# Supervisory Controller for Task Assignment and Resource Dispatching in Mobile Wireless Sensor Networks

*Vincenzo Giordano, Frank Lewis, Prasanna Ballal & Biagio Turchiano*

## 1. Introduction

Wireless sensor networks are one of the first real-world examples of pervasive computing [1, 14, 24]. Small, smart, and cheap sensing devices will eventually permeate a certain environment and, suitably coordinated, will automatically recognize the present context and accordingly readjust their behavior. Smart environments represent the next evolutionary development step in building, utilities, industrial, home, shipboard, and transportation systems automation. Although this technology is still in its early days, the range of potential applications is mind-boggling – Robotics, health monitoring, defense systems, habitat monitoring etc. Sensor networks would greatly help monitor the environment and detect occurrences of natural calamities. For example, sensor networks that measure seismic activity from remote locations and provides tsunami warnings to coastal areas. However they present a range of challenges as they are closely coupled to the physical world with all its unpredictable variation, noise, and asynchrony; they involve many energy-constrained, resource-limited devices operating in concert; they must be largely self-organizing, adaptable to different environmental sensing applications and robust to sensor losses and failures.

To meet these challenges, recently there has been increased research interest in systems composed of autonomous mobile robotic sensors exhibiting cooperative behavior [13]. Sensor mobility, in fact, holds out the hope to support self-configuration mechanisms [5], guaranteeing adaptability, scalability and optimal performance, since the best network configuration is usually time-varying and context dependent. An example of mobile wireless sensor network is a warehouse guarded by mobile robots constantly interacting with strategically placed ground sensors to keep the risk of fire, robbery, etc to a minimum, while reducing the personnel costs.

In related literature, effective coordination strategies have been proposed to coordinate mobile robotic sensing units cooperating for a common goal. In decentralized coordination approach, the robots rely only on information about their local neighbors and the behavior of the overall network of robots emerges from the interactions of single units. In [4] and in [7] decentralized coordination algorithms for robots teams are proposed, to guarantee minimum exploration time and complete coverage respectively. In [10] a potential field approach to reach uniform deployment of sensors in an unstructured environment is proposed. It is implicitly assumed that the network cannot be configured for different missions apart from blanket coverage. [2, 3] propose behavior based formation control for



Source: Cutting Edge Robotics, ISBN 3-86611-038-3, pp. 784, ARS/plV, Germany, July 2005 Edited by: Kordic, V.; Lazinica, A. & Merdan, M.

exploration purposes, but no specific performances can be guaranteed and changes in the mission plans are not straightforward. In all the before mentioned decentralized approaches, robots possess similar functionalities, perform similar tasks and just one mission at a time is usually implemented.
To overcome the inherent limitations of decentralized approaches, supervisory (centralized) control techniques have been studied. In [8], a supervisory controller is proposed which reschedules the mission planning in response to uncontrollable events (node failures) using computationally efficient algorithms. Also the use of a centralized coordinator can ensure that the group possesses certain desired properties and remains within the bounds of pre-specified behavioral constraints. Some significant results in supervisory control have also been obtained using Petri nets [11]. Nevertheless the implementation of high-level mission specifications is not straightforward, the dynamical description of the system is incomplete and a new design stage, almost from scratch, is required if objectives or resources change. Summarizing, in related literature, there is a lack of supervisory control techniques which can sequence different missions according to the scenario (adaptability) and reformulate the missions if some of the robots fail (fault tolerance) in a predictable way and using a high-level interface.
In this chapter we present a discrete-event controller (DEC) for the centralized coordination of cooperating heterogeneous wireless sensors, namely unattended ground sensors (UGSs) and mobile robots. The DEC sequences the most suitable tasks for each agent according to the current perception of the environment. Priority rules for efficiently dispatching shared resources and handling simultaneous missions can also be easily taken into account. A novel and easy to implement matrix formulation makes the assignment of the mission planning straightforward and easily adaptable if agents or applications change. It represents a complete dynamical description which allows one to simulate and implement the system with the same controller software, simplifying the implementation of a mobile WSN in real world scenarios (e.g. when hostile terrains and huge areas have to be monitored).

## 2. Discrete Event Controller (DEC)

This section presents a matrix-based discrete event controller for modeling and analysis of complex interconnected DE systems with shared resources and dynamic resource management. Its matrix formulation gives a very direct and efficient technique for both computer simulation and actual on-line supervisory control of DE systems. It provides a rigorous, yet intuitive mathematical framework to represent the dynamic evolution of DE systems according to linguistic *if-then* rules:

*Rule i: If* <conditions$^i$ hold > *then* <consequences$^i$>

For coordination problems of multi-agent systems (e.g. mobile robots and wireless sensors), we can write down a set of if-then rules to define the mission planning of the sensor agents, such as:

*Rule i: If* <sensor1 has completed task1, robot2 is available and a chemical alert is detected > *then* <robot 2 starts task4 and sensor 1 is released>

These linguistic rules can be easily represented in mathematical form using matrices. Let $r$ be the vector of resources used in the system (e.g. mobile robots and UGSs), $v$ the vector of tasks that the resources can perform (e.g. *go to a prescribed location*, *take a measurement*, *retrieve and deploy UGS*), $u$ the vector of input events (occurrence of sensor detection events, node failures, etc.) and $y$ the vector of completed missions (outputs). Finally, let $x$

be the state logical vector of the rules of the DE controller, whose entry of '1' in position *i* denotes that rule *i* of the supervisory control policy is currently activated.
Then we can define two different sets of logical equations, one for checking the conditions for the activation of rule *i* (matrix controller state equation), and one for defining the consequences of the activation of rule *i* (matrix controller output equation). In the following, all matrix operations are defined to be in the or/and algebra, where + denotes logical or and 'times' denotes logical and.
The controller state equation is

$$\overline{x} = F_v \overline{v} + F_r \overline{r} + F_u \overline{u} + F_{ud} \overline{u}_d \tag{1}$$

where $x$ is the task or state logical vector, $F_v$ is the task sequencing matrix, $F_r$ is the resource requirements matrix, $F_u$ is the input matrix. $F_{ud}$ is the conflict resolution matrix and $u_d$ is the conflict resolution vector. They are used to avoid simultaneous activation of conflicting rules, as will be shown later. The current status of the DE system includes task vector *v*, whose entries of `1' represent `completed tasks', resource vector *r*, whose entries of `1' represent `resources currently available', and the input vector *u*, whose entry of 1 represent occurrence of a certain predefined event (fire alarm, intrusion etc.). The overbar in equation (1) denotes logical negation so that tasks complete or resources released are represented by '0' entries.

$F_v$ is the task sequencing matrix (used by Steward [22] and others in manufacturing), and has element *(i,j)* set to '1' if the completion of task $v_j$ is an immediate prerequisite for the activation of logic state $x_i$.
$F_r$ is the resource requirements matrix (used by Kusiak [12] and others in manufacturing) and has element *(i,j)* set to '1' if the availability of resource *j* (robot or UGS) is an immediate prerequisite for the activation of logic state $x_i$.
On the ground of the current status of the DE system, equation 1 calculates the logical vector *x*, i.e. which rules are currently activated. The activated rules determine the commands that the DEC has to sequence in the next iteration, according to the following equations

$$v_s = S_v x \tag{2}$$

$$r_s = S_r x \tag{3}$$

$$y = S_y x \tag{4}$$

$S_v$ is the task start matrix and has element *(i,j)* set to '1' if logic state $x_j$ determines the activation of task *i*.

$S_r$ is the resource release matrix and has element *(i,j)* set to '1' if the activation of logic state $x_j$ determines the release of resource *i*.

$S_y$ is the output matrix and has element *(i,j)* set to '1' if the activation of logic state $x_j$ determines the completion of mission *i*.
The task start equation (2) computes which tasks are activated and may be started, the resource release equation (3) computes which resources should be released (due to completed tasks) and the mission completion equation (4) computes which missions have been successfully completed.
Vector $v_s$, whose `1' entries denote which tasks are to be started, and vector $r_s$, whose `1' entries denote which resources are to be released, represent the commands sent to the DE

system by the controller. '1' entries in vector $y$ denote which missions have been successfully completed.
Equations 1-4 represent the rule-base of the supervisory control of the DE system. All the coefficient matrices are composed of Boolean elements and are sparse, so that real time computations are easy even for large interconnected DE systems.
The task sequencing matrices ($F_v$ and $S_v$) are direct to write down from the required operational task sequencing. On the other hand, the resource requirements matrices ($F_r$, $S_r$) are written down based on the resources needed to perform the tasks and are assigned independently of the task sequencing matrices.
Given the presence of shared resources, simultaneous activation of conflicting rules may arise. Matrix $F_{ud}$ in equation (1) is used to resolve conflicts of shared resources, i.e. conflicts deriving by the simultaneous activation of rules which start different tasks requiring the same resource. Matrix $F_{ud}$ has as many columns as the number of tasks performed by shared resources. Element *(i,j)* is set to '1' if completion of shared task $j$ is an immediate prerequisite for the activation of logic state $x_i$. Then an entry of '1' in position $j$ in the conflict resolution vector $u_d$, determines the inhibition of logic state $x_i$ (rule $i$ cannot be fired). It results that, depending on the way one selects the conflict-resolution strategy to generate vector $u_d$, different dispatching strategies can be selected, in order to avoid resource conflicts or deadlocks.
In the conversion of linguistic rules into the matrix formulation of the DEC, the following assumptions must be considered:

1. A resource cannot be removed from a task until it is complete.
2. A single resource can be used for only one task at a time.
3. A process holds the resource allocated to it until it has all the resources required to perform a task.
4. Resource is released immediately after it has executed its task.

## 3. Control Architecture

To use the DEC as a Supervisory Controller for task assignment and resource dispatching in mobile wireless sensor networks, we need to have an architecture that is modular, flexible and adaptable. It consists of three layers [20] with distinct functionalities, namely agent control layer, network control layer and supervisor control layer (figure 1). In this way improvements and updates on one layer results in minor changes in the other layers.
The first layer (agent control level) deals with the control of each agent (being either a UGS or a mobile robot), keeping into account its peculiar functionalities. At this level we define the processing capabilities of the UGSs (e.g. signal processing) and the control algorithms for the behavior of each robot (e.g. reach the target, follow another robot etc.).
The second layer (network control level) deals with the implementation of communication protocols for energy efficient data transmission between the UGS, robots and the supervisor.
The third layer (supervisor control level) consists of the matrix-based DE supervisory controller whose matrix formulation allows one to employ a high-level human interface to define the mission planning, the resource allocation and the dispatching rules. The supervisor is in charge of sequencing the tasks each agent has to perform according to the perception of the environment, assuming that the agent level controllers correctly perform the assigned tasks and that the communication protocol for each agent perfectly works. The feedback information about the evolution of the scenario is provided by the "sensor to context mapping" modules (e.g. see [25]), constituted by sensor fusion and decision

making algorithms implemented on some (or all) of the agents. In order to keep track of the dynamic behavior of the network, the supervisor also receives notification from each agent about the completed tasks and the released resources.

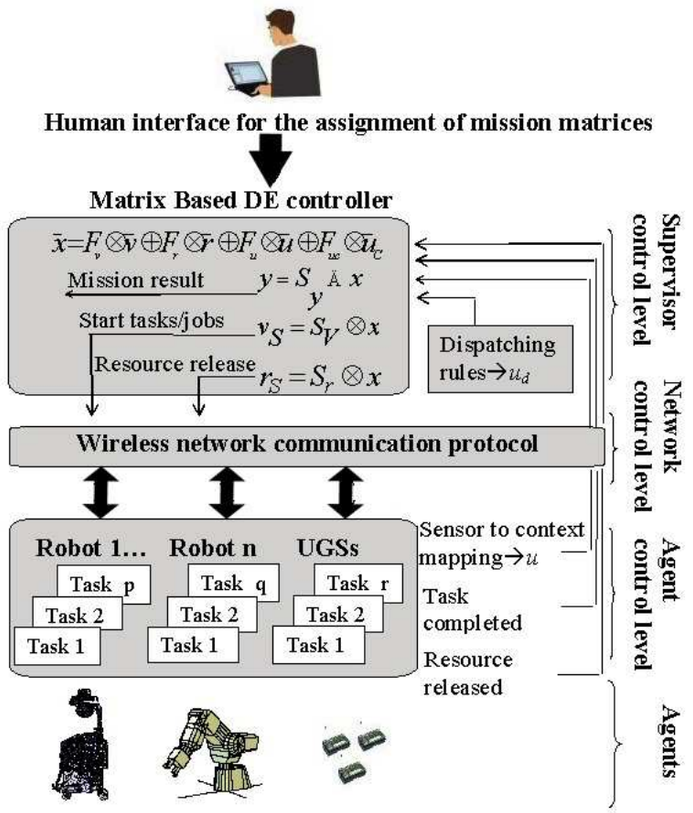
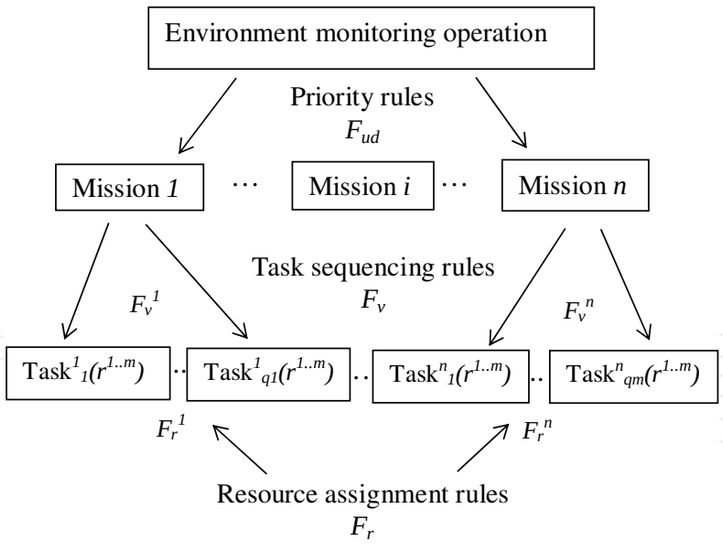



Figure 1. Complete control architecture

Thus, using this architecture, a complex system can be decomposed into missions, tasks and rules for task sequencing, resource dispatching and conflict resolution (figure 2a).
This block diagram can be represented using block matrices in the framework of our DEC. Suppose that we have $m$ resources $r_j$ $j=1…m$ (mobile robots and stationary sensors) each one capable of performing $p_j$ tasks, and define $n$ different missions, each one composed of $q_i$ tasks. For each mission, we define the corresponding set of matrices $F_v^i$, $F_r^i$, $S_v^i$, $S_r^i$ which represent the coordination rules of the agents in the execution of the tasks. In order to take into account the priority among missions, we derive the global conflict resolution matrix $F_{ud}$ according to the following procedure. After assigning a priority order $k$ to each mission, we calculate, for every resource $j$ and every mission $i$, a matrix ($F_{ud}^i(r_j)$), creating a new column for every '1' appearing in the $jth$ column of $F_r^i$. Then we construct the global conflict resolution matrix of resource $r_j$ ($F_{ud}(r_j)$) inserting each $F_{ud}^i(r_j)$ matrix in position $(i,k)$.
As shown in figure 2b, the matrix formulation of the overall environment monitoring operation is then obtained by stacking the set of matrices together. The correspondence between figure 2a and 2b is straightforward.




(a)

$$F_{ud} = \left[F_{ud}(r_1)...F_{ud}(r_j)...F_{ud}(r_m)\right]$$

$$F_v = \begin{matrix} q_1 & \dots & q_i & \dots & q_n \end{matrix} \begin{bmatrix} F_v^1 & & & & \\ & \dots & & & \\ & & F_v^i & & \\ & & & \dots & \\ & & & & F_v^n \end{bmatrix} \qquad F_r = \begin{matrix} m \end{matrix} \begin{bmatrix} F_r^1 \\ \dots \\ F_r^i \\ \dots \\ F_r^n \end{bmatrix}$$

(b)

Figure 2. Decomposition of the environment monitoring operation (a) and its matrix representation (b)

## 4. Dec for Mobile Wireless Sensor Networks

Differently from manufacturing systems where discrete event controllers have been already successfully applied, the implementation of a DEC for a mobile WSN is indeed much more challenging. In fact a WSN has in general a variable topology, is composed of heterogeneous resources and operates in highly unstructured environments. An efficient centralized control policy has to guarantee efficient and automatic responses to changes in the operating conditions (adaptability) and to changes in the dimension of the mobile WSN (scalability). In particular, to accomplish mission goals and to guarantee optimal performances, the supervisory controller must be able to automatically reschedule missions, reassign resources to tasks and redefine the mission priorities as long as the WSN topology and the surrounding environment evolve. In the following, we will show how these issues can be efficiently tackled using our matrix-based DEC.

### 4.1 Adaptability

Adaptability is the ability of an agent team to change its behavior according to the dynamical evolution of the environment. In the following, we provide some ideas to make the system adaptable using our DEC.

*Implementation of distributed algorithms*

In related literature, coordination of teams of robots through the implementation of distributed algorithm is a common strategy, because it guarantees robustness to environment uncertainties and disturbances. Every robotic agent performs a certain task relying on local information only (e.g. keep a certain distance from the neighbors), in such a way that a predefined aggregate objective (e.g. complete environment coverage) is achieved. In the framework of the DEC, these operations can be considered as a generic (fully decentralized) mission *i* (or part of it) with a certain goal composed of simultaneous tasks. Enhanced adaptability can be obtained deciding, at the supervisor level (on the ground of the present situation), which decentralized mission (optimal sensor placement for environmental monitoring, search and recovery operations etc.) has the priority (changing $F_{ud}^{i}$) and which resources should be used (changing $F_{r}^{i}$ and $S_{r}^{i}$).

*Combining multiple plans for the same mission*

In certain circumstances, different sequences of tasks can be used to implement the same mission. A computationally efficient algorithm has been recently proposed to combine the plans together and derive one single compact matrix representation for the DEC [9]. In this way, the DEC automatically sequences the most suitable succession of tasks to achieve a certain goal, depending on the current available resources.

*Dynamic reallocation of resources*

A dynamic reallocation of the agents to missions can be performed by rearranging the '1' relative to similar resources in the matrices $F_r$ and $S_r$ when new missions (or new agents) are added. Due to the matrix representation of the mission plans, these objectives can be pursued using computationally efficient algorithms.

In figure 3 we have reported an example of reallocation of resources ($r_1$, $r_2$ and $r_3$) to the tasks of two missions. The number of tasks each resource has to perform is equal to the number of '1' in the corresponding column. After the reallocation, the resources have a more balanced workload, i.e. they perform a similar number of tasks.

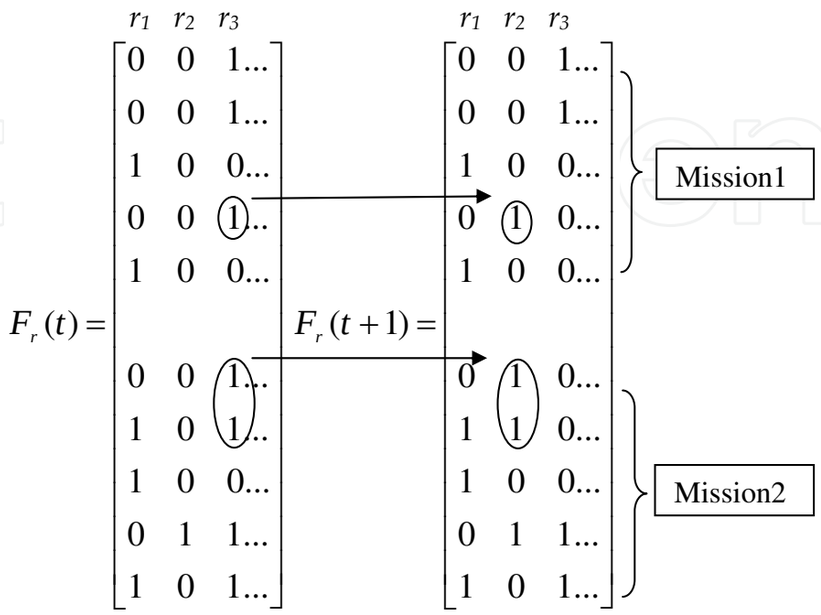



Figure 3. Reallocation of resources through matrix operations

*Priority among missions*

In order to further tailor the WSN to the present control scenario, the set of mission priority rules can be made adaptable.

For example, suppose that resource $r_1$ is shared among three different missions whose priority rank is *3, 1, 2*. After defining the conflict resolution matrix of $r_1$ for each mission $(F_{ud}^1(r_1), F_{ud}^2(r_1), F_{ud}^3(r_1))$, the overall conflict resolution matrix of $r_1$ $(F_{ud}(r_1))$ is built as

$$F_{ud}(r_1) = \begin{array}{c} \\ mission^1 \\ mission^2 \\ mission^3 \end{array} \begin{array}{c} \begin{array}{ccc} priority^1 & priority2 & priority3 \end{array} \\ \begin{bmatrix} 0 & F_{ud}^1(r1) & 0 \\ 0 & 0 & F_{ud}^2(r1) \\ F_{ud}^3(r1) & 0 & 0 \end{bmatrix} \end{array}$$

If the priority of the missions change in *2, 3, 1* then we have

$$F_{ud}(r_1) = \begin{array}{c} \\ mission^1 \\ mission^2 \\ mission^3 \end{array} \begin{array}{c} \begin{array}{ccc} priority^1 & priority2 & priority3 \end{array} \\ \begin{bmatrix} 0 & 0 & F_{ud}^1(r_1) \\ F_{ud}^2(r_1) & 0 & 0 \\ 0 & F_{ud}^3(r_1) & 0 \end{bmatrix} \end{array}$$

Thus, a change of priority results in a simple permutation of the block matrices $F_{ud}^i$ for each resource.

### 4.2 Scalability

Scalability is the ability of an agent team to reorganize its overall behavior in response to a change in the number of the agents. We can use the DEC to tackle scalability at the supervisor level, updating the matrix based representation of the missions to take into account the failure of agents as well as the adding of new ones.

If a new agent is added to the system, a new column is added in the matrices $F_r$ and $S_r^{'}$ ($S_r$ transpose). Then, dispatching algorithms (based on matrix operations) can be applied to rearrange the tasks among resources. In a similar fashion, an agent failure can be tackled rearranging the tasks among the resources so that the columns relative to the failed resources in $F_r$ and $S_r$' are null. Just to give an idea, in the following example, we describe a simple algorithm for reallocating (off-line, i.e. when no missions are in progress) resources after agent failure. The mission planning is revised in such a way that predefined back-up agents execute the tasks of the failed agents. In the matrix formulation this is equivalent to move the elements equal to one in the matrices $F_r^i$ and $S_r^i$ from the column of the failed resource to the column of the back-up resource. This can be achieved through a simple linear combination of the columns of $F_r^i$ and $S_r^i$ respectively. We have

$$F_r^{i,new} = F_r^{i,old} \cdot B^i$$
$$S_r^{i,new} = S_r^{i,old} \cdot B^i$$

where $B^i$ is a square matrix of dimension equal to the number of the resources of the system. The diagonal elements of $B^i$ $(a_j)$ are parameters which are equal to 1 if resource $r_j$ is working properly and 0 otherwise. On row $j$ the element $(j,j)$ is equal to $a_j$ and the element $(j,k)$ is equal to $1- a_j$, where $k$ is the column of matrix $F_r^i$ corresponding to the back-up resource of agent $j$ for mission $i$. If $a_j$ =0, the $jth$ column of $F_r^{i,new}$ will be null (meaning that agent $j$ is not supposed to perform any task) and the $kth$ column will have '1's in correspondence of the tasks for which resource $j$ was required. If no failure occurs, $B^i$ is the identity matrix and mission plans are not changed. Clearly, in the definition of the matrix $B^i$ we have to make sure that each back-up agent does not perform any simultaneous task with the resource they are supposed to substitute. For example, suppose we have three agents, and that, for a certain mission $i$, the resource requirement matrix and the back-up matrix $B^i$ are

$$F_r^{i,old} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad B^i = \begin{pmatrix} a_1 & 1-a_1 & 0 \\ 0 & a_2 & 0 \\ 1-a_3 & 0 & a_3 \end{pmatrix}$$

The $B^i$ matrix corresponds to the case where, in mission $i$, agent 2 is the backup of agent 1, agent 2 has no back-up and agent 1 is the back up of agent 3. If agent 1 fails ($a_1$=0) whereas agent 2 and 3 work properly ($a_2$=$a_3$=1), we get

$$B^i = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad F_r^{i,new} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

i.e., in mission $i$, agent 1 has been replaced by agent 2.
A more effective alternative to cope with agent failure is to introduce routing resources [17], which allow one to define a priori a set of multiple resource choices for certain critical tasks. The routing resources automatically assign to the task the first available resource of the corresponding set providing redundancy and robustness against agent failures. In the case of WSN, they are logical, fictitious resources and can be considered as local dispatchers. Our matrix formulation supports task routing efficiently, since there is no need to distinguish between physical and logical resources.

### 4.3 Complete Dynamical Description

It is well known that a matrix approach can be used to describe the marking transitions of a Petri Net [18] using the PN transition equation

$$m(t+1) = m(t) + (S - F') \cdot x(t) \tag{5}$$

where $S$ and $F$ are the output and input incidence matrix respectively. This equation gives a useful insight on the dynamics of discrete event systems but does not provide a complete dynamical description of DE systems.
If we observe that the vector $x$ in equation (5) is the same as in equation (1), then we may identify $x$ as the vector associated with the PN transitions and $u$, $v$, $r$, $u_d$ as associated with the places. Then it follows that [23]

$$m(t) = [u(t)', v(t)', r(t)', u_d(t)']'$$
$$S = [S_u', S_v', S_r', S_{u_d}', S_y']'$$
$$F = [F_u', F_v', F_r', F_{u_d}', F_y']'$$

Therefore, we can use equation (1) to generate the allowable firing vector to trigger transitions in equation (5). That is, the combination of the DEC (1) and the PN marking transition equation provides a complete dynamical description of the system.

In order to take into account the time durations of the tasks and the time required for resource releases, we can split *m(t)* into two vectors, one representing available resources and current finished tasks ($m_a(t)$) and the other representing the tasks in progress and idle resources ($m_p(t)$)

$$m(t) = m_a(t) + m_p(t) \tag{6}$$

This is equivalent to introducing timed places in a Petri net and to dividing each place into two parts, one relative to the pending states (task in progress, resource idle) and the other relative to the steady states (task completed and resource available). As a consequence, we can also split equation (5) into two equations

$$m_a(t+1) = m_a(t) - F' \cdot x(t) \tag{7}$$
$$m_p(t+1) = m_p(t) + S \cdot x(t) \tag{8}$$

When a transition fires a token is moved from $m_p(t)$ to $m_a(t)$ where it may be used to fire subsequent transitions. Therefore equations (1), (7) and (8) represent a complete description of the dynamical behavior of the discrete event system and can be implemented for the purposes of computer simulations using any programming language (e.g. Matlab® or C). In the case of a mobile wireless sensor network, where experiments on wide and hostile areas can be really complex and challenging, it allows one to perform extensive simulations of the control strategies and then test experimentally only those which guarantee the most promising results.

A network consisting of two mobile robots and five wireless sensors is considered as experimental scenario. Two different missions have been implemented to show the potentialities of the proposed DEC. In the first mission, after one of the sensors launches a chemical alert, the network automatically reconfigures its topology to further investigate the phenomenon. In the second mission, considering that power constraints are a very serious concern in a WSN, one of the mobile robots is employed to charge the batteries of one of the UGSs.

The procedure for implementing the supervisory control policy consists of three different steps.

First of all we define the vector of resources *r* present in the system and the tasks they can perform. In our test-bed we have two robots ($R_1$ and $R_2$), each one able of performing certain number of tasks, and five stationary sensors ($UGS_1$, $UGS_2$, $UGS_3$, $UGS_4$, $UGS_5$), each one able of performing one task (i.e. taking measurement). The resource vector is $r$=[$R_1$, $R_2$, $UGS_1$, $UGS_2$, $UGS_3$, $UGS_4$, $UGS_5$].

Then for each mission *i*, we define the vector of inputs $u^i$, of outputs $y^i$ and of tasks $v^i$, and the task sequence of each mission (refer table I and II for mission 1 and mission 2), and write down the if-then rules representing the supervisory coordination strategy to

sequence the programmed missions (table III and table IV). In the definition of the rule bases particular attention has to be devoted to the definition of consecutive tasks performed by the same resources. If the consecutive tasks are interdependent (e.g. *go to sensor 2* and *retrieve sensor2*), the corresponding resource should be released just at the end of the last of the consecutive tasks. This is the case of the task groups (*R1gS2, R1rS2*) and (*R1gS1, R1dS2, R1m*) in mission1 and of (*R1gS3* and *R1cS3*) in mission2. Instead, if the tasks are not interdependent, before starting the new consecutive task, the DEC releases the corresponding resource and makes sure that no other missions are waiting for it. If after a predetermined period of time no other missions request that resource, the previous mission can continue. This is the function of the "*Robot 1 listens for interrupts*" task in mission 1.

Finally we translate the linguistic description of the coordination rules into a more convenient matrix representation, suitable for mathematical analysis and computer implementation.

As an example, in the following we derive the matrix formulation of mission 1 from the rule-base reported in table III. We can easily write down the $F_v^1$ and $F_r^1$ matrices considering that $F_v^1(i, j)$ is '1' if task *j* is required as an immediate precursor to rule *i* and $F_r^1(i, j)$ is '1' if resource *j* is required as an immediate precursor to rule *i* (see figure 4). For example, the conditions for firing rule 6 ($x_6^1$), are the completion of tasks $R2m^1$ and $R1gS1^1$ (task 7 and 8 respectively, see table I). We therefore have two '1' entries in position (6,7) and (6,8) in matrix $F_v^1$. Resource 1 is required for the execution of the two macro-tasks defined previously, which start when rules 2 and 5 respectively are triggered. Therefore, as shown in figure 4b, we have two '1' in position (2,1) and (5,1) of $F_r^1$.

An analysis of matrix $F_r^1$ reveals that only robot 1 and robot 2 are shared resources (due to multiple '1's in the corresponding column of $F_r^1$), therefore we need to calculate just $F_{ud}^1(R1)$ and $F_{ud}^1(R2)$ (figure 4c).

| **mission1** | **notation** | **description** |
|---|---|---|
| *Input 1* | $u^1$ | *UGS1* launches chemical alert |
| *Task 1* | $S4m^1$ | *UGS4* takes measurement |
| *Task 2* | $S5m^1$ | *UGS5* takes measurement |
| *Task 3* | $R1gS2^1$ | *R1* goes to *UGS2* |
| *Task 4* | $R2gA^1$ | *R2* goes to location A |
| *Task 5* | $R1rS2^1$ | *R1* retrieves *UGS2* |
| *Task 6* | $R1lis^1$ | *R1* listens for interrupts |
| *Task 7* | $R1gS1^1$ | *R1* gores to *UGS1* |
| *Task 8* | $R2m^1$ | *R2* takes measurement |
| *Task 9* | $R1dS2^1$ | *R1* deploys *UGS2* |
| *Task 10* | $R1m^1$ | *R1* takes measurement |
| *Task 11* | $S2m^1$ | *S2* takes measurement |
| output | $y^1$ | Mission 1 completed |

Table 1. Mission1- Task sequence

| Mission2 | notation | description |
|---|---|---|
| *input* | $u^2$ | *UGS3* batteries are low |
| *Task 1* | $S1m^2$ | *UGS1* takes measurement |
| *Task 2* | $R1g\ S3^2$ | *R1* goes to *UGS3* |
| *Task 3* | $R1cS3^2$ | *R1* charges *UGS3* |
| *Task 4* | $S3m^2$ | *UGS3* takes measurement |
| *Task 5* | $R1dC^2$ | *R1* docks the charger |
| *output* | $y^2$ | Mission 2 completed |

Table 2. Mission 2-Task sequence

| Mission1-operation sequence | |
|---|---|
| *Rule1* $x_1^1$ | If $u^1$*occurs and S4 available and S5available then start* $S4m^1$ *and* $S5m^1$ |
| *Rule2* $x_2^1$ | If $S4m^1$ *and* $S5m^1$ *completed* and *R1 and R2 available* then *start* $R1gS2^1$ *and* $R2gA^1$ and *release S4 and S5* |
| *Rule3* $x_3^1$ | If $R1gs2^1$ *and* $R2gA^1$ *competed* then *start* $R1rS2^1$ and *release R2* |
| *Rule4* $x_4^1$ | If $R1rS2^1$ *completed* then *start* $R1lis^1$ and *release R1* |
| *Rule5* $x_5^1$ | If $R1lis^1$ *completed* and *R1 and R2 available* then *start* $R2m^1$ *and* $R1gS1^1$ |
| *Rule6* $x_6^1$ | If $R1gS1^1$ and $R2m^1$ *completed* then *start* $R1dS2^1$ and *release R2* |
| *Rule7* $x_7^1$ | If $R1dS2^1$ *completed* then *start* $R1m^1$ |
| *Rule8* $x_8^1$ | If $R1m^1$ *completed* and *S2 available* then *start* $S2m^1$ and *release R1* |
| *Rule9* $x_9^1$ | If $S2m^1$ *completed* then *release S2* and *terminate mission1* $y^1$ |

Table 3. Mission 1-Rule-base

The $S_v^1$ matrix (figure 5a) is built considering which tasks should be executed after a rule fires. For example, after robot1 listens for interrupts (this fires rule 5), $R_1$ should go to $UGS_1$ (*R1gS1*, task 7) and $R_2$ should start taking measurements with its onboard sensors (*R2m*, task 8). Accordingly, the elements of $S_v^1$ in position (7,5) and (8,5) are equal to 1.

The $S_r^1$ matrix (figure 5b) is built considering that $S_r^1(i,j)$ is 1 if resource *i* has to be released after rule *j* has been fired. For example, since the firing of rules 4 and 8 releases robot1 (resource1), we have entries of 1 in positions (1,4), and (1,8) in matrix $S_r^1$.

| **Mission2- operation sequence** | |
|---|---|
| *Rule1* $x_1^2$ | If $u^2$ *occurs* and *S1 available* then *start* $S1m^2$ |
| *Rule2* $x_2^2$ | If $S1m^2$ *completed* and *R1 available* then *start* $R1gS3^2$ and *release S1* |
| *Rule3* $x_3^2$ | If $R1gS3^2$ *completed* then *start* $R1cS3^2$ |
| *Rule4* $x_4^2$ | If $R1cS3^2$ *completed* and *S3 available* then *start* $S3m^2$and *release R1* |
| *Rule5* $x_5^2$ | If $S3m^2$ *completed* and *R1 available* then *start* $R1dC^2$ and *release S3* |
| *Rule6* $x_6^2$ | If $R1dC^2$ *completed* then release *R1* and *terminate mission2* $y^2$ |

Table 4. Mission 2-Rule-base

$$F_v^1 = \begin{array}{c} \\ x_1^1 \\ x_2^1 \\ x_3^1 \\ x_4^1 \\ x_5^1 \\ x_6^1 \\ x_7^1 \\ x_8^1 \\ x_9^1 \end{array} \begin{array}{c} \begin{array}{cccccccccc} S5m & R2gA & R1rS2 & R2m & S2m & & & & & \\ S4m & R1gS2 & R1lis & R1gS1 & R1dS2 & R1m & & & & \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad \text{(a)}$$

$$F_r^1 = \begin{array}{c} \\ x_1^1 \\ x_2^1 \\ x_3^1 \\ x_4^1 \\ x_5^1 \\ x_6^1 \\ x_7^1 \\ x_8^1 \\ x_9^1 \end{array} \begin{array}{c} \begin{array}{ccccccc} R1 & R2 & S1 & S2 & S3 & S4 & S5 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad \text{(b)}$$

$$F_{ud}^1(R1) = F_{ud}^1(R2) = \begin{array}{c} \\ x_1^1 \\ x_2^1 \\ x_3^1 \\ x_4^1 \\ x_5^1 \\ x_6^1 \\ x_7^1 \\ x_8^1 \\ x_9^1 \end{array} \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{(c)}$$

Figure 4. Mission1- job sequencing matrix $F_v^1$ (a), resource requirement matrix $F_r^1$ (b) and conflict resolution matrix $F_{ud}^1$ (c)

$$S_v^1 = \begin{array}{c} \\ S4m \\ S5m \\ R1gS2 \\ R2gX \\ R1lis \\ R1rS2 \\ R1gS1 \\ R2m \\ R1dS2 \\ S2m \\ R1m \end{array} \begin{array}{c} \begin{array}{ccccccccc} x_1^1 & x_2^1 & x_3^1 & x_4^1 & x_5^1 & x_6^1 & x_7^1 & x_8^1 & x_9^1 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$ (a)

$$S_r^1 = \begin{array}{c} \\ R1 \\ R2 \\ S1 \\ S2 \\ S3 \\ S4 \\ S5 \end{array} \begin{array}{c} \begin{array}{ccccccccc} x_1^1 & x_2^1 & x_3^1 & x_4^1 & x_5^1 & x_6^1 & x_7^1 & x_8^1 & x_9^1 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$ (b)

Figure 5. Mission1- Task start matrix $S_v^1$ (a) and resource release matrix $S_r^1$ (b)

$$F_v^2 = \begin{array}{c} \text{S1m R1cS3 R1dC} \\ \text{R1gS3 S3m} \\ x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \\ x_5^2 \\ x_6^2 \end{array} \begin{array}{c} \\ \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$ a)

$$F_r^2 = \begin{array}{c} \text{R1 R2 S1 S2 S3 S4 S5} \\ x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \\ x_5^2 \\ x_6^2 \end{array} \begin{array}{c} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$ b)

$$F_{ud}^2(R1) = \begin{array}{c} \\ x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \\ x_5^2 \\ x_6^2 \end{array} \begin{array}{c} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array}$$ c)

Figure 6. Mission2- Task sequencing matrix $F_v^2$ (a), resource requirement matrix $F_r^2$ (b) and conflict resolution matrix $F_{ud}^2(R1)$ (c)

$$S_v^2 = \begin{array}{c} \\ S1m \\ R1gS3 \\ R1cS3 \\ S3m \\ R1dC \end{array} \begin{array}{c} \begin{array}{cccccc} x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 & x_6^2 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$ a)

$$S_r^2 = \begin{array}{c} \\ R1 \\ R2 \\ S1 \\ S2 \\ S3 \\ S4 \\ S5 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 & x_6^2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$ b)

Figure 7. Mission2- Task start matrix $S_v^2$ (a) and resource release matrix $S_r^2$ (b)

$$F_v = \begin{matrix} x^1 \\ x^2 \end{matrix}\begin{pmatrix} F_v^1 & 0 \\ 0 & F_v^2 \end{pmatrix} \quad F_r = \begin{matrix} x^1 \\ x^2 \end{matrix}\begin{pmatrix} F_r^1 \\ F_r^2 \end{pmatrix}$$

$$S_v = \begin{matrix} v^1 \\ v^2 \end{matrix}\begin{pmatrix} S_v^1 & 0 \\ 0 & S_v^2 \end{pmatrix} \quad S_r = r\begin{pmatrix} S_r^1 & S_r^2 \end{pmatrix}$$

$$F_{ud} = \begin{bmatrix} \overbrace{0}^{R1} & \overbrace{F_{ud}^1(R_1)}^{R1} & \overbrace{F_{ud}^1(R_2)}^{R2} \\ F_{ud}^2(R_1) & 0 & 0 \end{bmatrix}$$

Figure 8. Overall monitoring operation- Matrix formulation

In the same way, the set of matrices relative to mission 2 can be built (figure 6 and 7). Then the matrix formulation of the overall monitoring operation is easily obtained by stacking together the matrices of mission 1 and mission 2 (figure 8). Also, as stated before, this matrix formulation can be used to represent a Petri Net. Therefore, for the sake of clarity, in figure 9 we have reported the Petri Net which corresponds to the matrix description of the two implemented missions.

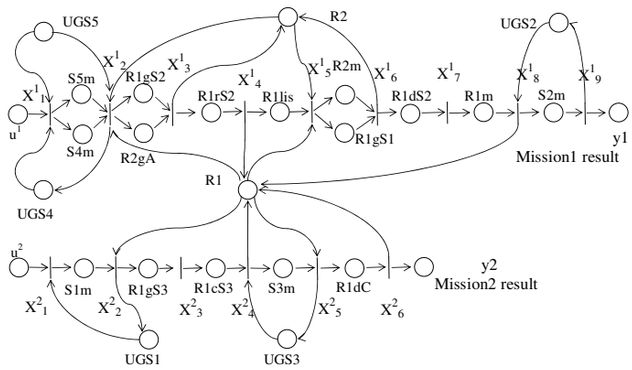



Figure 9. Petri net representation of the implemented missions, corresponding to the matrixes Fv, Fr, Sv, Sr

## 5. Simulation Results

Simulation of this system for the given missions using equations 1, 7 and 8 can be done using Matlab®. Figure 10, shows the utilization time trace of the resources and the execution time trace of the tasks. In these time traces, busy resources and tasks not in progress are denoted by low level, whereas idle resources and tasks in progress are denoted by high level. At time instants 5s and 15s the events $u^2$ ($UGS_3$ has low batteries) and $u^1$ ($UGS_1$ detects a chemical agent) occur respectively and mission 2 and mission 1 are triggered. Since mission 1 has a lower priority, after the first two tasks are completed ($S4m^1$ and $S5m^1$), the mission is temporarily interrupted because resource $R_1$ is assigned to mission2 for the execution of task *R1gS3*. At time instant 75 s, $R_1$, after listening for interrupts (task 6, mission1), is reassigned to mission 2 which can then come to an end with the execution of task $R1dC^2$. Finally, mission 1 can be successfully terminated. From the time traces of figure 10, it is interesting to note that whenever possible, the missions are executed simultaneously, and the shared resources are alternatively assigned to the two missions.

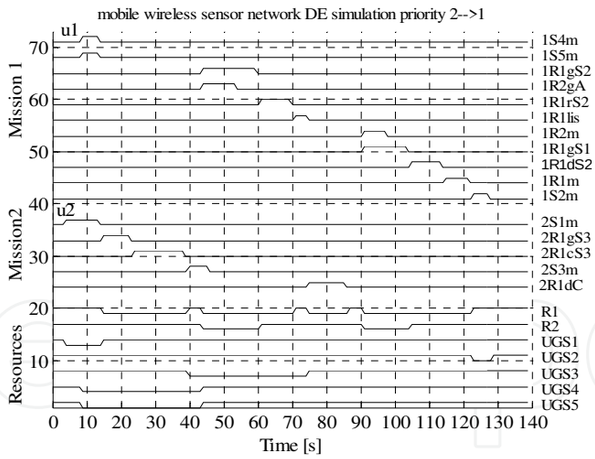



Figure 10. Event time trace simulation results of the WSN

If we change the priority order of the two missions, permuting the block matrices $F_{ud}^{1}$ and $F_{ud}^{2}$, we get the utilization time trace of figure 11. Mission 2 is executed more fragmentarily, since resource 1 is preferentially assigned to mission1. To further increase the priority of mission1 we could eliminate the listening task *R1lis*, reducing the possibilities of mission 2 to get resource 1 while mission1 is active.

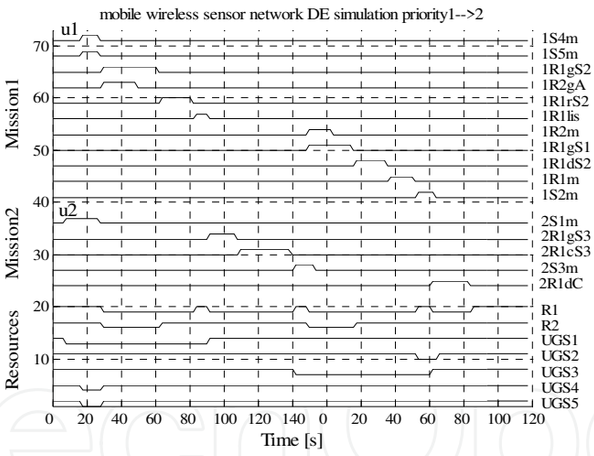



Figure 11. Event time trace simulation results of the WSN after changing the priority order between the two missions

## 6. Experimental Results

After performing extensive simulations, we can implement the control system directly on the WSN test-bed. Figure 12 shows the actual experimental utilization time trace of the agents, assigning higher priority to mission 2. Notice that the time duration of the real WSN runs in terms of discrete-event intervals, whereas the simulation results shown in figure 10 is in terms of time. It is interesting to note the similarity and fidelity of the dispatching sequences in both the simulation and experimental cases. This is a key result since it shows that the DEC allows one to perform a "simulate and experiment" approach for a WSN, with noticeable benefits in terms of cost, time and performance.

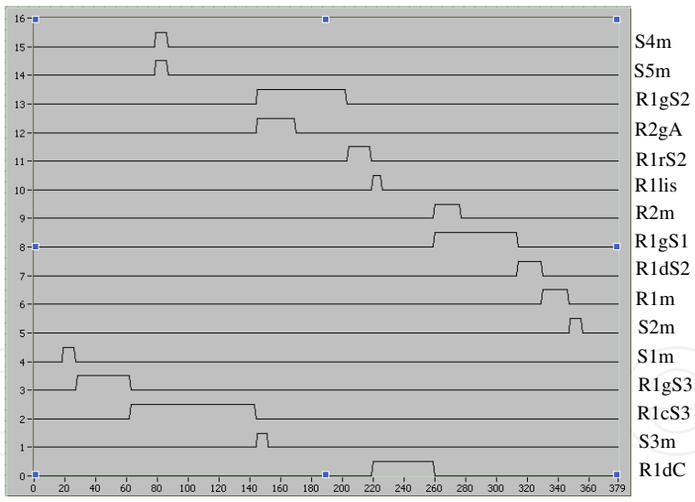



Figure 12. Utilization time trace of the WSN- Experimental results

In figure 13, for the sake of clarity, we have depicted the configuration of the agents in the environment, showing how the topology of the network evolves along time (13a) and finally reaches the configuration shown in figure (13b). In figure 14 and 15, the same results are proposed showing a panoramic view of the mobile WSN.

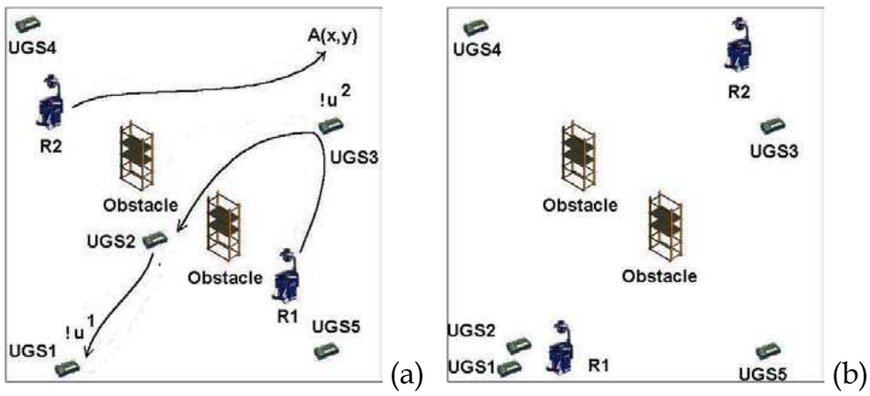



Figure 13. Starting configuration and trajectory followed by the mobile robots (a) - Final configuration after execution of mission 1 and 2 (b)

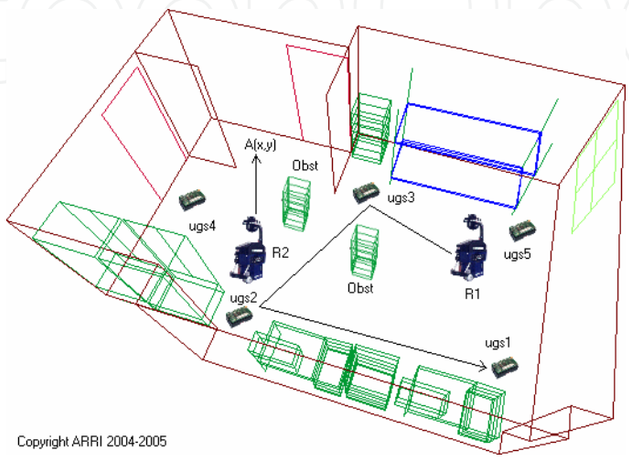



Figure 14. Panoramic view of the configuration of the mobile WSN during real-world experiments

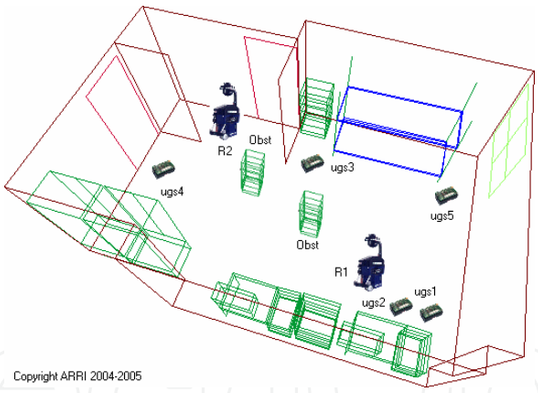



Figure 15. Panoramic view of the final configuration of the mobile WSN

*Reassignment of resources*

Since the topology of a mobile WSN evolves with time, it is necessary to adapt the mission planning to the changed operating conditions. We want to show now how the matrix formulation of the DEC allows one to easily reconfigure the dispatching rules of the system. Suppose that robot 2 has the same functionalities of robot1 and that mission 2 is triggered after the completion of mission 1 (figure 16a). Since at the end of mission1, robot 2 is actually closer to UGS3, we might consider employing robot2 for mission 2 instead of robot 1 (figure 16b). This change in mission planning would result in simple matrix operations. The '1's relative to the tasks of mission2 are shifted from column 1 to column 2 of matrix $F_r^2$ and from row 1 to row 2 of matrix $S_r^2$ (i.e. tasks are shifted from resource1 to resource2). $F_{ud}$ is changed redefining the conflict resolution matrix of each resource for each mission (in this case we have to define $F_{ud}^2(R_2)$ whereas $F_{ud}^2(R_1)$becomes null) and accordingly reassembling the block matrices in $F_{ud}$ (figure 17).

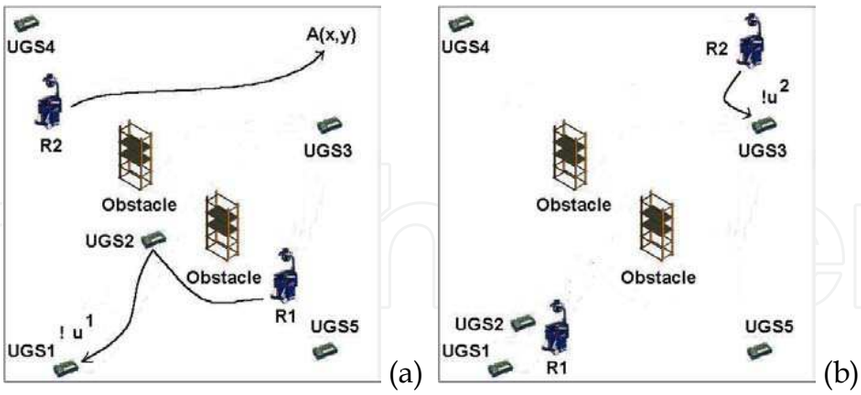



Figure 16. Execution of mission1 (a), change of plans and execution of mission2 b)

$$F_{ud} = \begin{bmatrix} \overbrace{0}^{R1} & F_{ud}^1(R_1) & \overbrace{F_{ud}^1(R_2)}^{R2} \\ F_{ud}^2(R_1) & 0 & 0 \end{bmatrix} \text{a)}$$

$$F_{ud} = \begin{bmatrix} \overbrace{F_{ud}^1(R_1)}^{R1} & 0 & \overbrace{F_{ud}^1(R_2)}^{R2} \\ 0 & F_{ud}^2(R_2) & 0 \end{bmatrix} \text{b)}$$

Figure 17. Conflict resolution matrix when resource1 (a) or resource 2 (b) are assigned to execute mission2

## 7. Conclusions

In this chapter we have presented a discrete-event coordination scheme for sensor networks composed of both mobile and stationary nodes. This architecture supports high-level planning for multiple heterogeneous agents with multiple concurrent goals in dynamic environment. The proposed formulation of the DEC represents a complete dynamical description that allows efficient computer simulation of the WSN prior to implementing a given DE coordination scheme on the actual system. The similarity between simulation and experimental results shows the effectiveness of the DEC for simulation analysis. The obtained results also prove the striking potentialities of the matrix formulation of the DEC, namely: straightforward implementation of missions on the ground of intuitive linguistic descriptions; possibility to tackle adaptability and scalability issues at a centralized level using simple matrix operations; guaranteed performances, since the DEC is a mathematical framework which constraints the behaviour of the single agents in a predictable way. Future research will be devoted to the development of high-level decision making algorithms for the dynamic updates of the matrices of the DEC, in order to automatically reformulate the missions on-line according to the current topology of the network and the current perception of the environment.

## 8. References


Akyldiz I., Su W., Sankarasubramaniam Y, Cayirci E., "A survey on sensor networks", IEEE Communications magazine, August 2002

Balch T., Hybinette M., "Social potentials for scalable multi-robot formations", Proceedings of the International Conference of Robotics and Automation, April 2000

Balch T., Arkin R., "Behavior-based formation control for multirobot teams", IEEE Transactions on Robotics and Automation, vol. 14, no.6, December 1998

Burgard, W.; Moors, M.; Fox, D.; Simmons, R.; Thrun, S.; "Collaborative multi-robot exploration", Proceedings of the IEEE International Conference on Robotics and Automation, 2000, vol. 1 , 24-28 April 2000 Pages:476 - 481

Butler Z., Rus D., "Event-based motion control for mobile-sensor network", IEEE Transactions. on Pervasive Computing, vol.2 issue 4, October-December 2003

Christensen T., Noergaard M., Madsen C., Hoover A., "Sensor networked mobile robotics", Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, June 2000

Cortes J., Martinez S., Karatas T., Bullo F., "Coverage control for mobile sensing network", IEEE Trans. on Robotics and Automation, vol.20, no.2, April 2004

Gordon-Spears A., Kiriakidis K., "Reconfigurable robot teams: modeling and supervisory control", IEEE Transactions on control system technology, vol. 12, no. 5, September 2004

Harris B., Lewis F., Cook D., "Machine planning for manufacturing: dynamic resource allocation and on-line supervisory control", Journal of Intelligent Manufacturing, pp. 413-430, vol. 9, 1998

Howard, A.; Mataric, M.J.; Sukhatme, G.S.; "An incremental deployment algorithm for mobile robot teams", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System, vol.30, Pages:2849 – 2854, October 2002

King J., Pretty R., Gosine R., "Coordinated execution of tasks in multiagent environment", IEEE transactions on Systems, man and cybernetics- Part A: Systems and Humans, vol. 33, no.5, September 2003

Kusiak A. Intelligent scheduling of automated machining systems. In Intelligent design and Manufacturing. A. Kusiak (ed.) Wiley, New York (1992)

Lee J., Hashimoto H, "Controlling mobile robots in distributed intelligent sensor network", IEEE Transactions on Industrial Electronics, vol.50, no.5, October 2003

Lewis F., "Wireless sensor networks", Smart environments: technologies, protocols, and applications, ed. D. J. Cook and S. K. Das, John Wiley, New York, 2004.

McMickell M., Goodwine B., Montestruque L., "MICAbot: a robotic platform for large-scale distributed robotics", Proceedings of the International conference of robotics and automation, September 2003

Mireles J., Lewis F., "Intelligent material handling: development and implementation of a matrix-based discrete event controller IEEE Transactions on Industrial Electronics, , vol. 48 , Issue: 6 , Dec. 2001 Pages:1087 – 1097

Mireles J., Lewis F., "Deadlock analysis and routing on free-choice multipart reentrant flow lines using a matrix-based discrete event controller" Proceedings of the IEEE International conference on Decision and Control, December 2002

Murata, T. "Petri nets: properties, analysis and applications." Proceedings of the IEEE, vol.77, no.4, April 1989, pp.541-80

Petriu E., Whalen T.,Abielmona R., Stewart A., "Robotic sensor agents: a new generation of intelligent agents for complex environment monitoring", IEEE Magazine on Instrumentation and Measurement, vol.7 issue 3, September 2004

Saridis G., "Intelligent robotic control", IEEE Transactions on Robotics & Automation, vol. 28, no.5, May 1983

Sibley G., Rahimi M., Sukhatme G., "Robomote: a tiny mobile platform for large-scale ad-hoc sensor networks", Proceedings of the International conference of robotics and automation, May 2002

Steward D. V., "The design structure system: a method for managing the design of complex systems", IEEE Transactions on Engineering Management, pp. 45-54, Aug. 1981

Tacconi D., Lewis F., "A new matrix model for discrete event systems: application to simulation", IEEE Control System Magazine

Tilak S., Abu-Ghazaleh N., Heinzelman W., "A taxonomy of wireless micro-sensor network models," ACM Mobile Computing and Communications Review, Vol. 6, No. 2,pp. 28-36, 2002

Wu H., Siegel M., Stiefelhagen R., Yang J., "Sensor fusion using Dempster-Shafer theory", Proceedings of IEEE Instrumentation and Measurement Technology Conference,May


**Cutting Edge Robotics**

Edited by Vedran Kordic, Aleksandar Lazinica and Munir Merdan

ISBN 3-86611-038-3

Hard cover, 784 pages

**Publisher** Pro Literatur Verlag, Germany

**Published online** 01, July, 2005

**Published in print edition** July, 2005

This book is the result of inspirations and contributions from many researchers worldwide. It presents a collection of wide range research results of robotics scientific community. Various aspects of current research in robotics area are explored and discussed. The book begins with researches in robot modelling & design, in which different approaches in kinematical, dynamical and other design issues of mobile robots are discussed. Second chapter deals with various sensor systems, but the major part of the chapter is devoted to robotic vision systems. Chapter III is devoted to robot navigation and presents different navigation architectures. The chapter IV is devoted to research on adaptive and learning systems in mobile robots area. The chapter V speaks about different application areas of multi-robot systems. Other emerging field is discussed in chapter VI - the human- robot interaction. Chapter VII gives a great tutorial on legged robot systems and one research overview on design of a humanoid robot.The different examples of service robots are showed in chapter VIII. Chapter IX is oriented to industrial robots, i.e. robot manipulators. Different mechatronic systems oriented on robotics are explored in the last chapter of the book.



**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen