Received October 21, 2017, accepted November 26, 2017, date of publication December 8, 2017,
date of current version February 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2781459

Multiagent Architecture for Distributed Adaptive Scheduling of Reconfigurable Real-Time Tasks With Energy Harvesting Constraints

WIEM HOUSSEYNI[1,2,3], (Student Member, IEEE), OLFA MOSBAHI[1,2], MOHAMED KHALGUI[1,2], ZHIWU LI[4,5], (Fellow, IEEE), AND LI YIN[4,5]

[1]School of Electrical and Information Engineering, Jinan University (Zhuhai Campus), Zhuhai 519070, China
[2]National Institute of Applied Sciences and Technology, University of Carthage, Tunis 1080, Tunisia
[3]Tunisia Polytechnic School, University of Carthage, Tunis 1080, Tunisia
[4]Institute of Systems Engineering, Macau University of Science and Technology, Macau 999078, China
[5]School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China

Corresponding authors: Mohamed Khalgui (khalgui.mohamed@gmail.com) and Zhiwu Li (zhwli@xidian.edu.cn)

This work was supported by the Science and Technology Development Fund, MSAR, under Grant 078/2015/A3.

ABSTRACT This paper presents new challenges for the real-time scheduling of distributed reconfigurable embedded systems powered by a renewable energy. Reconfigurable computing systems have to deal with unpredictable events from the environment, such as activation of new tasks and hardware or software failures, by adapting the task allocation and scheduling in order to maintain the system feasibility and performance. The proposed approach is based on an intelligent multiagent distributed architecture composed of: 1) a global agent "coordinator" associated with the whole distributed system and 2) four local agents, such as supervisor, scheduler, battery manager, and reconfiguration manager, belong to each subsystem. The efficiency and completeness of the reconfiguration adaptative strategy is proved as all possible reconfiguration forms are considered to guarantee a feasible system with a graceful quality of service. Two communication protocols, such as an intra-subsystem communication protocol and an inter-subsystem communication protocol, are proposed to ensure the effectiveness of the proposed reconfiguration strategy. Extensive simulations show the effectiveness of the proposed intelligent multiagent distributed architecture in terms of the number of exchanged messages, deadline success ratio, and the energy consumption.

INDEX TERMS Distributed embedded system, energy harvesting, multiagent, reconfiguration, real-time scheduling.

NOMENCLATURE

NREEHS	Networked Reconfigurable Embedded Energy Harvesting System.
DAG	Directed Acyclic Graph.
MAS	Multi-Agent System.
DMH	Decomposition Migration Heuristic.
DH	Degradation Heuristic.
RH	Removal Heuristic.
QoS	Quality of Service.
EDF	Earliest Deadline First.
SDA	Semi-Dynamic Algorithm.
EH-EDF	Energy-Harvesting Earliest Deadline First.
WSNs	Wireless Sensor Networks.
DVS	Dynamic Voltage Scaling.
DMS	Dynamic Modulation Scaling.
CPU	Central Processing Unit.
DVFS	Dynamic Voltage Scaling Selection.
TSM	Task slack Management.
EDH	Earliest Deadline Harvesting.
UTB	Utilization Based.
ILP	Integer Linear Programming.
DRDECS	Distributed Reconfigurable Discrete Event Control System.
WCET	Worst Case Execution Time.
WCEC	Worst Case Energy Consumption.
R	Set of all simultaneous reconfiguration requests.
$R(t)$	Set of all simultaneous requests received at time t.
B_{Cr}	Charging rate B_{Cr} of the battery calculated as the difference of the regenerated energy from the harvesting device E_{h_j} and the consumed energy by the embedded system E_{c_j}.

Sys	Networked reconfigurable real-time embedded system.
Σ	Set of m networked subsystems in Sys.
σ_j	Subsystem $\sigma_j, j \in \{1, .., m\}$.
P_j	Processor $P_j, j \in \{1, .., m\}$.
B_j	Battery B_j associated to processor P_j in σ_j, $j \in \{1, .., m\}$.
E_{d_j}	Energy demand of tasks set ψ_j in time interval calculated by $\sum_{i=1}^{n} En_i$.
$P_{h_j}(t)$	Instantaneous power of harvesting energy of the battery associated to processor P_j.
$E_{h_j}(t_1, t_2)$	Harvested energy in time interval $[t_1, t_2]$ in battery B_j.
$E_{B_j}\ (t)$	Energy available in $B_j, j \in \{1, .., m\}$ at time t.
$P_{c_j}(t)$	Instantaneous power consumption of processor $P_j, j \in \{1, .., m\}$ expressed in watts.
$E_{c_j}(t\text{-}1, t)$	Energy required between t-1, and t for the execution of jobs related to the tasks assigned to processor P_j.
B_{C_j}	Capacity of battery B_j expressed in units of energy.
N	Number of tasks that can implement Sys.
Γ	Software platform to handle N tasks that can implement Sys.
ψ	Set of N tasks to be executed in Sys.
ψ_j	Set of n tasks assigned to processor P_j, j $\in \{1, .., m\}$.
τ_i	i-th task in ψ, $i \in \{1, .., N\}$.
C_i	Worst case execution time (WCET) of task τ_i, $i \in \{1, .., N\}$.
T_i	Period of task τ_i, $i \in \{1, .., N\}$.
D_i	Relative deadline of task τ_i, $i \in \{1, .., N\}$.
En_i	Worst case energy consumption of task τ_i, $i \in \{1, .., N\}$.
U_{τ_i}	Utilization factor of task τ_i, $U_{\tau_i} = \frac{C_i}{T_i}$.
d_{c_i}	Emergency execution level of task τ_i, $i \in \{1, .., N\}$.
γ_i	Density of task τ_i, $i \in \{1, .., N\}$.
G_i	Graph corresponding to task τ_i, $i \in \{1, .., N\}$.
n_i	Number of all subtasks in graph G_i.
$\tau_{i,k}$	k-th subtask in G_i, $i \in \{1, .., N\}$, $k \in \{1, .., n_i\}$.
V_i	Set of nodes in G_i that presents the subtasks of task τ_i, $i \in \{1, .., N\}$.
E_i	Set of directed edges in G_i, $i \in \{1, .., N\}$.
$\mathcal{F}_i$	Set of all possible execution flows of G_i, $i \in \{1, .., N\}$.
m_i	Number of all possible execution flows in graph G_i.
$F_{i,l}$	l-th execution flow of graph G_i, $i \in \{1, .., N\}$, $l \in \{1, .., m_i\}$.
$V_{i,l}$	Set of nodes associated to execution flow $F_{i,l}$, $i \in \{1, .., N\}$, $l \in \{1, .., m_i\}$.
$E_{i,l}$	Set of edges associated to execution flow $F_{i,l}$, $i \in \{1, .., N\}$, $l \in \{1, .., m_i\}$.
$C_{F_{i,l}}$	WCET of $F_{i,l}$, $i \in \{1, .., N\}$, $l \in \{1, .., m_i\}$.
F_i^c	Critical execution flow in G_i, $i \in \{1, .., N\}$.
Λ	Distributed multi-agent architecture.
C_{Sys}	Coordinator agent.
A_{Sup}	Supervisor agent.
A_{Sched}	Scheduler agent.
A_{Reconf}	Reconfiguration manager agent.
A_B	Battery manager agent.
U_{P_j}	Utilization factor of processor P_j, $j \in \{1, .., m\}$.
U_{e_j}	Energy load of tasks set ψ_j assigned to processor P_j.
$U_{m,k}$	Utilization processor factor with (m,k)-firm requirements.

I. INTRODUCTION

Distributed embedded systems have drawn substantial interest and the number of their application domains is varying and increasing ranging from all objects of our daily life to industry production. Most of these applications are real-time constrained where the timing behavior is of paramount importance and is a part of their performance or correctness criteria. The correctness of real-time systems depends not only upon their accurate results, but also upon the imposed deadlines in which the results are delivered [1]. An increasing trend in embedded systems is towards implementing multiple functionalities with different levels of criticality upon a common platform. The degree of criticality is defined as the functional and operational importance of a task. The designer of the system defines (manually) the criticality degree of each task in the system. Some of these functionalities are hard real-time where the treatments must absolutely respect all time constraints, only one failure to meet deadlines can have serious consequences and the task is considered to be critical, whereas others may be soft real-time where failure to respect temporal constraints will have no catastrophic effect on the controlled environment and the task can be considered to be non critical [2].

A major constraint in the design of real-time embedded systems today is the battery lifetime. Obviously, these batteries have limited energy storage capacity and therefore, finite useful life. As a result, there is tremendous interest in the energy harvesting technology that emerges as a promising alternative to enhance the system's lifetime and to achieve energy autonomy [3]–[5]. Several technologies are proposed for environmental energy harvesting, such as solar cells, piezoelectric vibration generators, and energy drawn from thermal and acoustic noise [6], [7]. In particular, solar energy harvesting provides relatively higher power densities which make it increasingly deployed to design the new generation of embedded devices.

Reconfigurable computing systems have the potential to greatly satisfy the simultaneous demand for application performance and flexibility [8]. Reconfigurable computing systems have pervaded nearly all research work from both academia and industry [9]–[11]. Reconfiguration is usually performed in response to both user requirements and dynamic changes in its environment such as unpredictable activation of new tasks, and hardware or software failures. Some examples of reconfigurable systems are multirobot systems [12] and wireless sensor networks [13]. At run-time, the occurrence of unpredictable task's activation makes the static schedule no longer optimal and may evolve the system towards an infeasible state due to energy and processing overloads. Thereafter, some existing or added tasks may violate deadlines. The system has to dynamically adjust and adapt the task allocation and scheduling in order to cope with unpredictable new task's arrival. Classical scheduling approaches mostly ignore the dynamic nature of the systems. Multiagent systems (MAS) appear as a promising approach for automatic reconfiguration in distributed systems such as sensor networks [14]–[18].

In this paper, the proposed contribution exposes new challenges for the development of a networked reconfigurable embedded energy harvesting system (NREEHS). The system can be reconfigured at run-time where additional tasks may arrive on a given processor. We assume that the execution frequency of reconfiguration scenarios is lower than that of system tasks. It means that the periods of system tasks are in seconds or minutes whereas the reconfigurations are in hours or days. Therefore, the processing time and energy overhead involved by any reconfiguration are considered to be negligible than those involved by the system tasks. In addition, the migration overhead is assumed to be neglected since the execution code of tasks resides in all the processors from initialization.

Initially, the computing distributed system is assumed to be schedulable. In other terms, every task initially assigned to a given processor is guaranteed to meet its timing requirements. At any instant, external unpredictable new task's activation may occur on a given processor. The latter may become faulty due to processor overload and/or energy starvation. In what follows, we consider that a processor is faulty if the schedulability cannot be guaranteed, i.e., deadline missing may occur. The reconfiguration is motivated by the unschedulability which appears because of processor overload and/or energy starvation on a processor.

The objective of the reconfiguration process is to optimize the global quality of service (QoS) measured in terms of deadline success ratio and the degree of criticality. This paper proposes a solution with three successive adaptation strategies to be applied in a hierarchical step by step order: (i) decomposition and migration which decomposes software tasks and migrates their branches from a faulty processor to a non-faulty one, (ii) degradation heuristic that modifies the scheduling mode, and (iii) removal heuristic which deletes branches or tasks. We propose an efficient protocol for NREEHS deploying an MAS that comprises a global agent denoted as "Coordinator" for coordination between networked subsystems, and four local agents: supervisor, scheduler, battery manager, and reconfiguration manager belonging to each subsystem.

The proposed solution is a complete methodology that deals with all possible reconfiguration forms to guarantee a feasible system with a graceful QoS. Simulation results are presented to demonstrate the effectiveness of the proposed multiagent distributed architecture and the three proposed reconfiguration scenarios measured in terms of deadline miss ratio and energy savings. Moreover, the effectiveness of the proposed communication protocols is evaluated in terms of the number of exchanged messages.

The remainder of the paper is structured as follows. Section II summarizes the state of the art that deals with (i) the real-time scheduling in energy harvesting based embedded systems, and (ii) multiagent architectures for reconfigurable embedded systems. Section III gives a formal presentation of the NREEHS context. Section IV details the proposed solution for NREEHS. Section V describes a new multiagent architecture dedicated to networked reconfigurable energy harvesting systems. The results of the conducted experiment to evaluate the proposed solutions are reported in section VI. Finally, the paper is concluded with a summary of the contributions and the presentation of the future work in Section VII.

II. STATE OF THE ART

This section, presents a state of the art dealing first with energy harvesting oriented architectures, and then with the scheduling in reconfigurable embedded systems based on distributed multiagent architectures.

A. REAL-TIME SCHEDULING IN ENERGY HARVESTING BASED EMBEDDED SYSTEMS

Uniprocessor real-time scheduling for energy harvesting based systems has been the focus of many works from one decade only, including [19]–[21]. In [20], the earliest deadline-harvesting (ED-H) scheduling algorithm is proved to be optimal. ED-H is an extension of the EDF (Earliest Deadline First) scheduler with energy awareness capabilities. By using the notions of slack-time and slack-energy, ED-H not only makes scheduling decisions based on the relative urgency of the deadline constrained tasks, it also provides dynamic power management capabilities. The idea behind ED-H is to order the tasks according to the EDF rule. In contrast to EDF, tasks are not systematically executed as soon as possible due to possible energy shortage. The difference between ED-H and classical EDF is to decide when to execute a task and when to let the processor be idle. Before authorizing any task to execute, the energy level of the storage must be sufficient such that all future occurring tasks execute timely with no energy starvation, considering both their energy consumption and the replenishment rate of the storage unit. Recently, a research work has been done on the multiprocessor case. The work in [22], presents an energy management approach based on epoch in performance-constrained WSNs (Wireless Sensor Networks) that utilize energy harvesting. The proposed approach utilizes two energy management techniques, Dynamic Voltage Scaling (DVS) and Dynamic Modulation Scaling (DMS). In order to satisfy performance requirements, the approach adjusts radio modulation levels and CPU frequencies. Several simulations show that the proposed algorithms achieve significantly higher performance than a baseline approach under both normal and emergency situations. The work in [23] presents a task mapping, scheduling and power management method for multicore real-time embedded systems with energy harvesting. The proposed method is based on the concept of task CPU utilization, which is defined as the worst-case task execution time divided by its period. This method combines with a new dynamic voltage and frequency selection (DVFS) algorithm with energy harvesting awareness and task slack management (TSM), forms the proposed utilization based (UTB) algorithm. Moreover, UTB was extends to support multicore platforms by allocating a subset of tasks to each core and executing the single-core UTB algorithm separately on each core. It introduces a deadline-aware scheduling algorithm with energy migration strategies specifically designed to manage distributed super-capacitors in sensor networks.

B. RECONFIGURABLE REAL-TIME EMBEDDED SYSTEM BASED ON MULTIAGENT DISTRIBUTED ARCHITECTURE

Several research works have been done in recent years, focusing on reconfigurable embedded systems [24], [25]. Recently, the multiagent distributed architectures have attracted considerable attention from the community of reconfigurable embedded technologies [26], [27]. The work in [27] reports a decentralized supervision policy for a Petri net through collaboration between a coordinator and subnet controllers. Then, a coordinator is selected from subnet controllers by using integer linear programming (ILP) to reduce the communication cost. The research in [28] develops a new coordination method for a distributed reconfigurable discrete event control system (DRDECS) where each subsystem is modeled by a reconfigurable timed net condition/event system. The paper develops a virtual coordinator and a communication protocol in order to treat all concurrent reconfiguration requirements using judgment matrices while the exchanged messages are reduced. However, all of these works cope with reconfigurable multiagent distributed embedded systems but no one among them deals with energy requirements. Power consumption and energy requirements for the reconfigurable distributed embedded systems have received much less attention. The research in [29] proposes a multiagent based architecture consisting of: (i) a master agent defined for the whole control in the distributed multiprocessor architecture, and (ii) a slave agent assigned to each processor for the local control of energy and memory. In addition, it defines a communication protocol between the different proposed agents to guarantee the respect of memory capacity while minimizing the energy consumption. The research work in [30] deals with a software-agent-based architecture that provides three virtual processors and four solutions to reconfigure the system at run-time in order to reduce the system's power consumption.

To the best of our knowledge, most of the previous studies consider a centralized architecture where the whole system depends on the decision of the coordinator agent. As far as we know, the intelligent multiagent distributed architecture for networked reconfigurable embedded systems with energy harvesting requirements where tasks are represented by DAG (Directed Acyclic Graph) is reported for the first time in this research. The main advantages of our multiagent distributed architecture and the two proposed communication protocols are: i) by performing the proposed coordination strategy, the exchanged messages among agents are reduced significantly, and ii) by applying the new solution with four adaptative strategies, the percentage of satisfied deadlines and energy saving are increased.

III. FORMALIZATION OF NETWORKED RECONFIGURABLE SYSTEM

In this section we formally describe the system model of NREEHS composed of multiple networked subsystems. Each subsystem consists of one processor and one rechargeable energy storage unit with limited capacity supplied by a renewable energy source. We assume that the system is composed of a set of identical processors in which the preemption and migration of tasks are authorized. Each subsystem performs a set of periodic and independent tasks. The system can be reconfigured at run-time where additional tasks may arrive on a given processor.

This paper proposes to address the scheduling problem in NREEHS through a multiagent distributed architecture. The agents are categorized into two categories: i) a global agent denoted as "Coordinator" for coordination between networked subsystems, and ii) four local agents: supervisor, scheduler, battery manager, and reconfiguration manager. Every subsystem has local agents in order to maintain its feasibility whenever possible after any external reconfiguration scenario. Two communication protocols are proposed: i) an intra-subsystem communication protocol for communication between agents inside each subsystem, and ii) an inter-subsystem communication protocol for communication between subsystems. Fig. 1 shows the overview of the NREEHS considered in this paper which consists of networked reconfigurable real-time subsystems, a middleware based on the proposed multiagent architecture, and a set of real-time periodic and independent tasks.

A. HARDWARE ARCHITECTURE

Let $Sys = (\Sigma, \Gamma, \Lambda)$ be an NREEHS composed of m networked subsystems, where $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$ is the set of subsystems, Γ is the software platform, and Λ is a distributed multiagent architecture. Each subsystem $\sigma_j \in \Sigma$, $j \in \{1, .., m\}$, is composed of: (i) processor P_j that performs a set ψ_j of tasks where the preemption and migration of tasks are authorized; and (ii) a rechargeable energy storage with limited capacity B_j.

B. ENERGY CONSIDERATIONS

The energy produced by the source is not considered controllable. Let $P_{h_j}(t)$ be the instantaneous power of harvesting

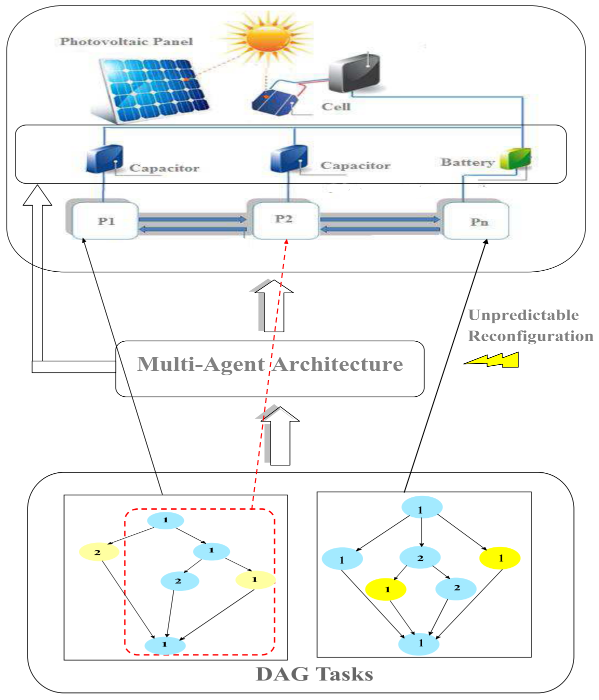

FIGURE 1. Proposed system model for NREEHS.

energy of the battery associated with processor P_j that incorporates all losses. The harvested energy in the interval time $[t_1, t_2]$ in the battery B_j denoted by $E_{h_j}(t_1, t_2)$ is calculated as follows:

$$E_{h_j}(t_1, t_2) = \int_{t_1}^{t_2} P_{h_j}(t)\, \mathrm{d}t$$

We assume that the energy production times can overlap with the consumption times. While the source power is not necessarily a constant value, we assume that we can predict it accurately for near future with negligible time and energy cost. Our system uses an ideal energy storage unit (supercapacitor or battery) with a nominal capacity B_{C_j} expressed in watt. The energy available in the storage B_j at time t is denoted by $E_{B_j}(t)$. We also assume that each energy storage can be charged up to its capacity. In addition, each processor P_j in the embedded system is characterized by instantaneous power consumption $P_{c_j}(t)$, expressed in watts where $0 \leq P_{c_j}(t)$, and by power demand P_{d_j} expressed in watts corresponds to the power needed by tasks' jobs when executing in processor P_j considering EDF scheduling. The charging rate B_{Cr} of the battery state of charge is calculated as the difference of the regenerated energy from the harvesting device E_{h_j} and the energy consumption of the embedded system E_{c_j}.

$$B_{Cr} = E_{h_j} - E_{c_j} \tag{1}$$

C. REAL-TIME TASKS

We consider a software platform Γ composed of a set ψ of N periodic tasks, i.e., $\psi = \{\tau_1, \ldots, \tau_N\}$. We assume that *Sys* performs two classes of tasks: soft and hard. The task τ_i, $i \in \{1, .., N\}$, is characterized by: i) Worst case energy consumption (WCEC) En_i expressed in Joules, the energy consumption of τ_i is considered at the worst case and corresponds to the largest amount of energy that τ_i can consume when executed on a processor, ii) Worst case execution time (WCET) C_i, iii) Period T_i, and iv) A degree of criticality dc_i that defines its applicative importance. The degree of criticality is defined as the functional and operational importance of a task. The designer of the system defines (manually) the degree of criticality of each task in the system. It is considered that tasks have implicit deadlines, i.e., deadlines are equal to periods. A task τ_i is characterized by (T_i, C_i, En_i, dc_i). Moreover, a non critical task with soft deadline is characterized also by a (m_i, k_i) parameter which indicates the tolerance of at least m among k consecutive instances that meet their deadlines for task τ_i. The utilization factor of task τ_i is denoted U_{τ_i} and is defined as $U_{\tau_i} = \frac{C_i}{T_i}$.

D. REAL-TIME SCHEDULING

In this paper, the semi-partitioned approach is considered. Tasks are initially allocated to processors, and every task set assigned to a processor is scheduled according to the EDF policy. The scheduling problem in a reconfigurable distributed embedded system based on energy harvesting falls into two constraints which should be respected.

1) TIME FEASIBILITY

Without considering energy requirements, exact schedulability tests for uniprocessor EDF-scheduling are given by

$$U_{P_j} \leq 1 \tag{2}$$

where U_{P_j} is the utilization factor of the processor P_j calculated by $\sum_{i=1}^{n} U_{\tau_i}$.

2) ENERGY FEASIBILITY

The energy demand E_{d_j} of each processor P_j in the embedded system must be less than the total energy provided by both the battery B_{C_j} and the energy generator E_{h_j}, i.e.,

$$E_{d_j}(t_1, t_2) \leq B_{C_j} + E_{h_j}(t_1, t_2)$$

where $E_{d_j}(t_1, t_2)$ is the energy demand of tasks set ψ_j in time interval $[t_1, t_2]$, calculated by $\sum_{i=1}^{n} En_i$, and $E_{h_j}(t_1, t_2)$ is the harvested energy in the time interval $[t_1, t_2]$. The energy load $U_{e_j}(t_1, t_2)$ of the task set ψ_j assigned to processor P_j is given by:

$$U_{e_j}(t_1, t_2) = \frac{E_{d_j}(t_1, t_2)}{B_{C_j} + E_{h_j}(t_1, t_2)} \leq 1$$
$$U_{e_j} = \sup_{0 \leq t_1, t_2 \leq H} U_{e_j}(t_1, t_2)$$

where H is the hyper period.

$$U_{e_j} \leq 1 \tag{3}$$

Proof: Since ψ_j is energy-feasible, we consider an energy-valid schedule. The amount of energy demanded in each interval of time $[t_1, t_2]$, $E_{d_j}(t_1, t_2)$, is necessarily less than or equal to the actual energy available in $[t_1, t_2]$ given by $E_{B_j}(t_1) + E_{h_j}(t_1, t_2)$. An upper bound on $E_{B_j}(t_1)$ is the maximum storable energy at time t_1, that is B_{C_j}. Consequently, $E_{d_j}(t_1, t_2)$ is lower than or equal to $B_{C_j} + E_{h_j}(t_1, t_2)$. This leads to $[t_1, t_2]$, $E_{d_j}(t_1, t_2) \leq B_{C_j} + E_{h_j}(t_1, t_2)$ i.e. U_{e_j} (t_1, t_2). Thus, $U_{e_j} \leq 1$. □

Proposition 1: The set of tasks ψ_j assigned to processor P_j is feasible only if

$$U_{P_j} \leq 1 \quad and \; U_{e_j} \leq 1. \tag{4}$$

Proof: Suppose that ψ_j is feasible. Thus, ψ_j is time-feasible and energy feasible. From constraint (2) and constraint (3), constraint (4) is satisfied. □

E. DAG MODEL

1) MOTIVATION

A recurring task requests the execution of infinite sequential pieces of code called jobs. Therefore, real-time tasks are usually modeled as a sequence of recurrent jobs. Tasks are released several times and have a job to do for each release. In other words, a task starts a job for each release time. Thus, a job can be seen as an instance of a real-time task associated with a temporal deadline relative to its arrival time. Each task should complete its current job before it has been released for the next one. In the real application scenarios the execution flow of tasks is characterized by multiple conditional structure such as the (if-then-else, statement). Two jobs $\tau_{i,h}$ and $\tau_{i,k}$ of task τ_i may execute different parts of the code. Hence, an "execution flow" is defined as the path used by a job throughout its execution. To the best of our knowledge, the recurring real-time task model proposed by Baruah [31] represents the first attempt that permits the presentation of conditional real-time code. The conditional structure within the code may mean that different activations of the task cause different parts of the code to be executed.

2) DAG TASK MODEL DESCRIPTION

Each task τ_i, $i \in \{1, .., N\}$, is represented by a task graph $G_i(V_i, E_i)$, as depicted in Figure 2, where $V_i = \{\tau_{i,1}, ..., \tau_{i,n_i}\}$ is the node set that represents the subtasks of τ_i, n_i is the number of subtasks in G_i, and E_i is the directed edge set that represents the dependencies between the nodes in G_i. This task graph is a DAG with a unique source vertex, i.e., a vertex with no incoming edge, and a unique sink vertex, i.e., a vertex with no outgoing edge. Each vertex represents a subtask and each edge defines a possible flow of control. Each subtask $\tau_{i,k}$ is labeled by a WCET $C_{i,k}$. The total execution requirement of task τ_i is calculated as the sum of the WCET of all subtasks of the critical path in G_i. The critical path is the longest path in G_i.

DAG G_i corresponding to task τ_i is characterized by set $\mathcal{F}_i = \{F_{i,1}, ..., F_{i,m_i}\}$ which denotes the m_i possible execution flows in G_i. Each execution flow $F_{i,l} = (V_{i,l}, E_{i,l})$ is characterized by: i) set of nodes $V_{i,l}$, and ii) set of edges $E_{i,l}$. The semantics of this task DAG are as follows.

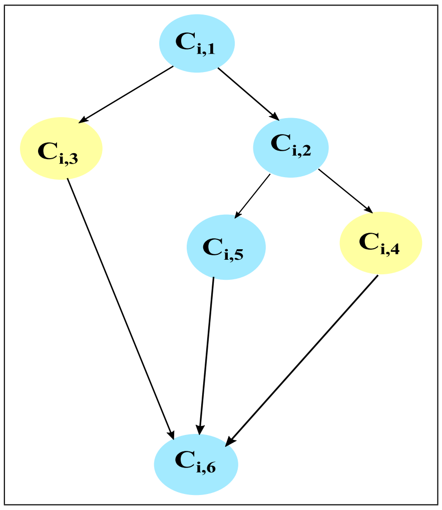

FIGURE 2. DAG task model.

Whenever subtask $\tau_{i,1}$ is released, depending upon the outcome of $\tau_{i,1}$ either $\tau_{i,2}$ with WCET $C_{i,2}$, or subtask $\tau_{i,3}$ with WCET $C_{i,3}$ are executed. If $\tau_{i,2}$ is executed, depending upon the outcome of this subtask, either $\tau_{i,4}$ with WCET $C_{i,4}$ or subtask $\tau_{i,5}$ with WCET $C_{i,5}$ are executed. A single subtask $\tau_{i,6}$ with WCET $C_{i,6}$ is executed. Therefore, task τ_i $(i \in \{1, .., N\})$ is characterized by the sixtuplet $(G_i, C_i, T_i, E_{ni}, dc_i)$. We introduce the following notation and terminology.

Definition 1: The WCET of execution flow $F_{i,l}$ of task τ_i, $i \in \{1, .., N\}$, $l \in \{1, .., m_i\}$, is defined as the cumulative amount of WCET of $V_{i,l}$ subtasks

$$C_{F_{i,l}} = \sum_{\tau_{i,k} \in V_{i,l}} C_{\tau_{i,k}} \tag{5}$$

The critical execution flow F_i^c in graph G_i is defined as the execution flow with the longest execution time.

IV. RECONFIGURATION APPROACH

This section details the proposed reconfiguration solution used to reestablish the system feasibility in NREEHS.

A. RECONFIGURABLE REAL-TIME SCHEDULING

To adjust the framework to cope with any unpredictable external event such as hardware faults or new task arrivals, we characterize a reconfiguration as any procedure that permits to reconfigure the system to be feasible, i.e., satisfying its real-time and energy constraints with the consideration of system performance optimization. This research presents a solution with three successive adaptation strategies to reconfigure the system at run-time. These strategies are performed in a hierarchical order as depicted in Fig. 3.

- Decomposes the task DAG of the faulty processor to a set of branches and to be migrated to other non-faulty processors,
- Degrades the QoS on each faulty processor. Non-critical tasks with the lowest degree of criticality execute in degrade mode according to (m,k)-firm constraints,

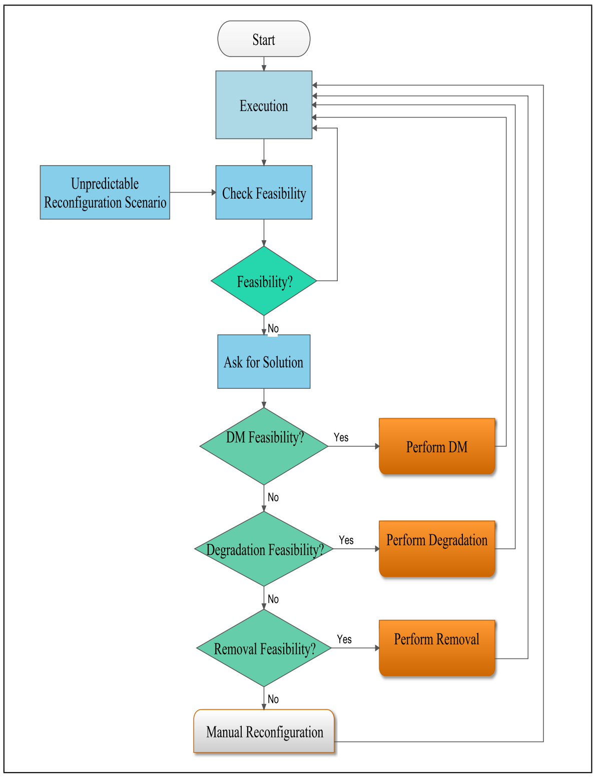

FIGURE 3. **Flow chart of the proposed methodology.**

- Deletes branches or tasks DAG so as to minimize the global deadline miss ratio.

1) DECOMPOSITION AND MIGRATION MODULE

A two steps strategy:

- First step: Branch Selection Heuristic. Selects a branch or a group of branches to be migrated to other non-faulty processors in order to reestablish the system feasibility. The task with the lowest degree of criticality will be decomposed into a set of branches. Then the critical execution flow will be removed from the DAG.
- Second step: Processor Selection Heuristic. Selection of a processor into which the migrant branches will be assigned. Sorts the set of candidate processors in an increasing order of energy availability in storage unit.

2) DEGRADATION MODULE

Degrades the scheduling in each faulty processor. In this case, the tasks with the lowest degree of criticality may be executed under (m,k)-firm constraints according to user requirements which indicate that the deadlines of at least m among any k consecutive instances of a task must be met. According to [32], a given task set ψ_j is assumed to be schedulable with (m,k)-firm constraints if the utilization processor factor with a (m,k)-firm requirement, defined by $U_{m,k} = \sum_{i=1}^{n} U_{\tau_i} * \frac{m_i}{k_i}$, is no greater than 1 defined by

$$U_{m,k} = \sum_{i=1}^{n} U_{\tau_i} \times \frac{m_i}{k_i} \leq 1 \qquad (6)$$

3) REMOVAL MODULE

Deletes branches or DAG tasks with the highest densities so as to minimize the global deadline miss ratio. With each task τ_i, $i \in \{1, .., N\}$ is associated a density denoted by

$$\gamma_i = \frac{En_i}{T_i} \qquad (7)$$

We sort all the tasks in an ascending order of densities such that we can reject those with higher densities one by one until the remaining utilization factor and energy consumption of the faulty processor satisfy (3) and (4).

B. RECONFIGURATION ALGORITHM

Algorithm 1 Reconfiguration Solution With Three Adaptation Strategies

```
Input : $\psi = \{\tau_1, .., \tau_N\}$; $\psi_j$: Set of tasks assigned to
        processor $P_j$; $\psi_r = \{\tau_{r,1}, .., \tau_{r,k}\}$: Set of
        reconfiguration tasks; $\Sigma = \{\sigma_1, ..\sigma_m\}$: Set of
        subsystems; Sched: boolean.
Begin;
Sched < − true;
/*$\psi$ is schedulable on $\Sigma$*/
; if Event $(\tau_r - > P_j)$ then
    $\psi_j < -\psi_j \bigcup \{\tau_r\}$;
    if Set $\psi_j \bigcup \{\tau_r\}$ not schedulable in $P_j$ then
        Sched < − false;
        if Decomposition migration($\psi_j$, $\Sigma$) is
        schedulable in $P_j$ then
            Perform Decomposition migration($\psi_j$, $\Sigma$);
        else
            if Degradation((m,k),$\psi_j$) is schedulable in
            $\psi_j$ then
                Perform Degradation((m,k), $\psi_j$)
            else
                Perform removal();
            end
        end
    end
end
Sched < − true;
Output: $\psi \bigcup \psi_r$ schedulable on $\Sigma$.
```

The proposed solution with the three adaptation strategies is described in Algorithm 1. The system performs set ψ of tasks assigned to multiprocessor platform Σ. At run-time an unpredictable event occurs and adds task τ_r to processor P_j. Thereafter, the proposed solution performs the feasibility analysis to the set $\psi_j \bigcup \{\tau_r\}$ in processor P_j. If the system is infeasible due to processor overload and/or energy starvation, then the proposed algorithm performs the reconfiguration solution.

V. MULTIAGENT ARCHITECTURE FOR NREEHS

This section details the proposed multiagent architecture for networked reconfigurable energy harvesting systems.

A. MOTIVATIONS FOR THE USE OF THE MAS PARADIGM

The NREEHS works in dynamic environment where unpredictable events occur such as activation of new tasks and software or hardware failures. Thereafter, the static scheduling is no longer optimal and the system may evolve towards a situation of processor or energy overloads. An occurred problem such as processor or energy overloads in a particular processor P_j is resolved by a global or local reconfiguration. For this purpose, this paper proposes the use of a distributed system decentralizing the control, and more precisely the use of an MAS. A distributed control system is built to connect all the processors for information exchange, and to conduct a global control of the entire system with guaranteed correctness and optimized performance. We aim through the use of MAS to represent as near as possible the real behavior of the physical NREEHS thanks to the developed simulator.

An intelligent reconfiguration agent is developed to provide the proposed solution with the three adaptation strategies. The software agents are helpful to perform some tasks such as the supervision to detect unpredictable events, feasibility analysis, and battery management. Motivated by these considerations, we choose to deploy the intelligent agents to simulate the dynamic behavior of NREEHS.

B. CLASSIFICATION OF AGENTS

We propose a new multiagent architecture consisting of:

- Three agents belonging to each subsystem $\sigma_j \in \Sigma$, $j \in \{1, .., m\}$: (i) supervisor agent A_{Sup_j}, (ii) scheduling agent A_{Sched_j}, (iii) reconfiguration manager agent A_{Reconf_j}, and (iv) battery manager agent A_{B_j}.
- A coordinator agent C_{Sys} defined to coordinate between the networked reconfigurable subsystems and to handle all concurrent reconfiguration requirements.

1) SUPERVISOR AGENT

A supervisor agent for $\sigma_j \in \Sigma, j \in \{1, .., m\}$, that plays the role of a coordinator in the subsystem. The supervisor establishes two kinds of interactions:

a) Intra-subsystem interaction with agents from the same subsystem in order to 1) control reconfiguration scenarios and check the system feasibility, and 2) establish useful solutions to reobtain the system feasibility,

b) Inter-subsystem interaction with the coordinator agent in order to obtain an authorization to apply a global reconfiguration scenario.

2) SCHEDULING AGENT

A scheduling agent is assigned to each subsystem of the execution environment in order to perform the feasibility analysis. Each subsystem σ_j composed of processor P_j and battery B_j to perform the tasks set ψ_j, should satisfy the following schedulability test:

$$U_{P_j} \leq 1 \quad and\ Ue_j \leq 1$$

3) RECONFIGURATION MANAGER AGENT

The automatic reconfiguration scenarios are classified into two categories: 1) internal reconfiguration scenarios where each subsystem handles its own reconfiguration scenarios without the permission of the coordinator agent, and 2) external reconfiguration scenarios where the supervisor agent needs the permission from the coordinator agent. Furthermore, a reconfiguration manager agent affected to each subsystem $\sigma_j \in \Sigma, j \in \{1, .., m\}$ is defined, to handle automatic reconfigurations in order to maintain the system feasibility. The reconfiguration manager agent performs the proposed solution with the three successive adaptation strategies in a hierarchical order. Therefore, the reconfiguration manager agent is decomposed into three modules: i) decomposition and migration, ii) degradation, and iii) removal modules.

4) BATTERY MANAGER AGENT

In the distributed architecture, a battery manager agent is associated with each subsystem $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$. Indeed, the main role of this agent is to control the energy level in the storage unit (battery/ supercapacitor) and to predict the availability of the energy in the future. Further, the battery manager agent performs the solar energy harvesting prediction algorithm proposed in [33].

5) COORDINATOR AGENT

The coordinator agent C_{Sys} is defined to control all concurrent reconfiguration scenarios and to guarantee safe, coherent and adequate distributed reconfigurations as well as a feasible execution in the whole system. The role of the coordinator is to reach an agreement and to broadcast this decision value to all the other supervisor agents. The coordinator agent affects priority to the different concurrent reconfiguration requests according to the criticality of the migrated tasks and manages all concurrent reconfiguration requests. The role of the coordinator is to accept or reject a reconfiguration request. In addition, when the coordinator agent accepts a reconfiguration request to migrate a task from a faulty processor to another, it sends a token to all supervisor agents associated with the different subsystems. When the coordinator receives multiple candidates, it selects a winner according to the criterion which permits to balance the workloads and energy consumption of the processors.

C. COMMUNICATION PROTOCOL FORMALIZATION

To guarantee a feasible execution in the NREEHS architecture, two communication protocols are defined:

- Intra-subsystem communication protocol that manages the communication between agents in the same subsystem,
- Inter-subsystem communication protocol that manages concurrent reconfigurations between subsystems to define coherent behaviors.

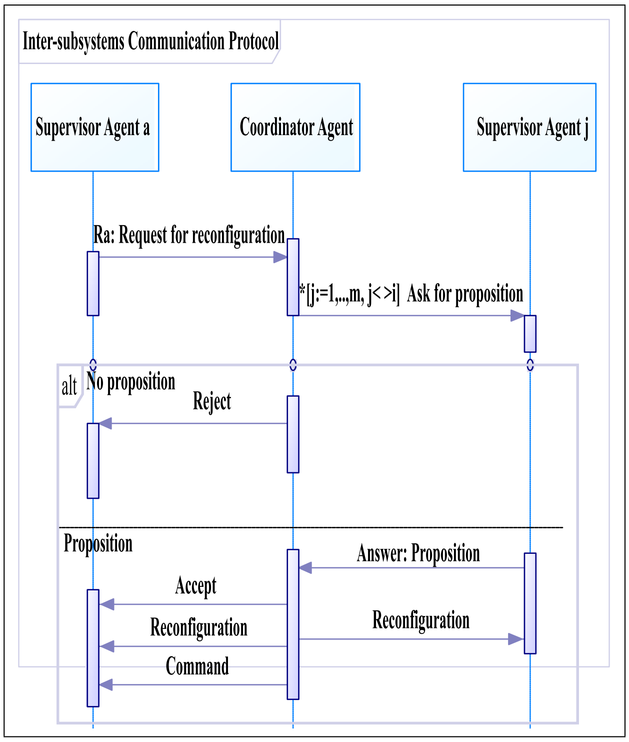

FIGURE 4. Inter-subsystems communication protocol.

1) INTER-SUBSYSTEM COMMUNICATION PROTOCOL

We propose a communication protocol between the supervisor agent associated with each subsystem $\sigma_j \in \Sigma$, $j \in \{1, .., m\}$, and the coordinator C_{Sys}. It defines the interaction rules between subsystems and this coordinator in order to guarantee a feasible execution of the whole distributed system. Hence, any subsystem cannot apply Decomposition-Migration strategy until it receives the permission from the coordinator agent. The communication between supervisors and the coordinator is performed through exchanged messages. Fig. 4 depicts the interaction between the coordinator and supervisor agents. When a particular reconfiguration manager agent A_{Reconf_a}, $a \in \{1, .., m\}$, should apply a Decomposition-Migration reconfiguration scenario, the supervisor agent A_{Sup_a} sends the following request R_a to C_{Sys} to obtain its authorization, i.e.,

$$R_a\text{: request(from-}A_{Sup_a}\text{, to-}C_{Sys}\text{, ID, }F_{i,l})$$

In this case, the supervisor A_{Sup_a} sends a request to the coordinator for migrating the branch $F_{i,l}$ of the task identified by ID. If R_a has the highest priority between all requests in *R(t)*, then C_{Sys} broadcasts a token for each supervisor agent $A_{Sup_j}(j \in \{1, .., m\}/\{a\})$ by sending the following message, i.e.,

$$\text{Send(T[}A_{Sup_j}, F_{i,l}, \text{ID])}$$

A_{Sup_a} wants to migrate the branch $F_{i,l} \in \mathcal{F}_i = \{F_{i,1}, ..., F_{i,m_i}\}$, $i \in \{1, .., N\}$, $l \in \{1, .., m_i\}$. When supervisor A_{Sup_j}, $j \in \{1, .., m\}$, receives the token, it verifies the schedulabilty conditions (real-time and energy constraints) by accepting the migrating branch. If it is possible for the subsystem to accept the branch, then it answers by sending a proposition to C_{Sys} as follows, i.e.,

$$\text{Answer(P[}A_{Sup_j}, U_{e_j}, U_{p_j}\text{])}$$

The coordinator receives all propositions and selects a winner A_{Sup_w}, $w \in \{1, .., m\}/\{a\}$, according to the criterion which permits to balance the workloads and energy consumption of the processors. The coordinator sends its permission to the supervisor for migrating the branch $F_{i,l}$ of the task identified by ID, i.e.,

$$\text{Accept(from-}C_{Sys}\text{, to-}A_{Sup_a}\text{, ID, }F_{i,l})$$

If no supervisor sends a proposition to the coordinator, then it is impossible to migrate the branch and the coordinator rejects the supervisor request, i.e.,

$$\text{Reject(from-}C_{Sys}\text{, to-}A_{Sup_a}\text{, ID, }F_{i,l})$$

When the coordinator C_{Sys} accepts the request of the particular agent A_{Sup_a}, it sends a reply to inform the concerned subsystems as follows, i.e.,

$$\text{Reconfiguration(from-}C_{Sys}\text{, to-}A_{Sup_a}\text{, to-}A_{Sup_w})$$

The coordinator informs the target supervisor agent A_{Sup_w} that it will receive a migrating branch from agent A_{Sup_a}. Finally the coordinator sends a command to A_{Sup_a} to perform the Decomposition-Migration reconfiguration, i.e.,

$$\text{Command(from-}C_{Sys}\text{, to-}A_{Sup_a}\text{, to-}A_{Sup_w}\text{ ID, }F_{i,l})$$

2) INTRA-SUBSYSTEM COMMUNICATION PROTOCOL

We propose a communication protocol between the different agents associated with each subsystem. The protocol defines interaction rules between agents in order to verify the system feasibility and to guarantee a feasible execution. The supervisor agent plays the role of the coordinator in each subsystem. Initially, the supervisor agent is in a listening state, whenever it detects an unpredictable external event, it interacts with the different agents from the same subsystem so as to:

1) Control reconfiguration scenarios and check the system feasibility,
2) If not satisfied, establish solution in order to reobtain the system feasibility.

Fig. 5. depicts the interaction between the supervisor, reconfiguration manager, scheduling and battery manager agents.

Algorithm 2 Reconfiguration of the NREEHS

```
Input : $\Sigma$: set of subsystems; $\Gamma$: the software platform;
        $\Lambda$: the intelligent multiagent architecture.
Begin
$A_{Sup_a}$listening();
if Event () then
    Test-Feasibility(from-$A_{Sup_a}$, to-$A_{sched}$);
    if Feasibility-Answer(from-$A_{Sup_a}$, to-$A_{sched}$, NO, $U_{p_a}$ ) then
        Ask-for-Solution(from-$A_{Sup_a}$, to-$A_{Reconf}$);
        Solution( $S_D$, $S_{DM}$, $S_R$);
        if $S_f$ = ( $S_D$, $S_R$) then
            command(from-$A_{Sup_a}$, to-$A_{Reconf}$, $S_f$);
        end
        else if $S_f$ = ($S_{DM}$) then
            $R_a$: request(from $- A_{Sup_a}$, to $- C_{Sys}$, ID, $F_{i,l}$);
            if $P_{R_a} = P_H$ then
                Send(T[$A_{Sup_j}$, $F_{i,l}$, ID]);
                A:= Answer(P[$A_{Sup_j}$, $En_j$, $U_{pj}$];
                if $A = \emptyset$ then
                    Reject(from-$C_{Sys}$, to-$A_{Sup_a}$, ID, $F_{i,l}$);
                else
                    Accept(from-$C_{Sys}$, to-$A_{Sup_a}$, ID,
                    $F_{i,l}$);
                    $A_{Sup_w}$ := Select-winner(A);
                    Reconfiguration(from-$C_{Sys}$,
                    to-$A_{Sup_a}$, to-$A_{Sup_w}$);
                    Command(from-$C_{Sys}$, to-$A_{Sup_a}$,
                    to-$A_{Sup_w}$, ID, $F_{i,l}$);
                end
            end
        end
    end
end
Output: Feasible system
```

The interaction between the supervisor and the three other different agents is ensured through exchanged messages as implemented in Algorithm 2. When the supervisor agent A_{Sup_a} detects an external event in the associated subsystem $\sigma_a, a \in \{1, .., m\}$, it sends a request to the scheduling agent to check the subsystem feasibility, i.e.,

$$\text{Test-Feasibility(from-}A_{Sup_a}\text{, to-}A_{Sched_a}\text{)}$$

The scheduling agent performs the feasibility analysis, and according to related results, it sends one of the following answers:

$$\text{Feasibility-answer(from-}A_{Sched_a}\text{, to-}A_{Sup_a}\text{, YES, }U_{p_a}\text{)}$$

YES means that the system is feasible,

$$\text{Feasibility-answer(from-}A_{Sched_a}\text{, to-}A_{Sup_a}\text{, NO, }U_{p_a}\text{)}$$

NO means that the system is infeasible. In this case, the supervisor A_{Sup_a} sends the following request to the reconfiguration

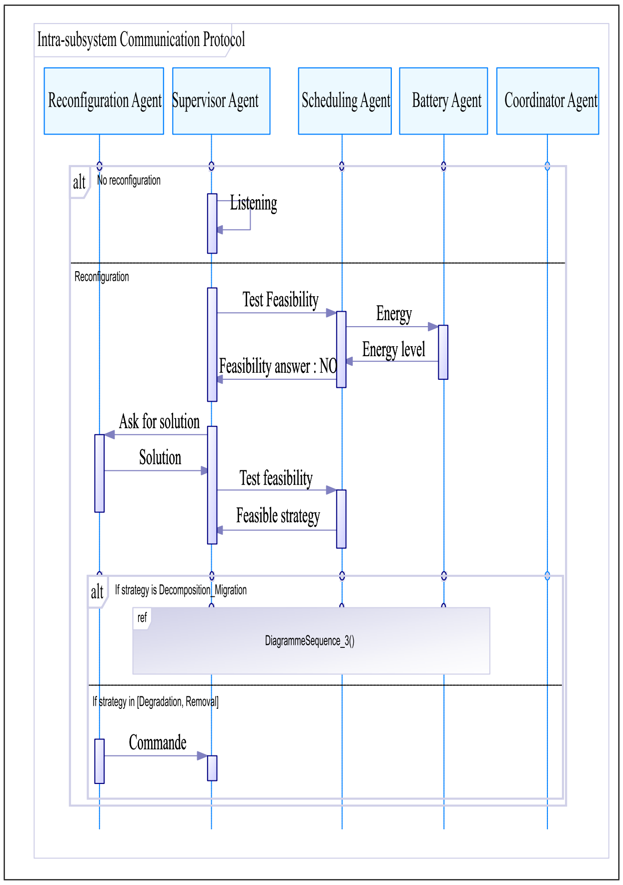

FIGURE 5. Intra-subsystem communication protocol.

manager agent to establish the required solution, i.e.,

$$\text{Ask-for-Solution(from-}A_{Sup_a}\text{, to-}A_{Reconf_a}\text{)}$$

When the reconfiguration manager agent receives a request for establishing solution, it interacts with the three reconfiguration modules and then it sends a token that contains the proposed solution with the three adaptation strategies in a hierarchical order. Indeed, this order should be respected at run-time where the subsystem should start by the first proposed strategy DM. Nevertheless, if the first strategy DM does not involve feasibility, then the system proceeds to the second strategy. Therefore, the reconfiguration manager agent sends the following message, i.e.,

$$\text{Solution(}S_{DM}, S_D, S_R\text{)}$$

If the required strategy is the Decomposition-Migration, then the supervisor agent sends a request to the coordinator for migrating the branches $F_{i,l}$ of the task identified by ID to apply an external reconfiguration as follows, i.e.,

$$R_a\text{: request(from-}A_{Sup_a}\text{, to-}C_{Sys}\text{, ID, }F_{i,l}\text{)}$$

Algorithm 3 depicts the communication protocol: inter-subsystem and intra-subsystem in the proposed multiagent architecture.

D. COMPLETENESS PROOF OF THE RECONFIGURATION SOLUTION

The proposed reconfiguration solution with the three strategies has the advantage of completeness. Thus, if a solution exists, the proposed MAS will find it. We denote the solution set as $S_f = \{S_{DM}, S_D, S_R\}$ for a faulty processor P_f where the real-time and/or the energy constraints are violated. Thus, the system feasibility is reestablished by a global reconfiguration decomposition migration or local reconfiguration degradation or removal strategies.

Proposition 2: The reconfiguration protocol is complete.

For each subsystem in *Sys*, if an unpredictable event occurs and evolves the system towards an infeasible state where the real-time and/or the energy constraints are not respected, and if there exists a local or global solution S to reestablish the system feasibility, then the reconfiguration process will necessarily find it.

Proof: We perform a reasoning by absurdity to prove the completeness of the proposed protocol. Let us suppose that the protocol is not complete. That is, there is no possible solution neither local nor global for the faulty subsystem. Thus, we have

$$U_{P_j} > 1 \quad and \; U_{e_j} > 1 \tag{8}$$

Since the reconfiguration process is assumed by absurdity to be not complete, we conclude that there exists a solution S such that

$$U_{P_j} \leq 1 \quad and \; U_{e_j} \leq 1 \tag{9}$$

S is not found by the proposed reconfiguration protocol but by another one. We recall that a software solution S consists with decreasing processor resource and energy requirements while satisfying the QoS. Such a solution can be either the degradation of the execution mode, migration, or removal of tasks.

Thus, we conclude that no other algorithm provides a solution not yet found by the protocol. In fact if a solution S exists, it is included in $S_f = \{S_{DM}, S_D, S_R\}$. Thus, S belongs to either local or global strategy. □

VI. CASE STUDY

This section investigates a running example in order to explain the proposed methodology using theoretical tasks. Suppose that *Sys* is a networked reconfigurable system composed of three subsystems such that $Sys = (\Sigma, \Gamma, \Lambda)$, where $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$. Initially, the batteries $B_1 = 45$ energy units, $B_2 = 110$ energy units, and $B_3 = 100$ energy units are fully charged. The system *Sys* is composed of five tasks as depicted in Table 1. The tasks τ_1, τ_2, and τ_3 are assigned to processor P_1, task τ_4 is assigned to processor P_2, and task τ_5 is assigned to processor P_3. The energy consumption is equal to $U_{e_1} = 20$, $U_{e_2} = 30$ and $E_{e_3} = 2$ energy units. Due to the cheddar [34] implementation, the feasible scheduling result of the system *Sys* is shown in Figure 7. After applying different reconfiguration scenarios as depicted in Table 2, the system may evolve towards an infeasible state where the energy consumption may increase and/or some tasks violate their deadlines.

TABLE 1. System configuration.

Initial System Configuration					
Task	C_i	T_i	D_i	En_i	dc_i
τ_1	11	50	50	4	A
τ_2	8	25	25	7	B
τ_3	9	50	50	2	C
τ_4	14	25	25	30	D
τ_5	12	75	75	2	C
System Reconfiguration Scenarios					
Task	C_i	T_i	D_i	En_i	dc_i
Reconfiguration scenario 1					
τ_6	11	15	15	15	A
Reconfiguration scenario 2					
τ_7	9	15	15	18	B
Reconfiguration scenario 3					
τ_8	7	15	15	18	E

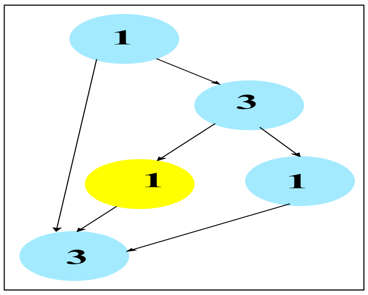

FIGURE 6. DAG G_4 associated to the task τ_4.

VII. EXPERIMENTS

This section explores the performance of the proposed intelligent multiagent distributed architecture that allows feasible executions after any external reconfiguration scenario that may evolve the system towards an infeasible state. Extensive simulation experiment has been performed to validate the proposed scheme in energy efficiency and deadline miss rate. In order to evaluate this architecture, an NREEHS composed of eight subsystems is considered. Each subsystem is composed of a processor and a rechargeable energy storage with limited capacity supplied by a renewable energy source. The software platform consists of 100 tasks to be schedulable on the eight identical processors. The parameters of the tasks are randomly generated where every period T_i ($i \in [1..100]$) is randomly chosen in the range [100, 200], every WCET C_i ($i \in [1..100]$) is randomly chosen in the range [6, 10], and every degree of criticality dc_i ($i \in [1..100]$) is randomly chosen in the range [A, F]. We use a DAG generator to generate graphs of tasks $G_i = (V_i, E_i)$ ($i \in [1..100]$) according to C_i and T_i. We assume that the intelligent multiagent distributed architecture consists of 33 agents. A coordinator agent is affected to the whole system and each subsystem gathers four agents: Reconfiguration manager, scheduling, battery manager and supervisor agents. We assume a set of unpredictable reconfiguration scenarios is applied repeatedly.

TABLE 2. Reconfiguration scenarios.

Reconfiguration Scenario	Processor Utilization Factor	Energy Consumption	Applied Strategy	New Processor Utilization Factor	New Energy Consumption
Reconfiguration scenario 1: Add task τ_6 to processor P_2	U_{P_2}= 1.293	U_{e_2}= 195	Decomposition-Migration of branch $F_{4,1}$ of graph G_4 Fig. 6 to processor P_3	U_{P_2} = 0.89, U_{P_3}= 0.72	U_{e_2} = 108, U_{e_3}= 0.72.
Reconfiguration scenario 2: Add task τ_7 to processor P_3	U_{P_3}= 1.32	U_{e_3}= 182	Degradation strategy is applied on processor P_3 the task set may be executed under (1,2)-firm constraint	U_{P_3} = 0.66	U_{e_3} = 91
Reconfiguration scenario 3: Add task τ_8 to processor P_3	U_{P_3}= 1.37	U_{e_3} = 181	The removal is selected to be applied. Tasks τ_4, τ_7, and τ_8 have the same density which is higher than the τ_5's density. Therefore, task τ_4 is selected to be removed since it has the lower degree of criticality	U_{p_3} = 0.81	U_{e_3} = 91.

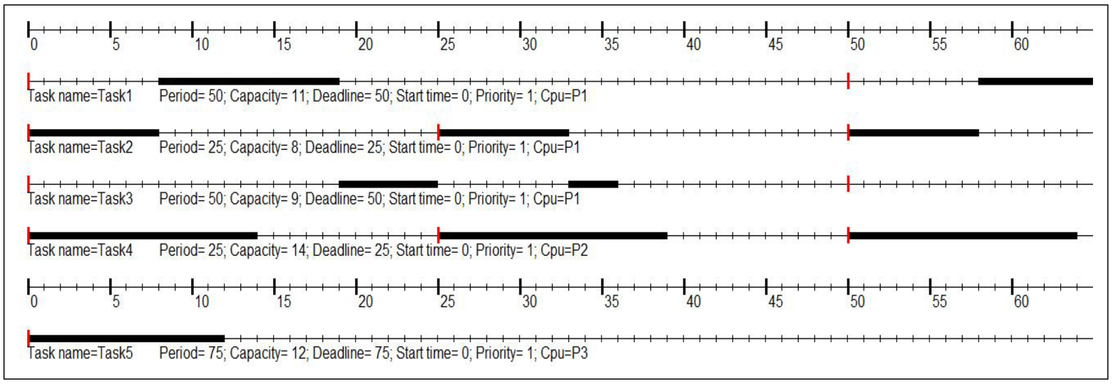

FIGURE 7. Scheduling of the initial system.

Each scenario adds a set of n tasks such that n is randomly chosen in the range [10, 40].

A. COMPARISON OF DEADLINE MISS RATE

We perform a simulation in order to prove the performance of the proposed intelligent multiagent architecture in terms of the percentage of satisfied deadline.

1) COMPARISON WITH AND WITHOUT MAS

In this first set of experiments, we investigate the performance of the proposed multiagent architecture. For this purpose, we compare overall miss rates with and without MAS. Fig. 8 presents the percentage of succeeded deadlines. When we apply the intelligent multiagent distributed architecture, the percentage of the succeeded deadlines increases from 52% to 73% if the number of concurrent reconfiguration requests is equal to 35.

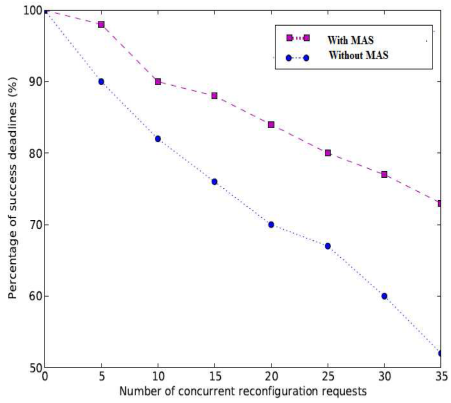

FIGURE 8. Percentage of satisfied deadlines.

2) COMPARISON WITH PREVIOUS APPROACHES

The proposed algorithm is compared with the well known state-of-the-art techniques EDH [20], UTB [23],

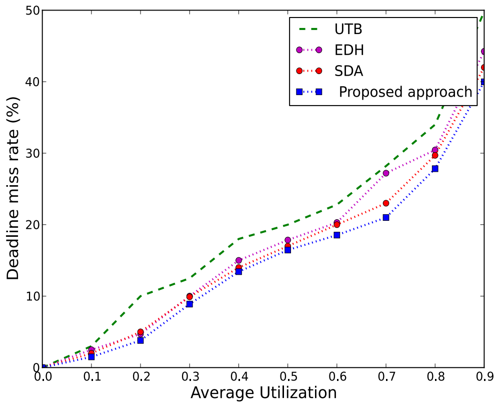

FIGURE 9. Overall miss rate comparison.

and SDA [35]. As described in Fig. 9 UTB has a much higher miss rate as it uses an isolated task dropping scheme on each processor, which is based on energy availability prediction for one upcoming task, ignoring workload on other processors that compete for the same energy source.

For the other two techniques, EDH and SDA have a lower miss rate percentage than UTB. However, EDH before authorizing any task to execute, the energy level of the storage must be sufficient such that all future occurring tasks execute timely with no energy starvation, considering both their energy consumption and the replenishment rate of the storage unit. On the other hand, SDA performs task rejection before assigning accepted tasks to different processors thus, the workload is adapted to a system-wide energy budget that has been predicted.

It is clear that the proposed approach outperforms the other techniques. One reason for this trend is that the proposed approach exploits the flexibility to perform dynamically a solution with three successive adaptation strategies: migration from one processor to another one, degradation of the execution mode, and removal which may increase the percentage of succeeded deadlines. In contrast to SDA and EDH, the proposed approach allows the execution of requested tasks while maintaining a graceful QoS.

B. COMPARISON OF ENERGY GAIN

The approach proposed in [30] presents a software-agent-based architecture where an intelligent software control agent is developed to perform four solutions. The study in [30] considers a reconfigurable real-time system that processes periodic and probabilistic tasks. In order to compare with this approach, we consider that the system processes only periodic tasks. Since the gains of the four proposed solutions in [30] are independent of the considered scenarios, Solution A is selected in the performed experimentation. Fig. 10 presents the percentage of energy gain when a set of reconfiguration scenarios is applied repeatedly during the execution time of both solution A from [30] and the proposed heuristic RH.

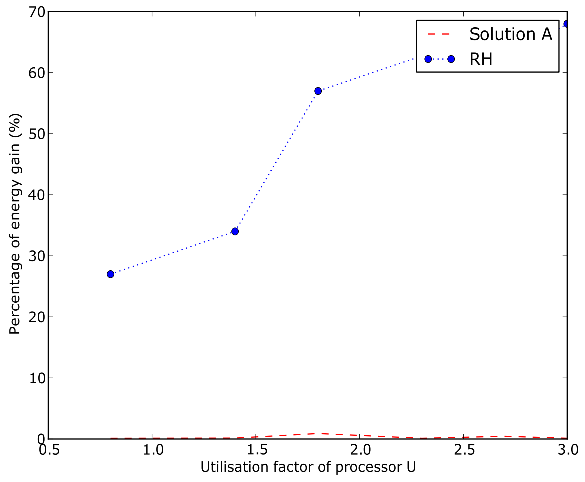

FIGURE 10. Percentage of energy gain.

C. COMPARISON OF NUMBER OF EXCHANGED MESSAGES

The rate of exchanged messages is an important criterion to guarantee an acceptable level of safety and robustness in real-world industry such as distributed applications. First of all, we compare the inter-subsystem communication protocol defined to treat the Decomposition-Migration reconfiguration and the intra-subsystem communication protocol defined to perform the DH and RH strategies. We compare the proposed work with the research reported in [28] in terms of the number of exchanged messages. Fig. 11 shows that the DMH outperforms all the other heuristics. In fact, when the number of concurrent reconfiguration requests is equal to 100, DMH provides 10600 exchanged messages, whereas the DH and RH heuristics provide similar results and the corresponding number of the exchanged messages exceeds 20000.

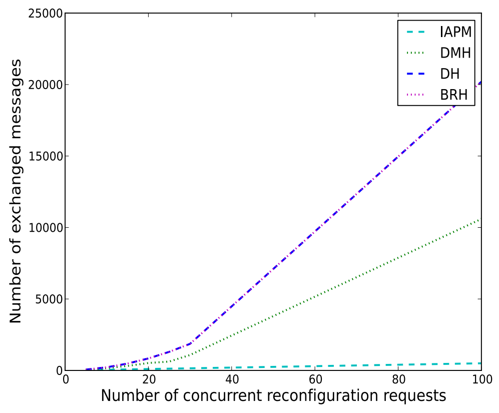

FIGURE 11. Number of exchanged messages in the inter-subsystem and intra-subsystem communication protocol.

In order to show the performance of the proposed communication protocol, we perform a simulation to compare it with the one reported in [28]. Let us assume that *Sys* is a

TABLE 3. Comparative study.

Work	MAS architecture	Communication	Energy harvesting	Strategy	DAG model
[20]	No MAS.	No communication.	Ambient energy is harvested and converted into electrical power.	Extends the well-known EDF algorithm and adds energy awareness capabilities by using the notion of slack-time and slack-energy.	No DAG .
[36]	Multiagent model: Security Agent, Reconfiguration Agent, Execution Agent, and Scheduling Agent.	Inter-agent communication.	No energy harvesting.	Proposes a middleware for the execution of secured software reconfigurations while meeting deadlines and keeping an optimum level of safety.	No DAG.
[28]	Virtual coordinator to coordinate subsystems.	Inter-agent communication protocol based on exchanged messages. The number of exchanged messages: 4n in the best case, and 3n(n+1) in the worst case.	No energy harvesting.	Computes the optimal coordination solution using judgment matrices.	No DAG.
[30]	An intelligent software control agent with four solutions is developed, in addition to three virtual processors to provide reconfiguration of the system at run-time.	Not mentioned.	No energy harvesting.	An intelligent software control agent with four solutions consists to modify the temporal parameters of the probabilistic tasks dynamically; and provides three virtual processors by dynamically extending the periods of the periodic tasks.	No DAG.
This work	Intelligent multiagent distributed architecture: A coordinator agent associated to the whole system, and four local agents associated to each subsystem.	Two communication protocols: Intra-subsystem communication, and Inter-subsystem communication. Exchanged messages: In the worst case is estimated to be n*(n+2).	The system is supplied by a renewable energy source.	A reconfiguration agent developed to handle three solutions: DMB, DH, RH.	New task model called Probabilistic DAG.

networked reconfigurable system based on energy harvesting and has n subsystems and a coordinator. Then, in order to evaluate the performance of the system, we consider the worst case where n subsystems need to perform reconfiguration scenarios. We admit that the concurrent reconfiguration requests are accepted in n steps such that only one request is accepted in each step. According to the communication protocol proposed by Zhang *et al.* [28], the number of exchanged messages is equal to $3*n*(n+1)$. Besides, the number of exchanged messages in the proposed communication protocol in this paper is equal to $n*(n+2)$.

Fig. 12 shows clearly that the proposed communication protocol outperforms the one reported in [28]. Especially when the number of concurrent reconfiguration requests exceeds ten, the gap increases considerably. Hence, the number of exchanged messages exceeds 2700 when the number of concurrent reconfiguration requests is equal to 30, whereas it is equal to 960 when we perform the proposed protocol in this paper. The results of this experimentation show that the proposed communication protocol reduces the number of exchanged messages by 64.44% than that the work reported

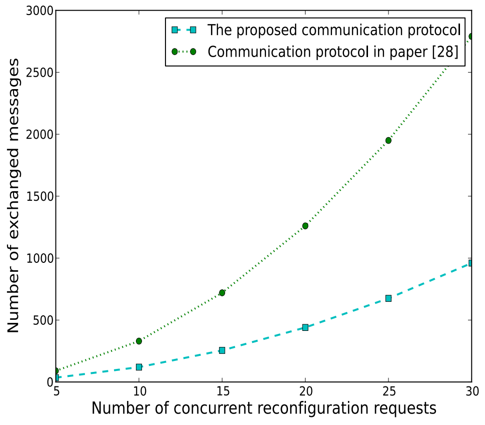

FIGURE 12. Number of exchanged messages.

in [28]. This is justified by the fact that in the related work the authors assume that the rejected subsystems during a distributed reconfiguration process will send again the same requirements to the coordinator until they are accepted in the future and no new reconfiguration requirements arise before all the requirements are accepted. On the contrary, in the current work we assume that all the reconfiguration requirements are sorted in $R(t)$ and treated according to their emergency calculated in function of the degree of criticality of migrated tasks. Therefore, a subsystem will only send one message for a reconfiguration requirement which will reduce the number of exchanged messages compared with the related work.

D. SCHEDULING OVERHEADS ANALYSIS

To compare scheduling overhead between UTB, EDH, SDA and the proposed approach, we executed the scheduling procedures of these schemes on the gem5 simulator [37] with a single thread at 1 GHz to observe average execution time overhead averaged over all task instances when managing a 8 subsystems that run 160 periodic tasks with a scheduling granularity of 1 ms. The results of this paper are shown in Fig 13. It can be seen that the obtained execution time and energy overheads are lower than overheads for UTB, EDH, and SDA. This result is consistent with the time complexity of EDH in the worst-case which is pseudo-polynomial and might be a serious drawback in practice. The complexity of EDH comes mainly from slack-time and slack-energy computations. The main reason for the lower overhead with the proposed approach is that it is based on the intelligent multiagent architecture which avoids computation overheads.

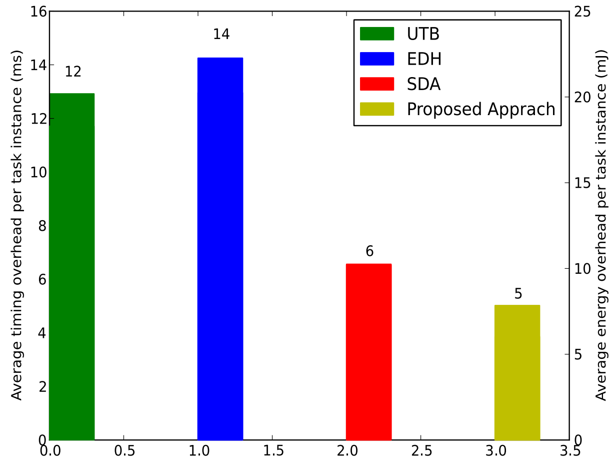

FIGURE 13. **Comparison of scheduling overhead.**

Table 3 compares the proposed approach in this paper with related works. The originality lies in treating a more challenging problem that combines different and independent problems in the related works. In fact, this paper is the first to deal with the adaptive scheduling of real-time DAG tasks with energy harvesting. The technical solution based on the migration of probabilistic branches as well as the proposed multiagent architecture is original. The discussed approach develops a new efficient solution to resolve the encountered problem.

VIII. CONCLUSION AND FUTURE WORK

This paper developed a new intelligent distributed multiagent architecture for networked distributed reconfigurable systems based on energy harvesting. The agents are classified into two categories: i) coordinator agent associated with the whole distributed system to coordinate between the networked reconfigurable subsystems and to treat all concurrent reconfigurations, and ii) local agents associated with each subsystem in order to keep feasible executions after any external reconfiguration scenario. A reconfiguration manager agent is proposed to perform the proposed solution with three adaptation strategies: Decomposition Migration Heuristic, Degradation Heuristic, and Branch Removal Heuristic in order to reestablish feasible executions. Two communication protocols are proposed: i) an intra-subsystem communication protocol to manage the communication between agents in the same subsystem, and ii) an inter-subsystem communication protocol to manage concurrent reconfigurations between subsystems. Extensive simulation experiments show the effectiveness of the proposed intelligent multiagent distributed architecture compared with a previous work in terms of the percentage of succeeded deadlines. Indeed, the experimental results show that when we apply the intelligent multiagent distributed architecture, the percentage of the succeeded deadlines increases from 52% to 73% when the number of concurrent reconfiguration requests is equal to 35. Moreover, the results prove the effectiveness of the multiagent architecture and communication protocols compared with related works from the state of the art in terms of the number of exchanged messages and energy saving.

The authors are now working on an extension of the current research by implementing the proposed approach in a practical distributed system based on multiagent architecture. We will also deal with the hardware aspect by proposing a software-hardware solution based on the XILINX FPGA technology [38].

REFERENCES

[1] J. A. Stankovic, "Real-time computing system: The next generationcoins," Dept. Comput. Inf. Sci., Univ. Massachusetts, Amherst, MA, USA, Tech. Rep. 88-06, 1988.

[2] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[3] F. Yao, H. Wu, Y. Chen, Y. Liu, and T. Liang, "Cluster-based collaborative spectrum sensing for energy harvesting cognitive wireless communication network," *IEEE Access*, vol. 5, pp. 9266–9276, 2017.

[4] J. Liu, K. Xiong, P. Fan, and Z. Zhong, "RF energy harvesting wireless powered sensor networks for smart cities," *IEEE Access*, vol. 5, pp. 9348–9358, 2017.

[5] M. Ashraf, A. Shahid, J. W. Jang, and K.-G. Lee, "Optimization of the overall success probability of the energy harvesting cognitive wireless sensor networks," *IEEE Access*, vol. 5, pp. 283–294, 2017.

[6] B. Buchli, F. Sutton, J. Beutel, and L. Thiele, *Towards Enabling Uninterrupted Long-Term Operation of Solar Energy Harvesting Embedded Systems* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2014, pp. 66–83.

[7] G. Gatti, M. J. Brennan, M. G. Tehrani, and D. J. Thompson, "Harvesting energy from the vibration of a passing train using a single-degree-of-freedom oscillator," *Mech. Syst. Signal Process.*, vols. 66–67, pp. 785–792, Jan. 2016.

[8] M. G. Valls and P. B. Val, "Comparative analysis of two different middleware approaches for reconfiguration of distributed real-time systems," *J. Syst. Archit.*, vol. 60, no. 2, pp. 221–233, 2014.

[9] A. Gharbi, M. Khalgui, and M. A. Khan, "Functional and operational solutions for safety reconfigurable embedded control systems," in *Embedded and Real Time System Development: A Software Engineering Perspective*. Berlin, Germany: Springer, 2014, pp. 251–282.

[10] R. M. da Silva *et al.*, "Modeling of mechanisms for reconfigurable and distributed manufacturing control system," in *Technological Innovation for Cloud-Based Engineering Systems*. Cham, Switzerland: Springer, 2015, pp. 93–100.

[11] X. Wang, Z. Li, and W. Wonham, "Dynamic multiple-period reconfiguration of real-time scheduling based on timed des supervisory control," *IEEE Trans. Ind. Informat.*, vol. 12, no. 1, pp. 101–111, Jan. 2016.

[12] Y. Chen, X. Mao, F. Hou, Q. Wang, and S. Yang, "Combining re-allocating and re-scheduling for dynamic multi-robot task allocation," in *Proc. IEEE Int. Conf. Syst., Man, Cybern (SMC)*, Oct. 2016, pp. 000395–000400.

[13] H. Grichi, O. Mosbahi, M. Khalgui, and Z. Li, "RWiN: New methodology for the development of reconfigurable WSN," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 109–125, Jan. 2017.

[14] M. Gasmi, O. Mosbahi, M. Khalgui, L. Gomes, and Z. Li, "New pipelined approach for an effective reconfigurable wireless sensor node," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2016.2625817.

[15] S. B. Meskina, N. Doggaz, M. Khalgui, and Z. Li, "Multiagent framework for smart grids recovery," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1284–1300, Jul. 2017.

[16] N. Luo, W. Zhong, F. Wan, Z. Ye, and F. Qian, "An agent-based service-oriented integration architecture for chemical process automation," *Chin. J. Chem. Eng.*, vol. 23, no. 1, pp. 173–180, 2015.

[17] G. Michalos, P. Sipsas, S. Makris, and G. Chryssolouris, "Decision making logic for flexible assembly lines reconfiguration," *Robot. Comput.-Integr. Manuf.*, vol. 37, pp. 233–250, Feb. 2016.

[18] E. M. Shakshuki, H. Malik, and T. Sheltami, "WSN in cyber physical systems: Enhanced energy management routing approach using software agents," *Future Generat. Comput. Syst.*, vol. 31, pp. 93–104, Feb. 2014.

[19] M. Chetto and A. Queudet, "A note on edf scheduling for real-time energy harvesting systems," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 1037–1040, Apr. 2014.

[20] M. Chetto, "Optimal scheduling for real-time jobs in energy harvesting computing systems," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 2, pp. 122–133, Jun. 2014.

[21] H. E. Ghor, M. Chetto, and R. H. Chehade, "A real-time scheduling framework for embedded systems with environmental energy harvesting," *Comput. Elect. Eng.*, vol. 37, no. 4, pp. 498–510, 2011.

[22] B. Zhang, R. Simon, and H. Aydin, "Harvesting-aware energy management for time-critical wireless sensor networks with joint voltage and modulation scaling," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 514–526, Jan. 2013.

[23] J. Lu and Q. Qiu, "Scheduling and mapping of periodic tasks on multi-core embedded systems with energy harvesting," in *Proc. Int. Green Comput. Conf. Workshops (IGCC)*, 2011, pp. 1–6.

[24] M. Khalgui, O. Mosbahi, Z. Li, and H.-M. Hanisch, "Reconfigurable multiagent embedded control systems: From modeling to implementation," *IEEE Trans. Comput.*, vol. 60, no. 4, pp. 538–551, Apr. 2011.

[25] J. Zhang *et al.*, "Modeling and verification of reconfigurable and energy-efficient manufacturing systems," *Discrete Dyn. Nature Soc.*, vol. 22, Mar. 2015, Art. no. 813476.

[26] S. Ostroumov, L. Tsiopoulos, J. Plosila, and K. Sere, "Formal approach to agent-based dynamic reconfiguration in networks-on-chip," *J. Syst. Archit.*, vol. 59, no. 9, pp. 709–728, 2013.

[27] J. Ye, Z. Li, and A. Giua, "Decentralized supervision of Petri nets with a coordinator," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 6, pp. 955–966, Jun. 2015.

[28] J. Zhang, M. Khalgui, Z. Li, G. Frey, O. Mosbahi, and H. B. Salah, "Reconfigurable coordination of distributed discrete event control systems," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 1, pp. 323–330, Jan. 2015.

[29] I. Khemaissia, O. Mosbahi, and M. Khalgui, "New automatic agent-based solutions for feasible reconfigurable mp-soc architectures," in *Proc. 14th Int. Conf. Appl. Concurrency Syst. Design (ACSD)*, Tunis, Tunisia, 2014, pp. 152–158.

[30] X. Wang, I. Khemaissia, M. Khalgui, Z. Li, O. Mosbahi, and M. Zhou, "Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 258–271, Jan. 2015.

[31] S. K. Baruah, "A general model for recurring real-time tasks," in *Proc. IEEE Real-Time Syst. Symp.*, Washington, DC, USA, Dec. 1998, p. 114.

[32] P. Ramanathan, "Overload management in real-time control applications using (m, k)-firm guarantee," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 6, pp. 549–559, Jun. 1999.

[33] W. Housseyni, O. Mosbahi, M. Khalgui, and M. Chetto, "Real-time scheduling of reconfigurable distributed embedded systems with energy harvesting prediction," in *Proc. IEEE/ACM 20th Int. Symp. Distrib. Simulation Real Time Appl. (DS-RT)*, Sep. 2016, pp. 171–178.

[34] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: A flexible real time scheduling framework," *ACM SIGAda Ada Lett.*, vol. 24, no. 4, pp. 1–8, 2004.

[35] Y. Xiang and S. Pasricha, "Run-time management for multicore embedded systems with energy harvesting," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 12, pp. 2876–2889, Dec. 2015.

[36] R. Idriss, A. Loukil, and M. Khalgui, "New middleware for secured reconfigurable real-time systems," in *Intelligent Software Methodologies, Tools and Techniques*. Cham, Switzerland: Springer, 2015, pp. 469–483.

[37] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.

[38] T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "An end-to-end multi-standard ofdm transceiver architecture using FPGA partial reconfiguration," *IEEE Access*, vol. 5, pp. 21002–21015, 2017, doi: 10.1109/ACCESS.2017.2756914.

WIEM HOUSSEYNI was born in Tunis, Tunisia, in 1989. She received the engineering degree in computer science from Tunis El Manar University, Tunis, in 2014.

She is currently pursuing the Ph.D. degree with the Computer Laboratory for Industrial Systems, National Institute of Applied Science and Technology, and the Research Institute of Communications and Cybernetics.

Her interests focus on real-time scheduling of reconfigurable distributed embedded systems with energy harvesting requirements.

OLFA MOSBAHI received the B.S. degree in computer science and the M.S. degree from Tunis El Manar University, in 1999 and 2002, respectively, the Ph.D. degree from the French Polytechnic Institute of Lorraine, France, in 2008. She did her Ph.D. thesis in computer science with inria, France. She was a part time Researcher with inria, and a temporary Lecturer with Nancy 2 University. She was also a Researcher with Martin Luther University, Germany.

She is currently an Assistant Professor in computer science with INSAT, Carthage University, Tunisia. She is actively involved in several European projects and also in other interesting international collaborations.

Dr. Mosbahi is a TPC Member of many conferences and different boards of journals.

MOHAMED KHALGUI received the B.S. degree in computer science from Tunis El Manar University, Tunis, Tunisia, in 2001, the M.S. degree in telecommunication and services from Henri Poincaré University, Nancy, France, in 2003, the Ph.D. degree from the National Polytechnic Institute of Lorraine, Nancy, in 2007, and the Habilitation Diploma degree in information technology (computer science) from the Martin Luther University of Halle-Wittenberg, Halle, Germany, in 2012, with Humboldt Grant.

He was a Researcher in computer science with the Institut National de Recherche en Informatique et Automatique, Rocquencourt, France, the ITIA-CNR Institute, Vigevano, Italy, the Systems Control Laboratory, Xidian University, Xi'an, China, and the KACST Institute, Riyadh, Saudi Arabia, a Collaborator with SEG Research Group, Patras University, Patras, Greece, the Director of the RECS Project, O3NEIDA, Canada, the Director of the RES Project, Synesis Consortium, Lomazzo, Italy, the Manager of the Cyna-RCS Project, Cynapsys Consortium, France, and the Director of the BROS and RWiN Projects, ARDIA Corporation, Germany.

He is currently a Professor with Jinan University, China. He has been involved in various international projects and collaborations. He is a TPC member of many conferences and different boards of journals.

ZHIWU LI (M'06–SM'07–F'16) received the B.S. degree in mechanical engineering, the M.S. degree in automatic control, and the Ph.D. degree in manufacturing engineering from Xidian University, Xi'an, China, in 1989, 1992, and 1995, respectively. He was with Xidian University in 1992.

He is currently with the Macau Institute of Systems Engineering, Macau University of Science and Technology, Macau, China.

He has listed in the book of Marquis entitled *Who's Who in the World* (27th edition, 2010). He is currently the Founding Chair of the Xi'an Chapter of the IEEE Systems, Man, and Cybernetics Society. He serves as a Frequent Reviewer for over 50 international journals, including *Automatica* and a number of the IEEE Transactions and many international conferences.

LI YIN received the B.E. degree in remote sensing from Wuhan University, Wuhan, China, in 2007, and the M.S. degree in GIS from the Institute of Remote Sensing Technology Application, CNNC Beijing Research Institute of Uranium Geology, Beijing, in 2014.

He is currently pursuing the Ph.D. degree in system control with the Macau University of Science and Technology, Macau, China.

His research interests include discrete-event systems and fault-tolerant dynamic systems with applications to manufacturing.