Models and Tradeoffs in WSN System-Level Design

Simone Campanoni, William Fornaciari
Politecnico di Milano, Dipartimento di Elettronica e Informazione
campanoni@elet.polimi.it, fornacia@elet.polimi.it

Abstract

System-level design of WSNs includes the selection of the sensing nodes and their dissemination in the environment to be monitored. Many design choices have to be taken during this stage of the development of the application. The goal of this paper is to present a methodology to specify formally the desired behavior of the sensing application and to derive an optimal selection and placement of the network nodes. The approach is flexible and powerful, since it allows the designer to analyze the impact of clustering sensors onto a reduced set of boards (nodes) and to perform sensitivity analysis on parameters such as type of sensor, position of the nodes, observation time, presence of faults, etc. The paper introduces the concepts with some representative examples as well as by considering bigger use cases extracted from international projects.

1. Introduction

The technologies to realize wireless sensor networks (WSNs) are becoming so mature to enable a "LEGO-like" approach to the deployment of applications. At the prices of consumer electronics, it is available the bare hardware and some middleware to simplify the building of applications [1] [2]. The attention of the designer can be moved towards system-level design issues, concerning the software side of the application and the organization of the sensors set.

Despite of such simplifications, many other questions are still open or partially neglected, like optimization of overall costs (sensors, communication infrastructure, networks deployment, etc), feasibility analysis to understand suitability and effectiveness of the WSN against real application goals, lifetime (especially in the case of battery operating sensor nodes), robustness, etc [1]. The main need is an overall analysis and design framework, to enable a quantitative evaluation of the above properties, taking into account not only the networking-related issues or the distributed software system itself, but also the cross relations existing among the network topology, the nodes, the environment where the WSN is embedded and the events to be monitored, namely the real and comprehensive functional goal of the WSN.

Since a few years, in literature appeared a number of proposals regarding simulation and deployment of WSNs. A not exhaustive list of the more mature and publicly available results is: TOSSIM, NS-2, Avrora, J-Sim, SENSE, OMNeT++, VisualSense, SensorSim, EmStar, OPNET, ATEMU, Ptolemy, etc. Each of these proposals addresses with significant results some specific simulation or implementation aspects of WSN analysis, covering hardware, software and networking. However, from the best of our knowledge, none of them is addressing with a proper and formal extent the capability of the network to capture the events to be monitored. Their focus is frequently related to the optimization of the cost or to verify other properties like power consumption, robustness of the connection layer or the analysis of the models of computation (middleware).

As far as the offline planning is concerned, which is the case this paper is focusing on, the proposals usually deal with only one single objective (e.g., coverage) or in some cases with lifetime in terms of power consumption. The sensing model is normally built around flat squares, and only few proposals cope with simple obstacles [3] [4]. From a more abstract standpoint, the problem of designing WSNs produced noticeable solutions identifying data-centric high level representations of the overall network behavior. For example, TinyDB [5] has been a pioneer effort enabling a SQL-based interface to the sensed data, while considering the need of achieving a power efficient processing and routing of query data. GSN [6] is another proposal based on XML and SQL as data specification and data manipulation languages, taking into account the problem of dynamic reconfiguration of the system. A declarative approach to the network

978-0-7695-3277-6/08 $25.00 © 2008 IEEE
DOI 10.1109/DSD.2008.96

description has been considered in [7], where a dialect of Datalog is used for both data acquisition and transmission management. On top of such internal data representations, an engine to recognize events can be implemented. In [8], Symblic Aggregate approXimation (SAX) is used as an algorithm for detecting complex events by analyzing the patterns related to the sensed basic parameters.

The scope of the work here presented is a wide class of applications where, in addition to the typical monitoring capabilities, it is also required a prompt highlight of the occurrence of particular events. Under these assumptions, our methodology to tackle the problem of designing a sensor network requires to:

- specify the characteristics of the events of interest;
- select a proper set of sensors tailored to catch such events;
- embed the sensors in the environment so to ensure the capturing of the desired events while optimize some design goals, and
- map the sensor set onto realistic WSN board architectures.

First of all it is crucial to make sure a priori that it exists a feasible solution to the sensing problem with the accuracy required by the application. Then, by exploiting the capabilities of the SWORDFISH optimization engine [9], it is possible to derive the WSN by refining the architecture according to design constraints and user's goals. Other important issues addressed in this paper are:

- the possibility to forecast the impact of clustering many sensors onto the same board to keep under control the realization cost;
- the availability of a toolset simplifying the sensitivity analysis of the WSN behavior, both disregarding or considering the need of grouping the sensors in boards.

The paper is organized as follows. Section 2 summarizes the overall architecture of SWORDFISH. Section 3 discusses the models of the events to be recognized and the design flow to create a WSN ensuring that all the events can be sensed. Some of the capabilities of SWORDFISH are discussed in Section IV, where it is shown how it is possible to explore the design space taking into account both abstract and functional requirements. Sections 5, 6 and 7 are more related to the physical constraints/optimizations of actual implementations. It is shown how the different sensors can be grouped in board to decrease costs while maintaining acceptable WSN performance degradation and the impact of design choices onto the capability of the WSN to tolerate temporary faults. Concluding remarks are drawn in Section 8.

2. The design environment

The architecture of SWORDFISH is conceived to support the users during the system-level design of the WSN-based application. The main problems addressed and an outlook of the toolset implementation are discussed in the following of this Section.

2.1 Framework architecture and design flow

The general architecture of SWORDFISH is depicted in Fig. 1. It is composed of a set of modules allowing the users to describe the main actors (sensors, network, events, and environment) and the design goals of the systems (properties of the network and optimization parameters). The coarse grain supported activities are:

Verification. The goal is to determine the occurrence of a set of events (e.g., fire in a defined region, temperature and humidity over a certain threshold for a time window, etc.) by exploiting the *potential* of a given sensor network.

Sensitivity Analysis. Evaluation of the impact of some variation of sensors, environment and network properties, onto the performance of a WSN. Examples are fault tolerance w.r.t. sensors and network errors, effect of sensor aging or moving of their location, influence of the observation time, etc.

Design/Planning. Given a set of events and some constraints/goals, the task is to discover the optimal sensor network capable to identify the events while maximizing user-controlled goal functions.

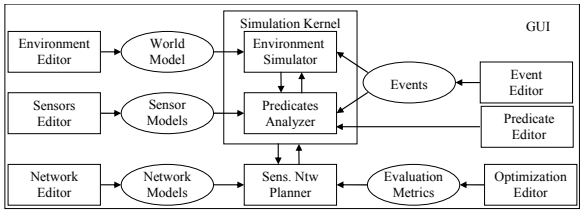

Figure 1. The modules composing SWORDFISH.

The overall framework is encapsulated in a graphical user interface connecting all the different modules, whose main characteristics are outlined in the following (more details can be found in [9]).

Environment Editor. This module allows defining a model of the environment where the WSN will be embodied, with graphical views of the associated physical parameters (e.g., temperature, humidity, 3D-spatial representation, obstacles, etc) and the

possibility to specify constraints such as position and type of some sensors, if relevant for the users.

Sensor Editor. It is the mean to obtain the analytic representation of the sensing nodes, which is a modeling of the relation existing between the sensed physical parameters and the signal produced. The model of the node includes additional information like cost, type of sensors, energy consumption, accuracy, speed, etc.

Network Editor. In addition to the node features, a model of the available connection channels among nodes is specified. This model can cover both wired and wireless links, although current implementation focuses on wireless only.

Predicate Editor. This editor allows the user to specify via logic formulas the properties to be verified in the case a given event occurs. This is of paramount importance to verify that a WSN is actually capable to argue if an event is recognized, or, dually, to select the proper set of sensors to recognize the events. Such a concept more abstract and powerful then a simple measurement-based analysis.

Event Editor. The purpose of this editor is to support the description of the events to be captured in terms of variation of some physical parameters to be sensed, along with their timing characteristics. These models are flexibly implemented via software plugins.

Simulation Kernel. It is the engine which, based on a simulation of the event occurring, modifies the configuration of the world model accordingly. This allows feeding the sensor node models with the real (location aware) data of the world, including their dynamics. Hence, both the physical parameters of the environment and the events to be monitored can be jointly modeled and verified by the Predicate Analyzer (Fig.1)

Optimization Editor. It is an editor allowing the designer to specify and tune the goal functions and the formal model of the network properties/constraints.

Planner. This is the main module for both verification and network design. It allows to formally verify that a given WSN is able to capture a set of events as well as to support the building and optimization of the overall network according to the selected policies and goals.

The focus of this paper is on the sensitivity analysis and on some planning strategies enabled by SWORDFISH. Based on the application requirements, the first steps for the user are defining formally the events to be captured and possibly some optimization goals/constraints. Network properties and sensor behavior can be also specified, in the case of default settings are not considered suitable. According to the existing model of the environment, the events are then "fired" to get a profiling of the evolution of the physical parameters corresponding to the events. Such results are then used as a testbench to compare the performance of alternative WSNs in terms of sensing capabilities. Useful information for optimization can be gathered by analyzing the sensitivity of the network over the variation of parameters like observation time, clustering of sensors or temporary faults, as shown in the following sections.

2.2 Software implementation

The entire SWORDFISH software system has been developed in C under GNU/Linux (Debian distribution), using the following libraries:

- XanLib library ver. 0.1.5: to manipulate data structures like hash tables, trees, pipes, etc.
- Gtk library ver. 2.0: for the GUI development.
- GNU C library ver. 2.3: to interface with the GNU system.
- Flex: to make a lexical analysis of the user's input describing the sensing goal.
- Bison: to generate a parser for the grammar which describes the multi value logic used to express the sensing goal.
- Libglut: for writing the world in a 3D vision.
- Libgtkglext: to embed the openGL objects inside the GTK GUI of SWORDFISH.
- Graphviz: to draw the direct graph representing the sensing goal written by the user and each of its derivatives.

The software architecture is composed of five main modules (Simulator; Planner; Logic_manager; Sensors; Events) whose role is sketched below.

Simulator has the goal to emulate the behavior of the supported physical events (e.g., a fire or an atmospheric phenomenon), while the *Planner* has the role to design the WSN by satisfying the input constraints.

A crucial module is the *Logic_manager*, capable to manage the multi-value logic which is the base for writing the sensing goals and the constraints. Such a module is invoked by the Planner to calculate the truth of a predicate, as a result of certain spatial distribution of the sensors, as well as to calculate the derivative of the logic functions.

Sensor is the manager of the sensor models implemented in SWORDFISH. It is realized via a standard interface based on dynamically loadable shared libraries (plugins). Thanks to this choice, it is allowed to manage efficiently a wide range of sensors with no impact on SWORDFISH code, since the only contact is through the functionality exposed by the

interface. Sensor models can be added incrementally as well as obtained interfacing other libraries, without changing SWORDFISH software.

Events is in charge of managing the implemented available types of events. Its implementation is similar to Sensor, since it uses plugins to decouple the implementation of the events from the rest of the SWORDFISH software.

The current version performs the design of WSNs with a dozen of predicates composing the SG and a similar amount of sensors with runtimes of less than a minute, running on a 1.8 MHz Centrino Laptop. Bigger WSNs (tens of predicates and sensors) requires 1-2 minutes to produce the result. The execution time of SWORDFISH is considerably influenced by the time window (observation period) chosen by the user.

3. Preliminary steps of the design

The model of the environment is 3-D, so that each point is represented by using (x,y,z) coordinates belonging to a user-defined grid. Before starting the exploration of the WSN design space, there are three preliminary steps to be carried out: i) definition of the *purpose* of the network; ii) identification of the *benchmark*; iii) modeling of the *hardness* to recognize physical parameters corresponding to an event.

The first activity turns to the definition of an overall **Sensing Goal** (SG) for the WSN, that is a multi-value logic formula composed of some predicates Pr (implemented via plugins), each corresponding to an event. For example Water(x, y, z, magn, trend) is a plug-in modeling the presence of *water* in the point (*x,y,z*), starting from a given *magnitude* and with a specified *trend* over the time. A predicate Pr is an instance of Water applied to a specific point. A catalog of plugins (e.g., Fire, Water, Humidity, …) is available, and its extension is straightforward. An example of sensing goal is (1).

$$SG = \text{Water}(3,2,1,\ 30,\ \text{const}) \text{ AND } \text{Water}(5,6,7,\ 20,\ \text{const}) \quad (1)$$

Such SG means that the WSN has the goal to discover the concurrent presence of the events of having a certain amount (30 and 20) of water in two points (3,2,1), (5,6,7) of the environment.

The second step is the characterization of the changing in the environment whenever the events occur, namely the identification of a testbench to evaluate the WSN performance. To this purpose, based on the (user defined) *fp* sampling rate of the environment simulator, a profiling stage is triggered by firing each one of the defined events, namely running the Pr-related plugins. At the end, $\forall(x,y,z)$, and $\forall$ Pr of SG, all the data patterns are obtained.

There are at least other two problems the designer has to face with during WSN system-level design. The first concerns the selection of the type of sensor, while the second is the sensors placement. In fact, the target is to discover a positioning of the sensors, maximizing the capability of the WSN to recognize the events, i.e. maximizing the SG. The former question impacts mainly on the feasibility of designing a WSN capable to recognize the events encompassed by the SG. The latter is related to the dissemination of sensors in order to enhance their possibility to satisfy the Pr composing the SG, i.e. improving the performance of the system.

In the current implementation of SWORDFISH, we followed an approach producing results in the order of seconds, so as to actually enable sensitivity analysis. In Section 4 and Section 7, the main benefits of sensitivity analysis are addressed through some representative examples.

Our first concern in the design flow is ensuring that a solution to the SG can exist, by using a proper set of sensors that is incrementally built up and significantly optimized by sharing sensors among the set of Pr (specified in the SG) to be verified. Then, this set of candidate sensors are placed in the environment taking into account the information coming from a configurable *hardness* function. In such a way, it is guaranteed to obtain a WSN formally satisfying the SG with a quasi-optimal cost, with runtimes in the order of a few seconds.

As far the positioning of the sensors is concerned, we defined a *hardness* function Hard(x, y, z, Pr) modeling the difficulty in evaluating Pr in a given point (x,y,z).

$$\text{Hard}(x,y,z,Pr) = \text{Hs}(PPr,t) / C\{(PPr,t), Pr\} \quad (2)$$

Calling PPr the *profiling output* of Pr, i.e. the data pattern associated to Pr obtained during the initial profiling, Hs(PPr,t) depends on the type of sensor (corresponding model) and relates to the difficulty to recognize the event Pr within the time frame of a profiler sampling rate (1/fp). For example for a slow temperature sensor can be hard (or even impossible) recognizing T-ramps moving faster than its cutting frequency. The term $C\{(PPr(x,y,z), Pr\}$ is the confidence to infer the truth of Pr based on the sequence of the physical variations defined via PPr.

Of course, any positioning strategy for the sensors attempts to place the sensor where *Hard* is low, i.e. where it is easier and more reliable recognizing the Pr composing the SG. More formally, it is selected the Si to be assigned to the predicated Pj, such that $\lfloor dSG/dPj \rfloor \forall$ Si available, is minimum.

The implemented algorithm actually starts considering only the models of the available types of sensors and the predicate Pr to be satisfied, with possibly additional constraints (e.g., cost figures) that can be provided by the users within the sensor plugins. Then, the minimum set of sensors capable to recognize physical parameters to satisfy all the Pr is discovered and initially allocated to the most relevant predicates (in the SG sense). Based on this initial allocation, that is a pre-condition to satisfy SG, the sharing of the sensor proceeds as described in the above example. The end of the process produces a solution employing the minimum set of sensors covering all the predicates, using a quick heuristic producing a configuration that in most of the cases it is also the absolute optimum.

To represent how a given sensor is actually capable to capture its target events from a position (xp,yp,zp), a proper metric (3) has been defined, called *confidence*.

$$\text{Confidence} = 1-(\text{Hard}(xp,yp,zp,Pr))/\max \text{Hard}(x,y,z,Pr)) \quad (3)$$

Where Pr is the predicate corresponding to the event, Hard (xp,yp,zp,Pr) is the hardness calculated in the candidate point for the sensor positioning and max Hard (x,y,z,Pr) is the maximum hardness within the considered environment. Note that values of confidence closer to one means that the position of the sensor is approaching the best existing in the environment to satisfy Pr, while lower values corresponds to critical points; this latter case can trigger the search for a better positioning or the increasing of the sensor set cardinality.

In summary, Fig.2 depicts the pseudo-code steps of the WSN planning implemented in SWORDFISH.

1. **Analysis** of the inputs (sensing goal parsing and constraints processing)
2. Storing of the **initial condition** for the environment simulation
3. **Profiling** of the events composing the sensing goal (storing of the data for each physical parameters and point, given an observation window and a user defined sampling rate of the simulation)
4. Computation of the **hardness** grid for each predicate composing the sensing goal
5. for(numSensors=1; numSensors < maxSensors; numSensors++) {
   a) choice of the target predicate for the sensors (depending on numSensors and sensing goal)
   b) computation of the sensors positions (based on Hardness and numSensors)
   c) if (check_WSN()==OK) break}

Figure 2. Steps of the WSN planning strategy.

4. Sensitivity analysis

First of all this section shows some practical usages of SWORDFISH and demonstrates, by using simple examples, its flexibility and the value added even when the complexity of the application seems to be manageable. In this section we still consider abstract architectures with one sensor per node. Section 7 figures out how similar analysis can be carried out also at board level, i.e. with architectures closer to real implementations.

4.1 Sensor set and observation time

This example shows the influence of the number of sensors and of the observation time onto the truth value of the sensing goal, namely the confidence on the capability of the WSN to correctly recognize the events. We considered a linear model for the sensor and the sensing goal (5) corresponding to the identification of three events.

$$SG=\text{Water}(9,9,9) \text{ AND } \text{Water}(0,0,0) \text{ AND } \text{Water}(4,4,0) \quad (5)$$

The analysis result is depicted in Fig.3, showing how vary the SG when changing the observation time (time windows) and the number of sensors.

The obtained result reveals that using at least three sensors it is possible to realize a WSN capturing all of the three events disregarding the observation period.

Conversely, using less than three sensors, the time window influences the performance. With two sensors the observation time must be grater than 3 seconds: such sensors (S0, S1) will be able to recognize more than one event with the following positioning: S0=(9,9,4), S1=(1,2,0). In such a case, S1 can capture most of the events Water(0,0,0) and Water(4,4,0), so that S0 and S1 can cover the entire SG.

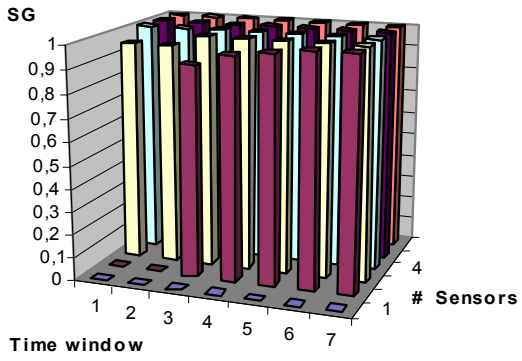

Figure 3. Influence of the numer of sensors and time window on the Sensing Goal.

It is worth nothing that S0 is not positioned in (9,9,9), as in the case where more than three sensors are available. In fact, under this more severe "restrictions", S0 contributes to the identification of the rest of the events, though its position denotes a major value added for Water(9,9,9).

4.2 Impact of the sensor position

To figure out the cross relation existing between the type of event and the position of a sensor, we considered two sensing goals (6) with a model of the sensor is still linear:

SG1 = Water(4,4,4); SG2 = Fire(4,4,4); (6)

The value of SG1 has been computed considering the following space: X=[3..5], Y=[2..4] and Z=[0..3], with a time window of two seconds). The obtained data show that, to recognize a *Water* event, the sensor has to be located close to the point of interest.

Conversely, for the *Fire* event, the positioning of the sensor seems to be less important w.r.t. the previous case. This result makes sense: it possible to recognize fire events even by positioning sensors far away from the critical area. Within the entire analyzed space for the positioning of the sensors (X=[4..8], Y=[4..7] and Z=[0..8]), SG2 has been always satisfied.

4.3 Sharing of sensors

This example addresses the search for a WSN capable to recognize an event with a scarce amount of resources (sensors). The user have to specify the max number of sensors, the min value of the SG considered acceptable and other data regarding the observation time window for the sensors. We have chosen the SG (7), corresponding to the presence of water on the ground in two positions:

SG = water(0,0,0) AND water(2,2,0),
with min SG=0.2 (7)

Tab.1 reports the output of SWORDFISH (sensor position) along with the confidence for each predicated. In this simple case the SG is close to one, pretty over the 0.2 threshold.

The truth value of the single predicates are similar and close to one (water(0,0,0)=water(2,2,0)=0.99999) so that the SG=0.998 is fairly acceptable. In summary, the system discovers an intermediate position for a single sensor (1,2,0) ensuring the meeting of the SG with sensor sharing. In the case our goal is modified (8) to recognize the presence of water in two points more distant as above and with an observation window of 5 seconds, i.e.:

SG = water(0,0,0) AND water(9,9,9),
with min SG=0.2 (8)

The system fails in using only one sensor and find out automatically a new WSN using two sensors, now satisfying the SG. Because of we have two sensors for two events, the suggested positioning of the sensors are obviously overlapped to the event locations (Tab.2).

Note that Tab.2 highlights a (negligible in this case) contribution of S1 also to water(0,0,0) recognition. Such type of information can be useful to identify *Achilles' heel* of more complicated WSNs, where the amount of sensors makes hard identifying their ordering of relevance in contributing to the overall SG.

Table 1. Positioning of the set of sensors.

Sens	Pos	Confidence		
		Water(0,0,0)	Water(2,2,0)	Total
S0	(1,2,0)	0.999	0.999	0.998

Table 2. Solution with two sensors.

Sens	Pos	Confidence		
		Water(0,0,0)	Water(9,9,9)	Total
S0	(0,0,0)	0.9999	0.0	0.999
S1	(9,9,0)	0.1	0.9999	0.999

4.4 Influence of the type of sensor

The WSN we are designing has the responsibility to report the presence of two different events (water and fire) having different sensing requirements. In particular, in our modeling environment sensing the water it is harder than recognizing the fire. The SG is:

SG = water(0,0,0) AND fire(5,5,0),
with min SG=0.4 (9)

In this case, as shown in Tab.3, the positioning of the sensor is closer (0,3,0) to the water, because of the sensing of fire is considered easier than recognizing the water itself. In more complicated scenarios, but even in this simple case, the typical design approach to place the sensors in intermediate positions disregarding the type of events to be considered, does not allows to take full advantages from the WSN intrinsic capability.

Table 3. Influence of type of sensor on the WSN placement.

Sens	Pos	Confidence		
		Water(0,0,0)	Fire(5,5,0)	Total
S0	(0,3,0)	0.99	0.99	0.99

5. Modeling of nodes

The output of the SWORDFISH-based feasibility analysis is a set of {sensor, position} tailored to optimize cost-effectiveness and performance (sensing goal) of the WSN. On the other hand, realistic design and deployment typically requires simplifying both the hardware and the architecture of the network, while maintaining effectiveness and performance.

To cope with such problem, in the following section the PESCA (Pareto Efficient Solution Clustering Algorithm) approach is outlined. The goal of PESCA is to produce a quasi-optimal clustering of the sensors by identifying an optimal mapping of the sensor set onto boards, like those available on the market [2]. In general, each board hosting sensors includes the following sections: PCB/package; power supply and energy management; radio (RX/TX); control/processing Unit (CU); connectors/Interfaces; one or more sensors. Based on our experience in realizing PCB-level embedded systems and on market availability of sensing modules, we found reasonable adopting the model (10) for the cost of each board (*node*).

$$NodeCost = Const + K \times \log(N) + \sum_{j=1}^{SensorTypes} (SC_j \times NumS_j) \quad (10)$$

Where, for each board, N is the overall number of sensors, $NumSj$ is the number of sensors of a given type j, $SensorTypes$ is the number of possible types of sensor, and SCj is a cost of a sensor of type j.

In other words, there exist a variable cost which is related to the type and number of sensors in a linear manner and a processing cost that is logarithmic, due to the typical price trends of CPU and microcontrollers. Furthermore, the cost of PCB and packaging is less than linear against the number of sensors (close to a constant), while radio and power supply is fairly stable over a wide range of possible sensors cardinality.

As far the cost function is concerned, to increase the generality over different suppliers, we partitioned the available sensors into classes, to capture their relative cost, instead of considering the absolute values. Concerning the cost of the network, we assume a constant value for wireless connections (typically built-in in commercial nodes).

Actually, some influence of the network topology should be considered in the case of some gateway nodes, mastering hierarchies of sensors patches, were identified. In such a case there is an additional cost related to the wired connection or the use of other long-range radio communication standards/modules (e.g., GPRS), but this analysis is not included due to the lack of space.

6. Board-level design

The clustering of the set of sensors identified by the SWORDFISH framework is a multi-stage process, including the following topmost activities: compatibility analysis between all the possible pairs of sensors; identification of the boundaries of the clustering problem (worst and best case); generation and evaluation of the candidate solutions.

6.1 Compatibility graph

At the beginning, the user (e.g., by accepting default settings) is required to specify constraints on the possible clustering of different sensors onto the same board. Based on these information, an Interference Graph G=<N,E> is built, where nodes *n* are sensors and an edge *e* among the nodes represents a possible sensor interference to be avoided. Note that the interference is not only related to the nature of the sensors, as specified by the user. In fact, two sensors can have no shared position where both sensors are still able to discover the associated set of events. This latter case can be discovered by using the Hard function; in fact, the Hardness of a point to discover two events *e1*, *e2* is the sum of the Hardness of both sensors computed in that point.

Once the interference graph is completed, it is possible to identify the compatibility graph G', which is its complementary graph G'=<N,E'>, gathering all the feasible solutions. Note that any possible clustering of sensors, cannot but be a clique of the compatibility graph. In fact, all the sensors hosted by the same board must be cross-compatible. The next step is the computation of all the maximal cliques of the compatibility graph G'. Since this type of activity is recognized to be a NP-hard problem [10] we adopted some heuristics to speedup the computation.

6.2 Coverage of the sensor graph

To better understand the optimality of placing a certain group of sensors onto a board, we focused at the beginning on the boundaries solutions, considering the cardinality of the cliques (not their cost). At the lowest level, each sensor corresponds to a board. As far the best case is concerned, the biggest clique has been computed, and then the same action has been performed on the remaining graph, and so on. At the end we obtain a partitioning of the compatibility graph,

where its cliques are those clustering the maximum number of compatible nodes.

The design space spanning between the two boundaries cases so identified, contains a number of possible solutions that is exponential with the sensor cardinality. We attacked the analysis of a so wide solution space through an heuristic structured into a pair of consecutive steps.

1. Starting from the best case above described, solutions are generated by creating a new list of cliques where one sensor has been extracted from the biggest clique, to create a new single-sensor board. The same activity is then repeated, considering the biggest cliques at any iteration. At the end of such a process, a wide set of possible solutions is generated, ordered for relevance i.e., in terms of cardinality of the biggest clique.
2. All the possible pairs of the above solutions are taken into account for possible merging, while verifying the compatibility (taboos) of the new sensor set.

At the end of the second steps, the recombining of cliques not maximal in terms of cardinality, allows the optimizer to consider solutions more *orthogonal*, possibly characterized by a lower board-level cost.

6.3 Selection of solutions

This step takes into account the candidate WSNs under the Pareto standpoint. The task of the PESCA algorithm is to find out a solution to the multi-objective clustering problem, considering two metrics: the *cost* of the solution and the functional quality, i.e. its *performance*. The cost of a solution (set of boards) is evaluated through the cost model (10) described in Section 5, which is depending on the number and type of sensors associated with each partition. Concerning the performance, the badness of a solution is computed by exploiting the Hardness functions *Hij(x,y,z)* of the event *i* covered by the sensor *j* belonging to the same board. Thus, the hardness of the entire WSN is:

$$H(x,y,z) = \sum_{j \in WSN} \sum_{i \in J} H_{ij}(x,y,z) \qquad (11)$$

The badness of the WSN is evaluated, and its minimum corresponds to a point where the positioning of the board is optimal. This new location, which is shared by all the sensors on the same node, is the best to ensure that all the events associated with the sensors can still be captured after clustering. Note that the solution so discovered is a Pareto efficient solution. In fact, all the solutions "dominated" within the <*cost, performance*> optimization space of the WSN are discarded during the population of the design space.

7. Experiments

The PESCA approach and some capabilities to analyze the degradation of performance in the case of temporary faults have been considered.

7.1 Mapping on boards

Some validation scenarios extracted from multi-partner projects [11] [12] have been considered. Figure 4. reports the analysis of a complex WSN: SG of 16 basic predicates producing an optimal network with 18 ideal sensors. The plot reports the Pareto solutions and figures out the influence of the fixed costs (linked to the volume/standardization of boards) against the overall cost and performance (1/Hardness) of the clustered WSN. It is possible to observe the impact on performance and cost associated with the spreading vs clustering of sensors. By following such a quantitative analysis, the design driver may be not only the cost, but also the capability of the WSN to fulfill the initial application requirements.

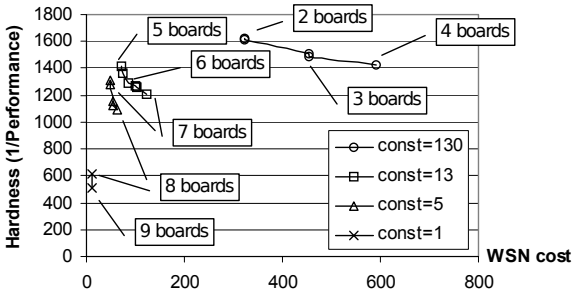

Figure 4. Pareto frontier of the cost-performance tradeoff (varying the fixed cost of the board).

7.2 Impact of temporary faults

The analysis framework allows also to quantitatively put in evidence the impact of some design parameters onto the fault tolerance of the WSN. Let us consider the two examples reported in Tab.4 and Tab.5.

Table 4. Fault tolerance vs sensor sampling rate.

Fp\Sf	10	12	14	16	20
0%	0.814	0.814	0.814	0.814	0.814
10%	0.081	0.629	0.698	0.752	0.814
20 %	0	0.316	0.358	0.391	0.65
30%	0	0.14	0.165	0.184	0.44
40%	0	0	0.065	0.075	0.211
50%	0	0	0	0	0.089
>50%	0	0	0	0	0

Both depict how the SG changes in the case of the probability Fp of temporary fault of one sensor, whose duration produces the loss of a single sample, varies from 0% to 100%. Gray regions correspond to SG greater than 0.5, but of course the choice of such threshold is up to the user. In both cases the time windows (Tw) to recognize the events is fixed to 2. Considering the minimum number of samples to correctly recognize the events (Smin) equal to 10, Tab.4 figures out the impact of varying the sampling rate Sf from 10 to 20 for a WSN with a SG composed of 6 predicates, each assigned to a separate sensor. It is manifest the capability of the network to tolerate increasing error probabilities as the sampling rate rises up. Tab.5 addresses a different analysis, where the number of predicates is still 6, but the number of sensors (#sens) ranges from 6 to the double of the predicates (12). As it can be seen, such redundancy (#sens>6) allows the WSN to tolerate significant fault probabilities, while maintaining high sensing goals.
In summary, the framework significantly simplify a comparative (and quantitative) analysis of the solutions to enhance the fault tolerance of the WSN.

Table 5. Fault tolerance vs # of sensors.

Fp\#sens	5	6	8	10	12
0%	0.814	1	1	1	1
10%	0.081	0.533	1	1	1
20%	0	0.131	1	1	1
30%	0	0	1	1	1
40%	0	0	0.74	1	1
50%	0	0	0.45	1	1
60%	0	0	0.26	1	1
70%	0	0	0.13	0.888	1
80%	0	0	0	0.609	1
90%	0	0	0	0.403	0.51
100%	0	0	0	0	0

8. Conclusions

The paper described the design methodology of SWORDFISH and on some aspects related to the sensitivity analysis. The presented approach is complementary to the typical simulation-based analysis, since its emphasis is on the system-level steps of the design, where a broad design space has to be extensively and efficiently explored, and on the formal modeling and verification of the WSN objectives.
The obtained results are promising, and some of the verification and top-level analysis and design capabilities have been addressed by considering simple but representative examples and stressed with the use-cases of [11] [12]. It has been shown how it is possible to optimize the WSN while ensuring that the original user' goal has been fulfilled. The examples reveal that many side-effects of changing the behavior of the WSNs and sensor positioning (or clustering) produces significant changing in the sensing goal that are hard to be managed by a human designer without the support of a proper methodology and a toolsuite. At the same time, it becomes affordable the effort to quantitatively compare alternative solutions in terms of fault tolerance capability of the WSN.

10. References

[1] Akyildiz I.F.; Weilian Su; Sankarasubramaniam Y.; Cayirci E. 2002. A survey on sensor networks,IEEE Comm. Mag., vol. 40, n. 8, pp. 102-114, Aug 2002.

[2] www.moteiv.com, www.xbow.com

[3] S.Dhillon, K.Chakrabarty, S.Iyngar. Sensor placement for grid coverage under imprecise detections. Proc. Of Int. Conf. on Information Fusion, July 2002, pp 1581-1587.

[4] A.Howard, M.Mataric, G.Sukhatme, An incremental self-deployment algorithm for mobile sensor networks. Autonomous Robots-Special Issue on Intelligent Embedded Systems, Vol.13(2), 2002. pp 113-126.

[5] S.Madden, M.Franklin, J.Hellerstain, W.Hong, TinyDB: an acquisitional query processing system for sensor networks, ACM Trans. on Database Sys, vol.30 (1), 2005.pp122-173.

[6] K.Aberer, M.Hauswirth, A.Salehi, A middleware for fast and flexible sensor deployment, Proc of 32nd Int. Conf. on VLDB, 2005. pp 1199-1202.

[7] D. Chu, L.Popa, A.Tavakoli, J.Hellerstein, P.Levis, S.Shenker, The design and implementation of a declarative sensor network system, Proc. of SenSys'07, November, 2007. Sydney, Australia.

[8] M. Zoumboulakis, G.Roussos, Escalation: Complex Event Detection in Wireless Sensor Networks, LNCS Smart Sensing and Context, Vol. 4793/2007, October 2007.

[9] S.Campanoni, W.Fornaciari, SWORDFISH: a Framework to Formally Design WSNs Capturing Events, In Proc. of IEEE SoftCOM'07, Split-Dubrovnik, Croatia, September, 2007.

[10] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences), W. H. Freeman Publisher, 1979. ISBN 0716710455.

[11] Adaptive Infrastructure for Decentralized Organizations - ARTDECO, Min. for the National Research, 2006-2009. http://artdeco.elet.polimi.it

[12] WASP (Wirelessly Accessible Sensor Populations), EC-Funded IST project, http://www.wasp-project.org/