consider that the solar panel KL-SUN3W is located titled on the surface, with an inclination angle given by the standard condition of 41.81° between the sun and the surface. Figure 2 shows the average daily power output from KL-SUN3W generated from the data of irradiance in Madrid (Spain) for all months of the year (on the left) and the hourly solar irradiance computed for 1 day of the months of February, June, September, and December (on the right). Figure 3 presents the same parameters for the city of Hamburg (Germany). The comparison of the figures enables to easily visualize how the geographical location where the solar cell is deployed affects to the energy production. The amount of power output generated from the sun is larger in regions with latitude lower, since the angle of incidence of the sun rays with regard to the Earth's surface increases from the Ecuador towards the poles; in the figures, Madrid (latitude: 40.437944; longitude: -3.679536) receives more amount of sun than Hamburg (latitude: 53.558869; longitude: 9.927821).

For a given power output P_{out}, the corresponding amount of energy that can be produced is computed as $\frac{P_{out}}{Vmp}$, where Vmp is the voltage at maximum power. We are particularly interested in knowing the hourly curve of the energy production for the solar module KL-SUN3W. We compute the curve by approaching the distribution of the daily power output among the hours of a day. For approaching the distribution function, we consider the solar light intensity incident on the surface (irradiance) during a day, which depends on the climatic conditions at the moment and on the solar zenith angle Θ_Z, the angle between the Sun and the vertical axis of the Earth surface. The rotational movement of the Earth on its own axis and around the Sun makes Θ_Z varying along the 365 days of the year

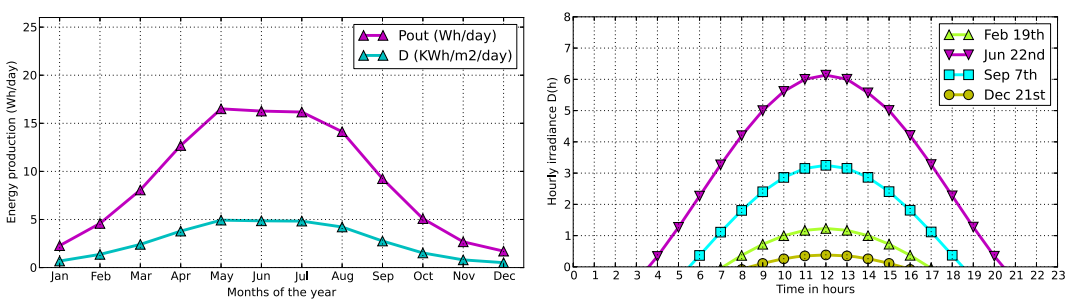


Fig. 3 Daily power output in Hamburg (Germany) and its distribution per hour

and time of the day. Three parameters are required to compute Θ_Z: (1) the latitude of the point geographic Φ; (2) the solar declination angle δ; and (3) the time of day H. Following the model described in Hartmann (1994) and Jacobson (2005), we have computed the zenith angle as $\Theta_Z = \cos^{-1}(\sin(\Phi)\sin(\delta) + \cos(\Phi)\cos(\delta)\cos(H))$, where according to Cooper (1969) δ is computed as $23.45°(360°(\frac{284+d}{366}))$ (being d the number of the day of the year to be computed, e.g., $d \in [1, 365]$) and H is the hour angle or angle of radiation due to time of day, which is computed as $H = 15°(t - 12)$, with t given as solar time, due to the fact that the Earth shifts approximately 15° each hour. Knowing the zenith angle, the hourly irradiance can be estimated as a function of Θ_Z and the average daily irradiance D, as $D(t) = D\cos(\Theta_Z)$, where t is any hour of the day ($t \in [0, 23]$) and Θ_Z represents the zenith angle at time t. Finally, the hourly energy production is obtained as $E(t) = \frac{D(t)\times\eta\times S}{Vmp}$, $t \in [0, 23]$.

For the city of Madrid and for all days of the year, we have computed its zenith angle and the estimation of the hourly irradiance. Figure 2 on the right shows the hourly irradiance $D(t)$ for the days of February 19th, June 22th (summer solstice), September 7th, and December 21st (winter solstice) assuming the values of daily irradiance of 2.96, 7.09, 4.87, and 1.7 KWh/day/m^2 for the months of February, June, September, and December, respectively, (data provided by RETScreen).

Energy consumption model

We present here the energy consumption model for the platform Waspmote. As described before, the solar module KL-SUN3W recharges the battery of the Waspmote; on the other hand, the battery is drained along the time depending on the consumption of the application that is executed, which depends on the hardware components, the energy consumption state of each one, and the time that is being used in each state. The Waspmote platform is connected to a Lithium-ion battery (Li-Ion) with 3.7 V nominal voltage and a voltage ranging between 3.3 and 4.2 volts, which represent the minimum and maximum operational battery voltage, respectively. This battery can be recharged through USB at a voltage of 5 V providing a maximum charging current of 100 mA and, alternatively, through a solar panel, as KL-SUN3W, at 6–12 V and providing a maximum charging current of 280 mA. To compute the discharge time of the battery we manage two parameters: C is the capacity of the battery (in mAh) and I is the current draw required for the working of the application (in mA). The time (in hours) to completely discharge the Waspmote assuming that there is no energy production from the solar cell and a constant current draw I comes given then by $\frac{C}{I}$. As an example, considering the capacity of the Waspmote battery, $C = 6600$ mAh, and assuming a simple application that only makes usage of the microcontroller in state On, $I = 15$ mA, the time to discharge would be $\frac{6600}{15} = 440$ h. Note, however, that the current draw is generally not constant and vary with the application requirements. In this sense, the current draw to be applied for this computation is the sum of the partial consumptions of the hardware components use.

System model

Let us consider a WSN composed of n sensors $\{n_1, n_2, \ldots, n_n\}$, where n_n acts as a sink. Each sensor executes a subset of the available tasks θ. A task $T \in \theta$ is represented by a triple $T = \langle t, p, w\rangle$, where t is the time of execution of the task, p is the period of execution of the task (note that must be $p > t$, otherwise the period of execution of the task cannot be met), and w is the energy consumption of the node per unit of time, due to the execution of task T. For each task T, its duty cycle, expressed as the percentage of system time required, is computed as $dc = \frac{t}{p} \times 100$. There also exists a special task T_0 (also called Idle task) which is executed when there is no other task to execute.

In a given period of time, a sensor may be assigned to execute a scheduling plan, which is defined by a finite subset of tasks in θ. When a scheduling plan is selected to be executed, all of its tasks are cyclically executed in order. Each sensor i, with $i \in [1, n)$, has a set of m scheduling plans to be selected for execution. For simplicity, we assume that m is the same for all of the sensors. However, note that it is not a limitation of the model and its generalization is straightforward. We define $P_i[]$ as the m elements vector containing all the available scheduling plans for sensor i. Given two tasks T_1 and T_2, T_1 is equal to T_2 iff $t_1 = t_2$, $p_1 = p_2$, and $w_1 = w_2$ and it is denoted as $T_1 = T_2$. Two scheduling plans A and B are equal, and it is denoted

as $A = B$, iff $T_1^A = T_1^B$, $T_2^A = T_2^B \ldots T_n^A = T_n^B$, where n is the number of tasks of A and B.

For any given scheduling plan P, we define its energy consumption per unit of time $c(P)$ as the sum of the energy consumptions of the tasks in P, and its duty cycle $dc(P)$ as the sum of the duty cycles of the tasks in P (note that must be $dc(P) \leq 100\%$, otherwise the scheduling plan is not feasible as the sensor does not have enough processing capacity to run it). We also define the quality level $q(P)$ of the scheduling plan to express the degree in which this scheduling plan fulfills the application requirements. Specifically, quality levels enable the representation of different user-defined metrics for the application, such as the sampling rate, the duty cycle, or the communication pattern employed. As an example, let us consider two scheduling plans A and B, intended for monitoring an area through a video camera. The definition for A is $A = \{T_1^A, T_2^A, T_3^A, T_0^A\}$, where T_1^A is the task for sampling the camera at a rate of 10 s with a high resolution, T_2^A is the task for processing the sample, and T_3^A is the communicating task that uses a routing protocol with guaranteed-delivery. Alternatively, the definition for B is $B = \{T_1^B, T_2^B, T_3^B, T_0^B\}$, where T_1^B is the task for sampling the camera at a rate of 60 seconds with a low resolution, T_2^B is the task for processing the sample, and T_3^B is the communicating task that uses a best-effort routing protocol (T_0^A and T_0^B are the Idle tasks for scheduling plans A and B, respectively). Generally speaking, a higher quality in the performing of a task, for instance the sampling or the transmission, implies a better quality of such activity as well as a higher cost. Specifically, $q(A) > q(B)$ and, subsequently, $c(A) > c(B)$.

The sink also has a set of scheduling plans to be selected for execution. For any given scheduling plan $P_i[j]$ of sensor i ($j \in [1, m]$) there exists one (and only one) scheduling plan $Q_i[j]$ in the sink that enables the overall execution of the pair $\langle P_i[j], Q_i[j]\rangle$. In other words, $Q_i[j]$ is the only set of tasks for the sink that is compatible with the execution of $P_i[j]$ in sensor i. In the rest of the paper, we use function $p(P)$ that returns the only scheduling subplan for the sink compatible with a scheduling plan P of a sensor, i.e., $Q_i[j] = p(P_i[j])$.

The energy consumption per unit of time of $Q_i[j]$ is denoted by $c(Q_i[j])$ and the duty cycle of $Q_i[j]$ is defined similarly to the duty cycles of the scheduling plans for the sensors, and it is denoted by $dc(Q_i[j])$. Note that, given a pair $\langle P_i[j], Q_i[j]\rangle$, the duty cycles $dc(P_i[j])$ and $dc(Q_i[j])$ can be rather different from each other because the tasks executed by the sensor differ from the tasks executed by the sink, and because the sink and the sensor may have completely different processing power. Note that we do not need to define a quality level for the scheduling sub-plans of the sink as these can be considered as service scheduling plans for the correspondent scheduling plans executed on the sensors and therefore, the quality of $Q_i[j]$ is equal to the quality of $P_i[j]$.

The efficiency of a scheduling plan P is denoted by $\epsilon(P)$ and is defined as $\frac{q(P)}{c(P)+c(p(P))}$, that is, the ratio between its quality level $q(P)$ and the costs $c(P)$ and $c(p(P))$ (i.e., the cost of P and the cost of the scheduling sub-plan in the sink compatible with P). It should be noted that a scheduling plan P for which $q(P) < q(P')$ and $c(P) + c(p(P)) \geq c(P') + c(p(P'))$ for some other scheduling plan P' is inefficient, and it can be excluded a priori. For this reason, we assume that, without loss of generality, the higher the quality level, the higher the energy consumption.

Hereafter, we assume that the vector $P_i[]$ is initially ordered in terms of efficiency in descendant order, i.e., $\epsilon(P_i[j]) \geq \epsilon(P_i[k])$ for each $j < k$.

For the ease of notation, we also denote with $P[]|_q$ the vector obtained from vector $P[]$ by removing all elements that have quality level smaller or equal than q. Similarly, we denote with $P[]|^c$ the vector obtained from $P[]$ by removing all elements that have a cost larger or equal than c. Note that, if $P[]$ is sorted, the vectors $P[]|_q$ and $P[]|^c$ are also sorted according to the same order, and both operations can be performed in a linear time with the size of the vector. The operations $P[]|_q$ and $P[]|^c$ return an empty vector, denoted with $\emptyset$, if no elements in $P[]$ satisfy the requested property.

Both the sensors and the sink are powered by means of the solar cell-based energy harvesting system described in "Solar cells-based sensors" section. Since the energy that can be produced from a solar cell depends on different factors such as, for instance, the solar time and the geographic location, batteries are still necessary to feed the circuitry of the sensors and to keep their operations when there is no energy production. We assume that the energy production in the sink is sufficient to attend several sensors simultaneously (e.g., the sink could use larger solar cells and have batteries of larger capacity). Thus, the

energy harvesting system converts the solar energy into electricity which is used to increase (or decrease) the battery level in the sensors and the sink. It is also known that the solar energy production has a natural cycle of 24 h that, although uncontrollable, is predictable (Kansal et al. 2007). For this reason, we consider a time frame taken as reference of $\Delta = 24$ h and we discretize the time axis into v slots of duration $\frac{\Delta \times 60 \times 60}{v}$ seconds. For each slot k ($k \leq v$) and for each sensor i ($i \leq n$), we devise a strategy aimed at finding the assignment of scheduling plans to the time slots in order to optimize the overall quality of service.

Let S be the scheduling matrix of dimension $n \times v$ that represents the assignment of scheduling plans to the sensors and to the sink in every slot, specifically:

$$s_{i,k} = \begin{cases} \text{the scheduling plan assigned to sensor } i \text{ in slot } k \text{ if } i < n \\ \text{the scheduling plan assigned to the sink in slot } k \text{ if } i = n \end{cases} \quad (1)$$

Note that, for a given slot $k \in [1, v]$, the assignment of scheduling plans to all the sensors is represented by the k^{th} column of S. Note also that, for a given sensor $i \in [1, n)$, its assignment of scheduling plans to the slots is represented by the i^{th} row of S. In order to attend the selections done by the $n-1$ sensors in each slot k the sink must select for execution a scheduling plan that is compatible with the scheduling plans selected for all the sensors in the same slot. For this reason, the sink has to schedule several of its scheduling plans, one for each sensor with whom it interacts. Considering that a scheduling plan for a sensor is compatible with only one scheduling sub-plan for the sink, we have that the scheduling plan for the sink at slot k must always be $s_{n,k} = \bigcup_{i=1}^{n-1} p(s_{i,k})$. This scheduling plan has a cost given by $\sum_{i=1}^{n-1} c(p(s_{i,k}))$ and a duty cycle equal to $\sum_{i=1}^{n-1} dc(p(s_{i,k}))$. Note that the duty cycle of the scheduling plan for the sink must clearly be lower or equal than 100 % to be feasible (otherwise the sink cannot attend the scheduling plans selected by the sensors); this means that an assignment of scheduling plans to all the sensors in a slot is feasible only if the resulting scheduling plan for the sink has a duty cycle not above 100 %.

Let $G = g_{i,k}$ and $E = e_{i,k}$ be two $n \times v$ matrices that represent the amount of energy generated and estimated, respectively, $\forall i \in [1, n]$ and $\forall k \in [1, v]$ (note that the energy that is finally produced may not match with the expected one). Let also $B = b_{i,k}$ be the $n \times v$ matrix that represents the battery level for each sensor and for the sink in each slot k. Referring to the current time slot t, matrix B contains the past battery levels of the sensors and the sink in all the columns $k < t$, the current battery levels in column t, and the expected battery levels in all columns $k > t$. Finally, we denote as $B_i^{\max}$ and $B_i^{\min}$ to the maximum and minimum admissible levels of the batteries, respectively, and as B_i^0 to the initial battery level at the beginning of slot 1 for any sensor $i \in [1, n]$. We define the rectifier function $[x]^+$ as:

$$[x]^+ = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (2)$$

If the duration of the slots is small enough, we can assume that, if the charge tends to overflow the battery above its maximum $B_i^{\max}$, then the battery charge will be $B_i^{\max}$ also at the end of the slot. Similarly, if the charge tends to underflow the battery below $B_i^{\min}$, then the battery charge will be $B_i^{\min}$ also at the end of the slot. This assumption is motivated by the fact that the slots are short, the energy production will keep the same trend while the energy consumption will remain almost the same. Under these assumptions, disregarding the power leakage of the battery, and assuming that the actual energy production equals the expected energy production, we can express the battery level for each sensor i in slot k as:

$$b_{i,k} = \begin{cases} \max\{B_i^{\max}, b_{i,k-1} + \phi[e_{i,k} - c(s_{i,k})]^+ - [c(s_{i,k}) - e_{i,k}]^+\} & k \in [2, v] \\ B_i^0 & k = 1 \end{cases} \quad (3)$$

where ϕ is the efficiency of the battery recharge process. In order to simplify the formulation of the optimization problem, we provide the following definitions.

Definition 1 A sensor i satisfies the condition of energy neutrality if $b_{i,v} \geq b_{i,1}$, i.e., its battery level at the last slot v is always greater or equal than the battery level at slot 1. For evaluating this condition, we define the boolean function `Neutral(i)` that returns `true` if the sensor i holds this condition.

Definition 2 A sensor i satisfies the condition of minimal battery if $b_{i,k} \geq B_i^{\min}$, i.e., the battery level $\forall k \in [1, v]$ is always greater or equal than its minimum battery level $B_i^{\min}$. We define the boolean function `Minbat(i)` that returns `true` if the sensor i holds this condition.

Definition 3 An assignment of scheduling plans to the sensors in slot x is feasible if $\sum_{i=1}^{n-1} dc(s_{i,x}) \leq 100$ %, i.e., the sum of their duty cycles is lower or equal than the maximum possible (100 %); otherwise, these scheduling plans cannot run together because the sink has not enough capacity to run them simultaneously in the same slot. We define the boolean function `Sumdc(x)` that returns `true` if the assignment done in slot x holds this condition.

Definition 4 An assignment of scheduling plans represented by matrix S is feasible if holds that `Neutral(i)`, `Minbat(i)`, and `Sumdc(k)` $\forall i \in [1, n], k \in [1, v]$. We define the boolean function `Feasible(S)` that returns if the assignment is feasible or not. For the sake of simplicity, we also define the boolean function `Feasible(`$s_{i,k}$`)` that returns `true` if the assignment to sensor i in slot k is feasible.

Then, we can formulate the optimization problem as follows:

$$\max \sum_{\substack{i\in[1,n],k\in[1,v] \\ \texttt{Feasible(S)}}} q(s_{i,k}) \tag{4}$$

The constraint `Feasible(S)` states that the assignment of the scheduling plans to slots must be always feasible, i.e., each sensor i must be energy neutral and hold the condition of minimal battery, and the sum of the duty cycles of the assigned scheduling plans to slot k must be always less or equal than 100 %.

Energy management optimization

We present in this section the algorithms for energy management optimization. Firstly, we consider the initial assignment problem of scheduling plans to slots for each sensor, assuming that the energy produced in the sensors coincides with the energy estimated. To this purpose, we exploit the energy production model presented in "Initial assignment" section. Note that the use of this model lets the algorithm to estimate the ideal energy production for the entire day, and thus to assign scheduling plans to be executed at night that are admissible if the actual production meet the ideal production. On the other hand, the actual production could be different from expected. For example, there can be an excess or defect of energy production in the sensors with regard to the estimation done, or because some sensors may leave or join the network. To deal with these cases, our energy management algorithm re-schedules the current assignment. This section concludes by comparing centralized and distributed implementations of the algorithm.

Initial assignment

The purpose of the initial assignment algorithm is to find a feasible assignment of scheduling plans to slots for each sensor. Such initial assignment may later be upgraded or downgraded by increasing or reducing the quality level of the scheduling plans initially assigned. The pseudo-code for the initial assignment is presented in Algorithm 1.

Algorithm 1 Initial Assignment Algorithm

```
Ensure: S: the scheduling matrix.
  s_{i,k} = P_i[1]; s_{n,k} = ⋃_{i=1}^{n-1} p(s_{i,k}) ∀k ∈ [1, v]
  loop
    case 1: Feasible(S)
              if(b_{i,v} == b_{i,1}) ∀i then
                  exit; % the solution is optimal
              improved=Upgrade(S)
              break;
    case 2: !Feasible(s_{i,k}) for some i ≠ n and some k
              improved=Downgrade(S,i)
              break;
    case 3: !Feasible(s_{n,k}) for some k
              improved=DowngradeAll(S)
    if (improved ==false) then
              exit; % the solution cannot be upgraded/downgraded
  end loop
```

The algorithm starts assigning to each sensor i its most efficient scheduling plan, i.e., the scheduling

plan $P_i[1]$ to all of the slots of sensor i. The scheduling plan to be executed in the sink is computed in each scheduling plan replacement accordingly. Once the matrix S has been initialized, three cases may occur: (1) feasibility of all the assignments (case 1); (2) non-feasibility of the assignment for a sensor $i < n$ in some slot (case 2); and (3) non-feasibility of the assignment in the sink (case 3). The first case occurs when all the sensors (included the sink) are energy neutral, their battery levels at every slot are above B_i^{min} and the sum of the duty cycles of the scheduling plans assigned to all the sensors in each slot k is below 100%. In this case, if the battery level at last slot v is equal to the battery level at slot 1 for all sensors, then there is no excess of battery to improve the current solution and the algorithm finishes; otherwise, the solution can be improved and the algorithm invokes function `Upgrade` with the purpose of iteratively finding a better assignment than the current one. In the second case, the algorithm uses function `Downgrade` with the purpose of iteratively finding a cheaper (less energy demanding) assignment for the sensor that is not energy neutral. In the last case, the algorithm uses function `DowngradeAll` with the purpose of iteratively finding a cheaper assignment for all the sensors, in order to reach a feasible assignment for all the sensors. In any of the three cases, the algorithm iterates until no improvements are possible to the scheduling plans assignment.

Algorithm 2 shows the function `Upgrade`. This function takes the assignment matrix S and selects the scheduling plan of the sensor whose replacement provides the largest increase on the current quality (of course, the new assignment will also have a larger cost). To this purpose, the `Upgrade` function considers the *upgradable* scheduling plans, i.e., the assigned scheduling plans that can be replaced with a scheduling plan with higher quality. Specifically, given an assigned scheduling plan $s_{i,k}$ with a quality level $q = q(s_{i,k})$, it is said to be upgradable if the list $P_i[]|_q \neq \emptyset$, i.e., there exists a scheduling plan that improves the quality level of the current assignment. We define the function `up(`$s_{i,k}$`)` that returns the scheduling plan in $P_i[]|_q$ with minimum cost (and with a larger quality than q) and that returns the empty set $\emptyset$ if such a scheduling plan does not exists. In practice, the `Upgrade` function looks for an assigned scheduling plan that is upgradable with minimum cost and upgrades it only if this results in an overall admissible assignment.

Algorithm 2 Upgrade Function

```
Require: S: the scheduling plans assignment matrix
Ensure: a boolean value that states if S has been upgraded
  Let U={s(i,k) s.t. i ∈ [1, n − 1], k ∈ [1, v] and s(i,k) is upgradable}
  % U is the set of assigned scheduling plans that are upgradable
  Let s(h,x) ∈ U be a plan s.t. c(up(s(h,x))) ≤ c(up(s(i,k))) ∀ s(i,k) ∈ U
  % s(h,x) is the assigned plan that is upgradable with minimum cost
  S'=S; s'(h,x)=up(s'(h,x)); % upgrades the plan of sensor h in slot x
  s'(n,x)=⋃_{i=1}^{n−1} p(s'(i,x)) % updates the state of the sink
  % considers an assignment in which s(h,x) is replaced with up(s(h,x))
  if Feasible(S') then
    S=S';
    return true
  return false
```

Algorithm 3 shows the function `Downgrade`. This function takes the assignment matrix S and a sensor i and attempts to reduce the cost only for the assignment of the sensor i (of course, the new assignment will also have a lower quality). To this purpose, the `Downgrade` function considers the *downgradable* scheduling plans, i.e., the assigned scheduling plans that can be replaced with a scheduling plan with lower cost. Specifically, given an assigned scheduling plan $s_{i,k}$ with a cost $c = c(s_{i,k})$, it is said to be downgradable if the list $P_i[]|^c \neq \emptyset$, i.e., there exists a scheduling plan that reduces the cost of the current assignment. We define the function `down(`$s_{i,k}$`)` that returns the scheduling plan of maximum quality in $P_i[]|^c$ (and with a cost lower than c) and that returns the empty set $\emptyset$ if such a scheduling plan does not exists. In practice, the `Downgrade` function looks for an assigned scheduling plan that is downgradable with maximum quality.

Algorithm 3 Downgrade Function

```
Require: S: the scheduling plans assignment matrix.
Require: α: the sensor to be downgraded.
Ensure: a boolean value that states if the assignment S of
        sensor i has been downgraded
  Let D={s(i,k) s.t. i = α, k ∈ [1, v] and s(i,k) is downgradable}
  % D is the set of assigned scheduling plans in S that
  are downgradable
  if D==∅ then
    return false% solution cannot be downgraded
  Let s(h,x)∈D be a plan s.t. q(down(s(h,x))) ≥ q(down(s(i,k)))
  ∀s(i,k)∈D
  % s(h,x) is the assigned plan that is downgradable with
  maximum quality
  s(h,x)=down(s(h,x)); % downgrades s(i,k)
  s(n,x)=⋃_{i=1}^{n−1} p(s(i,x)) % updates the state of the sink
  return true
```

Finally, Algorithm 4 shows the function `DowngradeAll`. This function takes the assignment matrix S and attempts to reduce the overall cost of the assignment (of course, the new assignment will

also have a lower quality). This function is similar to function `Downgrade`, with the difference that it is not limited to downgrade only a specific sensor.

Algorithm 4 DowngradeAll Function

Require: S: the scheduling plans assignment matrix.
Ensure: a boolean value that states if the assignment S has been downgraded
Let D={s(i,k) s.t. $i \in [1, n-1], k \in [1, v]$ and s(i,k) is downgradable }
% *D is the set of assigned scheduling plans in S that are downgradable*
if D==$\emptyset$ **then**
  return false% *solution cannot be downgraded*
Let s(h,x)$\in$ D be a plan s.t. q(down(s(h,x))) $\geq$ q(down(s(i,k)))
$\forall$s(i,k)$\in$ D
% *s(h,x) is the assigned plan that is downgradable with maximum quality*
s(h,x)=down(s(h,x)); % *downgrades s(h,k)*
s(n,x)=$\bigcup_{i=1}^{n-1}$ p(s(i,x)) % *updates the state of the sink*
return true

Re-optimization

The strategy described computes a (sub-)optimal scheduling of the scheduling plans to be executed along a time frame, such that the overall quality level is maximized both in the sensors and in the sink. This scheduling is based on estimations of the energy to be produced by the energy harvesting systems attached to the sensors and to the sink, as well as the energy consumption of each specific scheduling plan. However, the amount of energy that is actually harvested from the sun could differ from the amount of energy predicted at some slot along the time frame taken as reference. This may happen for different reasons, as for instance the unexpected change of solar conditions or a wrong prediction. Note that the amount of the real energy harvested can be obtained by reading the battery sensor at the instant of time appropriate. If it is observed a meaningful deviation between the estimated and the harvested energy, i.e., if the amount of estimated energy at some slot is higher (or lower) than the amount of energy that is really produced, the scheduling for the future slots is no longer valid and the re-optimization is triggered (note that small differences between the estimated and produced energy are still tolerable). If this is the case, re-optimization makes necessary only for the next slots, i.e., the slots between the following slot to the slot in which the energy estimation changed, let say v_0 and the last slot v. The re-optimization algorithm behaves equal to Algorithm 1 except that it starts the initial assignment of scheduling plans from slot v_0 instead of from slot 1.

Finally, we want to stress that in our task model we assume a cyclic execution of tasks with a uniform consumption in each slot. Under the event of the battery level is very low and the energy production drops dramatically between the beginning and the end of slot (which would lead the sensor to interrupt its operation), we consider that there exist a minimum admissible battery level (B_i^{min}) that enables sustaining the sensor operation during one slot, after which the re-optimization will be triggered for the next slots.

Centralized vs. distributed communication

The algorithms presented so far allow for both a centralized and distributed implementations. However, the choice of communication (centralized or distributed) impacts significantly on the efficiency of the solution. The main difference between the centralized and distributed model lies in *who* has the knowledge to make the decisions. For the sake of simplicity, we consider two extreme cases. One, centralized, in which the sensors do not have any knowledge (apart the scheduling plan assigned to them by the sink), and one (that we call *distributed*), in which the sensors know all the parameters of their tasks (basically quality and consumption), and they can perform the optimization of their own scheduling, while the sink do not have specific knowledge of the sensors' task parameters (energy consumption and quality). The distributed scenario models the case in which the set of sensors participating to the network is not defined a priori, new sensors with new sets of tasks may join the network. In these conditions, when a new sensor joins, the sink chooses the set of tasks that fit the sensor ones (e.g., tasks that publish the sensor data at the appropriate rate), but it does not know the specific parameters of the tasks of the sensor (consumption, production, and quality). Note, however, that there are many different ways to implement distributedly the algorithms, depending on the assumption made about the capability of the sensors and on the knowledge they have. Note also, that, regardless the choice of the communication and, according to our algorithms, the sink is always responsible of deciding about its own scheduling, based on the scheduling of the sensors, which can be made by the sensors (in the distributed case) or by the sink itself (in the centralized case).

Considering these two opposite scenarios, we compute the overhead of these approaches due to communication messages that the sink and the sensors interchange.

- **The centralized approach.** Following this approach, the sink acts as coordinator node by centralizing the global knowledge, making decisions for upgrading/downgrading each sensor with a scheduling plan, and driving the communication with the sensors. The sink knows the properties of the scheduling plans of the sensors, their initial battery levels, and their energy estimations, therefore, it may compute in advance the scheduling plan to be assigned to every slot in each sensor and, correspondingly, the scheduling plan to be executed in each slot in the sink. To communicate their scheduling the sink needs to send one message to each sensor. If the precision of the energy solar prediction algorithm is good enough, it will not be necessary to send more than $(n-1)$ messages. Otherwise, if re-optimization is necessary (for instance due to a non-tolerable difference between the energy estimated and produced), the sensor that detects such imprecision has to communicate to the sink the amount of energy that is really produced and the slot where it was detected, in such a way that the sink can proceed to recompute the assignment for the next slots (between v_0 and v) of that sensor, as well as to recompute its own scheduling. This involves two messages more for each slot: one from the sensor directed to the sink and the other one from the sink directed to the sensor. Thus, the number of messages interchanged in the centralized approach depends on the number of sensors $n-1$, the number of slots v but, more importantly, on the precision of the energy prediction algorithm, and it goes to range between $n-1$ messages (the best case, it occurs when no re-optimization must be done) and $3v(n-1)$ (the worst case, it occurs when re-optimization must be done for all slots in all sensors).
- **The distributed approach.** In the distributed approach, the initial assignment is triggered when the sink sends to each sensor the order to start assigning its most efficient scheduling plan to all slots. In turn, each sensor i responses to the sink if, after the assignment of the most efficient scheduling plan to the v slots, it is still feasible or not, e.g., $\texttt{Feasible}(s_{i,k})\ \forall k \in [1, v]$ as well as the scheduling plan assigned, since the sink has to adapt its scheduling plan accordingly. Therefore, in this first step, the number of messages interchanged between the sink and the sensors sum up to $2(n-1)$ messages. After that, three cases may happen: (1) upgrade all the sensors; (2) downgrade a subset of the sensors; and (3) downgrade all the sensors. In the first case, if the sink finds feasible all the assignments of all sensors, broadcasts a message ordering upgrade ($(n-1)$ messages). In the second case, if the sink finds that some assignment in some sensor (different to the sink) is not feasible, sends a message towards the subset of sensors whose assignment is not feasible (a number of messages lower than $(n-1)$). In the third case, the sink broadcasts a message ordering downgrade to all the sensors ($(n-1)$ messages). The reception of any of these messages implies that each sensor starts independently to upgrade/downgrade its solution and, after the replacement of one single scheduling plan in one single slot, each one must send back a new message to the sink indicating the possibly new scheduling plan assigned and if it improved or reduced the cost of its solution (note that if the solution was not improved or its cost was not reduced, the sensor undoes the replacement and leaves the old scheduling plan). This is necessary because the change of one scheduling plan in one sensor involves a change in the scheduling plan in the sink. For each message received, the sink selects its new scheduling plan accordingly to the new scheduling plan received from a sensor, and checks again if it is still energy neutral. At this point, the three cases previously mentioned could happen again. The process repeats until there are no more scheduling plans that improve the solution (or that reduce its cost). Therefore, the maximum number of messages interchanged between the sink and the sensors by using a distributed approach in the initial assignment is $2(n-1)+2(n-1)((m-1)v) = 2(n-1)(1+v(m-1))$. Thus, in this case, the number of messages interchanged between the sensors and the sink depends on the number of sensors $n-1$, the number of scheduling plans m, and the number of slots v. Note that the re-optimization does not impact on

Table 2 Maximum number of messages interchanged between the sensors and the sink for different values of n, m, and with $v = 300$ for centralized (best and worst case) and distributed communication

n	5	5	6	6	7	7	8	8	9	9
m	4	6	4	6	4	6	4	6	4	6
Central-best	4	4	5	5	6	6	7	7	8	8
Central-worst	3600	3600	4500	4500	5400	5400	6300	6300	7200	7200
Distributed	7208	12,008	9010	15,010	10812	18,012	12,614	21,014	14,416	24,016

the number of messages in the distributed communication, because the sensors always have to send any change done in their scheduling to the sink. For this reason, there is no a best and a worst case since the distributed scenario is always the same.

In order to compare the centralized approach against the distributed one, we provide values for $n \in [5, 9]$, $m = \{4, 6\}$, and $v = 300$, and we compute the number of messages in the three cases: distributed and centralized, best, and worst case. Table 2 shows the number of resulting messages; in the case of the distributed approach, the number of messages increases when n and m grow. In the case of the centralized approach, the number of messages increases with n and not with m, but depends on the accuracy of the prediction algorithm. As observed, the number of messages interchanged by using the centralized approach is much lower than in the distributed approach; this is especially true when re-optimization is not required, i.e. when the number of slots with a wrong energy estimation tends to 0.

Evaluation

We have evaluated the algorithms described in "Energy management optimization" section by simulation. With the purpose of evaluating a large number of scenarios the simulator first generates a valid test case and then proceeds to its optimization. This section presents the description of the simulator and the results obtained for a subset of selected test cases. We stress here that the hardware platform shown in Fig. 1 was used in this evaluation to define the simulation parameters and to configure the energy harvesting parameters. We use simulation for the evaluation of our energy efficiency strategy in order to have the flexibility of simulating the algorithm on a system with an arbitrary number of sensors.

The cases generator considers $n - 1$ sensors and m scheduling plans per sensor to produce $n - 1 \times m$ scheduling plans. For each scheduling plan, it randomly generates a set of tasks θ (i.e., the number of tasks is not fixed), and for each task $T \in \theta$ generates its amount of execution time t, its period p (with $p > t$), and its cost per unit of time w. Remember that the special task T_0 is executed when there is no other task to execute. Knowing the description of each task $T = \langle t, p, w \rangle$, the duty cycle of the scheduling plan as well as its cost can be computed as described in "System model" section while its quality is randomly generated by taking into account that a larger duty cycle and a larger cost involves necessarily higher quality. Conversely, the cases generator must also generate $n - 1 \times m$ scheduling plans for the sink, where each one corresponds to one and only one scheduling plan of the sensor. The cost of a scheduling plan in the sink is generated (quasi-)randomly by considering that it should not be less than the cost of its corresponding scheduling plan in the sensor (to model the additional data processing and aggregation work performed by the sink). Since the sink has to simultaneously attend to $n - 1$ sensors, its duty cycle comes given by the sum of the duty cycles of the corresponding scheduling plans executed by the sensors in the same instant of time.

We have considered a time frame of $\Delta = 24$ h, composed of 300 slots ($v = 300$) with a duration of $\frac{\Delta \times 60 \times 60}{v}$=288 s each one. The energy produced in an hour E(t) is equally distributed among all the slots of an hour.

In our simulations, the n sensors (including the sink) employ the energy harvesting system described in "Solar cells-based sensors" section. Our simulator estimates the energy production in the city of Madrid (latitude = 40.24 North) that provides the solar module KL-SUN3W, according to the model described in "Solar cells-based sensors" section. Specifically, for each day of the year, it first computes

the hourly irradiance $D(t)$ as the product between D and $cos\Theta_Z$ and then it calculates the energy production at time $t \in [0, 23]$ as $E(t) = \frac{D(t)\times\eta\times S}{Vmp}$. The values for the average daily irradiance D in Madrid that we took were the data provided by RETScreen: 2.03,2.96,4.29,5.11,5.95,7.09,7.2,6.34,4.87,3.13,2.13, 1.7 $KWh/m^2/day$, corresponding to the months of the year from January to December. Note that, at implementation time, the programmer must provide as an input the data of irradiance (D) corresponding to the specific geographical location where the sensors will be located in order to compute the amount of solar energy that is harvested from the sun at that location, according to our energy prediction model. To enable the sink attend simultaneously several sensors, we increase the energy production in the sink by assuming that the energy to be produced in the sink is f times the energy production in the sensors and, for the sake of simplicity, all the sensors produce the same amount of energy. The maximum capacity of the battery attached to the sensors is $B_i^{\max} = 6600$ mAh$\forall i \in [1, n)$ and for the sink is the double $B_n^{\max} = 13,200$ mAh; $B_i^{\min} = \frac{B_i^{\max}}{2} \forall i \in [1, n]$ and, in slot 1, the battery levels were configured to be $B_i^0 = B_i^{\max} - 500 \forall i \in [1, n]$.

The strength of our simulator is the diversity of scenarios that it can reproduce. Our simulator implements the optimization algorithms proposed with the capability of emulating different real-world and heterogeneous energy-harvesting platforms (i.e., the combination of a solar cell and a sensor node) that form the energy harvesting network. Each sensor node could be connected to a different solar cell, therefore with different opportunities of scavenging, and each sensor node executes a different set of potentially unlimited of scheduling plans with a variable number of applications and, therefore, with different consumption curves. The network size, the deployment place, and the number of slots (and, consequently, its duration) in the time frame of 24 h can also be configured.

We have generated and evaluated a set of experiments. From them, we have selected the next three, that correspond to the three cases distinguished by the Initial Assignment algorithm: (1) case 1: upgrade; (2) case 2: downgrade sensors that are not energy-neutral; and 3) case 3: downgrade all. The day used for estimating the energy production was December 21st, winter solstice ($d = 358$) and the starting hour was 0:00 am. For each case, we configure the cases generator to create a number of instances different, where each instance corresponds to a simulation of n sensors with m different scheduling plans each one. First, the next three subsections analyze separately one only instance of each case and later, in the last subsection we provide the global results for the complete set of instances.

Case 1: upgrade

We study the case of upgrading for $n = 5$ sensors (including the sink) and $m = 6$ scheduling plans. Table 3 shows the details of the scheduling plans of a specific instance for this case. For each scheduling plan, the columns from 2 to 6 present the definition of the tasks 1 to 5, respectively, (a maximum number of 6 tasks was fixed); the column 7 indicates the execution time for the Idle task; the next three columns indicate the cost per unit of time of the scheduling plan, its quality level, and its duty cycle, respectively. The last column indicates the number of order of the scheduling plan by efficiency (e.g., the lowest order number corresponds to the largest efficiency).

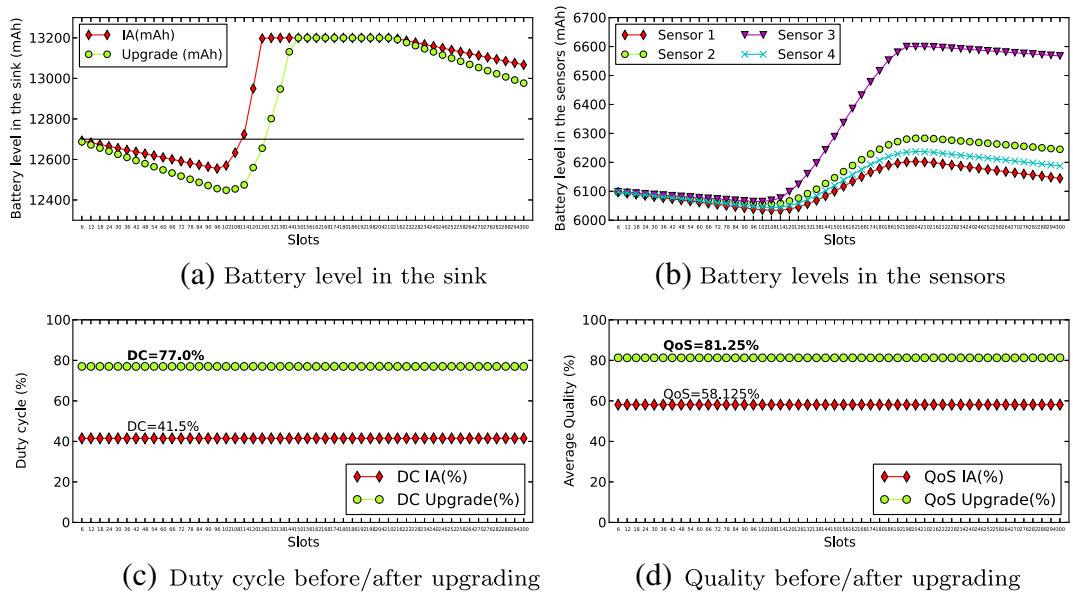
After assigning the most efficient scheduling plan to each sensor, the duty cycle is 41.5 % and all the sensors including the sink are energy-neutral. There is, therefore, a choice of improving the solution for which the Algorithm 1 starts upgrading the sensors. Figure 4a shows the battery level in the sink, and Fig. 4c, d depicts the duty cycle and the resulting quality level after upgrading, respectively. As observed, the sink keeps energy-neutral after the first scheduling plan is assigned (red line) and also after upgrading (green line). Since the process of upgrading improves the solution by increasing necessarily its cost, the battery level at slot $v = 300$ is slightly lower than the battery level when the initial scheduling plan, which is cheaper, is assigned. The effect of upgrading the solution increases its duty cycle and, in turn, its quality: the duty cycle grows from 41.5 % up to 77.0 % (see Fig. 4c) while the overall quality grows from 58.125 % up to 81.25 % (see Fig. 4d). Note that in these figures, we are overlapping the results of the assignments in two situations: after initial assignment (IA) and after upgrading/downgrading. It is important to point out that, in these cases, the axis x, which is showing the slots of time in a day, does

Table 3 Scheduling plans for case 1

	T_1	T_2	T_3	T_4	T_5	T_0	c	q	dc	O
Sensor 1										
P1	⟨5, 39, 13⟩	⟨8, 39, 17⟩	⟨9, 39, 13⟩	⟨6, 39, 8⟩	⟨9, 39, 7⟩	T_0^2	1.58	51.0	5.5	2
P2	⟨3, 20, 20⟩	⟨5, 20, 16⟩	–	–	–	T_0^{12}	3.0	74	9	3
P3	⟨4, 16, 1⟩	⟨4, 16, 18⟩	⟨4, 16, 11⟩	⟨3, 16, 11⟩	–	T_0^1	2.68	63	7	5
P4	⟨4, 17, 4⟩	⟨1, 17, 3⟩	⟨1, 17, 7⟩	–	–	T_0^{11}	2.11	53	6	4
P5	⟨4, 24, 4⟩	⟨6, 24, 3⟩	⟨3, 24, 8⟩	⟨3, 24, 5⟩	⟨6, 24, 6⟩	T_0^2	1.25	50	4.5	1
P6	⟨2, 11, 11⟩	⟨2, 11, 5⟩	⟨2, 11, 1⟩	⟨1, 11, 19⟩	⟨2, 11, 10⟩	T_0^2	4.54	87	15	6
Sensor 2										
P1	⟨6, 38, 6⟩	⟨6, 38, 20⟩	⟨9, 38, 15⟩	⟨5, 38, 2⟩	–	T_0^{12}	1.76	48.5	6	1
P2	⟨3, 21, 13⟩	⟨1, 21, 12⟩	⟨3, 21, 6⟩	⟨5, 21, 16⟩	–	T_0^9	3.09	77	16	2
P3	⟨3, 17, 11⟩	⟨1, 17, 12⟩	–	–	–	T_0^{13}	2.88	51	11	5
P4	⟨2, 26, 6⟩	⟨2, 26, 15⟩	–	–	–	T_0^{22}	2.5	50	9	4
P5	⟨1, 13, 17⟩	⟨2, 13, 3⟩	–	–	–	T_0^{10}	3.07	52	14	6
P6	⟨1, 31, 14⟩	–	–	–	–	T_0^{30}	2.38	49.5	7	3
Sensor 3										
P1	⟨6, 25, 2⟩	⟨1, 25, 15⟩	⟨6, 25, 6⟩	⟨5, 25, 19⟩	⟨2, 25, 14⟩	T_0^5	2.64	83	23	1
P2	⟨2, 32, 14⟩	⟨6, 32, 12⟩	–	–	–	T_0^{24}	2.31	62	9	3
P3	⟨5, 23, 7⟩	⟨5, 23, 14⟩	–	–	–	T_0^{13}	2.04	54	7	5
P4	⟨5, 40, 12⟩	⟨5, 40, 4⟩	–	–	–	T_0^{30}	1.9	51	5.5	4
P5	⟨1, 29, 20⟩	⟨3, 29, 9⟩	⟨3, 29, 2⟩	⟨6, 29, 7⟩	⟨3, 29, 6⟩	T_0^{13}	2.41	75	12	2
P6	⟨4, 23, 1⟩	⟨4, 23, 15⟩	–	–	–	T_0^{15}	2.0	53	6	6
Sensor 4										
P1	⟨8, 40, 1⟩	⟨9, 40, 8⟩	⟨9, 40, 10⟩	–	–	T_0^{14}	1.17	51	8	1
P2	⟨7, 36, 20⟩	⟨5, 36, 12⟩	–	–	–	T_0^{24}	2.22	54	11	3
P3	⟨1, 13, 3⟩	–	–	–	–	T_0^{12}	2.07	53	9	2
P4	⟨3, 13, 4⟩	⟨3, 13, 17⟩	⟨2, 13, 17⟩	–	–	T_0^5	3.69	68	21	5
P5	⟨4, 19, 19⟩	⟨1, 19, 7⟩	⟨1, 19, 16⟩	⟨3, 19, 2⟩	–	T_0^{10}	3.36	60	19	6
P6	⟨1, 13, 16⟩	⟨1, 13, 5⟩	⟨1, 13, 10⟩	–	–	T_0^{10}	3.92	78	23	4

Legend: *P* plan, *O* order, *q* quality, *dc* duty cycle given in percentage

not represent absolute time, since clearly the upgrading/downgrading process occurs later, but the relative time within the time frame in which the scheduling plan that upgrades/downgrades is assigned, and its consequent profit with regard to the scheduling plan assigned by the IA algorithm at the same slot. Figure 4b demonstrates that all sensors remain energy-neutral after upgrading. As observed, they experience a reduction of their battery levels during the hours of low (or null) energy production and an increase coinciding with the hours of solar light (remember that slot 1 coincides with the instant of time 0:00 am). The battery levels vary slightly among sensors, since each sensor executes a different scheduling plan with a different consumption (note that, for the sake of simplicity, we assume that the energy production is the same for all sensors). During the upgrading process, the sensors 1, 2, and 4 were selected for upgrading according to the Algorithm 2, i.e., first the scheduling plans that are upgradable are selected, and among them, it is selected the scheduling plan whose replacement provides the smallest increase in the cost of the solution. Note that the scheduling plans selected for replacement could not be the most efficient scheduling plans. For this case, we checked that the sensors 1, 2, and 4 transit progressively to each one of the



(a) Battery level in the sink (b) Battery levels in the sensors

(c) Duty cycle before/after upgrading (d) Quality before/after upgrading

Fig. 4 **Case 1**: Battery levels in the sink and in the sensors after upgrading (*above*); duty cycle and quality (*below*) after the initial assigning (IA) of the most efficient scheduling plan and after upgrading

$m = 6$ possible scheduling plans, until the solution cannot be improved, which occurs when the quality level achieves 81.25 % and the duty cycle achieves 77 %.

Case 2: downgrade sensors different to the sink

This case uses $n = 8$ sensors (including the sink) and $m = 5$ scheduling plans (see Table 4 for details). The assignment of the most efficient scheduling plan to each sensor may make some sensor (different than the sink) not energy-neutral; specifically, as shown in Fig. 5 on the left, these sensors are the sensor 1, 2, 3, and 7. Thus, the Algorithm 1 proceeds to downgrade all of them until finding a feasible solution. The `Downgrade` function (Algorithm 3) proceeds by selecting for each sensor to downgrade, a cheaper scheduling plan than the current one but whose replacement involves the lowest reduction on the quality of the solution. The transitions required for sensors 1, 2, 3, and 7 become energy-neutral can be viewed in Fig. 6, where we have used the red color to show the initially assigned scheduling plan, the green color to show the final scheduling plan assigned, and yellow color to show the intermediate transitions.

Note that downgrading the sensors involves also downgrading the sink as shown in Fig. 7a, where the battery level that is obtained after assigning the first scheduling plan and after the downgrading process is compared. As observed, the battery level after downgrading (green line) is slightly larger than the battery level after assigning the first scheduling plan; the reason for that is that also in the sink a cheaper scheduling plan has to be selected. The downgrading process impacts also on the duty cycle and the quality of the solution such as shown in Fig. 7b and c, respectively: the lower cost of the scheduling plan, the lower duty cycle and, in turn, the lower quality. Specifically, the duty cycle drops from 76 % to 39–52 %, depending on the set of scheduling plans selected for the sensors in each specific slot. Similarly, the quality drops from 67.14 % to 50.85–54.5 %, again depending of the scheduling plans selected for the sensors.

Case 3: downgrade all

The third case uses a number of $n = 10$ sensors (including the sink) and $m = 5$ scheduling plans. Table 5 shows the details of the scheduling plans generated.

Table 4 Scheduling plans for case 2

	T_1	T_2	T_3	T_4	T_5	T_0	c	q	dc	O
Sensor 1										
P1	⟨4, 27, 20⟩	–	–	–	–	T_0^{23}	2.44	49.5	7	3
P2	⟨7, 34, 12⟩	⟨8, 34, 18⟩	⟨4, 34, 14⟩	⟨5, 34, 5⟩	⟨7, 34, 3⟩	T_0^{3}	1.70	48.5	6	2
P3	⟨1, 13, 11⟩	–	–	–	–	T_0^{12}	2.69	51	20	5
P4	⟨2, 35, 20⟩	⟨3, 35, 16⟩	⟨1, 35, 9⟩	⟨8, 35, 18⟩	⟨3, 35, 8⟩	T_0^{18}	3.05	91	24	1
P5	⟨1, 24, 11⟩	⟨1, 24, 5⟩	–	–	–	T_0^{22}	2.5	50	10	4
Sensor 2										
P1	⟨2, 36, 15⟩	⟨2, 36, 12⟩	⟨2, 36, 3⟩	–	–	T_0^{30}	2.5	60	5.5	1
P2	⟨7, 35, 16⟩	–	–	–	–	T_0^{28}	2.05	51	3.5	2
P3	⟨3, 21, 14⟩	⟨2, 21, 4⟩	⟨1, 21, 16⟩	⟨1, 21, 6⟩	–	T_0^{14}	3.23	61	6	4
P4	⟨1, 22, 14⟩	⟨4, 22, 7⟩	⟨4, 22, 5⟩	⟨1, 22, 16⟩	⟨4, 22, 19⟩	T_0^{8}	3.5	64	8	5
P5	⟨2, 18, 10⟩	⟨3, 18, 5⟩	–	–	–	T_0^{13}	2.27	53	4.5	3
Sensor 3										
P1	⟨3, 28, 18⟩	⟨7, 28, 19⟩	⟨6, 28, 15⟩	–	–	T_0^{12}	2.71	70	17	2
P2	⟨3, 33, 12⟩	⟨1, 33, 7⟩	⟨7, 33, 12⟩	⟨6, 33, 17⟩	–	T_0^{16}	2.42	69	15	1
P3	⟨1, 15, 15⟩	⟨2, 15, 15⟩	⟨2, 15, 9⟩	⟨1, 15, 3⟩	⟨3, 15, 1⟩	T_0^{6}	3.66	72	23	5
P4	⟨6, 28, 9⟩	⟨5, 28, 14⟩	–	–	–	T_0^{17}	2.03	52	6	4
P5	⟨7, 37, 12⟩	⟨1, 37, 3⟩	–	–	–	T_0^{29}	1.97	51	5.5	3
Sensor 4										
P1	⟨2, 26, 2⟩	–	–	–	–	T_0^{24}	1.92	50	3.5	2
P2	⟨5, 27, 15⟩	⟨1, 27, 19⟩	⟨5, 27, 19⟩	–	–	T_0^{16}	3.14	73	6	3
P3	⟨1, 20, 3⟩	⟨5, 20, 13⟩	–	–	–	T_0^{14}	2.2	57	5.5	1
P4	⟨4, 16, 17⟩	⟨4, 16, 20⟩	⟨1, 16, 5⟩	⟨3, 16, 15⟩	⟨3, 16, 6⟩	T_0^{1}	4.06	80	14	5
P5	⟨6, 27, 2⟩	⟨2, 27, 20⟩	⟨4, 27, 2⟩	⟨6, 27, 16⟩	–	T_0^{9}	2.14	52	4.5	4
Sensor 5										
P1	⟨1, 17, 6⟩	⟨1, 17, 13⟩	⟨2, 17, 16⟩	–	–	T_0^{13}	3.58	50	9	5
P2	⟨6, 31, 4⟩	⟨6, 31, 4⟩	⟨5, 31, 2⟩	⟨2, 31, 16⟩	⟨4, 31, 2⟩	T_0^{8}	1.41	47.5	4.5	1
P3	⟨7, 35, 9⟩	⟨2, 35, 7⟩	⟨6, 35, 19⟩	–	–	T_0^{20}	2.14	49.5	6	3
P4	⟨3, 15, 15⟩	⟨3, 15, 18⟩	⟨3, 15, 12⟩	⟨1, 15, 13⟩	⟨3, 15, 3⟩	T_0^{2}	4.33	79	24	4
P5	⟨3, 24, 2⟩	–	–	–	–	T_0^{21}	1.83	48.5	5.5	2
Sensor 6										
P1	⟨3, 12, 1⟩	–	–	–	–	T_0^{9}	1.58	48.5	5.5	1
P2	⟨6, 37, 20⟩	⟨3, 37, 8⟩	–	–	–	T_0^{28}	2.27	50	7	3
P3	⟨6, 37, 16⟩	⟨5, 37, 10⟩	–	–	–	T_0^{26}	2.10	49.5	6	2
P4	⟨3, 16, 20⟩	⟨1, 16, 14⟩	–	–	–	T_0^{12}	3.62	71	14	5
P5	⟨4, 21, 15⟩	⟨5, 21, 15⟩	–	–	–	T_0^{12}	2.57	53	13	4
Sensor 7										
P1	⟨2, 17, 13⟩	⟨4, 17, 5⟩	⟨2, 17, 19⟩	⟨1, 17, 1⟩	–	T_0^{8}	3.17	50	9	5
P2	⟨2, 25, 17⟩	⟨3, 25, 19⟩	⟨5, 25, 10⟩	⟨4, 25, 18⟩	⟨3, 25, 18⟩	T_0^{8}	3.92	97	16	1
P3	⟨1, 29, 19⟩	⟨5, 29, 3⟩	–	–	–	T_0^{23}	2.34	49.5	7	4
P4	⟨3, 38, 4⟩	–	–	–	–	T_0^{35}	1.94	48.5	6	3
P5	⟨4, 20, 3⟩	–	–	–	–	T_0^{16}	1.75	47.5	5.5	2

Legend: *P* plan, *O* order, *q* quality, *dc* duty cycle given in percentage.

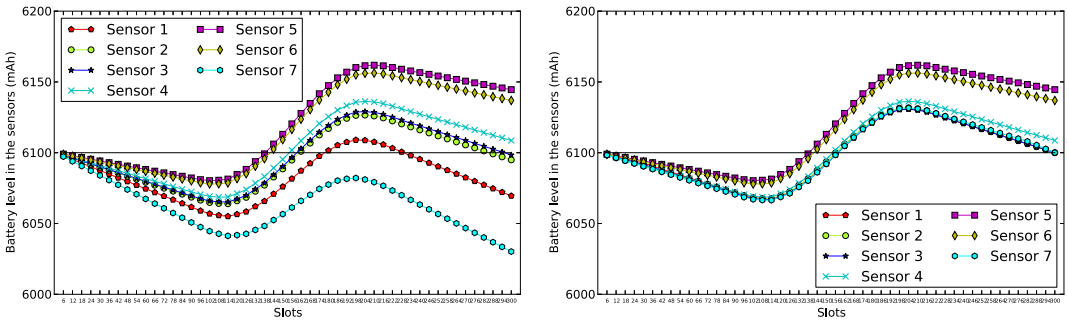


Fig. 5 **Case 2:** Battery levels in the sensors after the initial assignment (on the *left*) and after downgrading the sensors not energy-neutral (on the *right*)

The initial assignment of the most efficient scheduling plan of each sensor to the v slots results in a non-feasible assignment in which the sink does not hold the `Sumdc(k)` condition, $\forall k \in [1, v]$; this leads to downgrade all the sensors to find a feasible solution where the sum of the duty cycles be lower or equal than 100 %. Figure 8 shows the results obtained after downgrading the sensors. The initial assignment algorithm finds a feasible solution where the duty cycle is reduced from 119.5 % (which is the sum of the original duty cycles after assigning the most efficient scheduling plan) to 97.5 % (the sum of the duty cycles after downgrading the sensors) as shown in Fig. 8c. Observe that reducing the duty cycle reduces also the overall quality achieved (see Fig. 8d), which decreases from 53.55 to 44.33 %; a larger duty cycle involves a higher quality level since the sensor consumes more resources to perform better its tasks. We also observe in Fig. 8a that the battery level of the sink increases very slightly with regard to the battery level after assigning the most efficient scheduling plan; the reason for that is that the scheduling plans selected for the sensors during the downgrade must necessarily have a lower cost and, therefore, the corresponding scheduling plan selected in the sink, in turn, have a lower cost.

Figure 8b demonstrates that all the sensors keep energy-neutral after downgrading, following the same trend than in case 1 and 2: the battery level decreases when the solar production is low (or null) and increases during the hours of solar light. Note that sensors 6 and 2 present the lowest battery levels. Given that the assignment of the most efficient scheduling plan in each sensor (the first scheduling plan that is

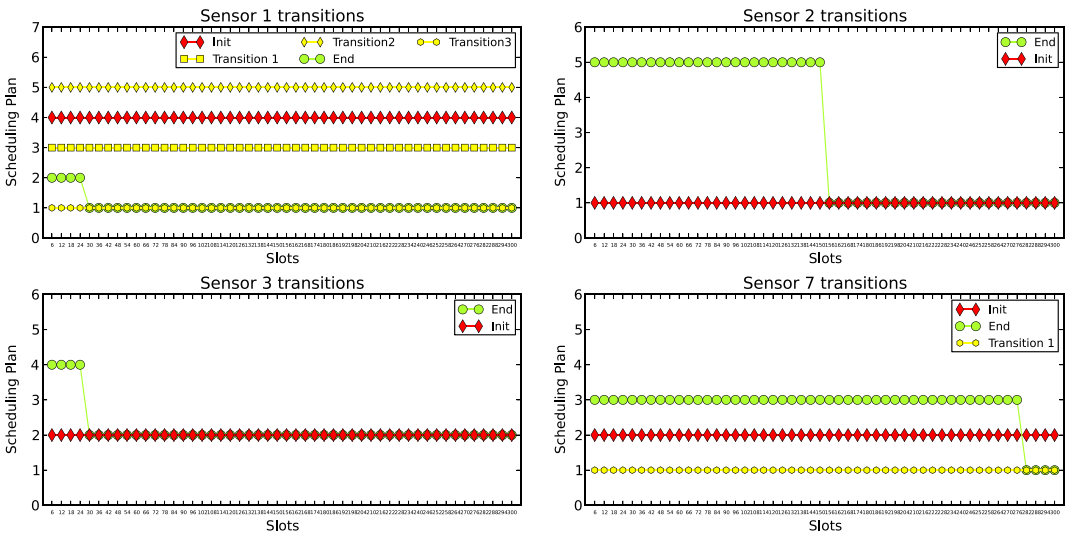
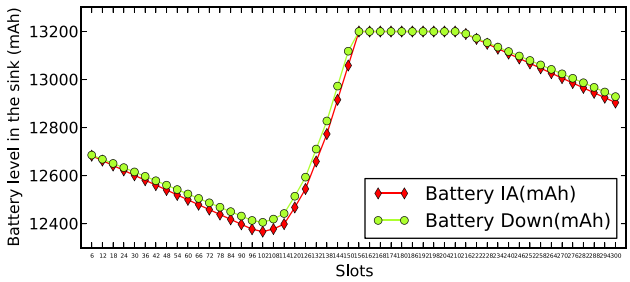
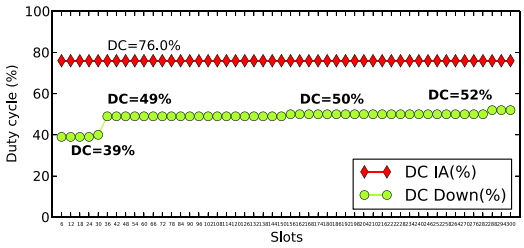


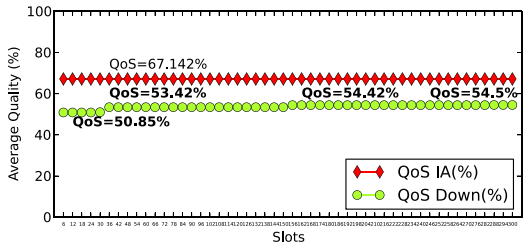
Fig. 6 **Case 2:** Transitions of the sensors 1, 2, 3, and 7 to achieve energy-neutrality



(a) Battery levels in the sink after initial assignment (IA) and after downgrading



(b) Duty cycle after downgrading



(c) Quality after downgrading

Fig. 7 Case 2: Battery level in the sink (above) and duty cycle and quality (below) after the initial assigning (IA) and after downgrading

assigned) would make the solution not feasible since the sum of the duty cycles would be larger than 100 %, the Algorithm 1 invokes `DowngradeAll`, which proceeds selecting first the scheduling plans of all sensors that are downgradable and, among them, selects the first scheduling plan whose replacement involves the lowest reduction on the quality. Among all the sensors, the scheduling plans assigned to sensors 2 and 6 were selected several times for downgrading until the solution became feasible. The transitions among scheduling plans in sensor 2 and sensor 6 can be viewed in Fig. 9, where the resulting duty cycle (after replacing the scheduling plan) is also displayed. As observed, the first sensor that downgrades is the sensor 2, where the scheduling plan 4 replaces the original scheduling plan 3; next, the sensor 6 downgrades two successive times: in the first one, all the slots are replaced by scheduling plan 4 and in the second one, this scheduling plan is replaced by scheduling plan 1; finally, the sensor 2 is selected again and downgrades other two times: the first one, in which the scheduling plan 4 is replaced by the scheduling plan 1; and the second one, in which the scheduling plan 1 is replaced by the scheduling plan 2; after these transitions, the value of duty cycle gets lower than 100 %, which stops downgrading.

Average optimization for each case

To compute the average optimization provided by our simulator, we first generate a set of instances for each specific case of upgrading, downgrading, and downgrading all, and run the Algorithm 1 for each one. Each instance of a case keeps the same values n, m, where m scheduling plans are generated randomly and have therefore different values. Table 6 shows the average optimization values obtained for 50 occurrences of the three cases, in terms of average battery level in the $n-1$ sensors and in the sink at the last slot ($v = 300$), the average init/end duty cycle (%), and the average init/end quality (%). Remember that the sensors and the sink start with an initial battery level of 6100 and 12,700 mAh, respectively. Note that to compute the values corresponding to duty cycle and quality, we had into account the average of the values in the $v = 300$ slots of each instance, since these values could differ from one slot to another. We observe that in the three cases both the sensors and the sink hold the feasibility condition. In the case of upgrade, as it is expected, the duty cycle and the quality grow an average of 42.6 and 14.5 %, respectively. The results for downgrading show that all sensors get set slightly above the minimum battery level (6100 mAh) which,

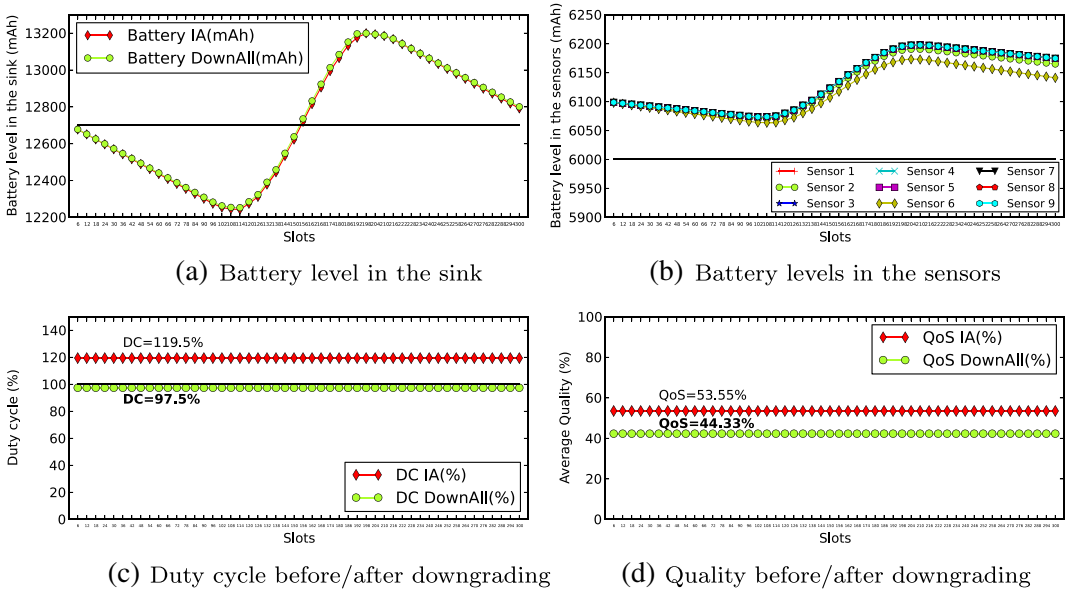
Table 5 Scheduling plans for case 3

	T_1	T_2	T_3	T_4	T_5	T_0	c	q	dc	O
Sensor 1										
P1	⟨3, 40, 4⟩	⟨10, 40, 18⟩	⟨2, 40, 17⟩	⟨10, 40, 6⟩	⟨1, 40, 3⟩	T_0^{14}	1.90	19.5	6	1
P2	⟨2, 27, 3⟩	⟨4, 27, 15⟩	⟨1, 27, 7⟩	⟨3, 27, 13⟩	–	T_0^{17}	2.66	22	10	3
P3	⟨7, 39, 18⟩	⟨3, 39, 13⟩	⟨9, 39, 6⟩	⟨6, 39, 20⟩	–	T_0^{14}	2.17	20	7	5
P4	⟨5, 40, 14⟩	⟨1, 40, 16⟩	⟨1, 40, 10⟩	–	–	T_0^{33}	2.65	21	8	2
P5	⟨2, 10, 20⟩	⟨2, 10, 9⟩	–	–	–	T_0^{6}	4.1	33	12	4
Sensor 2										
P1	⟨3, 29, 2⟩	⟨3, 29, 19⟩	⟨4, 29, 4⟩	⟨3, 29, 8⟩	⟨5, 29, 7⟩	T_0^{11}	2.13	38	15	3
P2	⟨2, 23, 9⟩	⟨5, 23, 9⟩	⟨4, 23, 6⟩	–	–	T_0^{12}	2.08	30	11	4
P3	⟨1, 26, 10⟩	–	–	–	–	T_0^{25}	2.30	77	24	1
P4	⟨7, 32, 17⟩	⟨8, 32, 18⟩	–	–	–	T_0^{17}	2.15	56	19	2
P5	⟨10, 40, 11⟩	⟨2, 40, 15⟩	⟨9, 40, 6⟩	–	–	T_0^{19}	1.75	21	10	5
Sensor 3										
P1	⟨8, 35, 6⟩	⟨3, 35, 2⟩	⟨3, 35, 17⟩	⟨5, 35, 9⟩	–	T_0^{16}	1.88	20	5.5	2
P2	⟨2, 28, 3⟩	⟨6, 28, 19⟩	⟨4, 28, 13⟩	⟨1, 28, 13⟩	⟨2, 28, 3⟩	T_0^{13}	2.75	50	8	3
P3	⟨2, 10, 4⟩	⟨1, 10, 18⟩	⟨2, 10, 10⟩	⟨2, 10, 13⟩	⟨1, 10, 13⟩	T_0^{2}	5.2	82	11	1
P4	⟨1, 20, 12⟩	⟨4, 20, 8⟩	⟨2, 20, 3⟩	–	–	T_0^{13}	2.45	22	6	4
P5	⟨2, 29, 18⟩	⟨2, 29, 9⟩	–	–	–	T_0^{25}	2.65	23	7	5
Sensor 4										
P1	⟨3, 12, 7⟩	⟨1, 12, 15⟩	⟨1, 12, 17⟩	⟨1, 12, 17⟩	⟨2, 12, 16⟩	T_0^{4}	6.66	35	12	3
P2	⟨6, 31, 17⟩	⟨4, 31, 20⟩	⟨3, 31, 8⟩	⟨2, 31, 14⟩	⟨3, 31, 2⟩	T_0^{13}	2.80	20	5.5	4
P3	⟨4, 34, 8⟩	⟨1, 34, 10⟩	–	–	–	T_0^{29}	2.23	18.5	3.5	2
P4	⟨1, 31, 13⟩	⟨3, 31, 16⟩	–	–	–	T_0^{27}	2.67	19.5	4.5	5
P5	⟨1, 16, 3⟩	⟨1, 16, 17⟩	⟨2, 16, 19⟩	⟨4, 16, 8⟩	–	T_0^{8}	3.93	23	6	1
Sensor 5										
P1	⟨3, 24, 11⟩	⟨4, 24, 13⟩	–	–	–	T_0^{17}	2.41	22	12	2
P2	⟨5, 23, 19⟩	⟨5, 23, 1⟩	⟨5, 23, 15⟩	–	–	T_0^{11}	2.47	57	15	4
P3	⟨3, 39, 4⟩	⟨9, 39, 15⟩	⟨4, 39, 20⟩	⟨5, 39, 19⟩	–	T_0^{18}	2.41	20	10	1
P4	⟨2, 14, 17⟩	⟨3, 14, 6⟩	⟨3, 14, 11⟩	⟨1, 14, 16⟩	–	T_0^{5}	4.28	81	19	5
P5	⟨4, 19, 10⟩	–	–	–	–	T_0^{15}	2.1	19.5	8	3
Sensor 6										
P1	⟨3, 12, 4⟩	⟨3, 12, 1⟩	⟨2, 12, 2⟩	⟨1, 12, 15⟩	–	T_0^{2}	2.75	42	18	3
P2	⟨5, 31, 10⟩	⟨2, 31, 20⟩	⟨5, 31, 6⟩	–	–	T_0^{19}	2.38	22	12	1
P3	⟨2, 22, 3⟩	⟨1, 22, 10⟩	⟨4, 22, 14⟩	⟨1, 22, 10⟩	–	T_0^{14}	2.95	93	24	4
P4	⟨6, 30, 17⟩	⟨4, 30, 7⟩	⟨3, 30, 20⟩	⟨1, 30, 19⟩	–	T_0^{16}	2.56	39	15	2
P5	⟨4, 30, 15⟩	–	–	–	–	T_0^{26}	2.23	20	6	5
Sensor 7										
P1	⟨2, 19, 3⟩	⟨4, 19, 20⟩	⟨4, 19, 19⟩	⟨2, 19, 15⟩	⟨3, 19, 16⟩	T_0^{4}	4.26	59	18	3
P2	⟨4, 31, 12⟩	⟨1, 31, 4⟩	⟨7, 31, 10⟩	–	–	T_0^{19}	2.06	18.5	4.5	1
P3	⟨3, 33, 15⟩	–	–	–	–	T_0^{30}	2.27	51	9	5
P4	⟨8, 32, 10⟩	⟨8, 32, 20⟩	⟨2, 32, 8⟩	⟨2, 32, 8⟩	–	T_0^{12}	2.18	20	6	4
P5	⟨3, 13, 15⟩	⟨1, 13, 4⟩	–	–	–	T_0^{9}	2.07	19.5	5.5	2

Table 5 (continued)

	T_1	T_2	T_3	T_4	T_5	T_0	c	q	dc	O
Sensor 8										
P1	⟨3, 15, 9⟩	⟨2, 15, 6⟩	⟨3, 15, 3⟩			T_0^7	2.13	19.5	6	4
P2	⟨3, 14, 3⟩	⟨1, 14, 8⟩	–	–	–	T_0^{10}	2.21	20	7	3
P3	⟨7, 36, 4⟩	⟨8, 36, 17⟩	⟨3, 36, 2⟩	–	–	T_0^{18}	1.63	18.5	5.5	5
P4	⟨3, 12, 3⟩	⟨2, 12, 8⟩	–	–	–	T_0^7	2.91	79	24	1
P5	⟨3, 23, 15⟩	⟨5, 23, 3⟩	⟨5, 23, 18⟩	–	–	T_0^{10}	2.43	25	12	2
Sensor 9										
P1	⟨1, 27, 18⟩	⟨1, 27, 11⟩	–	–	–	T_0^{25}	2.92	37	6	1
P2	⟨3, 39, 14⟩	–	–	–	–	T_0^{36}	2.20	22	4.5	5
P3	⟨5, 20, 16⟩	⟨4, 20, 18⟩	–	–	–	T_0^{11}	2.8	26.0	5.5	2
P4	⟨1, 16, 18⟩	⟨2, 16, 19⟩	⟨1, 16, 11⟩	⟨2, 16, 14⟩	–	T_0^{10}	5.12	39	8	3
P5	⟨4, 24, 6⟩	–	–	–	–	T_0^{20}	1.91	20	3.5	4

Legend: *P* plan, *O* order, *q* quality, *dc* duty cycle given in percentage



(a) Battery level in the sink (b) Battery levels in the sensors

(c) Duty cycle before/after downgrading (d) Quality before/after downgrading

Fig. 8 **Case 3:** Battery levels in the sink and in the sensors after downgrading (*above*); duty cycle and quality (*below*) after the initial assigning (IA) of the most efficient scheduling plan and after downgrading

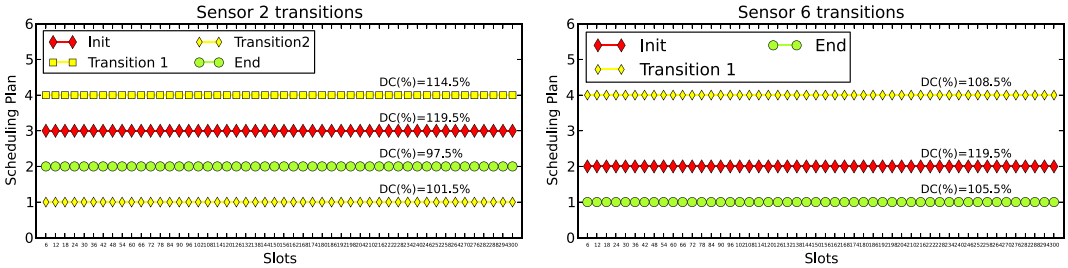


Fig. 9 **Case 3:** Transitions of the sensors 2 and 6 to achieve a feasible solution

Table 6 Average optimization for 50 instances of upgrade, downgrade, and downgrade all

		`Upgrade` (n = 5,m=6)	`Downgrade` (n = 8,m = 5)	`DowngradeAll` (n = 10,m = 5)
Avg. End Sensor Battery (mAh)	Sensor 1	6192.9	6100.03	6166.45
	Sensor 2	6212.5	6100.04	6161.05
	Sensor 3	6201.9	6100.02	6164.37
	Sensor 4	6194.3	6100.02	6165.46
	Sensor 5	–	6100.03	6167.72
	Sensor 6	–	6100.02	6163.07
	Sensor 7	–	6100.03	6162.81
	Sensor 8	–	–	6157.95
	Sensor 9	–	–	6166.35
Avg. End Sink Battery (mAh)		12,968.6	12,938.4	12,701.17
Avg. Init DC (%)		28.1	54.56	73.24
Avg. End DC (%)		60.7	44.07	55.71
Avg. Init q(%)		60.1	61.61	62.32
Avg. End q (%)		74.6	54.46	53.91

in turn, involves reducing both its duty cycle and its quality level, an average of 10.5 and 7 %, respectively. Finally, the last column depicts the results for the case downgrading all sensors, where the algorithm gets adjusting the average battery level in the sink slightly above its minimum level, 12,700 mAh. As a consequence of having a sink not energy-neutral (e.g., the sink is not able to attend simultaneously all the sensors), the algorithm downgrades all the sensors up to the system becomes feasible, which results in a reduction of the duty cycle and the subsequent loss of quality, an average of 18.53 and 8.4 %, respectively.

Discussion and conclusions

Energy harvesting systems have become a very appealing strategy to prolong the lifetime of sensor nodes. Generally, the energy sources employed by these systems are uncontrollable, which makes necessary the use of algorithms aimed at balancing the amount of energy that is produced and consumed by the node, in order to find a high overall quality level while conserving an energy level sufficient to maintain the sensors operations. The problem becomes more complicated in energy harvesting sensor net-

Table 7 Values for the test scenarios (on the *left*) and the results of the optimal assignments and of our algorithms (on the *right*)

		Scheduling plans										
		Plan 1			Plan 2			Plan 3			Slots	
		DC	Q	C	DC	Q	C	DC	Q	C	1 ...	10
Test 1	Sensor 1	43	51	1.58	67	74	3.0	59	63	2.7	1	
	Sensor 2	36	49	1.76	55	77	3.09	49	51	2.9	2	
Test 2	Sensor 1	42	50	2.44	37	49	1.7	59	51	2.7	3	
	Sensor 2	35	60	2.5	25	51	2.05	37	61	3.23	3	
Test 3	Sensor 1	42	42	2.75	37	22	2.38	65	93	3	3	
	Sensor 2	46	59	4.26	25	19	2.06	37	51	2.3	2	

Legend: DC: Duty Cycle (%); Q: Quality Level; C: Cost.

works, where all (or a subset) sensor nodes employ an energy harvesting system, possibly with different harvesting properties and different candidate applications to be elected, because a minimal change in the scheduling of a node may affect to the rest of the nodes leading to recompute their scheduling.

In this paper, we consider a scenario composed by n sensor nodes, one of them acting as the sink, and all of them equipped with a solar cell-based energy harvesting system. We address an optimization problem that consists in finding a (sub-)optimal assignment of execution plans at different instants of time (slots) within a reference period, both for the nodes and for the sink, such that it maximizes the overall quality level keeping their scheduling feasible (i.e., energy neutral, minimal battery, and sum of duty cycles below 100 %). To solve this problem, we propose an heuristic that proceeds assigning initially the most efficient scheduling plan of each sensor to all their time slots and, from here, upgrades/downgrades the solution according to the energy production and the energy consumption of the candidate scheduling plans, to compute scheduling that maximize the quality for the n sensors. We have evaluated our approach by simulation. Our simulator may generate test cases with different configurations and properties, and then it applies the algorithms proposed to find the optimal schedulings. Our algorithms find always a feasible solution even when the space of searching could be very large; we show how the nodes adapt dynamically their initial assignments to the actual solar conditions and improve/reduce the global quality level while respect the given constraints.

There are still two open issues that we discuss next: scalability and optimality. For the first issue, we stress that our model considers a star-connected WSN, where the sensors communicate wirelessly only with the sink and vice versa. The star topology is the typical one used in some scenarios of WSNs, as in Internet of Things applications and in Wireless Body Area Networks (WBAN), where the sensors transmit their data to some controller device through a wireless communication protocol, for instance, Bluetooth or Zigbee. The stronger constraint to scalability in a star-connected network is due to the technology of the radio communications that cannot support a scalable number of devices; for example, bluetooth supports up to seven slave devices communicating to a master. For this reason, the scalability issue is relatively marginal in star connected networks, which is the subject of this work. As a future work, we plan to address the problem of multi-hop energy harvesting sensor networks, connected by using different network topologies as, for instance, meshes and trees, where the issue of scalability becomes a primary concern.

The second open issue is the assessment of the sub-optimality of our approach. We propose low-complexity algorithms that use an heuristic to greedily achieve (sub-)optimal assignments for all sensors in the face of high cost computational algorithms that clearly cannot be embedded into resources constrained sensors. We do not guarantee the optimality of the solutions achieved since ensuring optimality involves traversing a search space composed by all possible combinations of assignments to find the optimum. However, as a first approach to evaluate the distance of our greedy approach from the optimal solution, we have implemented an algorithm based on brute force that computes all the possible assignments, on a small scale network, and selects the assignment feasible that maximizes the overall quality. For each sensor, the size of its search space are variations with repetition of a elements (scheduling plans) chosen of b different ways (slots), i.e., $VR_{a,b} = a^b$. Specifically, we consider 2 sensors, $a = 3$ scheduling plans, $b = 10$ slots with equal constant production for both sensors.

Table 7 shows the parameters used in the evaluation (on the left) and the optimal solutions provided by the brute force algorithm for each case (on the right), and that coincide with the solutions achieved by our algorithms, in the three cases. However, although these results are encouraging, we are far from ensuring sub-optimality since we are using a very small scale and a very limited set of cases. Clearly, it is not possible with exhaustive search to enlarge much the scale or to consider a significant number of cases, hence, we can present it just as a first step towards the future work, which is to assess analytically the sub-optimality of our approach.

Another future work is the implementation of our heuristic directly on the platform Waspmote connected to the solar module KL-SUN3W. This enables us to evaluate our heuristic on a real sensor in terms of feasibility and performance.

Acknowledgment This work has been funded by the Programme for Research and Innovation of University of Castilla-La Mancha, co-financed by the European Social Fund (Resolution of 25 August 2014).

References

Akkaya, K., & Younis, M. (2005). Energy and qos aware routing in wireless sensor networks. *Cluster Computing*, *8*(2–3), 179–188.

Alippi, C., Anastasi, G., Francesco, M.D., & Roveri, M. (2010). An adaptive sampling algorithm for effective energy management in wireless sensor networks with energy-hungry sensors. *IEEE Transactions on Instrumentation and Measurement*, *59*(2), 335–344. doi:10.1109/TIM.2009.2023818.

Baronti, P., Pillai, P., Chook, V.W., Chessa, S., Gotta, A., & Hu, Y.F. (2007). Wireless sensor networks: a survey on the state of the art and the 802.15.4 and zigbee standards. *Computer Communications*, *30*(7), 1655–1695. Wired/Wireless Internet Communications.

Barsocchi, P., Chessa, S., Furfari, F., & Potorti, F. (2013). Evaluating ambient assisted living solutions: the localization competition. *IEEE Pervasive Computing*, *12*(4), 72–79. doi:10.1109/MPRV.2013.23.

Benini, L., Bogliolo, A., & De Micheli, G. (2000). A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *8*(3), 299–316. doi:10.1109/92.845896.

Bergonzini, C., Brunelli, D., & Benini, L. (2009). Algorithms for harvested energy prediction in batteryless wireless sensor networks. In *3rd international workshop on advances in sensors and interfaces, 2009. IWASI 2009*. doi:10.1109/IWASI.2009.5184785 (pp. 144–149).

Bogliolo, A., Delpriori, S., Lattanzi, E., & Seraghiti, A. (2011). Self-adapting maximum flow routing for autonomous wireless sensor networks. *Cluster Computing*, *14*(1), 1–14. doi:10.1007/s10586-009-0115-x.

Chen, J., Díaz, M., Llopis, L., Rubio, B., & Troya, J.M. (2011). A survey on quality of service support in wireless sensor and actor networks: requirements and challenges in the context of critical infrastructure protection. *Journal of Network and Computer Applications*, *34*(4), 1225–1239. Advanced Topics in Cloud Computing.

Cooper, P. (1969). The absorption of solar radiation in solar stills. *Solar Energy*, *12*.

Escolar, S., Carretero, J., Marinescu, M.C., & Chessa, S. (2014). Estimating energy savings in smart street lighting by using an adaptive control system. *International Journal of Distributed Sensor Networks*, *2014*, 17.

Escolar, S., Chessa, S., & Carretero, J. (2012). Optimization of quality of service in wireless sensor networks powered by solar cells. In *10th ieee international symposium on parallel and distributed processing with applications* (p. 8). Madrid.

Escolar, S., Chessa, S., & Carretero, J. (2013). Energy management of networked, solar cells powered, wireless sensors. In *Proceedings of the 16th ACM international conference on modeling, analysis & simulation of wireless and mobile systems, MSWiM '13* (pp. 263–266). New York: ACM.

Escolar, S., Chessa, S., & Carretero, J. (2014). Energy management in solar cells powered wireless sensor networks for quality of service optimization. *Personal and Ubiquitous Computing*, *18*(2), 449–464. doi:10.1007/s00779-013-0663-1.

Escolar, S., Chessa, S., & Carretero, J. (2014). Energy-neutral networked wireless sensors. *Simulation Modelling Practice and Theory*, *43*, 1–15. doi:10.1016/j.simpat.2014.01.002. http://www.sciencedirect.com/science/article/pii/S1569190X14000033.

Fafoutis, X., & Dragoni, N. (2011). Odmac: an on-demand mac protocol for energy harvesting - wireless sensor networks. In *Proceedings of the 8th ACM symposium on performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, PE-WASUN '11*. doi:10.1145/2069063.2069072 (pp. 49–56). New York: ACM.

Felemban, E., Lee, C.G., & Ekici, E. (2006). Mmspeed: multipath multi-speed protocol for qos guarantee of reliability and. timeliness in wireless sensor networks. *IEEE Transactions on Mobile Computing*, *5*(6), 738–754. doi:10.1109/TMC.2006.79.

Hartmann, D.L. (1994). *Global physical climatology, international geophysics*, 1st edn. Vol. 56. Boston: Academic Press.

Inman, R.H., Pedro, H.T., & Coimbra, C.F. (2013). Solar forecasting methods for renewable energy integration. *Progress in Energy and Combustion Science*, *39*(6), 535–576. doi:10.1016/j.pecs.2013.06.002. http://www.sciencedirect.com/science/article/pii/S0360128513000294.

Iyer, R., & Kleinrock, L. (2003). Qos control for sensor networks. In *IEEE international conference on communications, 2003. ICC '03*. doi:10.1109/ICC.2003.1204230, (Vol. 1 pp. 517–521).

Jacobson, M.Z. (2005). *Fundamentals of atmospheric modeling*, 2nd edn. Cambridge University Press.

Jiang, X., Polastre, J., & Culler, D. (2005). Perpetual environmentally powered sensor networks. In *Proceedings of the 4th international symposium on information processing in sensor networks, IPSN '05*. Piscataway: IEEE Press.

Kansal, A., Hsu, J., Zahedi, S., & Srivastava, M.B. (2007). Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems*, *6*(4).

Kansal, A., Potter, D., & Srivastava, M.B. (2004). Performance aware tasking for environmentally powered sensor networks. *SIGMETRICS Perform Evaluation Review*, *32*(1), 223–234.

KL (2014). KL Solar company pvt ltd., http://www.klsolar.com/. India.

Lattanzi, E., & Bogliolo, A. (2011). WSN design for unlimited lifetime, chap. Sustainable energy harvesting technologies - past, present and future. 978-953-307-438-2. InTech.

Lattanzi, E., Regini, E., Acquaviva, A., & Bogliolo, A. (2007). Energetic sustainability of routing algorithms for energy-harvesting wireless sensor networks. *Computer Communications*, *30*(14–15), 2976–2986. doi:10.1016/j.comcom.2007.05.035. http://www.sciencedirect.com/science/article/pii/S0140366407002228. Network Coverage and Routing Schemes for Wireless Sensor Networks.

Libelium (2014). Waspmote. http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf (Document version: v4.7).

Lin, K., Yu, J., Hsu, J., Zahedi, S., Lee, D., Friedman, J., Kansal, A., Raghunathan, V., & Srivastava, M. (2005). *Heliomote: enabling long-lived sensor networks through solar energy harvesting*, (pp. 309–309). New York: ACM.

Moser, C., Chen, J.J., & Thiele, L. (2008). An energy management framework for energy harvesting embedded systems. *Journal on Emerging Technologies in Computing Systems*, *6*(2), 7:1–7:21.

Moser, C., Chen, J.J., & Thiele, L. (2009). Power management in energy harvesting embedded systems with discrete service levels. *International Symposium on Low Power Electronics and Design*, *0*, 413–418. doi:10.1145/1594233.1594338.

NASA (2013). Surface meteorology and solar energy (retscreen). https://eosweb.larc.nasa.gov/sse/RETScreen/.

Piorno, J., Bergonzini, C., Atienza, D., & Rosing, T. (2009). Prediction and management in energy harvested wireless sensor nodes. In *1st international conference on wireless communication, vehicular technology, information theory and aerospace electronic systems technology, 2009. Wireless VITAE 2009*. doi:10.1109/WIRELESSVITAE.2009.5172412 (pp. 6–10).

Polastre, J., Hill, J., & Culler, D. (2004). Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on embedded networked sensor systems, SenSys '04* (pp. 95–107). New York: ACM.

Sohrabi, K., Gao, J., Ailawadhi, V., & Pottie, G. (2000). Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, *7*(5), 16–27. doi:10.1109/98.878532.

Sudevalayam, S., & Kulkarni, P. (2011). Energy harvesting sensor nodes: survey and implications. *IEEE Communications Surveys and Tutorials*, *13*(3), 443–461.

Vithanage, M., Fafoutis, X., Andersen, C., & Dragoni, N. (2013). Medium access control for thermal energy harvesting in advanced metering infrastructures. In *EUROCON, 2013*. doi:10.1109/EUROCON.2013.6624999 (pp. 291–299): IEEE.

Vullers, R., Schaijk, R., Visser, H., Penders, J., & Hoof, C. (2010). Energy harvesting for autonomous wireless sensor networks. *IEEE Solid-State Circuits Magazine*, *2*(2), 29–38. doi:10.1109/MSSC.2010.936667.

Xia, F. (2008). Qos challenges and opportunities in wireless sensor/actuator networks. *Sensors*, *8*(2), 1099–1110. doi:10.3390/s8021099.

Yigitel, M.A., Incel, O.D., & Ersoy, C. (2011). Qos-aware MAC protocols for wireless sensor networks: a survey. *Computer Networks*, *55*(8), 1982–2004.