Joint Application Admission Control and Network Slicing in Virtual Sensor Networks

Carmen Delgado, María Canales, Jorge Ortín, José Ramón Gállego, Alessandro Redondi, Sonda Bousnina, and Matteo Cesana

***Abstract*—We focus on the problem of managing a shared physical wireless sensor network (WSN) where a single network infrastructure provider leases the physical resources of the networks to application providers to run/deploy specific applications/services. In this scenario, we solve jointly the problems of application admission control (AAC), that is, whether to admit the application/service to the physical network, and wireless sensor network slicing (SNS), that is, to allocate the required physical resources to the admitted applications in a transparent and effective way. We propose a mathematical programming framework to model the joint AAC-SNS problem which is then leveraged to design effective solution algorithms. The proposed framework is thoroughly evaluated on realistic WSNs infrastructures.**

***Index Terms*—Internet of Things, optimization, resource allocation, virtualization, wireless sensor networks (WSNs).**

I. Introduction

Wireless sensor networks (WSNs) are one of the key enabling building blocks for the Internet of Things. Looking back at the evolution of WSNs, a clear trend can be observed moving from standalone, application-specific deployments to highly integrated wireless sensor systems used to support heterogeneous ecosystems of services and applications. In this context, the premium deployment domains for WSNs are nowadays smart cities, smart home and buildings and intelligent transportation systems, which are all characterized by the coexistence of sensor nodes with heterogeneous sensing, processing and communication capabilities, which all together support multiple applications and services.

The aforementioned trend calls for novel design good practices to overcome the limits in flexibility, efficiency, and manageability of vertical, task-oriented, and domain-specific WSNs. In this field, virtualization appears to be the most promising approach to reach this goal. Virtualization is already a consolidated reality at the very heart of cloud-based services in data centers and the Internet core, and is also gauging momentum in the domain of wireless mobile networks under the push of network “softwareization”; as an example, one of the major innovation of the fifth generation (5G) of the mobile broadband systems under current standardization is network slicing, which allows allocating/partitioning the physical resources of the radio access network and the core network to multiple concurrent applications with heterogeneous requirements and constraints.

Similar virtualization approaches have been recently proposed in the domain of WSNs to ease up reconfigurability and manageability of network resources, eventually opening up for novel business opportunities where the roles of WSN infrastructure provider and WSN service/application provider are decoupled. WSNs virtualization includes all technologies to abstract the physical communication, sensing, and processing resources in a shared WSN to efficiently allocate them to multiple independent applications or clients. Generally speaking, the realization of virtual sensor networks requires technologies and solutions in different domains ranging from the node level, where the virtual sensor nodes must be able to support and run applications in a transparent way, up to the network level where effective platforms and solutions are required to manage and reconfigure on-the-fly the network resources.

We focus here on the problem of managing a shared physical WSN over time; we look at the reference scenario where a single shared sensor network infrastructure provider (SSN-IP) owns an heterogeneous infrastructure which can be accessed by multiple application providers which issue requests to deploy specific applications/services. In this scenario, we solve jointly the problems of application/service admission control (AAC), that is, whether to admit the application/service to the physical network, and WSN slicing (SNS), that is, to allocate the required physical resources to the admitted applications in a transparent and effective way. We propose a mathematical programming framework to model the joint AAC-SNS problem, which is then leveraged to design effective solution algorithms. The proposed framework is thoroughly evaluated on realistic heterogeneous WSNs.

This paper is organized as follows. Section II gives an overview of the background and the most relevant related literature, further commenting on the main contributions of this paper. Section III introduces the addressed problem and the reference scenario, whilst Sections IV and V introduce the

Manuscript received September 22, 2017; accepted October 26, 2017. Date of publication November 2, 2017; date of current version February 9, 2018. This work was supported in part by the Spanish Government through the Ministerio de Ciencia e Innovación under Grant TEC2014-52969-R, in part by the Gobierno de Aragón Research Group T98, in part by the European Social Fund, and in part by the Centro Universitario de la Defensa under Project CUD2016-17. *(Corresponding author: José Ramón Gállego.)*

C. Delgado, M. Canales, and J. R. Gállego are with the Aragón Institute of Engineering Research, Universidad de Zaragoza, 50018 Zaragoza, Spain (e-mail: jrgalleg@unizar.es).

J. Ortín is with the Centro Universitario de la Defensa, 50090 Zaragoza, Spain, and also with the Aragón Institute of Engineering Research, Universidad de Zaragoza, 50018 Zaragoza, Spain.

A. Redondi, S. Bousnina, and M. Cesana are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20131 Milan, Italy.

optimization framework to solve the joint problem of AAC and wireless SNS in the two cases where application arrival time and activation time are/are not known *a priori*, respectively. In Section VI, we introduce a heuristic to solve the joint AAC-SNS problem whose performance is evaluated in Section VII. Concluding remarks are finally reported in Section VIII.

II. Related Work

Virtualization is a widely used technique in the management of cloud-based services in the core network of the Internet [1]. Besides that, virtualization is also impacting the domain of wireless networks in general and mobile radio networks in particular with the ongoing standardization of novel virtualization-aware features for the 5G of mobile radio network including network slicing and Cloud-RAN [2].

Only recently virtualization technologies have also been studied in the domain of WSNs with the primary goal to improve the flexibility, the manageability and the return on investment of widespread and large WSNs deployments. Being the research field that recent, a common and shared terminology is still missing and the technical papers often use different wording for similar concepts; as an example, *virtual sensor networks*, *shared sensor networks*, *federated sensor networks*, and *multiapplication sensor networks* are often used almost interchangeably in the related literature. The interested reader may refer to the following surveys on the topic [3], [4].

In this paper, we will use the term *virtual sensor network* to define a physical sensor network heterogeneous infrastructure which can be used to support multiple concurrent applications and services, and where the ownership of the physical infrastructure is decoupled from the ownership of the applications and services; along the same lines, we will refer to *virtualization technologies* to define the different solutions and approaches to support and realize a *virtual sensor network*.

Virtualization technologies at different levels are needed to actually enable a virtual sensor network. One insightful classification of such technologies available in the literature distinguishes virtualization technologies at the node level and at the network level [3]. *Node-level virtualization* encompasses the design of abstraction layers and primitives on the single sensor node to overcome the problem of application-platform dependency and/or code modularization; in this field, architectures based on virtual machines are proposed to enable virtualization and reprogrammability. As an example, Maté [5], ASVM [6], Melete [7], and VMStar [8] are frameworks for building application-specific virtual machines over constrained sensor platforms. Similarly, ReLog [9] proposes a systematic approach consisting of a programming language, a compiler, and a virtual machine to make application programs concise and easy to modify.

Network-level virtualization technologies include two main building blocks which are usually tightly coupled: 1) management platforms to support multiple application sharing a common physical infrastructure and 2) tools/algorithms to allocate the physical resources to the multiple applications. Representative examples of management platforms are SenSHare [10] and UMADE [11], which create multiple overlay sensor networks which are "owned" by different applications on top of a shared physical infrastructure. Along the same lines, Fok *et al.* [12] and Li *et al.* [13] introduced middleware abstractions to support multiple applications in heterogeneous WSNs. A prototype of software defined WSN is proposed in [14] where a centralized control plane dynamically manages communication routes in the network with the goal of augmenting the energy efficiency.

As far as resource allocation in virtual sensor network is concerned, Xu *et al.* [15] focused on environmental monitoring applications and proposed a centralized optimization framework which allocates applications to sensor nodes with objectives to minimize the variance in sensor readings. The same authors address in a later work the case where the application assignment problem is no longer centralized but rather distributed by resorting to game-theoretic tools [16]. Ajmal *et al.* [17] proposed a decision algorithm to dynamically "admit" applications to physical sensor network infrastructure. Activity time maximization of the physical infrastructure is targeted in [18] and [19], which focus on the problem of scheduling applications in shared sensor nodes. In our past work [20], we study the problem of resource allocation in virtual sensor networks in a static case where the set of applications to be serviced is fixed and given. The research streamline on sensor mission assignment [21]–[24] also addresses a resource allocation problem in WSNs; namely, the problem is to jointly allocate the physical sensor resources to incoming applications while incorporating admission control policies. In this respect, the problem addressed in this paper bears similarities with sensor mission assignment; on the other side, our approach has the following distinctive additional features which make it more comprehensive with respect to the related work on sensor mission assignment: our framework considers networking-related aspects (e.g., routing, interference, and network capacity), includes the possibility to reconfigure the sensor network by moving applications which were previously deployed, and further allows modeling situations where multiple applications can be concurrently deployed at a sensor node.

This paper naturally fits in the network-level virtualization class and further extends the related literature in this field by considering a dynamic case where applications/services are "offered" to the physical infrastructure provider over time. Our proposal provides a more flexible resource allocation, aligned with the ideas of virtualization and network slicing, and a more detailed modeling of application demands, sensor resources and networking aspects than previous sensor mission assignment works. To the authors' knowledge, this paper is the first approach to model and analyze the joint problem of AAC and physical resource allocation in virtual sensor networks.

III. Problem Statement and System Model

The reference playground features one SSN-IP that owns a WSN composed of heterogeneous nodes in terms of processing, sensing and communication capabilities; the SSN-IP has full management control of the physical infrastructure

TABLE I
SET OF PARAMETERS OF THE OPTIMIZATION FRAMEWORK

Parameter	Definition
Sets, vectors, characteristics	
$S = \{s_1, s_2, \ldots, s_{\|S\|}\}$	Set of sensor nodes; subscript index i (or h) refers to sensor node s_i (or s_h)
S'	Set of nodes that are not Sinks (a subset of S)
$A = \{a_1, a_2, \ldots, a_{\|A\|}\}$	Set of applications; subscript index j refers to application a_j
$W = \{w_1, w_2, \ldots, w_{\|W\|}\}$	Set of test points; subscript index k refers to test point w_k
W_j	Set of test points of application j
S_{jk}	Set of sensor nodes covering test point k of application j
$o_j = \{c_j, m_j, l_j\}$	Requirement vector of application j (source rate, memory, processing load)
$O_i = \{C_i, M_i, L_i, E_i\}$	Resource vector of node i (bandwidth, storage, proccessing power, energy)
$T = \{t_1, t_2, \ldots, t_{\|T\|}\}$	Set of time instants; subscript index n refers to time instant t_n
τ_j	Instant time in which application j arrives
ϵ_j	Activity time of application j
T_j	Subset of T formed by the time instants that lay in the interval $[\tau_j, \tau_j + \epsilon_j)$
$A_q = \{a_j \| \tau_q \in [\tau_q, \tau_q + \epsilon_q)\}$	Set of applications that are running in the network at τ_q, including the *arriving* application
$Q = \{\tau_1, \tau_2, \ldots, \tau_{\|A\|}\}$	Subset of T containing the time instants of the arrival of the applications; subscript index q refers to time instant τ_q
R_i^s	Sensing range of node i
Coverage and application deployment	
z_j	Binary variable indicating if application j is successfully deployed
y_{ijkn}	Binary variable indicating if test point k of application j is deployed at sensor node i at time instant n
x_{in}	Binary variable indicating if sensor node i is active in the network at time instant n
h_{jkn}	Binary variable indicating if test point k of set T_j is sensed by a sensor node at time instant n
y_{ijk}	Binary variable indicating if test point k of application j is deployed at sensor node i
x_i	Binary variable indicating if sensor node i is active in the network
X_i	Constant equal to 1 if the node i was active before the arrival of the new application, and 0 otherwise
Y_{ijk}	Constant equal to 1 if the test point k of the application j was being sensed in the node i just before the arrival of the new application, and 0 otherwhise
N_{ij}	Maximum number of test points of application j actually covered by sensor node i
u_{in}	Binary variable: $u_{in} = x_{in}x_{in-1}$
v_{ijkn}	Binary variable: $v_{ijkn} = y_{ijkn}y_{ijkn-1}$

and receives requests from sensor network service providers (SN-SP) to have their services, called *applications* hereafter, deployed in the physical infrastructure. Each application request is characterized by an *arrival time*, that is, the time the application request is issued, an *activity time*, that is how long the application needs to be active in the network, and a *set of requirements*, that is, the service level required by the application in terms of sensing area to be covered, required processing and storage capabilities of the node(s) hosting the application, and required communication bandwidth to deliver application data remotely across the physical sensor network.

We set ourself in the perspective of the SSN-IP and address the problem on how to optimally manage the application requests coming from the SN-SPs to maximize our total revenue which is assumed to be proportional to the number of application requests which can be satisfied. In a nutshell, we introduce hereafter optimal and suboptimal techniques to let the SSN-IP decide jointly when to admit/not admit an application request and how to reconfigure consequently the current physical resource assignment to applications.

Tables I and II summarize the notation used through Sections III–V. Let $S = \{s_1, s_2, \ldots, s_{|S|}\}$ be a set of sensor nodes, $A = \{a_1, a_2, \ldots, a_{|A|}\}$ a set of applications which are to be deployed in the reference area, and $W = \{w_1, w_2, \ldots, w_{|W|}\}$ a set of test points in the reference network scenario. These test points are physical locations where the application's sensing parameters must be measured (e.g., a test point can be a physical location where a temperature monitoring application needs to collect a temperature sample). To simplify notation, in the following we will use the subscript index i (or h) to refer to a sensor node s_i (or s_h), the subscript index j to refer to an application a_j and the subscript index k to refer to a test point w_k.

Each application j requires to sense a given set of test points $W_j \subseteq W$. Formally, the application j has to be deployed in a subset of the sensor node set S such that all the test points in W_j are sensed. We consider that a test point is covered by a sensor node i if it is within its sensing range, R_i^s. Thus, given a test point, a set of sensor nodes can *cover* it (the test point can be in the sensing range of several nodes), but only one sensor node will *sense* it.

Therefore, it is also convenient to introduce the set S_{jk}, defined as the set of sensor nodes which cover the test point k, with $k \in W_j$. In other words, if the application j is deployed on

TABLE II
SET OF PARAMETERS OF THE OPTIMIZATION FRAMEWORK

Parameter	Definition
	Propagation model
P_{max}	Maximum transmission power
R^T_{max}	Maximum transmission range with P_{max}
R^I_{max}	Maximum interference range with P_{max}
g_{ih}	Channel gain from transmitter i to receiver h in the directional link (i, h)
d_{ih}	Distance from i to h
g_0	Constant dependent on antenna parameters
γ	Path loss index
α	Receiver sensitivity
μ	Interference sensitivity
p_i	Transmission power assigned to node i
$R^T(p_i)$	Transmission range for node i with transmission power p_i
$R^I(p_i)$	Interference range for node i with transmission power p_i
	Routing
f_{ihn}	Flow of data in bps transmitted from node i to node h at time instant n
f_{ih}	Flow of data in bps transmitted from node i to node h
f_{ihj}	Flow of data for application j in bps transmitted from node i to node h
r_{in}	Rate generated by node i at time instant n
K_{ih}	If nodes i and h are linked according to the routing tree, K_{ih} is a constant higher than the maximum transmission rate of a node. If not, K_{ih} is 0
	Bandwidth
C_{ih}	Capacity of the link (i, h)
	Energy
β_1	$\beta_1 = 50$ nJ/bit
β_2	$\beta_2 = 0.0013$pJ/bit/m^4
ρ	$\rho = 50$ nJ/bit
P^t_{in}	Power dissipation at the radio transmitter of node i at time instant n
P^r_{in}	Power dissipation at the radio receiver of node i at time instant n
P^t_{ij}	Power dissipation for the application j at the radio transmitter of node i
P^r_{ij}	Power dissipation for the application j at the radio receiver of node i
φ_i	Cost incurred every time a node i is activated
δ_{ij}	Cost incurred in node i due to the impact of receiving the bytecode of application j
λ_i	Variable indicating the residual energy that node i would have once the activity time of the applications deployed in it or forwarded by it expires
λ	Variable indicating the total residual energy of the network once the activity time of all the applications deployed in the network expires

any of the sensors in S_{jk}, then the target test point k is sensed for this application. A necessary condition for an application j to be successfully deployed is that all the test points in its target set W_j must be sensed.

To model the dynamics of the arrival and departure of applications, we assume that each application j arrives and leaves the system at time instants τ_j and $\tau_j + \epsilon_j$, where ϵ_j is its activity time. With this, we can define an ordered set of time instants $T = \{t_1, t_2, \ldots, t_{|T|}\}$, each of them corresponding to the moment at which the arrival or departure of an application happens. To simplify the notation, in the following we will use the subscript index n to refer to the time instant t_n. Each application j will be active in the network (if it is possible to deploy it) during a time interval that begins and ends at elements of T. We define T_j as the subset of T formed by the time instants that lay within the interval $[\tau_j, \tau_j + \epsilon_j)$, which corresponds to all the arrivals and departures of applications during the activity time of application j.

Each application j in A is further characterized by a requirement vector $o_j = \{c_j, m_j, l_j\}$ which specifies the generated source rate (bit/s), memory (bits), and processing load (MIPS) consumed by the application when it is deployed on a sensor node. The requirement vector can be interpreted as the amount of resources needed to accomplish the specific tasks required by the application (e.g., acquire, process, and transmit ten temperature samples, or acquire process and transmit one JPEG image, etc.). Additionally, each sensor node i in S is characterized by a given resource vector $O_i = \{C_i, M_i, L_i, E_i\}$, which specifies its available bandwidth, storage capacity, processing power and energy.

A protocol interference model with power control [25] is used to characterize the wireless communications among the sensor nodes. The maximum transmission power is $P_{\max}$. With this power, there are a maximum transmission range $R^T_{\max}$ and a maximum interference range $R^I_{\max}$. Given a directional link between a pair of nodes (i, h), the channel gain from transmitter i to receiver h is defined as $g_{ih} = g_0 d_{ih}^{-\gamma}$, being d_{ih} the distance from i to h, γ the path loss index and g_0 a constant dependent on antenna parameters. In order for a transmission to be successful, the received power must exceed a threshold α. Additionally, all the nodes under the interference range of a sensor node share the same transmission channel and therefore, the transmission time is shared among them. If p_i is the transmission power assigned to node i, a transmission toward h is successful if $p_i g_{ih} > \alpha$. Thus, the transmission range for node i with transmission power p_i can be obtained as $R_i^T(p_i) = (p_i g_0/\alpha)^{1/\gamma}$. Similarly, the interference resulting from node i with power p_i is non-negligible only if it exceeds a certain threshold μ. Then, the interference range is $R_i^I(p_i) = (p_i g_0/\mu)^{1/\gamma}$.

IV. General Optimization Framework

We start off by casting the AAC with wireless SNS (AAC-SNS) problem in the ideal case where the activity time and the arrival time of each application are known *a priori*. This leads to a performance benchmark which will be then used to evaluate the algorithms and solutions in the real case where application arrivals cannot be predicted *a priori*. In this reference scenario, we target the maximization of the overall serviced applications, subject to node-related and network-related constraints. Namely, this problem is subject to coverage constraints (the set of test points of each application must be sensed) and to application requirements (each application should be assigned enough processing and storage resources during its whole activity time to operate successfully). In addition, due to the multihop nature of WSNs, routing and link capacity constraints must be considered when the data generated by the applications have to be delivered remotely.

Let z_j be a binary variable indicating if application j is successfully deployed in the network. Let y_{ijkn} be a binary variable indicating if test point k of application j is deployed at sensor node i at time instant n. Let x_{in} be a binary variable indicating if sensor node i is active in the network at time instant n. Let h_{jkn} be a binary variable which indicates if test point k of application j is sensed at time instant n. The objective function aims to maximize the total number of deployed applications

$$\max \sum_{j \in A} z_j. \tag{1}$$

A. Constraints on Coverage and on Resources of the Sensors

Constraints (2)–(6) require that all the applications which are actually deployed do fulfill the coverage constraints, that is, they sense all the required test points during their corresponding activity time. Specifically, (2) indicates that a test point k of an application j is sensed at time instant n if the application j is deployed at a sensor node i belonging to S_{jk} at that instant. If so, it ensures that it is only sensed by sensor node i. Equation (3) ensures that if a sensor node i does not cover a test point k of an application j, then it cannot sense that test point. Equation (4) guarantees that the test points of each application are only tested during the required time interval of the application T_j. Depending on the application, it can be possible that the same sensor node can sense several of its test points (e.g., visual applications). If we define N_{ij} as the maximum number of test points of the same application j that a sensor i is able to sense, (5) guarantees that this threshold is not exceeded at any time instant. Equation (6) indicates that an application j is successfully deployed, i.e., $z_j = 1$, if and only if all its test points are sensed during the activity time of the application ($h_{jkn} = 1, \forall k \in W_j, \forall n \in T_j$). On the other hand, if the application is not successfully deployed, i.e., if $z_j = 0$, the constraint forces none of its test points to be sensed at all ($h_{jkn} = 0, \forall k \in W_j, \forall n \in T_j$ and consequently according to (2), $y_{ijkn} = 0, \forall k \in W_j, \forall i \in S_{jk}, \forall n \in T_j$). This guarantees that if the application cannot be deployed, resources are not wasted

$$\sum_{i \in S_{jk}} y_{ijkn} = h_{jkn} \qquad \forall j \in A, \forall k \in W_j, \forall n \in T \tag{2}$$

$$y_{ijkn} = 0 \qquad \forall i \notin S_{jk}, \forall j \in A, \forall k \in W_j, \forall n \in T \tag{3}$$

$$y_{ijkn} = 0 \qquad \forall i \in S_{jk}, \forall j \in A, \forall k \in W_j, \forall n \notin T_j \tag{4}$$

$$\sum_{k \in W_j} y_{ijkn} \le N_{ij} \qquad \forall i \in S, \forall j \in A, \forall n \in T_j \tag{5}$$

$$z_j = \frac{1}{|W_j||T_j|} \sum_{n \in T_j} \sum_{k \in W_j} h_{jkn} \qquad \forall j \in A. \tag{6}$$

Constraints (7) and (8) are budget-type constraints for the available storage and processing load of the sensor nodes. They must be ensured for all the nodes at any time instant $n \in T$

$$\sum_{j \in A} \sum_{k \in W_j} m_j y_{ijkn} \le M_i \qquad \forall i \in S, \forall n \in T \tag{7}$$

$$\sum_{j \in A} \sum_{k \in W_j} l_j y_{ijkn} \le L_i \qquad \forall i \in S, \forall n \in T. \tag{8}$$

B. Routing Constraints

Deployed applications require that the information generated locally is delivered remotely to collection points (sink nodes) through multihop paths. Note that these sensor nodes may run deployed applications or not. By resorting to a fluid model, it should be ensured that all the data produced by the sensors running applications at any time is received by the sink nodes. This fact can be conveniently expressed using the following constraints:

$$r_{in} = \sum_{j \in A} \sum_{k \in W_j} c_j y_{ijkn} \quad \forall i \in S, \forall n \in T \tag{9}$$

$$\sum_{\substack{h \in S \\ i \neq h}} f_{hin} - \sum_{\substack{h \in S \\ h \neq i}} f_{ihn} + r_{in} = 0 \quad \forall i \in S', \forall n \in T \tag{10}$$

$$\sum_{j \in A} c_j \sum_{k \in W_j} h_{jkn} = \sum_{h \in S \setminus S'} \left(\sum_{\substack{i \in S \\ i \neq h}} f_{ihn} + r_{hn} \right) \quad \forall n \in T \tag{11}$$

where S' is the set of nodes that are not sinks (a subset of S), r_{in} is a variable indicating the data generated by node i at time instant n, and f_{ihn} is a variable representing the flow of data in bps transmitted from node i to node h at time instant n. Constraints (9) and (11) enforce flow conservation at sensor nodes: the incoming flow rate into a node i plus the data generated by itself must be equal to the outcoming flow rate. When a node i runs application j to monitor its test point k, it generates traffic at a rate c_j bps.

Constraints (11) impose that the amount of data generated in the network at any time instant is equal to the amount of data collected by the set of sinks. The left term represents the total data rate generated in the network: for each active application

j, there may be different sensor nodes sensing the corresponding $|W_j|$ test points and therefore each one generating c_j bps. The right term represents the overall traffic received by the set of sink nodes plus the rate generated by themselves in case they are also running applications. This equality, together with the flow conservation in constraint (10), imposes that all the traffic generated in the network is finally collected by the set of sinks (delivered by other nodes or generated by the sinks themselves).

The following constraints set enforces that if a sensor node is either running an application or receiving data at time instant n, then it must be active in the network and inactive otherwise:

$$x_{in} \le \sum_{\substack{h\in S\\ h\ne i}} f_{hin} + r_{in} \le K x_{in} \qquad \forall i \in S, \forall n \in T \tag{12}$$

where K is a constant high enough (higher than the maximum transmission rate of a node).

Finally, in WSNs routes from each sensor node to a sink node follow typically a single path, such as the destination oriented directed acyclic graph (DODAG) of RPL [26]. To model this behavior, we build a DODAG for each sink using the number of hops as a metric (i.e., when there are several sinks, each node belongs to the DODAG that reaches a sink with the minimum number of hops). The following set of constraints forces all the traffic to be forwarded only through the links that belong to the predefined routes defined in the DODAGs:

$$f_{ihn} \le K_{ih} \qquad \forall i, h \in S, \forall n \in T \tag{13}$$

where for each node i, K_{ih} is 0 for all nodes h but for the father node in the routing tree toward the sink. In that case, K_{ih} is a constant higher than the maximum transmission rate of a node.

C. Bandwidth Constraints

The available bandwidth in the network is bounded and must be shared among sensor nodes. We assume that a fair medium access control scheme orchestrates the access to the shared medium. Given a directional link between a pair of nodes (i, h), let the capacity of the link be defined as $C_{ih} = \min(C_i, C_h)$. This aims to model that the transmission rate is limited by the most restrictive node in the link. Transmissions of other links where i or h are either transmitter or receiver cannot be simultaneously active with (i, h) (note that some combinations are not possible in this particular case due to routing constraints, i.e., another link with i as a transmitter).

According to the considered protocol interference model, the interfering links for link (i, h) are those whose receiver is within the interference range of node i or the links, where h is within the interference range of its transmitter. Although none of these links can be simultaneously active with (i, h), some of them (depending on their relative positions) could be simultaneously active with each other. Therefore, for each link in the network (i, h) it must be ensured that the fraction of time used by the link plus all its interferences at time instant n is less or equal to 1

$$\frac{f_{ihn}}{C_{ih}} + \sum_{g\in S} \frac{f_{gin}}{C_{gi}} + \sum_{\substack{g\in S\\ g\ne i}} \frac{f_{hgn}}{C_{hg}} + \sum_{\substack{g\in S\\ g\ne i}} \frac{f_{ghn}}{C_{gh}} + \sum_{\substack{g,t\in S\\ d_{it}<R_i^I(p_i)}} \frac{f_{gtn}}{C_{gt}} + \sum_{\substack{g,t\in S\\ d_{gh}<R_g^I(p_g)}} \frac{f_{gtn}}{C_{gt}} \le 1 \quad \forall i, h \in S, \forall n \in T. \tag{14}$$

Constraints (14) are the equivalent budget-type constraints for the available wireless capacity to the storage and processing load constraints given in (7) and (8).

D. Energy Constraints

Finally, energy constraints are included to ensure that the application deployment pattern does not exceed the energy budget of the network. Typically, energy consumption due to wireless communication (i.e., transmitting and receiving) has been considered the dominant factor in power consumption for WSNs [27]. While this is the case for traditional scalar applications, where processing is limited to simple operations, in multimedia applications the energy required to process data cannot be neglected [28].

Regarding wireless transceiver, the power dissipation at the radio transmitter P_{in}^t or at the radio receiver P_{in}^r of each node i at time instant n can be modeled as [29]

$$P_{in}^t = \sum_{h\in S, h\ne i} \left(\beta_1 + \beta_2 d_{ih}^{\gamma}\right) f_{ihn} \qquad \forall i \in S, \forall n \in T \tag{15}$$

$$P_{in}^r = \rho \sum_{h\in S, h\ne i} f_{hin} \qquad \forall i \in S, \forall n \in T. \tag{16}$$

Typical values for β_1, β_2, and ρ are $\beta_1 = \rho = 50$ nJ/bit and $\beta_2 = 0.0013$ pJ/bit/m^4, with $\gamma = 4$ the path loss index.

The estimation of the power dissipation due to the processing load is not so straightforward, since it depends on several factors such as the hardware architecture of the nodes or the specific implementation of the algorithm for each application. In the energy constraints set in (17), this power dissipation is left as a function f of the processing loads l_j of the applications. In Section VII, further details about the specific evaluated multimedia applications are given. In addition, we assume that the sinks do not have energy constraints since they can be plugged directly into the grid

$$\sum_{n\in T} \Delta t_n \left(P_{in}^t + P_{in}^r + f\left(\sum_{j\in A} \sum_{k\in W_j} y_{ijkn} l_j \right) \right) \le E_i \qquad \forall i \in S'. \tag{17}$$

To introduce the effect of the activation and deactivation of sensor nodes and the migration of active applications between sensor nodes of the network, we consider that both events decreases the energy of the affected nodes (i.e., the energy of a node decreases when it is booted-up or when it receives an application). In other words, there is a cost φ_i related to the amount of energy that the node i needs to wake up from the sleep mode. Besides, it is allowed moving applications from one active node to another as long as all the restrictions described previously are fulfilled. Nevertheless, it is assumed that moving an application j has a cost δ_{ij} due to the impact of moving the application code to the new node i. This modifies the previous set of restrictions as follows:

$$\sum_{n\in T}\Delta t_n\left(P^t_{in}+P^r_{in}+f\left(\sum_{j\in A}\sum_{k\in W_j}y_{ijkn}l_j\right)\right) + \varphi_i\sum_{n\in T}(x_{in}-x_{in-1})x_{in} + \sum_{j\in A}\delta_{ij}\sum_{k\in W_j}\sum_{n\in T}\left(y_{ijkn}-y_{ijkn-1}\right)y_{ijkn} \le E_i. \qquad (18)$$

Considering that x_{in} and y_{ijkn} are binary variables and defining the binary variables $u_{in}=x_{in}x_{in-1}$ and $v_{ijkn} = y_{ijkn}y_{ijkn-1}$, the previous expression can be reformulated as

$$\sum_{n\in T}\Delta t_n\left(P^t_{in}+P^r_{in}+f\left(\sum_{j\in A}\sum_{k\in W_j}y_{ijkn}l_j\right)\right) + \varphi_i\sum_{n\in T}(x_{in}-u_{in}) + \sum_{j\in A}\delta_{ij}\sum_{k\in W_j}\sum_{n\in T}\left(y_{ijkn}-v_{ijkn}\right) \le E_i \qquad \forall i\in S' \qquad (19)$$

with the variables u_{in} and v_{ijkn} fulfilling the following set of restrictions:

$$u_{in}\le x_{in} \qquad \forall i\in S, \forall n\in T \qquad (20)$$
$$u_{in}\le x_{in-1} \qquad \forall i\in S, \forall n\in T \qquad (21)$$
$$u_{in}\ge x_{in}+x_{in-1}-1 \qquad \forall i\in S, \forall n\in T \qquad (22)$$
$$v_{ijkn}\le y_{ijkn} \qquad \forall i\in S, \forall j\in A, \forall k\in W_j, \forall n\in T \qquad (23)$$
$$v_{ijkn}\le y_{ijkn-1} \qquad \forall i\in S, \forall j\in A, \forall k\in W_j, \forall n\in T \qquad (24)$$
$$v_{ijkn}\ge y_{ijkn}+y_{ijkn-1}-1 \qquad \forall i\in S, \forall j\in A, \forall k\in W_j, \forall n\in T. \qquad (25)$$

V. Optimization Model for Dynamic Resource Allocation

The general optimization framework set in the previous section assumes that the arrival and departure times of each application are known in advance. While this approach provides a benchmark for the best achievable performance, its application to real scenarios is limited to very specific situations where this information could be available.

Thus, to encompass a more realistic scenario, in this section we focus on the problem of dynamically addressing the deployment of new applications as they arrive to the network. Specifically, the problem can be described as follows: every time a new application arrives to the reference area, it will be deployed as long as the rest of applications that are running at that moment are not dropped out of the system. If this is not possible, the new application is discarded and the system remains unaltered so that the previous applications can be served. If it is possible, the applications should be deployed in such a way that the likelihood of deploying new applications in the future is maximized.

As one of the most limiting factors to deploy new applications is the remaining energy of the nodes (it will be shown in Section VII), three approaches are considered to select the way that applications are deployed in the network: 1) to maximize the global energy of the network; 2) to maximize the energy of the node with the lowest energy; and 3) a weighted sum of the first and the second approaches.

In the following, we define formally the optimization problem that must be solved every time a new application $q\in A$ arrives to the system. Let $Q=\{\tau_1,\tau_2,\ldots,\tau_{|A|}\}$ be the subset of T that contains all the arrival time instants. For every τ_q, we define the subset $A_q\subseteq A$ as the set of applications that are running in the network at time instant τ_q, *including* the arriving application, i.e., $A_q=\{a_j|\tau_q\in[\tau_j,\tau_j+\epsilon_j)\}$.

As the optimization problem that must be solved when the application q arrives only involves those applications that are active in the network at time instant τ_q (i.e., those is A_q), we can remove the dependency on n from the variables y_{ijkn} and x_{in} and redefine them as follows: y_{ijk} is a binary variable indicating if application j is deployed at a sensor node i covering test point k, and x_i is a binary variable indicating if sensor node i is active in the network.

The following sets of restrictions force all the applications in A_q to be deployed. It is worth noting that now the problem may be infeasible. If so, the system is left as it is to ensure that the current applications remain in the network and the new application is not deployed. Many of the following constraints are similar to the ones presented previously but without the subscript index n and restricted to the subset A_q instead of the set A.

A. Constraints on Coverage and on Resources of the Sensors

Constraints (26) require that all the applications in A_q sense all their required test points. To do so, it forces that for every test point k of an application j, the application is deployed at a sensor node i that can cover that test point k. Equations (27) and (28) are equivalent to the constraints presented in (3) and (5), respectively,

$$\sum_{i\in S_{jk}}y_{ijk}=1 \qquad \forall j\in A_q, \forall k\in W_j \qquad (26)$$
$$y_{ijk}=0 \qquad \forall j\in A_q, \forall k\in W_j, \forall i\notin S_{jk} \qquad (27)$$
$$\sum_{k\in W_j}y_{ijk}\le N_{ij} \qquad \forall i\in S, \forall j\in A_q. \qquad (28)$$

Constraints (29) and (30) are budget-type constraints, equivalent to the equations in (7) and (8)

$$\sum_{j\in A_q}\sum_{k\in W_j}m_jy_{ijk}\le M_i \qquad \forall i\in S \qquad (29)$$
$$\sum_{j\in A_q}\sum_{k\in W_j}l_jy_{ijk}\le L_i \qquad \forall i\in S. \qquad (30)$$

B. Routing and Bandwidth Constraints

Let f_{ihj} be the flow of data of application j in bps transmitted from node i to node h and f_{ih} the flow of data in bps transmitted from node i to node h. While f_{ih} is equivalent to the already shown f_{ihn}, f_{ihj} is needed to describe the corresponding power dissipation at radio transmitter and radio receiver as it will be shown in Section V-C.

Constraints (31)–(37) are equivalent to constraints (9)–(14). Now, in (34), since the problem forces all the applications to be active, the left term is the sum of all the possible data rate generated in the network by all the applications in A_q and it does not depend on any variable

$$r_{ij} = \sum_{k\in W_j} c_j y_{ijk} \quad \forall j \in A_q, \forall i \in S \tag{31}$$

$$f_{ih} = \sum_{j\in A_q} f_{ihj} \quad \forall i, h \in S \tag{32}$$

$$\sum_{\substack{h\in S\\ i\neq h}} f_{hij} - \sum_{\substack{h\in S\\ h\neq i}} f_{ihj} + r_{ij} \qquad \forall j \in A_q, \forall i \in S' \tag{33}$$

$$\sum_{j\in A_q} |W_j| c_j = \sum_{h\in S\setminus S'} \left(\sum_{\substack{i\in S\\ i\neq h}} f_{ih} + \sum_{j\in A_q} r_{hj} \right) \tag{34}$$

$$x_i \le \sum_{\substack{h\in S\\ h\neq i}} f_{hi} + \sum_{j\in A_q} r_{ij} \le K x_i \qquad \forall i \in S \tag{35}$$

$$f_{ih} \le K r_{ih} \qquad \forall i, h \in S \tag{36}$$

$$\frac{f_{ih}}{C_{ih}} + \sum_{g\in S} \frac{f_{gi}}{C_{gi}} + \sum_{\substack{g\in S\\ g\neq i}} \frac{f_{hg}}{C_{hg}} + \sum_{\substack{g\in S\\ g\neq i}} \frac{f_{gh}}{C_{gh}} + \sum_{\substack{g,t\in S\\ d_{it}<R_i^I(p_i)}} \frac{f_{gt}}{C_{gt}} + \sum_{\substack{g,t\in S\\ d_{gh}<R_g^I(p_g)}} \frac{f_{gt}}{C_{gt}} \le 1 \qquad \forall i, h \in S. \tag{37}$$

C. Energy Constraints

The power dissipation for the application j at the radio transmitter P^t_{ij} or at the radio receiver P^r_{ij} of each node i can be modeled as

$$P^t_{ij} = \sum_{h\in S, h\neq i} \left(\beta_1 + \beta_2 d^{\gamma}_{ih}\right) f_{ihj} \qquad \forall i \in S, \forall j \in A_q \tag{38}$$

$$P^r_{ij} = \rho \sum_{h\in S, h\neq i} f_{hij} \qquad \forall i \in S, \forall j \in A_q. \tag{39}$$

P^t_{ij} and P^r_{ij} are equivalent to the expressions seen in (15) and (16).

Note that now the power dissipation is defined per application j at each node i instead of only per node i. This differentiation is needed because the remaining activity time of each application in A_q may be different, which will impact on the energy consumption of the node as shown in the following restriction.

Let X_i be a constant equal to 1 if the node i was active before the arrival of the new application, and 0 otherwise. This constant is equivalent to the variable x_{in-1} shown in the general model. Analogously, let Y_{ijk} be a constant equal to 1 if the test point k of the application j was being sensed in the node i just before the arrival of the new application, and 0 otherwise. Thus, this constant is equivalent to the variable y_{ijkn-1} previously shown. $\Delta\tau_j$ is the remaining activity time of application j at time instant τ_q ($\Delta\tau_j = \tau_j + \epsilon_j - \tau_q$), $E_i(\tau_q)$ is the remaining energy that node i still has at τ_q, and λ_i is a variable indicating the residual energy that node i would have once the activity time of the applications deployed in it or forwarded by it expires. Again, sinks do not have energy constraints. With this, the energy constraints that must be ensured in every node are

$$\sum_{j\in A_q} P^t_{ij}\Delta\tau_j + \sum_{j\in A_q} P^r_{ij}\Delta\tau_j + f\left(\sum_{j\in A_q}\sum_{k\in W_j} y_{ijk} l_j\right)\Delta\tau_j + \varphi_i \cdot (x_i - X_i)\cdot x_i + \sum_{j\in A_q}\delta_{ij}\sum_{k\in W_j}(y_{ijk} - Y_{ijk})\cdot y_{ijk} + \lambda_i = E_i(\tau_q) \qquad \forall i \in S' \tag{40}$$

with $\lambda_i \ge 0$, which are equivalent to constraints defined in (18).

D. Objective Function

The sets of restrictions described above forces the deployment of the application arriving at time instant τ_q and also of all the applications that are already active at that moment. If the solution space described by these restrictions is null, then the new application cannot be deployed ensuring the presence of the previous applications and therefore the system rejects it. If the solution space contains several feasible solutions to deploy a new application, we should select a solution that increases the probability of accepting future applications. As stated before, the remaining energy of the nodes is one of the most limiting factors to deploy new arriving applications. Therefore, we propose three possible objective functions for the optimization problem, being all of them related to the residual energy of the network or its nodes. The quality of the three strategies will be analyzed in Section VII.

The first one is to maximize the total residual energy of the network, that is

$$\max \sum_{i\in S'} \lambda_i. \tag{41}$$

The second one is to maximize the residual energy of the node with the lowest energy

$$\max \lambda \quad \lambda \le \lambda_i \quad \forall i \in S'. \tag{42}$$

Finally, we also consider a weighted sum of the two previous alternatives

$$\max\left(\lambda + \frac{1}{|S'|}\sum_{i\in S'}\lambda_i\right) \quad \lambda \le \lambda_i \quad \forall i \in S'. \tag{43}$$

In the following, we will denote these three strategies as *total*, max–min, and *mixed*, respectively.

VI. Heuristic Algorithm for Dynamic Resource Allocation

We propose hereafter a heuristic based on a greedy algorithm to solve the dynamic resource allocation problem described in Section V. This heuristic is executed each time a new application a_q arrives to the reference area.

The objective of the heuristic is to deploy the arriving application in a set of sensor nodes that cover all its test points without dropping out any of the applications running at that moment. This implies that the solution found by the heuristic must satisfy the same constraints defined in Section V. Additionally, the solution should deploy the new application using the existing resources of the network efficiently, so that the probability of deploying new applications in the future is maximized.

As the most restricting factor to deploy new applications is the spare energy of the nodes, the main idea of the greedy algorithm is to deploy the new application trying to maximize the residual energy of the energy bottleneck node. To do so, the algorithm sorts all the nodes in the set S_{qk} in an decreasing order with respect to a *metric* e_i defined as

$$e_i = \min_{h \in P_i} \hat{E}_h(\tau_q) \tag{44}$$

where P_i is the set of nodes forming the path from node i to its corresponding sink (note that these nodes are known in advance since they depend on the routing algorithm used in the network), and $\hat{E}_h(\tau_q)$ is the remaining energy that node h would have once all the applications that are deployed on it at time instant τ_q (when application a_q arrives) leave the network. The term e_i represents therefore the "bottleneck" in terms of energy among the nodes in P_i. With this, the aim is to deploy the application in the sensor node with the highest value of e_i that has enough free resources to allocate it.

The detailed pseudocode of the proposed solution is reported in Algorithm 1. The inputs of the heuristic are the available memory and processing capacity of the nodes at time instant τ_q, which are denoted as $M_i(\tau_q)$ and $L_i(\tau_q)$, respectively; the terms $\hat{E}_i(\tau_q)$ defined previously; and the available transmission resources of the links between each node i and its predecessor in the routing graph, which are denoted as $B_i(\tau_q)$. Note that, before any application has arrived to the network, $M_i(\tau_q), L_i(\tau_q)$, and $\hat{E}_i(\tau_q)$ are equal to M_i, P_i, and E_i, and $B_i(\tau_q)$ are equal to 1 indicating that the whole airtime of all the links are fully available.

Once the new application a_q has arrived, the first step is to define the variables $M_i^{(r)}, P_i^{(r)}, E_i^{(r)}$, and $B_i^{(r)}$, which are used to store a copy of $M_i(\tau_q), L_i(\tau_q), \hat{E}_i(\tau_q)$, and $B_i(\tau_q)$. These variables allow obtaining the remaining resources of the network if application a_q can be effectively deployed (i.e., if there are enough resources in the network). We also define the lists S'_{qk} to store the nodes that cover each test point w_k of application a_q and have enough resources to sense effectively this test point. Each list S'_{qk} is initialized with the elements in the corresponding set S_{qk} (line 2).

Then, the algorithm enters into its main loop and tries to sense all the test points of a_q following the energy criteria defined previously (lines 3–30). To that end, we compute the terms e_i for all the nodes in S'_{qk} using (44) and sort the nodes in a decreasing order of e_i. Then, we look for the sensor node with the highest value of e_i (sensor node s_f in line 7) and try to deploy a_q on it. For simplicity, this part of the heuristic is detailed in Algorithm 2 and will be explained later. If this is feasible, we update the variables $M_i^{(r)}, P_i^{(r)}, E_i^{(r)}$ and $B_i^{(r)}$ (line 11) and we move to the next test point of a_q. If not, the next step depends on whether sensor node s_f is already active or not. If it is not active, s_f is removed from S'_{qk} and the next node with the highest value of e_i is selected (line 27). If it is active, we try to free some of its resources by moving one of the applications it hosts to another node (lines 14–26).

In order to decide which application to move, we compute for each application a_j on s_f and each node $s_i \in S_{jk} - \{s_f\}$, the cost of moving a_j to s_i. This cost is defined as $c(a_j, s_i) = h(s_i) - h(s_f)$, with $h(s_i)$ the number of hops from sensor node s_i to its corresponding sink. We store each pair of application a_j and alternative node s_i on the list L_f and sort it on increasing order of $c(a_j, s_i)$ (lines 14 and 15). Then, we try to move the first application in this list to its corresponding

Algorithm 1 Pseudocode of the Dynamic Greedy Solution

```
1:  Application $a_q$ arrives at time $\tau_q$
2:  Initialize variables $M_i^{(r)}, P_i^{(r)}, E_i^{(r)}, B_i^{(r)}$ and lists $S'_{qk}$
3:  for all $w_k \in W_q$ do
4:      Compute $e_i$ with (44) $\forall s_i \in S'_{qk}$
5:      Sort all the nodes in $S'_{qk}$ in decreasing order of $e_i$
6:      repeat
7:          $s_f \leftarrow \arg\max_{s_i \in S'_{qk}}(e_i)$
8:          Check if $a_q$ fits into $s_f$ (alg. 2)
9:          if it fits then
10:             Test point $w_k$ is sensed
11:             Update variables $M_i^{(r)}, P_i^{(r)}, E_i^{(r)}, B_i^{(r)}$
12:         else
13:             if $s_f$ is active then
14:                 Store in $L_f$ all the pairs $(a_j, s_i)$, with $a_j$
                    on $s_f$ and $s_i \in S_{jk} - \{s_f\}$
15:                 Sort the elements of $L_f$ in increasing order
                    of $c(a_j, s_i)$
16:                 repeat
17:                     $(a_c, s_d) \leftarrow \arg\max_{(a_j,s_i) \in L_f}(c(a_j, s_i))$
18:                     Check if $a_c$ fits into $s_d$ and $a_q$ fits into
                        $s_f$ without $a_c$ (alg. 3)
19:                     if they fit then
20:                         Test point $w_k$ is sensed
21:                         Update vars. $M_i^{(r)}, P_i^{(r)}, E_i^{(r)}, B_i^{(r)}$
22:                     else
23:                         Remove $(a_c, s_c)$ from $L_f$
24:                     end if
25:                 until $w_k$ is sensed or $L_f$ is empty
26:             end if
27:             Remove $s_f$ from $S'_{qk}$
28:         end if
29:     until $w_k$ is sensed or $S'_{qk}$ is empty
30: end for
31: if all $w_k \in W_q$ are sensed then
32:     Deploy $a_q$ and update $M_i(\tau_q)$, $L_i(\tau_q)$, $\hat{E}_i(\tau_q)$, $B_i(\tau_q)$
33: else
34:     Reject $a_q$
35: end if
```

Algorithm 2 Check if Application a_q Fits Into the Node s_f

```
1:  Initialize variables $M_i^{(a)}, P_i^{(a)}, E_i^{(a)}, B_i^{(a)}$
2:  $M_f^{(a)} \leftarrow M_f^{(a)} - m_q$
3:  $P_f^{(a)} \leftarrow P_f^{(a)} - p_q$
4:  for all $s_g \in P_f$ do
5:      if $s_g = s_f$ then
6:          $E_f^{(a)} \leftarrow E_f^{(a)} - \epsilon_q\Big(f(l_q) + c_q\Big(\beta_1 + \beta_2 d_{f\to s(f)}^{\gamma}\Big)\Big)$
7:      else
8:          $E_g^{(a)} \leftarrow E_g^{(a)} - \epsilon_q c_q\Big(\rho + \beta_1 + \beta_2 d_{g\to s(g)}^{\gamma}\Big)$
9:      end if
10:     if $s_g$ is off then
11:         $E_g^{(a)} \leftarrow E_g^{(a)} - \varphi_g$
12:     end if
13:     $B_g^{(a)} \leftarrow B_g^{(a)} - c_q/C_g$
14:     for all $l_h \in I_g$ do
15:         $B_h^{(a)} \leftarrow B_h^{(a)} - c_q/C_g$
16:     end for
17: end for
18: if any variable $M_i^{(a)}, P_i^{(a)}, E_i^{(a)}, B_i^{(a)}$ is $< 0$ then
19:     return $a_q$ does not fit into $s_f$
20: end if
21: return $a_q$ fits into $s_f$ and $M_i^{(a)}, P_i^{(a)}, E_i^{(a)}, B_i^{(a)}$
```

alternative node (a_c and s_d in line 17) and deploy a_q in s_f. For simplicity, this part of the heuristic is detailed in Algorithm 3 and will be explained later. If they fit, we update the variables $M_i^{(r)}, P_i^{(r)}, E_i^{(r)}$ and $B_i^{(r)}$ (line 21) and we move to the next test point of a_q. If not, we remove the pair (a_c, s_d) from L_f and repeat the process iteratively until the list is empty. When this happens, we remove s_f from S'_{qk} and the next node with the highest value of e_i is selected.

If after all this process we cannot sense a test point in W_q (i.e., the list S'_{qk} is empty for some $w_k \in W_q$), then the application is rejected and the system remains unchanged in a way that the previous applications can be served. On the contrary, if all the test points are sensed, we deploy the incoming application in the nodes selected during the execution of the algorithm, updating the resources of the involved nodes [i.e., the terms $M_i(\tau_q)$, $L_i(\tau_q)$, $\hat{E}_i(\tau_q)$, and $B_i(\tau_q)$].

Now, we describe the procedure followed to determine if the application a_q fits into the sensor node s_f (Algorithm 2). First, we define the variables $M_i^{(a)}, P_i^{(a)}, E_i^{(a)}$, and $B_i^{(a)}$, which are used to store a copy of $M_i^{(r)}, P_i^{(r)}, E_i^{(r)}$, and $B_i^{(r)}$ and are initialized with their values. These variables allow obtaining the remaining resources of the nodes if a_q can be effectively deployed on s_f.

To check this, we have to compute the remaining resources of all the nodes that would be affected if the application were deployed on s_f.

1) The remaining available memory and processing capacity of node s_f (lines 2 and 3).
2) The remaining energy of s_f. To compute it, we have to remove the energy that s_f requires to sense the test point, which corresponds to the term $\epsilon_q f(l_q)$, and the energy required to send the gathered data to the next node in the path to its sink, which corresponds to the term $\epsilon_q c_q(\beta_1 + \beta_2 d_{f\to n(f)}^{\gamma})$. In this expression $n(f)$ is the node corresponding to the next hop of s_f in the path calculated by the routing algorithm, and $d_{f\to n(f)}$ is the distance between s_f and $s_{n(f)}$ (line 6). Additionally, if s_f is off, we also have to consider the energy required to power it on (line 11).
3) The remaining energies of the rest of nodes in the path P_f. To compute them, we have to remove for each node s_g in the path (different from s_f) the energy that it requires to receive the sensed data from the previous node in the path, which corresponds to $\epsilon_q c_q \rho$, and the

Algorithm 3 Check if a_q Fits Into the Node s_f When a_c Is Moved to s_f

```
1:  Initialize variables $M_i^{(a)}, P_i^{(a)}, E_i^{(a)}, B_i^{(a)}$
2:  $M_d^{(a)} \leftarrow M_d^{(a)} - m_c$
3:  $P_d^{(a)} \leftarrow P_d^{(a)} - p_c$
4:  for all $s_g \in P_d$ do
5:      if $s_g = s_d$ then
6:          $E_d^{(a)} \leftarrow E_d^{(a)} - \delta_{cd}$
                $-\big(\tau_c + \epsilon_c - \tau_q\big)\Big(f(l_c) + c_c\Big(\beta_1 + \beta_2 d_{d\to s(d)}^{\gamma}\Big)\Big)$
7:      else
8:          $E_g^{(a)} \leftarrow E_g^{(a)}$
                $-c_q\big(\tau_c + \epsilon_c - \tau_q\big)\Big(\rho + \beta_1 + \beta_2 d_{g\to s(g)}^{\gamma}\Big)$
9:      end if
10:     if $s_g$ is off then
11:         $E_g^{(a)} \leftarrow E_g^{(a)} - \varphi_g$
12:     end if
13:     $B_g^{(a)} \leftarrow B_g^{(a)} - c_c/C_g$
14:     for all $l_h \in I_g$ do
15:         $B_h^{(a)} \leftarrow B_h^{(a)} - c_c/C_g$
16:     end for
17: end for
18: $M_f^{(a)} \leftarrow M_f^{(a)} - m_q + m_c$
19: $P_f^{(a)} \leftarrow P_f^{(a)} - p_q + p_c$
20: for all $s_g \in P_f$ do
21:     if $s_g = s_f$ then
22:         $E_f^{(a)} \leftarrow E_f^{(a)} - \epsilon_q\Big(f(l_q) + c_q\Big(\beta_1 + \beta_2 d_{f\to s(f)}^{\gamma}\Big)\Big)$
                $+\big(\tau_c + \epsilon_c - \tau_q\big)\Big(f(l_c) + c_c\Big(\beta_1 + \beta_2 d_{f\to s(f)}^{\gamma}\Big)\Big)$
23:     else
24:         $E_g^{(a)} \leftarrow E_g^{(a)} - \epsilon_q c_q\Big(\rho + \beta_1 + \beta_2 d_{g\to s(g)}^{\gamma}\Big)$
                $+c_c\big(\tau_c + \epsilon_c - \tau_q\big)\Big(\rho + \beta_1 + \beta_2 d_{g\to s(g)}^{\gamma}\Big)$
25:     end if
26:     $B_g^{(a)} \leftarrow B_g^{(a)} - (c_q - c_c)/C_g$
27:     for all $l_h \in I_g$ do
28:         $B_h^{(a)} \leftarrow B_h^{(a)} - (c_q - c_c)/C_g$
29:     end for
30: end for
31: if any variable $M_i^{(a)}, P_i^{(a)}, E_i^{(a)}, B_i^{(a)}$ is $< 0$ then
32:     return $a_q$ does not fit into $s_f$
33: end if
34: return $a_q$ fits into $s_f$ and $M_i^{(a)}, P_i^{(a)}, E_i^{(a)}, B_i^{(a)}$
```

energy required to retransmit them to the next node, which corresponds to $\epsilon_q c_q(\beta_1 + \beta_2 d^{\gamma}_{g\to n(g)})$ (line 8). Again, if s_g is off, we also have to consider the energy required to power it on (line 11).

4) The remaining transmission resources of each link of the path P_f (line 13). We name as l_g (or directly g) the link between the node s_g and the node being its next hop $s_{n(g)}$. For each link, the required transmission resources correspond to the airtime needed to transmit the sensed data, which is c_q/C_g.
5) The remaining transmission resources of the set of links that interfere or are interfered by link l_g (lines 14 and 15). We name this set as I_g and it is formed by the links whose receiver is within the interference range of the node s_g and the links where the node $s_{n(g)}$ is within the interference range of its transmitter.

If all the auxiliary variables $M_i^{(a)}, P_i^{(a)}, E_i^{(a)}$, and $B_i^{(a)}$ are higher than zero, we can state that a_q fits into s_f (line 21).

Finally, we describe the procedure followed to determine if it is possible to deploy application a_q into the sensor node s_f when we move application a_c from s_f to s_d (Algorithm 3). This procedure consists of two parts, being each of them very similar to the procedure described in Algorithm 2. In the first one, we check if sensor node s_d has enough resources to host application a_c (lines 4–17). The only difference with respect to the verifications done in Algorithm 2 is that the time that the application is in the network is not ϵ_c but $\tau_c + \epsilon_c - \tau_q$, which impacts on the energy requirements of the nodes (lines 6–8), as well as the energy impact of sending the bytecode of a_c to s_d. In the second one, we check if sensor node s_f can host a_q once the resources used by a_c are freed from s_f (lines 20–33). In this case, the difference with respect to Algorithm 2 is that we must add the new available resources that were previously used by a_c.

VII. Performance Evaluation

In this section, we evaluate in detail the performance of the proposed strategies. Unless otherwise stated, results have been obtained by solving the optimization models of Sections IV and V using CPLEX software [30]. The simulated scenarios consider several area sizes and different number of applications and nodes. In all the cases, results have been obtained averaging the outcome of 100 realizations. We consider a default sensing range of $R_i^s = 40$ m for all the sensors [31] and a two-ray ground path loss model with $\gamma = 4$ and $g_0 = 8.1 \cdot 10^{-3}$ [32]. $P_{\max}$ is set to -10 dBm and the receiver sensitivity α is fixed to -92 dBm [33], which implies a maximum transmission range $R^T_{\max}$ of 33 m. Similarly, the interference sensitivity μ is set to -104 dBm, which implies a maximum interference range $R^I_{\max}$ of 67 m.

A. Applications and Sensor Nodes

As a reference, we have focused on *multimedia* applications, which require the sensing, processing and delivery of multimedia content (images and video). Specifically, we consider visual sensor networks, i.e., WSNs designed to perform visual analysis (e.g., object recognition) [34]. In that work, a detailed characterization of transmission rates and energy consumption for these applications is provided. Based on this analysis, the transmission rate is set to 12 Kb/s, the required memory to 842 KB, the processing load to 69.23 MIPS and the associated power dissipation [function f in (17) and (40)] to 0.2 W. Details on how these numbers have been derived are shown in [20]. Thus, the requirement vector for visual applications is $o_j = \{12 \text{ kb/s}, 842 \text{ KB}, 69.23 \text{ MIPS}\}$.

To support these visual applications, we consider *high-level* sensor node hardware. Using as a reference BeagleBone platforms [35] or similar, the sensor nodes are assumed to have a 720 MHz super-scalar ARM Cortex-A8 processor (up to 720 MIPS) and 256 MB of RAM. In addition they are equipped with an IEEE 802.15.4 radio with an integrated antenna and a low-power USB camera. The reference resource vector is $O_i = \{250 \text{ kb/s}, 256 \text{ MB}, 720 \text{ MIPS}, 32400 \text{ J}\}$. The energy budget for all the nodes but sinks, which we assume that can be plugged directly into the grid, is 32400 J assuming that a node runs at 3 V with 3 Ah of battery supply (two AA batteries).

B. General Optimization Framework Results

We begin by assessing the general optimization framework of Section IV. The solution of this problem gives an upper-bound to the performance of the strategies evaluated in the following section and gives a hint on the value of "future information" (i.e., the knowledge about the applications that have not arrived to the system yet). Nevertheless, this model is neither realistic nor easy to solve computationally due to the extremely high number of variables needed to solve it.

Since the number of variables needed to solve the general problem increases quickly, we need to use a smaller scenario than the used in the following section. We have considered a scenario formed by 18 BeagleBone nodes, where 100 visual applications are generated according to a Poisson process with rate of 0.5 applications per hour and a constant activity time ϵ_j of 5 h. We assume that each sensor is able to cover $N_{ij} = 1$ test point of the same application and that there is 1 sink node in the network. Sensor nodes are deployed in a 100×100 m scenario.

TABLE III
Deployed Applications for Each Strategy

	Strategy					
	O.R.	Total	Max-min	Mixed	Heuristic	Global
1 TP. A.	97.85	99.65	99.65	99.65	99.64	99.94
2 TP. A.	76.71	75.36	91.71	95.64	87.64	98.57

TP. A. Test Point per Application

Table III gives the number of deployed applications achieved for the general optimization strategy seen in Section IV (*global*), the different proposed strategies with the dynamic model (*total*, max–min, and *mixed*) and the heuristic algorithm (*heuristic* in the legends). In addition, results are also presented for the case where no objective function is considered, and the dynamic optimization problem only tries

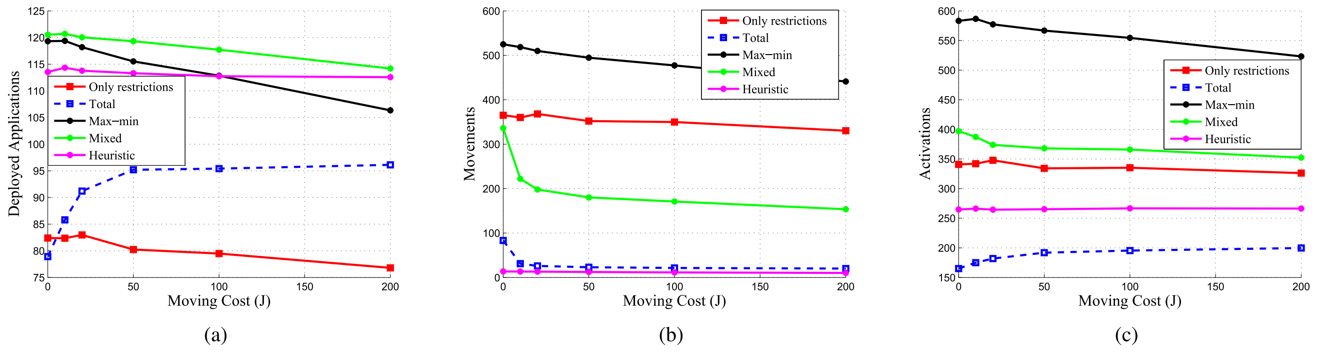


Fig. 1. Impact of moving cost. (a) Deployed applications. Number of (b) movements and (c) activations. $\varphi_i = 10$ J $\forall i \in S$.

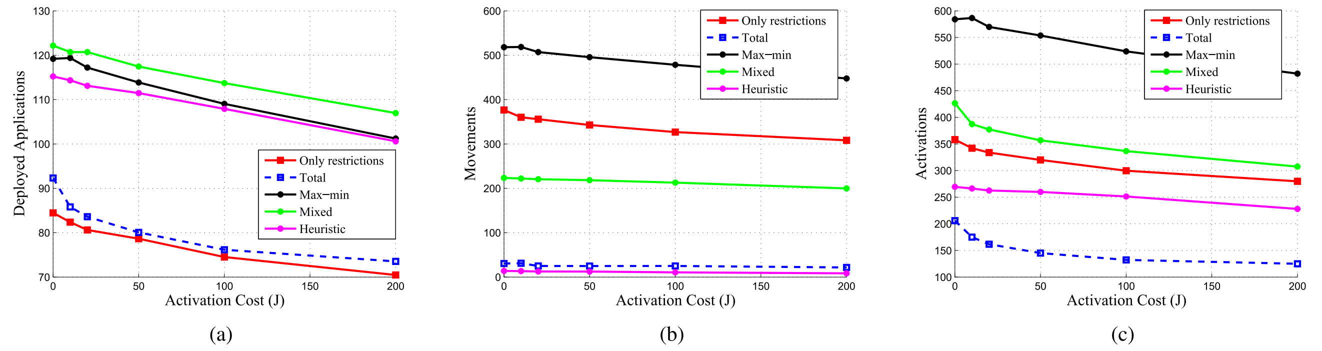


Fig. 2. Impact of activation cost. (a) Deployed applications. Number of (b) movements and (c) activations. $\delta_{ij} = 10$ J $\forall i \in S, \forall j \in A$.

to deploy the application satisfying all the restrictions [only restrictions (O.R.) in the legends].

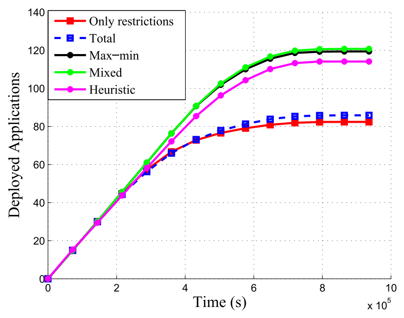
The number of test points for each application is 1 or 2. As can be seen, the improvement of solving the *global* problem is not significant, so the proposed solutions for optimizing online the energy of the network work properly.

C. Dynamic Resource Allocation Results

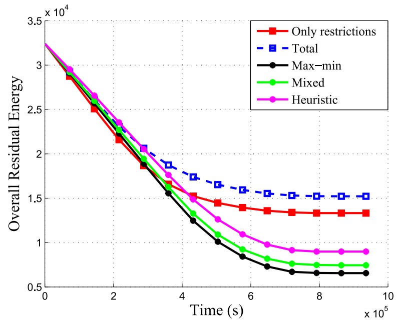
The following results have been obtained by solving the optimization model of Section V and the heuristic algorithm proposed in Section VI. We have considered a scenario formed by 36 BeagleBone nodes as a reference example to thoroughly evaluate the validity of the dynamic model, the performance of the different objective functions proposed in Section V-D and the heuristic greedy algorithm. In each realization, 200 visual applications are generated according to a Poisson process with rate of one application per hour and a constant activity time ϵ_j of 5 h. The number of test points is 3 for each application, and we assume that each sensor is able to cover $N_{ij} = 1$ test point of the same application and that there are two sink nodes in the network. Sensor nodes are deployed in a 141×141 m scenario.

Figs. 1 and 2 show the impact of the energy cost of moving an application j from one node h to another one i, δ_{ij}, and the energy cost of activating a node i, φ_i, for the different proposed strategies (*total*, max–min, *mixed*, *heuristic*, and O.R.). Fig. 1(a) shows that the total number of actually deployed applications in the system is higher for the *mixed* approach for all the moving cost values; as expected the worst results are obtained with the O.R. approach. The max–min and the greedy heuristic algorithm provide results close to the *mixed* approach. The near-flat behavior of the heuristic can be explained by the extremely low number of movements it performs: as there are barely any movement, the impact of the moving cost is negligible. More details about this behavior are given later, when explaining Figs. 5 and 6. On the other hand, when the total energy is maximized, the number of deployed applications rises with higher moving costs. This can be explained as follows: when the moving cost is low and a new application arrives at the system, it is preferred to move one current application from one active node to another, rather than activating a new node so as to maximize the overall residual energy. This makes new applications tend to be located in the activated nodes, making the energy of these nodes be spent faster. In the end, this leads to nodes running out of energy earlier, making the network disjoint and reducing the number of applications that can be deployed.

Fig. 1(b) and (c) shows that O.R. and max–min are the strategies that have more movements and activations. This is straightforward for O.R., since applications are deployed without any additional objective rather than fulfilling the constraints, and therefore the specific nodes where the applications are deployed are chosen randomly as long as the constraints are satisfied. A similar explanation can be applied to the max–min strategy: (40) and (42) only consider the node with the lowest energy, so the remaining nodes can be activated or receive an application without any penalty in the objective function. On the other hand, the lowest number of movements is obtained with the heuristic algorithm since in this case the



(a)



(b)

Fig. 3. Temporal evolution of system performance. (a) Deployed applications. (b) Overall residual energy. $\varphi_i = 10$ J. $\delta_{ij} = 10$ J. $\forall i \in S, \forall j \in A$.

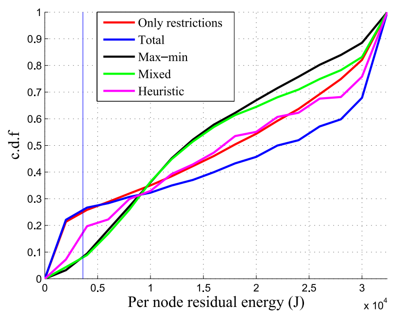
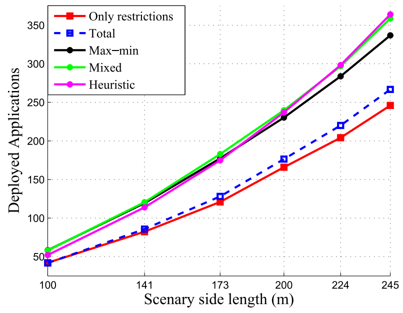
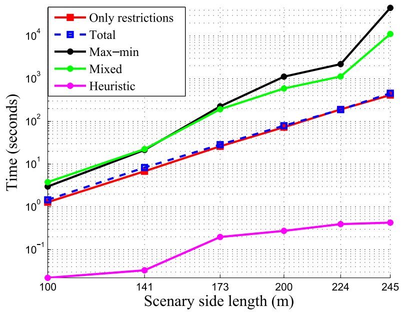


Fig. 4. Cumulative distribution function of the per node residual energy at time 360 000.



(a)



(b)

Fig. 5. Performance evaluation of the heuristic algorithm versus the dynamic scheme. (a) Deployed applications. (b) Computation time (in log scale).

number of possible movements that are tested is the lowest: we only move an application from one node to another so as to put the arriving application in its instead.

Finally, it must be noted [Fig. 1(c)] that the number of activations remains almost constant (and not rises) when the moving cost increases for the O.R., *mixed*, and *heuristic* strategies. This is because the activation of a new node also implies that this node has to receive the bytecode of the application, so the higher moving cost cannot be compensated by activating more nodes. However, for the *total* strategy, the number of activations increases as the moving cost increases. As noted in the explanation of Fig. 1(a), for low moving costs, this solution tends to move applications between active nodes instead of activating new nodes.

Fig. 2(a) shows that the *mixed* strategy keeps providing the best performance in terms of deployed applications for different values of the activation cost. Again, for the same reasons explained above, max–min is the strategy with more movements and activations [Fig. 2(b) and (c)]. In addition, it is worth noting again that for *heuristic*, the number of movements and activations keeps almost constant as the activation cost increases.

In Figs. 3 and 4, we focus on the temporal evolution of the system fixing both φ_i and δ_{ij} to 10 J. As can be seen in Fig. 3(a), at first (when all the nodes have a high residual energy), applications are equally deployed for all the strategies. However, as time passes and energy depletes, the fairer energy distribution obtained by the *mixed* and max–min strategies allows deploying more applications in the system. Comparing Fig. 3(a) and (b), an inverse relationship between the residual energy and the deployed applications is observed. Therefore, the key to design a proper strategy is not strictly maximizing the total energy of the network (which is done with the *total* strategy), but to ensure that the network has enough energy to keep being a connected graph as applications are deployed.

In Fig. 4, the cumulative distribution function of the residual energy per node at time 360 000 s is shown. The vertical blue line represents the minimum energy required for a node to sense an application and send the data at the maximum transmission power. As can be seen, the probability of a node not having enough energy to admit a new application is the lowest for the *mixed* and max–min strategies, which confirms the results shown in Fig. 3(a).

In order to assess the performance of the proposed solutions for different scenario sizes, we vary now the size of the reference network topology while maintaining the same node density and a maximum transmitted power of $P_{\max} = -10$ dBm. Both activation and moving costs are set to 10 J. Table IV summarizes the main features of the tested scenarios.

TABLE IV
SCENARIO TOPOLOGIES

	Scenario					
	1	2	3	4	5	6
Size	100 × 100 m	141 × 141 m	173 × 173 m	200 × 200 m	224 × 224 m	245 × 245 m
Number of nodes	18	36	54	72	90	108
Number of sinks	1	2	3	4	5	6
Number of applications	100	200	300	400	500	600
Arrival rate	0.5 app/hour	1 app/hour	1.5 app/hour	2 app/hour	2.5 app/hour	3 app/hour

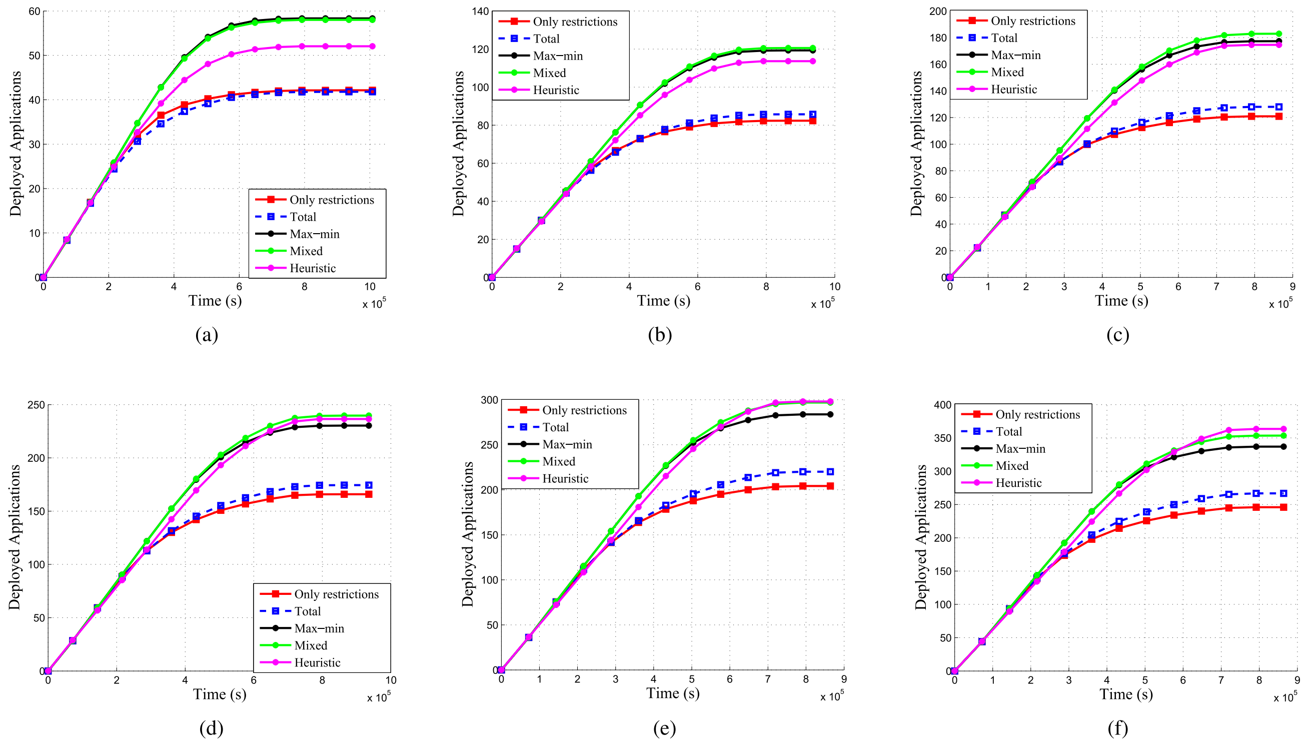


Fig. 6. Temporal evolution of the deployed applications for different scenarios. (a) 100×100 m. (b) 141×141 m. (c) 173×173 m. (d) 200×200 m. (e) 224×224 m. (f) 245×245 m.

For example, the first one is a 100 × 100 m scenario with 18 BeagleBones (one of them acting as a sink) where 100 ATC applications are generated according to a Poisson process with rate of 0.5 applications per hour and a constant activity time ϵ_j of 5 h. In all the cases, the number of test points per application is 3, and we assume that each sensor is able to cover $N_{ij} = 1$ test points of the same application.

Fig. 5 depicts the number of deployed applications and the solution time of the proposed solutions. The results have been obtained on 3.0-GHz Quad Core Intel Woodcrest (64 bits) machines with 8-GB RAM and 250-GB SATA storage, averaging over 100 randomly generated network topologies for each scenario of Table IV. These results show that for all the scenarios, the computation time of the *heuristic* approach is much lower. According to the number of deployed applications, the *heuristic* approach is always close to the *mixed* and max–min strategies. Therefore, we can consider that the proposed heuristic achieves near optimal results for all the scenarios considered so far (varying scenario sizes, moving cost, and activation size), with a negligible computational cost.

Additionally, in Fig. 5(a) we notice that for large scenarios, the deployed applications with the heuristic algorithm get slightly higher values than the other strategies. This can be explained as follows: in some cases, the heuristic algorithm rejects arriving applications that could have been deployed if a more exhaustive search had been performed. Typically, these rejected applications tend to consume more network resources than the accepted ones, so the rejections also save more resources for subsequent applications. Thus, later on, when nodes have little residual energy, new applications consuming few resources are more likely to be deployed with the heuristic algorithm. This effect of rejecting some high resource consuming applications to later accept more low resource consuming ones is more evident as the total number of applications rises, in large scenarios. This behavior is confirmed in Fig. 6, where the temporal evolution of the number of deployed applications is shown. For large scenarios [Fig. 6(e) and (f)], the *heuristic* goes below the *mixed* strategy at first (for $t < 6 \cdot 10^5$) because the *heuristic* does not find some feasible solutions that the *mixed* does since the *heuristic* does not examine the entire solution space whereas the *mixed* strategy solves the optimization problem, and therefore, always finds the feasible solution, if any. Nevertheless, this saved energy is used afterwards (for $t > 6 \cdot 10^5$), allowing the *heuristic* to deploy some applications that the *mixed* strategy cannot. Finally, coming back to Fig. 1(a), it was shown that the *heuristic* has a near-flat behavior with the moving cost while for the max–min and *mixed* strategies the performance degrades as the moving cost grows. This can be better explained now: the *heuristic* performs a low number of movements and some applications are rejected because of this. As the moving cost rises, the *heuristic* saves more energy with respect to the other strategies by rejecting those applications. Thus, later on, more applications consuming few resources that can no longer be admitted by the max–min or *mixed* strategies can still be deployed by the *heuristic*.

VIII. Conclusion

In this paper, we have studied the problem of virtual sensor network management from the perspective of an SSN-IP that leases its physical resources to multiple concurrent applications/application providers. Namely, a joint optimization framework has been introduced to solve the problem of AAC-SNS. The proposed framework optimally decides: 1) if/when to admit new applications to the use of the physical infrastructure and 2) how to allocate the physical resources of the shared infrastructure to the multiple concurrent applications, while considering constraints at the sensor node level (processing power and storage) as well as at the network level (available bandwidth, shared communication technologies, and routing). The results of the proposed optimization framework have been compared against the ideal scenario where the SSN-IP has full knowledge of the application/service request across time, which is introduced as a performance benchmark. Moreover, a greedy heuristic is also proposed to obtain close-to-optimal solutions of joint AAC-SNS problem in short computation time.

References

[1] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): Infrastructure and operations," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 640–656, 1st Quart., 2017.

[2] K. Samdanis *et al.*, "5G network slicing—Part 1: Concepts, principles, and architectures," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 70–71, May 2017.

[3] I. Khan *et al.*, "Wireless sensor network virtualization: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 553–576, 1st Quart., 2016.

[4] C. M. D. Farias *et al.*, "A systematic review of shared sensor networks," *ACM Comput. Surveys*, vol. 48, no. 4, pp. 1–50, May 2016, doi: 10.1145/2851510.

[5] P. Levis and D. Culler, "Maté: A tiny virtual machine for sensor networks," *SIGARCH Comput. Archit. News*, vol. 30, no. 5, pp. 85–95, Oct. 2002, doi: 10.1145/635506.605407.

[6] P. Levis, D. Gay, and D. Culler, "Active sensor networks," in *Proc. 2nd Conf. Symp. Netw. Syst. Design Implement. (NSDI)*, vol. 2. Berkeley, CA, USA, 2005, pp. 343–356. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251203.1251228

[7] Y. Yu, L. J. Rittle, V. Bhandari, and J. B. LeBrun, "Supporting concurrent applications in wireless sensor networks," in *Proc. 4th Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, Boulder, CO, USA, 2006, pp. 139–152, doi: 10.1145/1182807.1182822.

[8] J. Koshy and R. Pandey, "VMSTAR: Synthesizing scalable runtime environments for sensor networks," in *Proc. 3rd Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, San Diego, CA, USA, 2005, pp. 243–254. doi: 10.1145/1098918.1098945.

[9] X. Zhu, X. Tao, T. Gu, and J. Lu, "ReLog: A systematic approach for supporting efficient reprogramming in wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 102, pp. 132–148, Apr. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731516301861

[10] I. Leontiadis, C. Efstratiou, C. Mascolo, and J. Crowcroft, "SenShare: Transforming sensor networks into multi-application sensing infrastructures," in *Proc. 9th Eur. Conf. Wireless Sensor Netw. (EWSN)*, Trento, Italy, 2012, pp. 65–81.

[11] S. Bhattacharya, A. Saifullah, C. Lu, and G.-C. Roman, "Multi-application deployment in shared sensor networks based on quality of monitoring," in *Proc. 16th IEEE Real Time Embedded Technol. Appl. Symp. (RTAS)*, Stockholm, Sweden, Apr. 2010, pp. 259–268.

[12] C.-L. Fok, C. Julien, G.-C. Roman, and C. Lu, "Challenges of satisfying multiple stakeholders: Quality of service in the Internet of Things," in *Proc. 2nd Workshop Softw. Eng. Sensor Netw. Appl. (SESENA)*, 2011, pp. 55–60, doi: 10.1145/1988051.1988062.

[13] W. Li *et al.*, "Efficient allocation of resources in multiple heterogeneous wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 74, no. 1, pp. 1775–1788, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731513002104

[14] R. Huang, X. Chu, J. Zhang, and Y. H. Hu, "Energy-efficient monitor-ing in software defined wireless sensor networks using reinforcement learning: A prototype," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 10, pp. 1–12, Oct. 2015.

[15] Y. Xu, A. Saifullah, Y. Chen, C. Lu, and S. Bhattacharya, "Near optimal multi-application allocation in shared sensor networks," in *Proc. 11th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, Chicago, IL, USA, 2010, pp. 181–190, doi: 10.1145/1860093.1860118.

[16] C. Wu, Y. Xu, Y. Chen, and C. Lu, "Submodular game for distributed application allocation in shared sensor networks," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 127–135.

[17] S. M. Ajmal, S. Paris, Z. Zhang, and F. N. Abdesselam, "An efficient admission control algorithm for virtual sensor networks," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun. 6th Int. Symp. Cyberspace Safety Security 11th Int. Conf. Embedded Softw. Syst. (HPCC CSS ICESS)*, Paris, France, Aug. 2014, pp. 735–742.

[18] C. M. de Farias *et al.*, "A scheduling algorithm for shared sensor and actuator networks," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Bangkok, Thailand, Jan. 2013, pp. 648–653.

[19] D. Zeng *et al.*, "Energy minimization in multi-task software-defined sensor networks," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3128–3139, Nov. 2015.

[20] C. Delgado *et al.*, "On optimal resource allocation in virtual sensor networks," *Ad Hoc Netw.*, vol. 50, pp. 23–40, Nov. 2016.

[21] M. P. Johnson *et al.*, "Sensor-mission assignment in constrained environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 11, pp. 1692–1705, Nov. 2010.

[22] T. L. Porta, C. Petrioli, C. Phillips, and D. Spenza, "Sensor mission assignment in rechargeable wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 10, no. 4, pp. 1–39, Jun. 2014, doi: 10.1145/2594791.

[23] H. Cheng, R. Guo, Z. Su, N. Xiong, and W. Guo, "Service-oriented node scheduling schemes with energy efficiency in wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 2, 2014, Art. no. 247173, doi: 10.1155/2014/247173.

[24] N. Edalat and M. Motani, "Energy-aware task allocation for energy harvesting sensor networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2016, pp. 1–14, Jan. 2016, Art. no. 28, doi: 10.1186/s13638-015-0490-3.

[25] Y. Shi, Y. T. Hou, J. Liu, and S. Kompella, "Bridging the gap between protocol and physical models for wireless networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 7, pp. 1404–1416, Jul. 2013.

[26] T. Winter *et al.*, "RPL: IPv6 routing protocol for low-power and lossy networks," Internet Eng. Task Force, Fremont, CA, USA, RFC 6550, Mar. 2012.

[27] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, 2002.

[28] A. Redondi, M. Cesana, and M. Tagliasacchi, "Rate-accuracy optimization in visual wireless sensor networks," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Orlando, FL, USA, Oct. 2012, pp. 1105–1108.

[29] Y. T. Hou, Y. Shi, and H. D. Sherali, “Rate allocation and network lifetime problems for wireless sensor networks,” *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 321–334, Apr. 2008.

[30] (2015). *ILOG CPLEX*. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[31] A. Chen, S. Kumar, and T. H. Lai, “Designing localized algorithms for barrier coverage,” in *Proc. 13th Annu. ACM Int. Conf. Mobile Comput. Netw. (MobiCom)*, Montreal, QC, Canada, 2007, pp. 63–74.

[32] C. Suh, J.-E. Joung, and Y.-B. Ko, “New RF models of the TinyOS simulator for IEEE 802.15.4 standard,” in *Proc. IEEE WCNC*, Hong Kong, Mar. 2007, pp. 2236–2240.

[33] *2.4 GHz IEEE 802.15.4/ZigBee-Ready RF Transceiver*, CC2420 Datasheet, Texas Instrum. Inc., Dallas, TX, USA, Mar. 2003.

[34] A. Redondi, L. Baroffio, L. Bianchi, M. Cesana, and M. Tagliasacchi, “Compress-then-analyze versus analyze-then-compress: What is best in visual sensor networks?” *IEEE Trans. Mobile Comput.*, vol. 15, no. 12, pp. 3000–3013, Dec. 2016.

[35] G. Coley, *BeagleBone Rev a6 System Reference Manual*, Texas Instrum., Dallas, TX, USA, 2012.