On optimal resource allocation in virtual sensor networks

On Optimal Resource Allocation in Virtual Sensor Networks

Carmen Delgado[a], José Ramón Gállego[a,*], María Canales[a], Jorge Ortín[b], Sonda Bousnina[c], Matteo Cesana[c]

[a]*Aragón Institute of Engineering Research, Universidad de Zaragoza, Spain*
[b]*Centro Universitario de la Defensa, Zaragoza, Spain*
[c]*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy.*

Abstract

Sensor network virtualization is a promising paradigm to move away from highly-customized, application-specific wireless sensor networks deployment by opening up to the possibility of dynamically assigning general purpose physical resources to multiple stakeholder applications. In this field, this paper introduces an optimization framework to perform the allocation of physical shared resources of wireless sensor networks to multiple requesting applications. The proposed optimization framework aims at maximizing the total number of applications which can share a common physical network, while accounting for the distinguishing characteristics and limitations of the wireless sensor environment (limited storage, limited processing power, limited bandwidth, tight energy consumption requirements). Due to the complexity of the optimization problem, a heuristic algorithm is also proposed. The proposed framework is finally applied to realistic network topologies to provide a detailed performance evaluation and to assess the gain involved in letting multiple applications share a common physical network with respect to one-application, one-network vertical design approaches.

Keywords: Wireless Sensor Networks; Virtualization; Resource Allocation; Internet of Things; Optimization

*Corresponding author
Email address: jrgalleg@unizar.es (José Ramón Gállego)

1. Introduction

In the Internet of Things (IoT) vision, the Internet is "pushed down" to everyday objects which are equipped with sensing capabilities to gather information on the environment they are immersed in, processing/storage capabilities to locally filter and store date, and communication peripherals to deliver the collected/processed data remotely either directly, or through multi-hop paths leveraging the cooperation of other smart objects for traffic relaying. In this last case, network of smart objects, often referred to as Wireless Sensor Networks, are set up to collect and deliver data in specific areas. WSNs can be deployed in diverse scenarios and environments to support diverse application/services ranging from smart home or environmental monitoring based on scalar sensed data to more demanding applications based on multimedia sensors.

Usually, WSNs are designed and deployed in a "vertical", application-specific way, in which the hardware and network resources are customized to the specific application requirements. On one hand, such design paradigm allows to have "optimal" performance on the specific application, but, on the other hand, it precludes resources (hardware and software) reuse when other applications and services must be contemplated. In the end, this has led in the past to the proliferation of redundant WSNs deployments [1].

In this context, novel approaches are recently being investigated targeting the smart reuse of general purpose wireless sensor networks to dynamically support multiple applications and services. The key idea behind these approaches, which often go under the names of Virtual Sensor Networks (VSN) [2] or Software Defined Sensor Networks (SDSN) [3], is to decouple the physical infrastructure and resources from application ownerships which leads to more efficient resource utilization, lower cost and increased flexibility and manageability in WSN deployments [4]. Network virtualization technologies are used to abstract away "physical resources" including node processing/storage capabilities, available communication bandwidth and routing protocols, which can then be "composed" at a logical level to support usage by multiple independent users and even by multiple concurrent applications [5]. While network virtualization and software defined networks are already a reality in many communica-

tion networks [6, 7], research on sensor network virtualization is still in its infancy and comprehensive solutions still need to be found to cope with the specific characteristics of WSNs in terms of limited node capabilities and communication bandwidth.

In this work, we focus on the design of a virtualization engine for WSNs. Namely, we consider a general purpose WSN which can be used to support multiple applications and we propose a mathematical programming framework to optimally allocate shared physical resources to the requesting applications. In more details, the proposed framework allocates the physical resources of the general purpose WSN to multiple concurrent applications while accounting for the network- and hardware-specific constraints (processing, storage, available bandwidth, limited communication range) and the specific application requirements. Due to the high computational complexity of the resulting optimization model, a heuristic algorithm is also proposed. Numerical results are then obtained by applying the proposed framework to realistic WSN instances to assess the efficiency of the virtualization process.

The paper is organized as follows: Section 2 overviews the related work in the field of sensor network virtualization. Section 3 describes the proposed system model and the optimization problem for resource allocation in virtual sensor networks, including a complexity analysis of this problem. Section 4 explains the proposed heuristic algorithm. In Section 5, the proposed optimization model and heuristic algorithm are evaluated by simulation for a set of scalar and multimedia applications also with different types of sensor nodes. Finally, some conclusions are provided in Section 6.

2. Related Work

The emergence of shared sensor networks has stimulated research efforts in the field of novel programming abstractions at the node level and management framework at the network level to support multiple applications over a shared physical infrastructure [8],[9], [10], [11].

At the node level, architectures based on virtual machines are proposed to enable virtualization and re-programmability. As an example, Mate [12], ASVM [13], Melete [14] and VMStar [15] are frameworks for building application-specific virtual

machines over constrained sensor platforms.

At the network level, several virtualization management platforms have been introduced. SenSHare [16] creates multiple overlay sensor networks which are "owned" by different applications on top of a shared physical infrastructure. UMADE [17] is an integrated system for allocating and deploying applications in shared sensor networks based on the concept of Quality of Monitoring (QoM). Fok *et al.* [18] introduce middleware abstractions to represent multiple QoM requirements from multiple applications, whereas a service-oriented middleware is presented in [19] to address the challenges faced by running multiple applications onto heterogeneous WSNs. A prototype of Software Defined Wireless Sensor Network is proposed in [20] where a centralized control plane dynamically manages communication routes in the network with the goal of augmenting the energy efficiency.

Generally speaking, the aforementioned work provides "practical" building blocks to build up virtual sensor networks. Differently, we focus in this paper on the "intelligence" to properly allocate physical resources to virtual applications, which can be cast as a general resource allocation problem. Even if radio/network resource allocation is a widely debated topic in the literature, still very few works have appeared on the optimal resource allocation in the field of Virtual or Shared Sensor Networks.

In [21] the authors propose an optimization framework to maximize the Quality of Monitoring (QoM) in shared sensor networks. The proposed framework focuses on environmental monitoring applications whose reference "quality" can be modeled as dependent on the variance in the sensed data, and derives the application-to-sensors assignment which minimizes such variance. 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 [22]. Ajmal *et al.* leverage the concept of QoM and propose an admission control scheme to dynamically "admit" applications to be deployed on physical sensor networks. The authors of [23] focus on the problem of scheduling applications to shared sensor nodes with the ultimate goal of maximizing the sensor network lifetime. Along the same lines, Zeng *et al.* propose in [24] an optimization framework to prolong network lifetime by properly scheduling the tasks in a shared/virtual sensor network.

The problem of allocating physical resources to multiple application is also often also cast as an auction. In [25], the authors propose a reverse combinatorial auction, in which the sensor nodes act as bidders and bid cost values (according to their available resources) for accomplishing the subset of the applications' tasks. Optimal bidding strategies are then studied to make the auction effective and truthful.

This work extends our previous work in [26], where the preliminary optimization framework to allocate resources in virtual sensor networks is introduced; building on the aforementioned work, we further provide here a complexity analysis of the proposed optimization problem which is proven to be NP-hard, and we propose a heuristic iterative algorithm to obtain sub-optimal solutions of this resource allocation problem in reduced computation time. Finally, we provide here a comprehensive performance evaluation of the proposed approach: we analyze the impact of varying the main model parameters (number of scalar and multimedia nodes; number of sinks; lifetime; type of routing) in the system performance and we also evaluate the performance of the proposed heuristic algorithm.

3. System model and optimization framework

Let $S = \{s_1, s_2, \ldots, s_l\}$ be a set of sensor nodes, $A = \{a_1, a_2, \ldots, a_m\}$ a set of applications which are to be deployed in the reference area, and $T = \{t_1, \ldots, t_n\}$ a set of test points in the reference network scenario. 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 t_k.

Each application j requires to cover a given set of test points $T_j \subseteq T$. Formally, the application j has to be deployed on a subset of S such that all the test points in T_j are covered. A test point is covered by a sensor node i if it is within its sensing range, R_i^s. It is convenient to introduce as well the set S_{jk} defined as the set of sensor nodes which physically cover the test point k, with $k \in T_j$. In other words, if the application j is deployed on any of the sensors in set S_{jk}, then the target test point k is covered for this application. A necessary condition for an application j to be successfully deployed is that all the test points in its target set T_j must be covered.

Each application j in A is further characterized by a requirement vector $r_j = \{c_j, m_j, l_j\}$ which specifies the required 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 10 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 capabilities, processing power and energy store.

A protocol interference model with power control [27] 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 $g_{ih} = g_0 \cdot d_{ih}^{-\gamma}$, being d_{ih} the distance from i to h, γ the path loss index and g_0 a constant dependent on antenna parameters. A transmission is successful if the received power exceeds 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 must be divided between them. If p_i is the transmission power assigned to node i, a transmission towards h is successful if $p_i \cdot d_{ih} > \alpha$. Thus, the transmission range for node i with transmission power p_i can be obtained as $R^T_i(p_i) = (p_i \cdot g_0/\alpha)^{-\gamma}$. Similarly, the interference resulting from node i with power p_i is non-negligible only if it exceeds a certain threshold β. Then, the interference range is $R^I_i(p_i) = (p_i \cdot g_0/\beta)^{-\gamma}$.

Qualitatively, the application assignment problem for virtual sensor networks can be defined as follows: to maximize the weighted number of deployed applications subject to coverage constraints (the set of test points of each application must be covered) and application requirements (each application should be assigned enough bandwidth, and processing and storage resources 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 application has to be delivered remotely.

Further, let us assume that a preference vector across all the m applications is de-

fined $Q = [q_1, q_2, \ldots, q_m]^T$ where q_j represents the revenue for the network provider for having application j successfully deployed in the network. Let z_j be a binary variable indicating if application j is successfully deployed in the network. Let y_{ijk} be a binary variable indicating if test point k of application j is deployed at sensor node i. Let x_i be a binary variable indicating if sensor node i is active in the network. Let h_{jk} be a binary variable which indicates if test point k belonging to set T_j is actually covered by a sensor node which runs application j.

The objective function aims at maximizing the overall revenue out of the application deployment process while minimizing the cost related to activating sensor nodes:

$$\max \left(\sum_{j \in A} q_j z_j - \sum_{i \in S} \delta_i x_i \right) \tag{1}$$

where δ_i is the cost incurred in activating sensor node i.

3.1. Constraints on coverage and on resources of the sensors

Constraints (2)-(5) require that all the applications which are actually deployed do fulfill the coverage constraints, that is, they cover all the required test points. Specifically, Eq. (2) indicates if test point k of an application j is effectively covered. If so, it ensures that it is covered by only one sensor node i. Eq. (3) ensures that if a sensor i does not cover a test point k of an application j, then it will not sense that test point. Depending on the application, it can be possible that the same sensor node can cover 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 cover, Eq. (4) guarantees that this threshold is not exceeded. Eq. (5) indicates that if an application j is successfully deployed, i.e., if $z_j = 1$, then all the test points of application j must be covered. In addition, it guarantees that if the application cannot be deployed, none of its test points are covered so that no resources are wasted.

$$\sum_{i \in S_{jk}} y_{ijk} = h_{jk} \qquad \forall j \in A, \forall k \in T_j \tag{2}$$

$$y_{ijk} = 0 \qquad \forall i \notin S_{jk}, \forall j \in A, \forall k \in T_j \tag{3}$$

$$\sum_{k \in T_j} y_{ijk} \leq N_{ij} \qquad \forall i \in S, \forall j \in A \tag{4}$$

$$z_j = \frac{\sum_{k \in T_j} h_{jk}}{|T_j|} \qquad \forall j \in A \tag{5}$$

Constraints (6) and (7) are budget-type constraints for the available storage and processing load of the sensor nodes.

$$\sum_{j \in A} \sum_{k \in T_j} m_j y_{ijk} \leq M_i \qquad \forall i \in S \tag{6}$$

$$\sum_{j \in A} \sum_{k \in T_j} l_j y_{ijk} \leq L_i \qquad \forall i \in S \tag{7}$$

3.2. Routing constraints

The deployed applications will require most likely 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 are received by the sink nodes. This fact can be conveniently expressed using the following constraints:

$$\sum_{\substack{h \in S \\ i \neq h}} f_{hi} - \sum_{\substack{h \in S \\ h \neq i}} f_{ih} + \sum_{j \in A} \sum_{k \in T_j} c_j y_{ijk} = 0 \quad \forall i \in S \setminus SINK \tag{8}$$

$$\sum_{j \in A} |T_j| c_j z_j = \sum_{h \in SINK} \left(\sum_{\substack{i \in S \\ i \neq h}} f_{ih} + \sum_{j \in A} \sum_{k \in T_j} c_j y_{hjk} \right) \tag{9}$$

where $SINK$ is the set of sink nodes (a subset of S) and f_{ih} is a variable representing the flow of data in bps transmitted from node i to node h. Constraints (8) enforce flow conservation at sensor nodes. Constraint (9) imposes that all the generated data are

delivered to the set of sinks. The last term of this expression allows the sinks to be running applications as well.

The following constraint set enforces that if a sensor node is either running an application or receiving data, then it must be active in the network:

$$\sum_{\substack{h \in S \\ h \neq i}} f_{hi} + \sum_{j \in A} \sum_{k \in T_j} c_j y_{ijk} \leq K x_i \qquad \forall i \in S \tag{10}$$

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

$$f_{ih} \leq K l_{ih} \qquad \forall i, h \in S \tag{11}$$

where l_{ih} is a constant that indicates if there is a viable link between i and h, i.e., if the distance between both nodes is less than the maximum transmission range R^T_{max}, then $l_{ih} = 1$ and $l_{ih} = 0$ otherwise. Therefore, these constraints ensure that data must be transmitted exclusively along neighboring nodes.

These expressions allow flow splitting and multipath routing. In the sequel, we will denote this kind of routing as *multipath routing*.

However, 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 [28]. Therefore, we introduce the following restrictions to ensure that all the traffic flowing out of a sensor has only one possible route to a sink:

$$g_{ih} \leq l_{ih} \qquad \forall i, h \in S \tag{12}$$

$$\sum_{h \epsilon S} g_{ih} \leq 1 \qquad \forall i \in S \tag{13}$$

$$f_{ih} \leq K g_{ih} \qquad \forall i, h \in S \tag{14}$$

where g_{ih} is a binary variable which indicates if data are transmitted from node i to node h. Constraints (13) and (14) impose that only one link from sensor node i to any of its neighbors transports all the data that i must forward. In the sequel, we will denote this kind of routing as *singlepath routing*.

Including the route creation in the optimization framework may not be always feasible. In addition, since all the traffic in WSN is forwarded to a single or a limited number of sinks, the main bottleneck will be mainly the last hop to these sinks. For these reasons, we also consider the possibility of excluding the routing from the optimization process and assuming a predefined set of routes from each node to a sink. To that purpose, we build DODAGs 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). This implies that equations (12)-(14) must be excluded from the model and that for each node i, the constant l_{ih} is 1 just for a single h (the father node in the routing tree towards the sink) and therefore $f_{ih'} = 0$ for all $h' \neq h$. In the sequel, we will denote this kind of routing as *static routing*.

3.3. Bandwidth constraints

The available bandwidth in the network is limited and must be shared among sensor nodes. We assume that a fair medium access control schemes orchestrate 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 j 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, if we define IF_{ih} as the fraction of time that other links interfere the link (i, h), we have that:

$$IF_{ih} = \sum_{\substack{g \in S \\ g \neq h}} \frac{f_{ig}}{C_{ig}} + \sum_{g \in S} \frac{f_{gi}}{C_{gi}} + \sum_{\substack{g \in S \\ g \neq i}} \frac{f_{hg}}{C_{hg}} + \sum_{\substack{g \in S \\ g \neq i}} \frac{f_{gh}}{C_{gh}} + \sum_{\substack{g,t \in S \\ d_{it} < R_i^I(p_i)}} \frac{f_{gt}}{C_{gt}} + \sum_{\substack{g,t \in S \\ d_{gh} < R_g^I(p_g)}} \frac{f_{gt}}{C_{gt}} \tag{15}$$

Then, for each link (i, h) in the network it must be ensured that the fraction of time used by the link plus all its interferences is less or equal to 1:

$$\frac{f_{ih}}{C_{ih}} + IF_{ih} \leq 1 \qquad \forall i, h \in S \tag{16}$$

Constraints (16) are the equivalent budget-type constraints for the available wireless capacity to the storage and processing load constraints given in (6) and (7).

3.4. Energy constraints

Finally, energy constraints are included to ensure that the application deployment pattern guarantees a minimum lifetime L for the virtual sensor network. Typically, energy consumption due to wireless communication (i.e., transmitting and receiving) has been considered the dominant factor in power consumption for WSNs [29]. 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 can not be neglected [30].

Regarding wireless transceiver, the power dissipation at the radio transmitter P_i^t or at the radio receiver P_i^r of each node i can be modeled as [31]:

$$P_i^t = \sum_{h \in S, h \neq i} \left(\beta_1 + \beta_2 d_{ih}^{\gamma}\right) f_{ih} \qquad \forall i \in S \tag{17}$$

$$P_i^r = \rho \sum_{h \in S, h \neq i} f_{hi} \qquad \forall i \in S \tag{18}$$

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 lifetime constraints set in (19), this power dissipation is left as a function f of the processing loads l_j of the applications. In Section 5, further details about the specific evaluated multimedia applications are given.

$$P_i^t + P_i^r + f\left(\sum_{j\in A}\sum_{k\in T_j} y_{ijk} l_j\right) \leq \frac{E_i}{L} \qquad \forall i \in S \tag{19}$$

3.5. Complexity analysis

Theorem 1. *The application deployment problem is NP-complete.*

PROOF. The NP-completeness can be proved by restriction, that is by showing that an NP-complete problem reduces to our application deployment problem. The reference problem we use in the proof is the multiple Knapsack Problem which is known to be NP-complete. Let's consider the particular instance of the application deployment problem characterized by the following setting: δ_i =0,$\forall i \in S$ (negligible sensor activation cost), and $S_{jk} = S$, $\forall j \in A$, $T_j = \{1\}$, $\forall j \in A$ (all the applications need to cover one single test point which is reachable from all the sensor nodes). Let's further assume that routing is not needed, that is, formally $SINK = S$, and that sensor nodes do not have processing capability limitation, $L_i = \infty$, $\forall i \in S$. In such setting, since all the applications need to cover one single test point which is "reachable" from all the sensor nodes (from $T_j = \{1\}$ and $S_{jk} = S$), the index k can be safely dropped from variables y_{ikj} and h_{jk}. The application deployment problem can be re-written as follows:

$$\max \sum_{j\in A} q_j z_j$$

s.t.

$$\sum_{i \in S} y_{ij} = h_j \qquad \forall j \in A$$

$$y_{ij} \leq 1 \qquad \forall i \in S, \forall j \in A$$

$$z_j = h_j \qquad \forall j \in A$$

$$\sum_{j \in A} m_j y_{ij} \leq M_i \qquad \forall i \in S$$

which can be further simplified as:

$$\max \sum_{j \in A} \sum_{i \in S} y_{ij} q_j$$

$$\sum_{i \in S} y_{ij} \leq 1 \qquad \forall j \in A$$

$$\sum_{j \in A} m_j y_{ij} \leq M_i \qquad \forall i \in S$$

This last formulation is a multi-knapsack problem which is known to be NP-complete. By restriction, also the application deployment problem must be NP-complete [32].□

4. Heuristic algorithm

In order to obtain sub-optimal solutions of the resource allocation problem in reduced computation time, we introduce here a heuristic iterative algorithm which is based on LP relaxation the original problem. In short, we drop the integrality constraints on variables z_j, x_i, y_{ijk} and h_{jk} and focus on the *static routing* strategy defined in section 3.2, and iteratively solve simplified problems further checking feasibility of the obtained solution in the original problem at each iteration step. Fig. 1 shows the diagram block of the algorithm, whose steps are explained next.

First, the relaxed LP problem is solved (*problem1*). If all the variables in the obtained solution are integers, then the solution is also valid for the original MILP problem and the algorithm terminates. If not, we see if there is any application that is not active at all ($z_j = 0$) in the solution of the relaxed problem and we add new restrictions to *problem1* forcing this application to be inactive (if there are more than one inactive

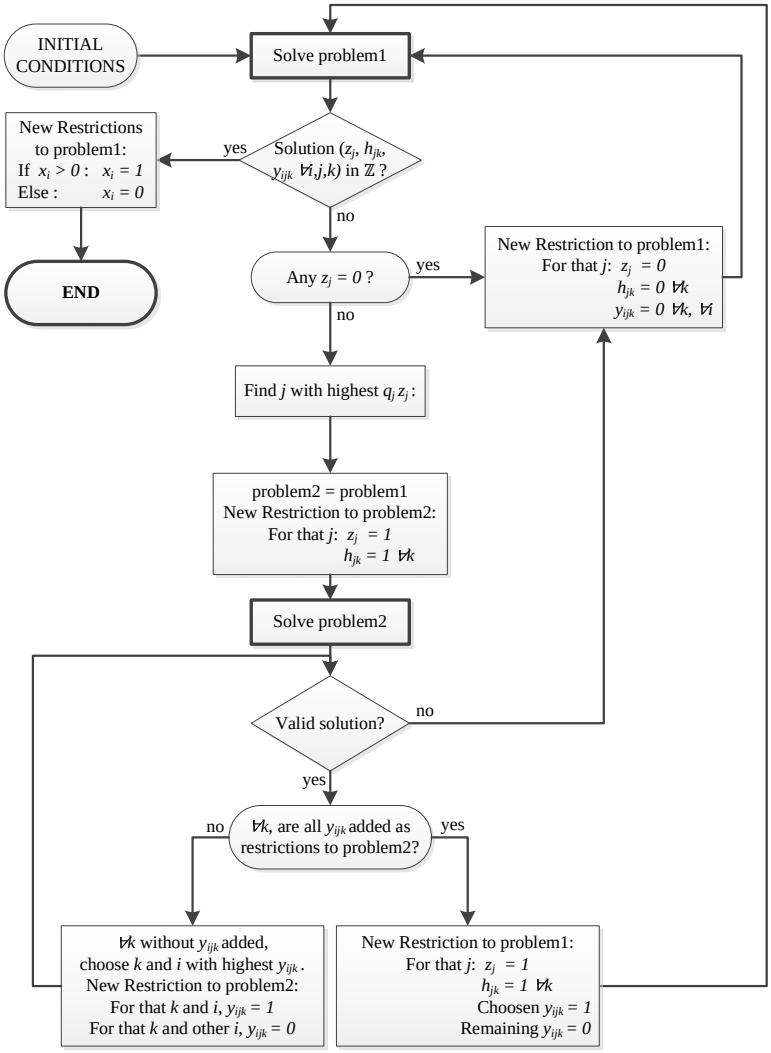


Figure 1: Diagram of the heuristic algorithm

applications, we choose one at random). We follow this procedure of solving *problem1* and adding these restrictions until $z_j > 0$ for all the remaining applications.

Then, we choose the application j with maximum value of $q_j z_j$ and we try to activate it when solving the optimization problem. To that purpose, we define a new temporary problem (*problem2*) equal to *problem1*, but with additional restrictions to force application j to be active. If *problem2* does not have a feasible solution, we dismiss application j and we add permanent restrictions to *problem1* forcing this application to be inactive. On the other hand, if *problem2* has a valid solution, then the values of y_{ijk} still have to be obtained: The feasibility of the solution guarantees that application j is active ($z_j = 1$) and therefore that all its test points are covered ($h_{jk} = 1$). However, it does not guarantee that each test point k is covered by a single node i ($y_{ijk} = 1$) as the variables y_{ijk} are still relaxed.

To solve this problem we proceed as follows: for each test point k we choose the node i with the highest value of y_{ijk} and we add to *problem2* the restriction $y_{ijk} = 1$ to force node i to cover completely test point k. If *problem2* with these new restrictions is feasible, then we add all the restrictions to *problem1* to guarantee that application j is active. On the contrary, if *problem2* does not have a valid solution, then application j is dismissed. Once the solution of *problem1* fulfills the integrality of variables z_j, h_{jk} and y_{ijk}, then we set $x_i = 1$ if $x_i > 0$ and 0 otherwise.

5. Performance evaluation

The proposed model leads to a mixed integer linear programming (MILP) problem, which has been solved using the CPLEX software [33]. To evaluate the model we have considered a scenario with two different types of sensor nodes and four different applications (two scalar and two multimedia). Next we define the main features of both sensor nodes and applications and the simulation parameters. Then, results are presented.

5.1. Sensor nodes

We have considered TelosB sensor motes [34] and BeagleBone nodes [35]. The energy budget for both nodes is 32400 J assuming that a node runs at 3 V with 3

Ah of battery supply (2 AA batteries). Each TelosB mote has integrated a temperature and a light sensor, an IEEE 802.15.4 radio with integrated antenna and a 8 MHz TI MSP430 microcontroller which can operate at 8 MIPS and with 10 KB RAM, although only 7 KB are available for applications [17]. Therefore, their resource vector is $o_i = \{C_i, M_i, L_i, E_i\} = \{250$ kbps, 7 KB, 8 MIPS, 32400 J$\}$. These motes are suitable for supporting scalar applications. BeagleBone is a low-power platform based on a Linux Computer that includes 720 MHz super-scalar ARM Cortex-A8 processor (up to 720 MIPS) and 256 MB of RAM. BeagleBone nodes should include a Shimmer Span IEEE 802.15.4-compliant transceiver, a low-power USB camera for multimedia applications and also scalar sensors. Their resource vector is $o_i = \{250$ kbps, 256 MB, 720 MIPS, 32400 J$\}$.

5.2. Applications

For the scalar applications, we have considered temperature and light monitoring. Temperature monitoring applications require 4462 bytes of RAM, while light monitoring applications require 1006 bytes [17]. This kind of applications has a low sample rate (0.017-1 Hz according to [36]). We have chosen a sample rate of 0.5 Hz for temperature monitoring and 1 Hz for light monitoring. Considering a packet size of 127 bytes per sample, the temperature application has a source rate of 0.5 kbps, whereas for the light application is 1 kbps. Given these parameters, we can assume that the processing load l_j is negligible in the application requirement vector, (i.e. memory usage, transmission rate or the energy consumed by the transmission will be more limiting factors than the processing load or the energy consumed by the processing). Thus, the requirement vector for temperature monitoring is $r_j = \{c_j, m_j, l_j\} = \{$ 0.5 kbps, 4462 B, -$\}$ whereas for light monitoring is $r_j = \{1$ kbps, 1006 B, -$\}$

For multimedia applications we focus on visual sensor networks, i.e. WSNs designed to perform visual analysis (e.g. object recognition) [30]. We consider two paradigms, the classic Compress-Then-Analyze (CTA) and the opposite approach, Analyze-Then-Compress (ATC) [30], [37]. CTA applications are those where images acquired from camera nodes are compressed and sent to a central controller for further analysis. On the other hand, ATC applications are those where camera nodes perform vi-

sual feature extraction and transmit a compressed version of these features to a central controller. In [37] a detailed characterization of transmission rates and energy consumption for both approaches is provided. In order to evaluate the model with some realistic parameters, we have chosen some specific values for both cases based on the aforementioned study. It is assumed that different techniques for the extraction of local visual features are used: CTA will use the SIFT (Scale Invariant Feature Transform) algorithm while ATC will use BRISK (Binary Robust Invariant Scalable Keypoints) algorithm. Assuming a desired Mean of Average Precision (MAP) of 0.6, the use of Zurich Building Database (ZuBuD) [38] and an application frame rate of $\lambda = 1$ query per second for both CTA and ATC paradigms, the needed capacity will be 20 kbps for CTA-SIFT and 12 kbps for ATC-BRISK [37].

For this kind of applications, the energy consumed to process the data is not negligible. In [37] a characterization of this energy on a BeagleBone-based visual sensor node is provided. The processing energy for the CTA paradigm can be computed as:

$$E_{cpu}^{CTA}(\rho) = P_{cpu} \cdot t_{cpu}^{CTA}(\rho) \tag{20}$$

where P_{cpu} is the power dissipated by the processor of the visual sensor node and has a value of 2.1 W for BeagleBone sensor nodes; and $t_{cpu}^{CTA}(\rho)$ is the time required to process an image, which depends on ρ, the amount of sent information per query (20 kbs in our scenario). According to the results in [37], the processing energy for an image for the CTA application in our scenario is 0.05 J. Therefore, assuming a frame rate of $\lambda = 1$ query per second, the power dissipation (function f in eq. (19)) is 0.05 W. In addition, we can estimate the required processing load l_j for a BeagleBone as the fraction of time used by the application ($t_{cpu}^{CTA} \cdot \lambda$) multiplied by the processing power of the sensor node, L_i. In this case, $24.5 \cdot 1 \cdot 720 = 17.64$ MIPS.

Similarly, the processing energy for the ATC paradigm can be computed as:

$$E_{cpu}^{ATC}(\rho) = P_{cpu} \cdot [\tau_{off} + M(\rho) \cdot (\tau_{det} + \tau_{desc})] \tag{21}$$

where τ_{off} is the time spent for initializing the detector and has a value of $1.6 \cdot 10^{-4}$ ms/pixel. With an image size of 640×480 pixels, τ_{off} is 49.152 ms. τ_{det} and τ_{desc} are the time spent for detecting and describing one BRISK feature of the image and their

values are 0.31 ms and 0.16 ms respectively. $M(\rho)$ is the optimal number of features that depends on the rate ρ. For $\rho = 12$ kb/query, the minimum value of M to provide a MAP of 0.6 is $M = 100$ features. Thus the processing energy for an image for the ATC application in our scenario is 0.2 J, and the power dissipation is 0.2 W. The processing load in this case is 69.23 MIPS.

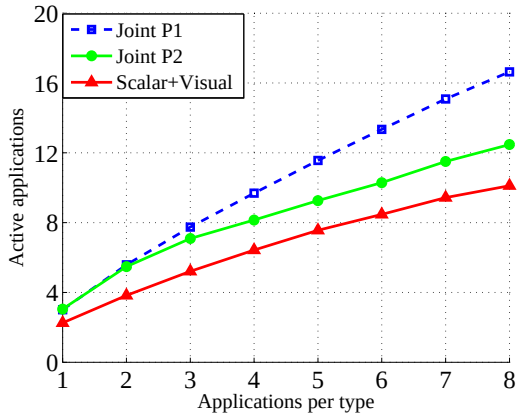
Regarding memory requirements, specific values have not been obtained for these applications. However, given the great difference in the amount of available memory in TelosB (10 KB) and BeagleBone (256 MB), we are assuming that due to memory constraints, multimedia applications could not be implemented in TelosB nodes and memory will not be a limiting factor in BeagleBone nodes, since processing or transmission rate will limit long before these applications rather than memory. Summing up, the requirements vector for CTA and ATC applications are respectively $r_j = \{20 \text{ kbps}, 10 \text{ KB} < m_j << 256 \text{ MB}, 17.64 \text{ MIPS}\}$ $r_j = \{12 \text{ kbps}, 10\text{KB} < m_j << 256 \text{ MB}, 69.23 \text{ MIPS}\}$.

5.3. Simulation Environment

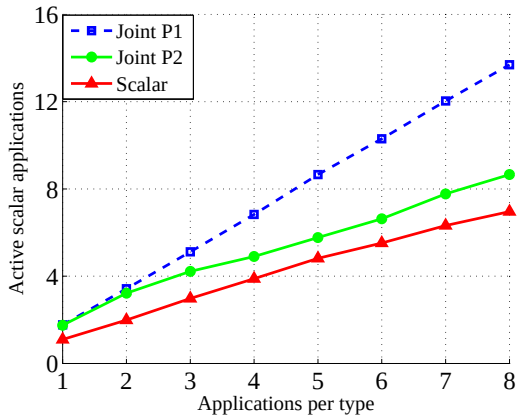
Sensor nodes are deployed in a 200×200 m scenario. We consider a default sensing range of $R_i^s = 30$ m for all of the sensors [39]. A two-ray ground path loss model with $\gamma = 4$ and $g_0 = 8.1 \cdot 10^{-3}$ [40] is considered. P_{max} is set to 0 dBm and the receiver sensitivity α is -92 dBm [41], which implies a maximum transmission range R_{max}^T of 59 m. Analogously the interference sensitivity is -104 dBm, which implies a maximum interference range R_{max}^I of 118 m.

5.4. Benefits of virtualization

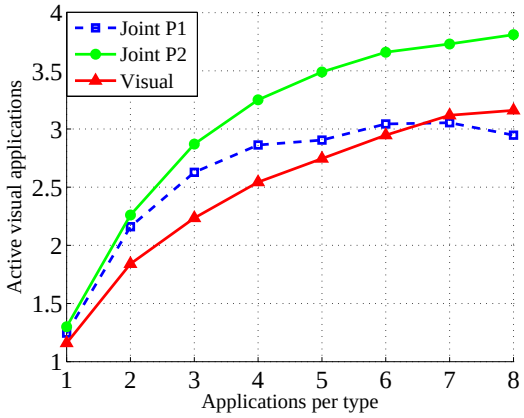
As a reference example to evaluate the validity of the model and the benefits of virtualization, we have considered a scenario with two different and overlapped WSNs: a *scalar* network, formed by 36 TelosB nodes and oriented to scalar applications (temperature and light monitoring), and a *multimedia* network, formed by 36 BeagleBone nodes and oriented to visual applications (CTA and ATC). The number of test points is 5 for the scalar applications and 3 for the visual ones. We assume that each sensor is able to cover $N_{i,j} = 1$ test points of the same application and that each network has a



(a)

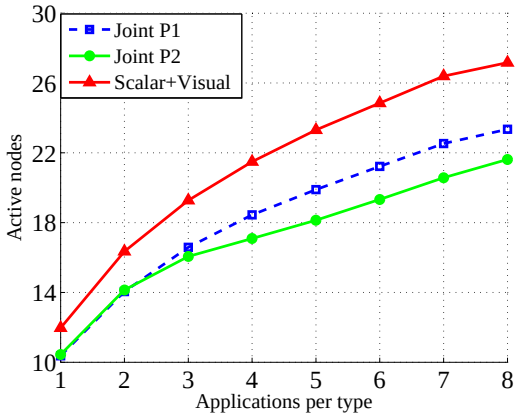


(b)

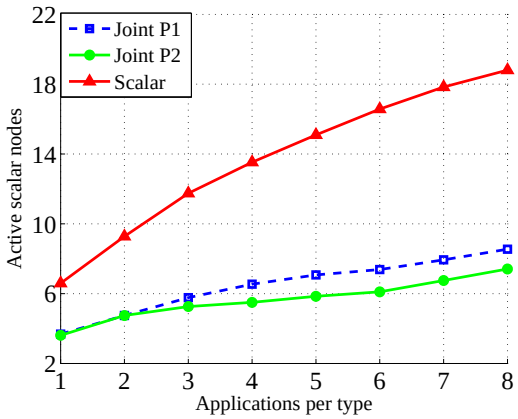


(c)

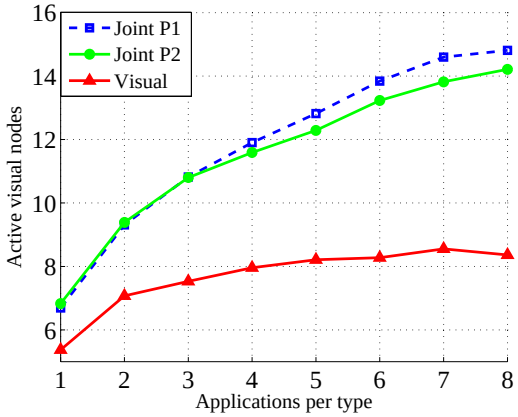
Figure 2: Number of active applications vs. offered applications per type. a) Total b) Scalar c) Visual



(a)



(b)



(c)

Figure 3: Number of active nodes vs. offered applications per type. a) Total b) Scalar c) Visual

sink node (one of the 36 nodes). The minimum lifetime for the virtual sensor network is $L = 1$ day.

Figs. 2 and 3 show the performance of both networks in terms of the number of active applications and the number of active nodes when the WSNs work isolated and also when the 72 nodes cooperate as a single network that gives support to all the applications. For each point in the curves, the same number of applications of each type (temperature, light, CTA and ATC) is generated, which is the value shown in the *x*-axis. For example, a 2 value in the *x*-axis represents a scenario where 2 temperature, 2 light, 2 CTA and 2 ATC applications try to be deployed. "Scalar" refers to the single *scalar* network, "Visual" to the single *visual* network and "Joint" to the cooperative whole network. In this latter case, two different preference vectors Q are included: P1 (all the applications have the same preference), and P2 (the preference vector depends on the application parameters). Since the main limiting factor for visual applications with regard to scalar ones is the bandwidth, preference values are approximately adjusted according to the demanded bandwidth: preference for scalar applications is 1 (they need 2.5 or 5 kbps $= 0.5$ or 1 kbps per test point $\times$ 5 test points); preference for ATC applications is 8 (they require 36 kbps $=$ 12 kbps $\times$ 3 test points); and preference for CTA applications is 12 (they 60 kbps $=$ 20 kbps $\times$ 3 test points).

Fig. 2(a) shows that the total number of active applications increases in the joint scenario, when compared to the sum of the independent networks (specially for the preference vector P1). As can be seen in Figs. 2(b) and 2(c), the main increase is due to the scalar applications. In fact, when visual applications are not prioritized (P1), the increase in the number of scalar applications eventually leads to an starvation of the visual applications (Fig. 2(c)). However, with the preference vector P2 both scalar and visual applications experience a more balanced improvement.

The reasons for this improvement are different in each case: according to our measurements, the probability of a test point not being covered with 36 nodes is about 0.15. Since an active application requires all its test points to be effectively covered, the probability of a scalar application with 5 test points not being fully covered (only as topology concerns) is about 0.55. As multimedia nodes can support scalar applications as well, in the joint scenario the 72 nodes can be used to sense scalar applications, re-

ducing these probabilities to 0.015 and 0.07 respectively. On the contrary, since scalar nodes do not support multimedia applications, this kind of gain cannot be obtained for visual applications. Nevertheless, as one of the main limitations in multimedia networks is bandwidth (specifically the bottleneck in the transmission to the sink node), the possibility of using two sink nodes in the joint scenario leads also to an improvement for this case. Since the resources consumed by scalar applications is much lower than by visual ones, prioritizing the latter (P2) in the objective function is useful to balance the amount of resources used by each of them.

Finally, Fig. 3(a) shows that the total number of active nodes when both networks works jointly is lower than when they work isolated. Additionally, as the number of active applications is also higher, the active nodes per active application are quite lower when the networks work jointly. Regarding the type of active nodes, Fig. 3(c) shows that the amount of active multimedia nodes increases for the joint scenario. The reasons for this effect are two: (1) multimedia nodes can support scalar applications in the joint scenario and (2) we have assumed that the cost for activating sensor nodes, δ_i, is the same for both kind of nodes. We have set $\delta_i = 0.01$ to ensure that in any case the cost of activating the 72 nodes is higher than the revenue of activating an application.

Once the main benefits of virtualization in the network performance have been shown, next sections present the impact of varying the most relevant model parameters. We take as basis the scenario with 36 scalar nodes and 36 visual nodes, preference vector P2, and 1 sink for each type of application (reference scenario).

5.5. Number of scalar and multimedia nodes

First, we vary the number of scalar and multimedia motes, maintaining the total number of nodes to 72. Fig. 4 and Fig. 5 show the number of active applications and the number of active nodes respectively. As can be seen, the probability of a visual application being fully covered increases with the number of multimedia motes since visual applications can only be deployed on multimedia motes. Therefore, the number of active visual applications grows from 0 (when there are not multimedia motes) until its maximum value (when the 72 motes are multimedia) (Fig. 4(b)). However, the growth rate of active visual applications decreases as the number of multimedia motes

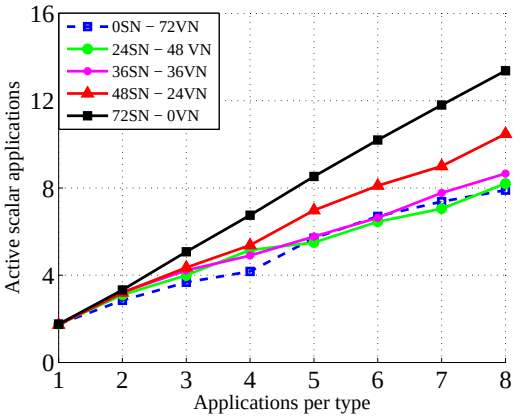
grows due to the bandwidth bottleneck at the sinks. The number of active scalar applications (fig. 4(a)) is not impacted by the number of scalar motes, since they can be deployed on any mote and are less limited by bandwidth constraints.

5.6. Number of sinks

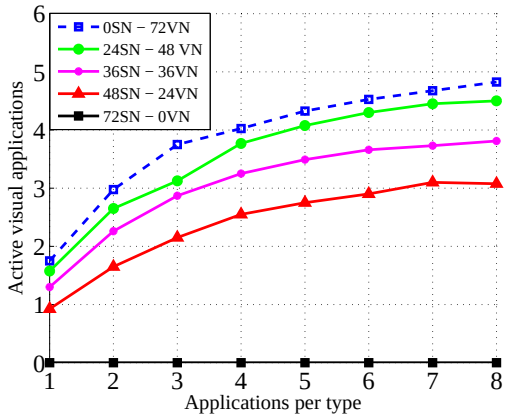
Since the number of active multimedia applications depends on the bottleneck access to the sink, we analyze next the impact of the number of sinks in the reference scenario. Figs. 6 and 7, that show the network performance as a function of the number of sinks, confirm that assumption. As can be seen, there is a great improvement in the number of active visual applications when the number of sinks is increased. Additionally, there is also an increase in the number of active scalar applications, which is not so relevant due to their lower bandwidth requirements. Consequently, the number of active scalar nodes (Fig. 7(a)) does not depend on the number of sinks, whereas the number of active multimedia nodes (Fig. 7(b)) increases with the number of sinks since the number of active visual applications grows.

5.7. Lifetime

In the reference scenario, the minimum lifetime of the virtual sensor network is set to $L = 1$ day. Next, we vary this parameter from $L = 1$ hour to $L = 8$ days. The results are shown in Figs. 8 and 9. Fig. 8(b) shows that with $L = 8$ days, visual applications cannot be activated since multimedia motes do not have enough energy to support any visual application. Logically, multimedia motes (fig. 9(b)) are still activated because they can be used by the scalar applications. In addition, there is a remarkable decrease in the number of active visual applications from $L = 1.75$ to $L = 2$ days. The reason is that ATC visual applications, which demand more energy, cannot be activated with $L = 2$ days and the only active visual applications are the CTA ones. It is also interesting to observe that from $L = 1$ to $L = 1.75$ days there is a slight decrease in the number of active visual applications whereas the number of active multimedia motes rises. This is due to the fact that nodes that were simultaneously sensing several test points when $L = 1$, do not have now enough energy when $L = 1.75$ and therefore, additional nodes must be activated.

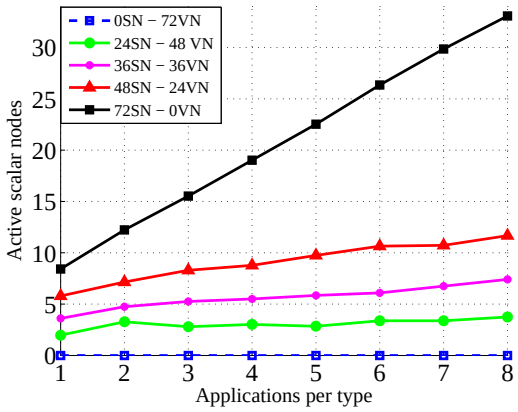


(a)

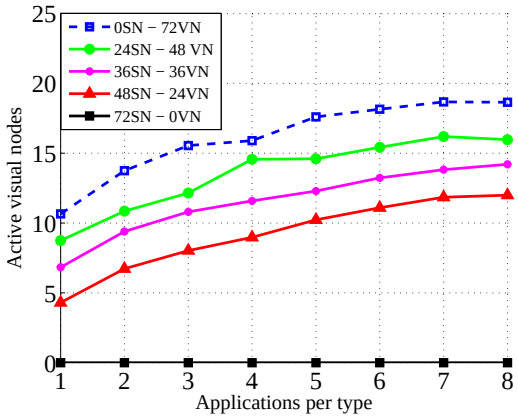


(b)

Figure 4: Number of active applications vs. offered applications per type varying the type of nodes. a) Scalar b) Visual

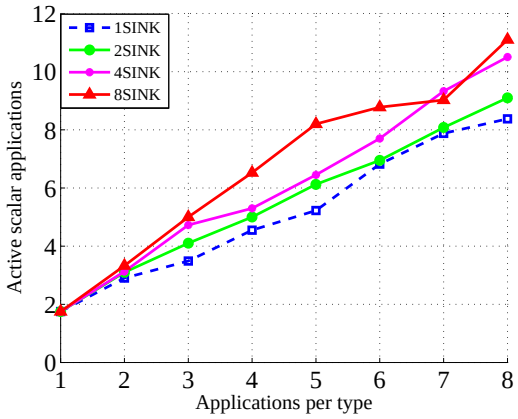


(a)

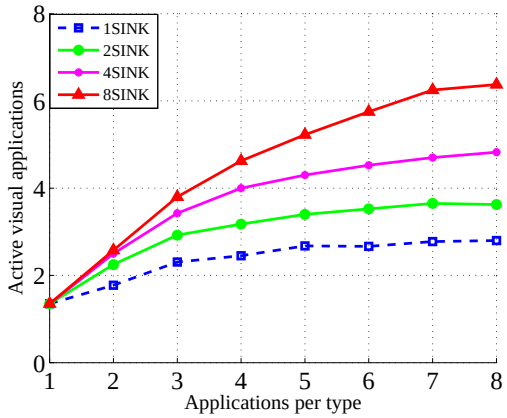


(b)

Figure 5: Number of active nodes vs. offered applications per type varying the type of nodes. a) Scalar b) Visual

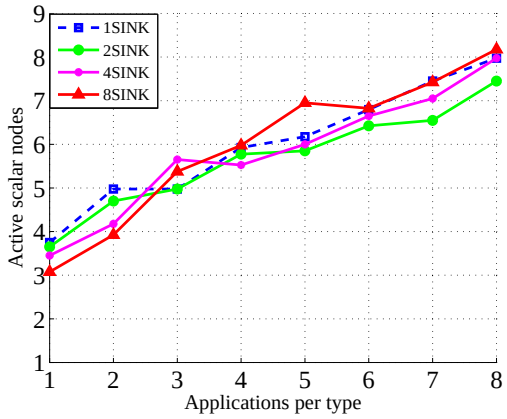


(a)

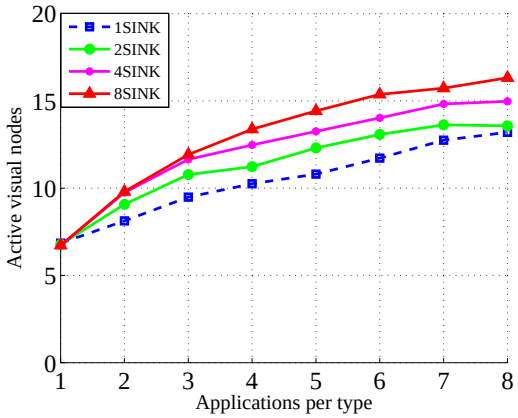


(b)

Figure 6: Number of active applications vs. offered applications per type varying the number of sinks. a) Scalar b) Visual

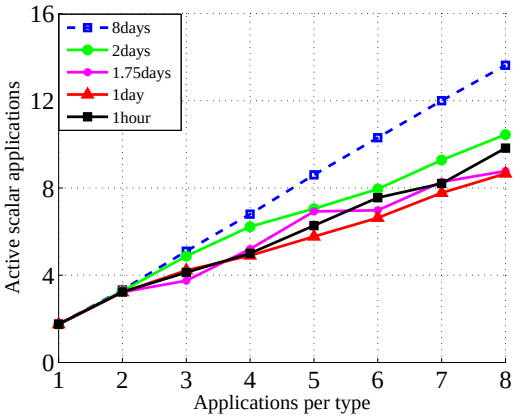


(a)

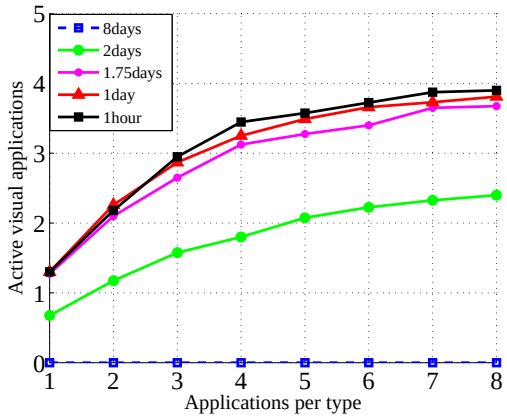


(b)

Figure 7: Number of active nodes vs. offered applications per type varying the number of sinks. a) Scalar b) Visual

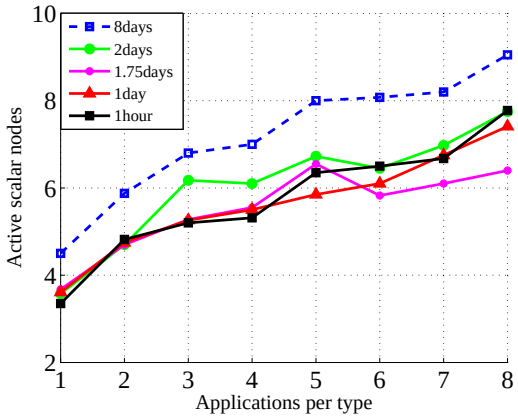


(a)

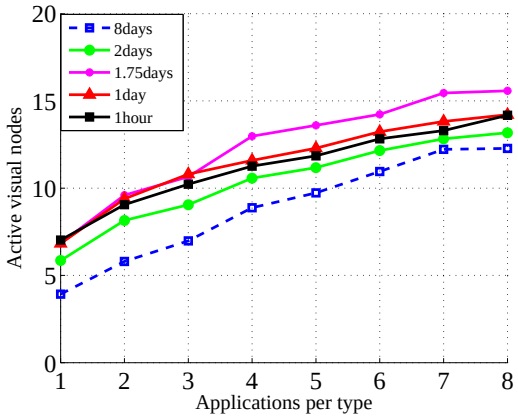


(b)

Figure 8: Number of active applications vs. offered applications per type varying the network lifetime. a) Scalar b) Visual

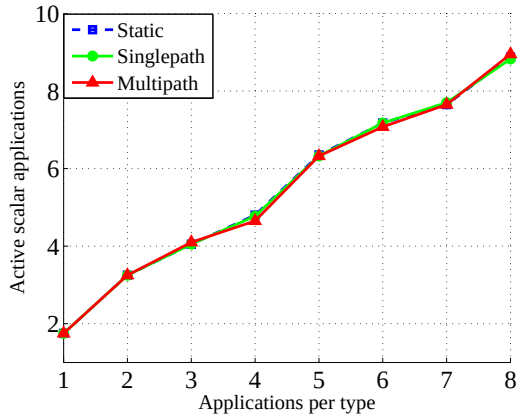


(a)

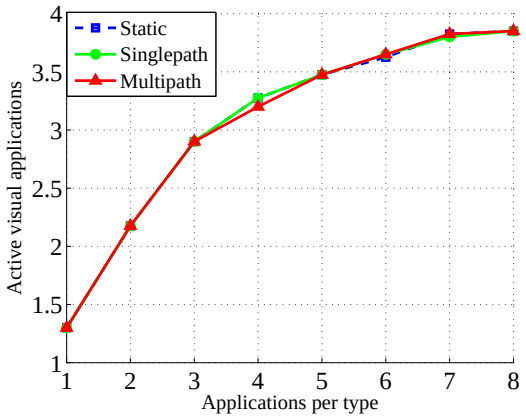


(b)

Figure 9: Number of active nodes vs. offered applications per type varying the network lifetime. a) Scalar b) Visual

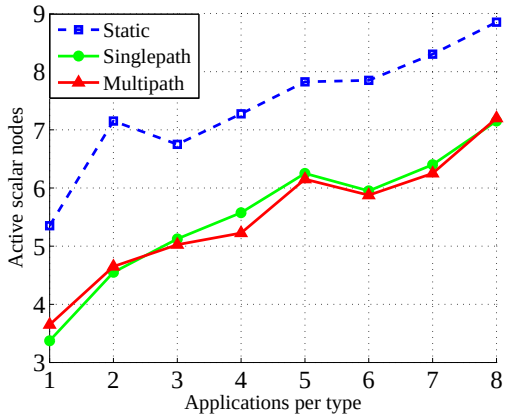


(a)

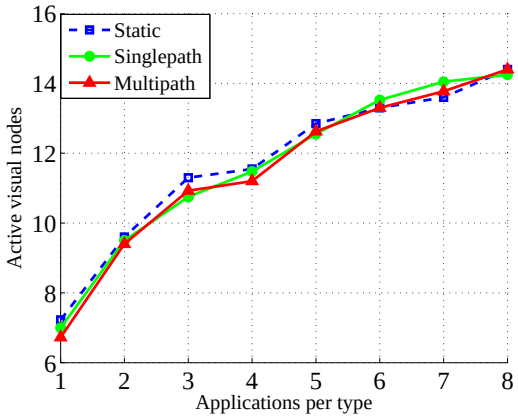


(b)

Figure 10: Number of active applications vs. offered applications per type varying the routing schemes. $P_{max} = 0dBm$. a) Scalar b) Visual

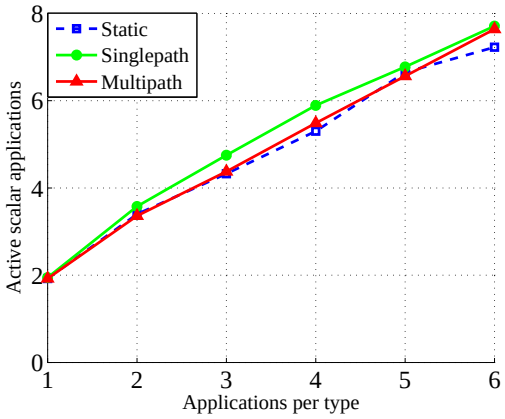


(a)

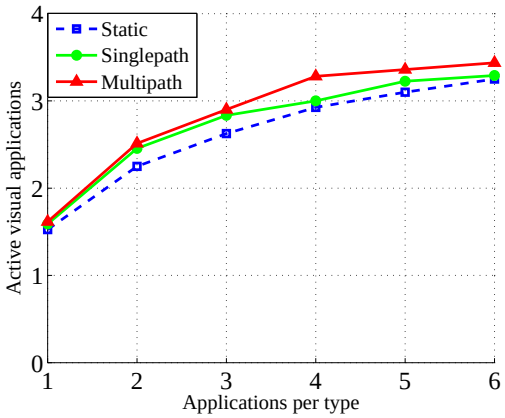


(b)

Figure 11: Number of active nodes vs. offered applications per type varying the routing schemes. $P_{max} = 0dBm$. a) Scalar b) Visual

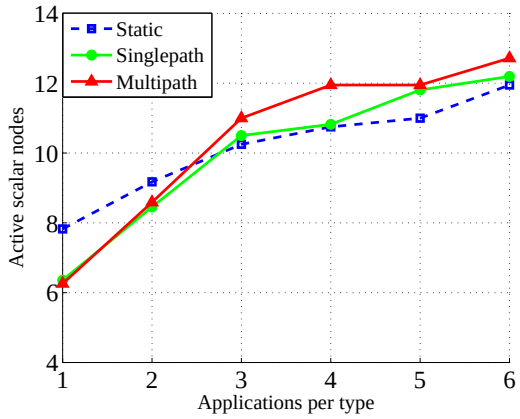


(a)

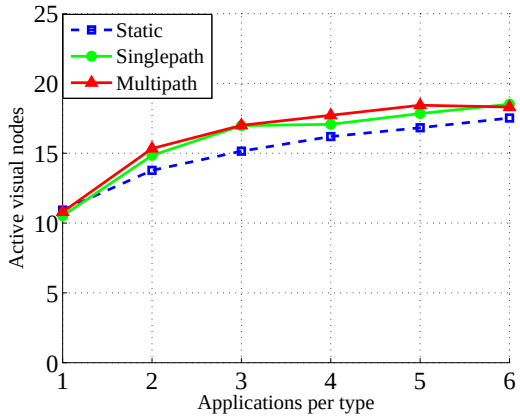


(b)

Figure 12: Number of active applications vs. offered applications per type varying the routing schemes. $P_{max} = -10dBm$. a) Scalar b) Visual



(a)



(b)

Figure 13: Number of active nodes vs. offered applications per type varying the routing schemes. $P_{max} = -10dBm$. a) Scalar b) Visual

5.8. Type of routing

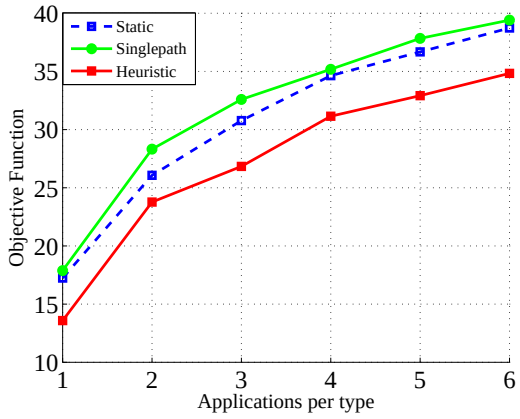
Finally, the three different types of routing described in section 3.2 are analyzed: *multipath*, *singlepath* and *static routing*. Figs. 10 and 11 show the number of active applications and the number of active nodes for the three different routing schemes in the reference scenario. As can be seen, the impact of the routing in the results seems very limited. Nevertheless, the reason behind this is that most of the routes only have one or two hops (with $P_{max} = 0$ dBm, the maximum transmission range is 59 m). To increase the number of hops of the routes, we decrease P_{max} to -10 dBm (maximum transmission range of 33 m). Figs. 12 and 13 show these new results. In this case, it can be observed that the *singlepath* routing achieves a performance very close to the upper bound provided by the ideal non-constrained *multipath* routing. In addition, the much simpler *static* routing is also close to the *singlepath*, which suggests its utilization in the proposed heuristic algorithm, whose performance is shown next. It can also be noted that the number of active nodes (Fig. 13) rises due to the increase in the number of hops.

5.9. Performance of heuristic algorithm

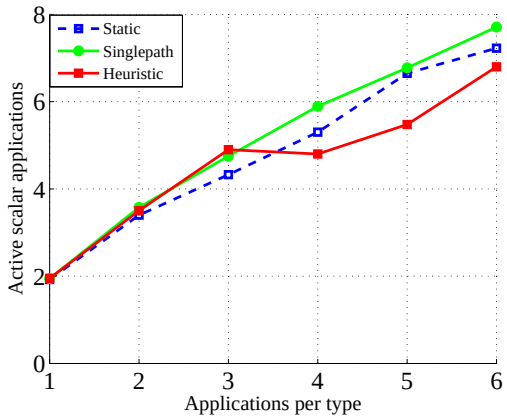
In this section, the heuristic algorithm is evaluated in the reference scenario with $P_{max} = -10$ dBm to see the impact of the use of the static routing in the heuristic. Fig. 14 compares the performance of the proposed heuristic algorithm with the optimum solutions achieved with the *singlepath* routing and with the *static* routing. As can be seen in Fig. 14(a), the degradation of the heuristic algorithm is about a 10% from the optimum value. A similar decrease is observed in the number of active scalar and visual applications (Figs. 14(b) and 14(c)). These results suggest the potential of this algorithm as a centralized resource allocation tool for virtual sensor networks.

6. Conclusion

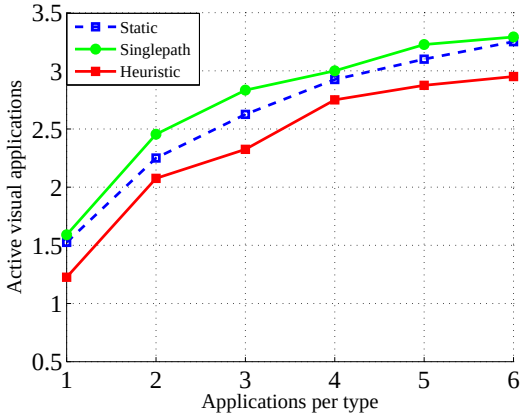
In this paper we have analyzed a virtual sensor network where different kinds of applications and sensor nodes coexist and cooperate. We have formulated mathematically the optimization problem of maximizing the overall revenue out of the application



(a)



(b)



(c)

Figure 14: Performance of heuristic algorithm vs the optimum scheme with *singlepath* and *static* routing. Number of active applications vs. offered applications per type. a) Objective function b) Active scalar applications c) Active visual applications

deployment process while minimizing the cost related to activating sensor nodes and we have analyzed its computational complexity. Constraints regarding sensor nodes capabilities (memory, computation, energy) and network limitations (topology, shared bandwidth) have been included. A heuristic algorithm has been proposed to reduce the computation time of the resource allocation pattern. Realistic parameters for both the sensor nodes and the supported applications have been considered in the evaluation of the model. Simulation results are further derived to assess the potential performance gains that the resource reuse achieved by virtualization can obtain: coverage enhancements, since there is a higher density of sensor nodes capable of supporting a given application, and capacity increase, due to the possibility of reusing several sink nodes to reduce congestion on bottleneck links.

Acknowledgment

This work has been supported by the Spanish Government through the grants TEC2011-23037 and TEC2014-52969-R from the Ministerio de Ciencia e Innovación (MICINN), Gobierno de Aragón (research group T98), the European Social Fund (ESF) and Centro Universitario de la Defensa through project CUD2013-05.

This work has also been partially supported by the Italian Ministry for Education, University and Research (MIUR) through the national cluster project SHELL, Smart Living technologies (grant number: CTN01 00128 111357).

References

[1] A. Jayasumana, H. Qi, T. Illangasekare, Virtual sensor networks - a resource efficient approach for concurrent applications, in: Proc. 4th International Conference on Information Technology (ITNG'07), Las Vegas, 2007, pp. 111–115.

[2] M. M. Islam, E.-N. Huh, Virtualization in wireless sensor network: Challenges and opportunities, Journal of Networks 7 (3) (2012) 412 – 418.

[3] T. Luo, H.-P. Tan, T. Quek, Sensor openflow: Enabling software-defined wireless sensor networks, Communications Letters, IEEE 16 (11) (2012) 1896–1899. doi:10.1109/LCOMM.2012.092812.121712.

[4] L. Sarakis, T. Zahariadis, H.-C. Leligou, M. Dohler, A framework for service provisioning in virtual sensor networks, EURASIP Journal on Wireless Communications and Networking.

[5] A. Merentitis, F. Zeiger, M. Huber, N. Frangiadakis, K. Mathioudakis, K. Sasloglo, G. Mazarakis, V. Gazis, Z. Boufidis, Wsn trends: Sensor infrastructure virtualization as a driver towards the evolution of the internet of things, in: Proc. Seventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies UBICOMM 2013, 2013, pp. 113–118.

[6] M. K. Chowdhury, R. Boutaba, A survey of network virtualization, Computer Networks 54 (5) (2010) 862 – 876.

[7] Z. Xiao, W. Song, Q. Chen, Dynamic resource allocation using virtual machines for cloud computing environment, IEEE Transactions on Parallel and Distributed Systems 24 (6) (2013) 1107 – 1117.

[8] A. Fischer, J. Botero, M. T. Beck, H. de Meer, Virtual network embedding: A survey, IEEE Communications Surveys and Tutorials 15 (4) (2013) 1888 – 1906.

[9] M. M. Islam, M. M. Hassan, G.-W. Lee, E.-N. Huh, A survey on virtualization of wireless sensor networks, Sensors 12 (2012) 2175 – 2207.

[10] S. Madria, V. Kumar, R. Dalvi, Sensor cloud: A cloud of virtual sensors, IEEE Software 31 (2) (2014) 70 – 77. doi:doi:10.1109/MS.2013.141.

[11] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, P. Polakos, Wireless sensor network virtualization: Early architecture and research perspectives, IEEE Network 29 (2015) 104 – 112.

[12] P. Levis, D. Culler, Mate: A tiny virtual machine for sensor networks, SIGARCH Comput. Archit. News 30 (5) (2002) 85–95. doi:10.1145/635506.605407.
URL http://doi.acm.org/10.1145/635506.605407

[13] P. Levis, D. Gay, D. Culler, Active sensor networks, in: Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2, NSDI'05, USENIX Association, Berkeley, CA, USA, 2005, pp. 343–356.
URL http://dl.acm.org/citation.cfm?id=1251203.1251228

[14] Y. Yu, L. J. Rittle, V. Bhandari, J. B. LeBrun, Supporting concurrent applications in wireless sensor networks, in: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys '06, ACM, New York, NY, USA, 2006, pp. 139–152. doi:10.1145/1182807.1182822.
URL http://doi.acm.org/10.1145/1182807.1182822

[15] J. Koshy, R. Pandey, Vmstar: Synthesizing scalable runtime environments for sensor networks, in: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys '05, ACM, New York, NY, USA, 2005, pp. 243–254. doi:10.1145/1098918.1098945.
URL http://doi.acm.org/10.1145/1098918.1098945

[16] I. Leontiadis, C. Efstratiou, C. Mascolo, J. Crowcroft, Senshare: Transforming sensor networks into multi-application sensing infrastructures, in: Proceedings of the 9th European Conference on Wireless Sensor Networks, EWSN'12, 2012, pp. 65–81.

[17] S. Bhattacharya, A. Saifullah, C. Lu, G. Roman, Multi-application deployment in shared sensor networks based on quality of monitoring, in: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE, 2010, pp. 259–268. doi:10.1109/RTAS.2010.20.

[18] C.-L. Fok, C. Julien, G.-C. Roman, C. Lu, Challenges of satisfying multiple stakeholders: Quality of service in the internet of things, in: Proceedings of the 2Nd Workshop on Software Engineering for Sensor Network Applications, SESENA '11, ACM, New York, NY, USA, 2011, pp. 55–60. doi:10.1145/1988051.1988062.
URL http://doi.acm.org/10.1145/1988051.1988062

[19] W. Li, F. C. Delicato, P. F. Pires, Y. C. Lee, A. Y. Zomaya, C. Miceli, L. Pirmez, Efficient allocation of resources in multiple heterogeneous wireless sensor networks, Journal of Parallel and Distributed Computing 74 (1) (2014) 1775 – 1788.
URL http://www.sciencedirect.com/science/article/pii/S0743731513002104

[20] R. Huang, X. Chu, J. Zhang, Y. H. Hu, Energy-efficient monitoring in software defined wireless sensor networks using reinforcement learning: A prototype, International Journal of Distributed Sensor Networks 2015. doi:10.1155/2015/360428.

[21] Y. Xu, A. Saifullah, Y. Chen, C. Lu, S. Bhattacharya, Near optimal multi-application allocation in shared sensor networks, in: Proceedings of the Eleventh ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc '10, ACM, New York, NY, USA, 2010, pp. 181–190. doi:10.1145/1860093.1860118.
URL http://doi.acm.org/10.1145/1860093.1860118

[22] C. Wu, Y. Xu, Y. Chen, C. Lu, Submodular game for distributed application allocation in shared sensor networks, in: INFOCOM, 2012 Proceedings IEEE, 2012, pp. 127–135. doi:10.1109/INFCOM.2012.6195490.

[23] C. de Farias, L. Pirmez, F. Delicato, W. Li, A. Zomaya, J. De Souza, A scheduling algorithm for shared sensor and actuator networks, in: Information Networking (ICOIN), 2013 International Conference on, 2013, pp. 648–653. doi:10.1109/ICOIN.2013.6496703.

[24] D. Zeng, P. Li, S. Guo, T. Miyazaki, J. Hu, Y. Xiang, Energy minimization in multi-task software-defined sensor networks, Computers, IEEE Transactions on 64 (11) (2015) 3128–3139. doi:10.1109/TC.2015.2389802.

[25] N. Edalat, W. Xiao, M. Motani, N. Roy, S. K. Das, Auction-based task allocation with trust management for shared sensor networks, Security and Communication

Networks 5 (11) (2012) 1223–1234. doi:10.1002/sec.631.
URL http://dx.doi.org/10.1002/sec.631

[26] C. Delgado, J. R. Gállego, M. Canales, J. Ortín, S. Bousnina, M. Cesana, An optimization framework for resource allocation in virtual sensor networks, in: Proceedings of IEEE Global Communications Conference, Globecom '15, IEEE, 2015.

[27] Y. Shi, T. Hou, J. Liu, S. Kompella, Bridging the gap between protocol and physical models for wireless networks, IEEE Transactions on Mobile Computing 12 (7) (2013) 1404 – 1416.

[28] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, R. Alexander, RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC 6550 (Proposed Standard) (Mar. 2012).

[29] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: A survey, Computer Networks 38 (4) (2002) 393 – 422.

[30] A. Redondi, M. Tagliasacchi, M. Cesana, Rate-accuracy optimization in visual wireless sensor networks, in: IEEE International Conference on Image Processing (ICIP2012), Orlando, Florida, 2012, pp. 1105–1108.

[31] T. Hou, Y. Shi, H. Sherali, Rate allocation and network lifetime problems for wireless sensor networks, IEEE/ACM Transactions on Networking 16 (2) (2008) 321 – 334.

[32] M. R. Garey, D. S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, USA, 1990.

[33] ILOG CPLEX, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ (2015).

[34] MEMSIC Inc., TelosB Mote Platform Datasheet.

[35] G. Coley, Beaglebone rev a6 system reference manual (2012).

[36] J. B. Javier Molina, Javier M. Mora-merchan, C. Leon, Wireless Sensor Networks: Application, InTech, 2010, Ch. Multimedia Data Processing and Delivery in Wireless Sensor Networks. doi:10.5772/13398.

[37] A. E. Redondi, Energy-aware visual analysis for wireless multimedia sensor networks, Ph.D. thesis, Politecnico Di Milano (2014).

[38] H. Shao, T. Svoboda, L. V. Gool, Zubud zurich buildings database for image based recognition, Tech. Rep. 260Computer Vision Laboratory, Swiss Federal Institute of Technology.

[39] A. Chen, S. Kumar, T. H. Lai, Designing localized algorithms for barrier coverage, in: Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking, MobiCom '07, ACM, New York, NY, USA, 2007, pp. 63–74.

[40] C. Suh, J.-E. Joung, Y.-B. Ko, New rf models of the tinyos simulator for ieee 802.15.4 standard, in: Proc. IEEE WCNC 2007, Hong Kong, 2007, pp. 2238–2242.

[41] CC2420: "Datasheet for Chipcon (TI) CC2420 2.4 GHz IEEE 802.15.4/ZigBee RF Transceiver".