IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 5, 2018, accepted June 1, 2018, date of publication July 2, 2018, date of current version July 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2845942

Coherence and Feasibility of Real-Time Software Tasks in Networked Adaptive Systems

IMEN KHEMAISSIA[1,2,3], OLFA MOSBAHI[1,2], MOHAMED KHALGUI[1,2], ZHIWU LI[4,5], (Fellow, IEEE), AND TING QU[1]

[1]School of Electrical and Information Engineering, Jinan University, Zhuhai 519070, China
[2]National Institute of Applied Sciences and Technology, University of Carthage, Tunis 1080, Tunisia
[3]King Khalid University, Abha 62529, Saudi Arabia
[4]Institute of Systems Engineering, Macau University of Science and Technology, Taipa 999078, Macau, 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 in part by the National Natural Science Foundation of China under Grant 61374068 and in part by the Science and Technology Development Fund, MSAR, under Grant 122/2017/A3.

ABSTRACT This paper deals with the dynamic reconfiguration of networked devices linked by a controller area network (CAN). Each device runs dependent periodic and aperiodic software tasks and can be adapted to any evolution in its environment. A reconfiguration is a dynamic scenario that activates–deactivates the deployed devices, adds–removes software tasks, or changes the network traffic according to user requirements. Nevertheless, such a scenario can trigger the execution of new–old tasks to violate real-time deadlines or to possibly increase the energy consumption. Moreover, a reconfiguration that adapts dependent tasks in different devices can modify the network traffic and some deadlines of frames can be violated too. To resolve all these problems that can happen after concurrent distributed reconfiguration scenarios, we propose a dynamic methodology called Cynapsys-reconfigurable control system that allows coherent distributed behaviors of devices after any scenario. This run-time automatic strategy based on a multi-agent architecture is achieved in five steps: 1) applying reconfiguration scenarios on the system devices to update their services; 2) coordination between devices after any reconfiguration for their coherence; 3) feasibility analysis of each reconfigured device; 4) verification of CAN feasibility; and 5) reconfigurable frame packing. A developed tool is applied to a case study for the evaluation of the proposed contribution.

INDEX TERMS Real-time control system, reconfiguration, CAN, low-power scheduling, multi-agent, frame-packing.

ABBREVIATIONS AND GLOSSARY

RCS	Reconfigurable Control System
Cyna-RCS	Cynaspsys Reconfigurable Control System
CAN	Controller Area Network
RCB	Reconfigurable CAN Bus
EDF	Earliest Deadline First
RM	Rate Monotonic
OPCP	Original Priority Ceiling Protocol
IPCP	Immediate Priority Ceiling Protocol
WCET	Worst Case Execution Time
WCTT	Worst Case Transmission Time
SRP	Stack Resource Policy
FP	Fixed Priority
AT	Architecture Token
ST	Scheduling Token
DT	Data Token

dev_i	Device i
Re^i	Reconfiguration in dev_i
Ω^i_{pb}	Set of periodic tasks before Re^i
Ω^i_{pa}	Set of periodic tasks after Re^i
Ω^i_{ab}	Set of aperiodic tasks before Re^i
Ω^i_{aa}	Set of aperiodic tasks after Re^i
Ψ^b_{mp}	Set of exchanged periodic messages before Re^i
Ψ^a_{mp}	Set of exchanged periodic messages after Re^i
Ψ^b_{ma}	Set of exchanged aperiodic messages before Re^i
Ψ^a_{ma}	Set of exchanged aperiodic messages after Re^i
n	Number of devices
nt^i_b	Number of periodic tasks in Ω^i_{pb} before Re^i

nt_a^i	Number of periodic tasks in Ω_{pa}^i after Re^i
mt_b^i	Number of aperiodic tasks in Ω_{ab}^i before Re^i
mt_a^i	Number of aperiodic tasks in Ω_{aa}^i after Re^i
mp_b	Number of periodic messages before Re^i
mp_a	Number of periodic messages after Re^i
map_b	Number of aperiodic messages before Re^i
map_a	Number of aperiodic messages after Re^i
U^i	Total utilization in dev_i
U_b^i	Utilization in dev_i before Re^i
U_a^i	Utilization in dev_i after Re^i
U_{pb}^i	Utilization of periodic tasks before Re^i
U_{pa}^i	Utilization of periodic tasks after Re^i
U_{ab}^i	Utilization of aperiodic tasks before Re^i
U_{aa}^i	Utilization of aperiodic tasks after Re^i
P^i	Energy consumed by dev_i
P_b^i	Energy consumed by dev_i before Re^i
P_a^i	Energy consumed by dev_i after Re^i
P_{pb}^i	Energy consumed by periodic tasks in dev_i before Re^i
P_{pa}^i	Energy consumed by periodic tasks in dev_i after Re^i
RA_i	Request agent in dev_i
CA_i	Coordination agent in dev_i
Ag_M	Master agent to control RCB
Ag_i	Slave agent in dev_i
τ_j^i	j-th task in dev_i
R_j^i	Release time of periodic task τ_j^i in dev_i
T_j^i	Period of periodic task τ_j^i in dev_i
T_Δ^i	Constant period of tasks in dev_i after Re^i
D_j^i	Deadline of periodic task τ_j^i in dev_i
C_j^i	WCET of periodic task τ_j^i in dev_i
C_Δ^i	Constant WCET of tasks in dev_i after Re^i
S_j^i	Static priority of periodic task τ_j^i in dev_i
d_k^i	Deadline of aperiodic task τ_k^i in dev_i
r_k^i	Release time of aperiodic task τ_k^i in dev_i
c_k^i	WCET of aperiodic task τ_k^i in dev_i
$\lambda_r^{i,k}$	Release time rate of aperiodic task τ_k^i in dev_i
$\lambda_c^{i,k}$	Worst-case execution time of aperiodic task τ_k^i in dev_i
B_j^i	Blocked time of task τ_j^i in dev_i
Pr_j^i	Priority assigned to each task τ_j^i in dev_i
$m_{\tau_k^i,\tau_l^j}$	Message to be exchanged between τ_k^i and τ_l^j
$T_m(\tau_k^i, \tau_l^j)$	Regular inter-arrival time of $m_{\tau_k^i,\tau_l^j}$
$T_{m,\Delta}$	Constant period of messages after Re^i
$S_m(\tau_k^i, \tau_l^j)$	Spent time to transmit $m_{\tau_k^i,\tau_l^j}$
$S_{m,\Delta}$	Constant spent time to transmit messages after Re^i
$D_m(\tau_k^i, \tau_l^j)$	Absolute deadline of $m_{\tau_k^i,\tau_l^j}$
$Z_m(\tau_k^i, \tau_l^j)$	Relative size of $m_{\tau_k^i,\tau_l^j}$
$Pr_m(\tau_k^i, \tau_l^j)$	Static priority of $m_{\tau_k^i,\tau_l^j}$
$U_{C,b}^{pm}$	Utilization of periodic messages before Re^i
$U_{C,a}^{pm}$	Utilization of periodic messages after Re^i
$U_{C,b}^{am}$	Utilization of aperiodic messages before Re^i
$U_{C,a}^{am}$	Utilization of aperiodic messages after Re^i
$U_{C,b}$	Total utilization in RCB before Re^i
$U_{C,a}$	Total utilization in RCB after Re^i
$Inc_{\tau_h^k}$	Set of devices that can execute τ_h^k
$U_{\tilde{a}}$	Utilization average of all devices

I. INTRODUCTION

Nowadays, the new generations of control systems [7], [8], [21], [41] and real-time control systems face new industrial challenges in terms of energy [23]–[26]. In fact, networked devices are widely used in these systems such as current vehicles that use controller area networks (CAN) [27] to link their distributed software tasks in order to offer more functionalities and services [28]. A system device is considered to be composed of dependent periodic and aperiodic software tasks allowing the control of physical processes. Due to possible external or also internal disturbances that may occur as an effect of a software/hardware failure, a system can be automatically adapted by adding/removing/updating software tasks to/from/in the devices. A reconfiguration is considered as a modification of the system behavior as a response to this evolution. A hardware or software reconfiguration scenario [29] is considered to be any run-time operation that adapts the system's behavior in each device and also on the network [30].

A reconfiguration is considered as a mode change [10]: Any reconfiguration scenario changes at run-time the initial set of tasks to another one. Also, the execution frequency of reconfiguration scenarios is low relative to that of control tasks. In fact, the power consumed during reconfiguration is assumed to be negligible compared with the power consumed by the execution of control tasks. We note also that some real-time constraints such as end to end response time bounds may be violated and the system becomes infeasible. The current paper deals with an original issue which is the automatic concurrent reconfigurations of distributed real-time dependent tasks under energy constraints. This paper aims to reconfigure networked devices for coherent execution of periodic and aperiodic dependent software tasks under energy and real-time constraints. We deal in particular with the feasible construction of the reconfigurable traffic on the considered controller area network (CAN).

The concurrent reconfigurations should be treated correctly at run-time to avoid any incoherence between the networked devices of a system. Many projects and studies are dealing today with the reconfiguration of real-time control systems under real-time and energy constraints [4], [6], [31], [32], [37]. Several works and algorithms are proposed for

low-power scheduling [33]. A large number of algorithms are developed to schedule real-time tasks [34]. This paper reports a comprehensive and complete methodology which is partially based on the following initial technical advancements that are stated as follows: The work reported in [30] suggests a solution for the real-time monoprocessor scheduling of reconfigurable tasks sharing resources, the study in [38] resolves the coordination problem (without real-time constraints) and the result in [29] manages the real-time scheduling of messages on a network (with real-time and energy constraints). The current paper deals with the global feasibility since the correctness of tasks in devices depends on that of messages on the network. Indeed, the problem considered in our research is different from those in previous papers since we deal with the feasibility of distributed reconfigurable real-time tasks under energy constraints and end to end response time bounds where the feasibility is used to mention whether the real-time constraints are respected or not. If the tasks do not miss their deadlines, then the system is considered as feasible.

As far as we know, the feasibility of the considered distributed system after any reconfiguration depends on the coherence between devices, the feasibility of tasks in each device, the correct generation of frames and their feasibility on the network. In fact, if a deadline is missed, then the cause can be related to a coordination problem, to a software task assigned to a particular device, or to a message on the network. The problem is global and needs new solutions that consider the internal behaviors of devices or their exchanged traffic on the network. We aim in this paper to propose a complete dynamic methodology to be applied automatically at run-time after any reconfiguration scenario. This methodology is original since it resolves the whole problem by arranging the execution of tasks in devices and messages on the network. The feasibility of any task depends on its coherence with related tasks and also on the feasibility of the exchanged messages. So far, no studies are found to resolve the global problem addressed in the current paper.

We consider n reconfigurable devices linked by CAN and run under energy and real-time constraints. These devices are implemented by dependent periodic and aperiodic tasks [30] which can exchange messages on the network. The messages are generally loaded in packets according to several solutions such as the frame-packing [39], [40]. In order to improve the performance and reliability of these devices, reducing their power consumption becomes an ever-increasing major concern. The reliability means that each device should continue its execution after any scenario. Nevertheless, the addition of tasks/messages can increase the energy consumption which can be a problem when the system is obliged to stop its execution. Therefore, in this work, we deal with the following problems: i) Coordination between the different devices of the system, ii) Management of the real-time scheduling of reconfigurable periodic/aperiodic tasks sharing resources, and iii) Management of the real-time scheduling of messages on a network under energy constraints.

In order to resolve all these problems that can happen after different concurrent distributed reconfiguration scenarios, the proposed run-time methodology Cyna-RCS allows coherent distributed behaviors of controllers after any reconfiguration scenario. This methodology is the scientific result of the European project Mobidoc that is called ''Low-power reconfigurations of real-time embedded systems LR-RTES'' and handled by Cynapsys corporation where it is applied to real industrial case studies.[1] It makes the following contributions: 1) Coordination between devices: A new communication protocol based on the token passing protocol [22] is developed, 2) System/CAN feasibility: New technical solutions are offered to satisfy the real-time constraints such as the modification of parameters and Bin-packing relocation under energy constraints, 3) Reconfigurable frame-packing: An algorithm is proposed to construct dynamically the frames under the bandwidth minimization, and 4) Energy constraints satisfaction: after any reconfiguration scenario that adds new tasks and messages, the energy is evaluated. Note that the real-time constraints are satisfied before any reconfiguration. Then before effectively applying any new scenario, the proposed multi-agent architecture computes (if needed) the new solutions that allow for the new and old tasks to meet the related deadlines. In fact, the new and old tasks cannot start their execution with the new parameters before being sure that the related deadlines are satisfied. To the best of our knowledge, no one in all related works deals with dependent-indepdent periodic-aperiodic reconfigurable tasks sharing resources and exchanging periodic-aperiodic messages in networked devices. Our methodology which is composed of different steps is original since it gives functional, real-time and energy guarantees that no one in all related works deals with.

The rest of the paper is organized as follows: Section 2 exposes some related works and Section 3 formalizes the reconfigurable control system considered in this paper. We present in Section 4 the proposed methodology for the system feasibility, reconfigurable CAN and coordination between the devices. Section 5 proves the Cyna-RCS correctness, and Section 6 presents the implementation of the paper's contribution. A case study, simulation and discussion are treated in Section 7. Section 8 concludes this research work.

II. RELATED WORKS

In the literature, there are a lot of successful studies addressing the scheduling problem of real-time tasks [34], [49]. The work in [34] suggests the earliest deadline first (EDF) and rate monotonic policies (RM) for the scheduling of periodic tasks. The original priority ceiling protocol (OPCP) and immediate priority ceiling protocol (IPCP) are presented in [13] in order to manage the tasks that share resources. The goal of the priority ceiling protocol is to prevent deadlocks and reduce the

[1] For confidentiality reasons, Cynapsys does not allow the authors to give more details on this project. The readers, if interested, can contact directly the company (www.cynapsys.de).

blocking to at most one critical section [48]. The work in [36] proposes a stack resource policy (SRP) that allows processes with different priorities to share a single run-time stack. The *(m,k)*-firm model [12] is used in a degraded mode to better characterize the timing constraints of real-time streams.

Also, various related works have been dedicated to develop reconfigurable real-time control systems. On the one hand, the research in [21] focuses on low-power dynamic reconfigurations of synchronous real-time control systems. The work in [42] proposes new solutions to schedule reconfigurable real-time systems implemented with independent periodic and probabilistic tasks under real-time constraints. The authors have proposed an agent-based architecture in order to reduce the power consumption after applying any reconfiguration scenario. This work does not treat the case of tasks that share resources. Moreover, it does not present distributed solutions and is not interested in traffic on the network. In [31], the authors define an architecture of reconfigurable multi-agent systems where a reconfiguration agent modeled by nested state machines is affected to each device of the execution environment to apply local automatic reconfigurations, and a coordination agent is proposed for any coordination between devices in order to guarantee safe and adequate distributed reconfigurations. In [43], a new algorithm is developed to guarantee the schedulability of the bandwidth minimization in automotive applications. Many techniques are proposed to adjust the CPU voltage and frequency dynamically to reduce the power consumption such as dynamic voltage and frequency scaling (DVFS) [14]–[16]. The research in [29] is interested in the dynamic reconfiguration of the frame packing as well as the traffic of real-time packets on a CAN network. The work in [11] addresses the problem of the shortage of energy in the case of the WCET's increase. As a solution, the mixed criticality (MC) paradigm is proposed to increase the processor speed if high level HI-criticality tasks need more than their WCET in a low level requirement. The work reported in [30] deals with a new middleware that is proposed to dynamically reconfigure RT-Linux based systems. An intelligent control agent is proposed in order to provide run-time solutions that reduce the power consumption while satisfying dynamically the real-time constraints. In order to control tasks, Mancuso *et al.* [9] describe a branch and bound algorithm for finding the optimal priority assignment. The research reported in [38] develops a new protocol for the feasible coordination between STM32F4 microcontrollers of a reconfigurable distributed system.

To the best of our knowledge, no one in all related works proposes a complete methodology for the adaptation of reconfigurable networked devices composed of dependent periodic and aperiodic tasks, and linked by CAN under real-time and power constraints. Moreover, no work is found to deal with the global problem of coherence between devices, feasibility of tasks, correct frame packing and also feasibility on the network. The contribution of the current paper is original and applied to a case study for the required evaluation of its performance.

III. RECONFIGURABLE DISTRIBUTED CONTROL SYSTEM

We formalize in this section a reconfigurable control system to be composed of n reconfigurable devices dev_i ($i \in [1..n]$) linked by RCB.

1) OS LEVEL

Each device is assumed to be composed of a set of periodic and aperiodic tasks under precedence and shared resources constraints. Each periodic task τ_j^i assigned to dev_i may produce many jobs [34] and is characterized by five parameters: release time R_j^i, period T_j^i, deadline $D_j^i = T_j^i$, WCET C_j^i, and static priority S_j^i. The static priorities of tasks are assigned by users according to their functional importance. Each aperiodic task τ_k^i assigned to dev_i ($i \in [1..n]$) is characterized by a deadline d_k^i, a release time r_k^i, and a WCET c_k^i. According to [18] and [42], the tasks have Markovian arrival and service processes on a unique processor. They request the processor according to a Poisson process and their WCETs are exponentially distributed. Thus, it arrives according to the Poisson process based on a Poisson distribution with a rate $\lambda_r^{i,k}$. Also, its WCET is exponentially distributed with the same mean value $\frac{1}{\lambda_C^{i,k}}$. We note that two dependent aperiodic tasks with precedence constraints have the same distributions. Let us consider also in this study that the task parameters can be changed at run-time without any reservation. Let Ω_{pb}^i and Ω_{ab}^i be the sets of periodic and aperiodic tasks respectively that implement dev_i ($i \in [1..n]$) before the application of a reconfiguration scenario Re^i. This scenario is assumed to change the software implementation of the devices as well as the traffic on the network. This device can be reconfigured dynamically by authorizing the addition/removal/update of tasks. The initial utilization U_b^i of dev_i ($i \in [1..n]$) before Re^i is the sum of U_{pb}^i and U_{ab}^i as given by the following equation:

$$U_b^i = U_{pb}^i + U_{ab}^i, \tag{1}$$

where $U_{pb}^i = \sum_{k=1}^{nt_b^i} \frac{C_k^i}{T_k^i} + \frac{B_k^i}{T_k^i}$ is the utilization of periodic tasks before Re^i, nt_b^i is the number of periodic tasks in Ω_{pb}^i before Re^i ($i \in [1..n]$) and B_k^i is the time spent by a task with a higher priority when blocked and awaits the termination of a task with a lower priority [17], [29]. Note that $C_k^i + B_k^i$ is equal to the sum of the execution time of any task τ_p^i and the waiting blocking time of any task with a lower priority than τ_p^i which stills blocked during B). When a task is blocked, the other tasks will be scheduled with a same defined policy that is the EDF algorithm [34]. Also, we use IPCP [13] to manage the shared resources since it guarantees just one blocking time. The processor utilization U_{ab}^i is formalized by:

$$U_{ab}^i = \sum_{k=1}^{mt_b^i} \frac{\lambda_r^{i,k}}{\lambda_C^{i,k}}, \tag{2}$$

where mt_b^i is the number of aperiodic tasks in Ω_{ab}^i. The energy consumed by dev_i is proportional to the utilization according to [21] and [44] (i.e., $P^i \propto U^i$). For the tasks with precedence constraints, we turn them into independent ones according to [30]. To manage the precedence constraints, we assign firstly a priority Pr_j^i to each task τ_j^i in dev_i. Note that the deadlines of two tasks τ_k^i and τ_h^i under precedence constraint are assigned as follows: if $S_k^i > S_h^i$, then $D_k^i = min\{D_k^i, (D_h^i - C_h^i)\}$. Finally, note that the total utilization of dev_i ($i \in [1..n]$) after Re^i is equal to $U_a^i = U_{pa}^i + U_{aa}^i$

2) RCB LEVEL

We formalize the exchanged messages on RCB between the different tasks assigned to the different devices of the system. We denote in the following by $m_{\tau_k^i,\tau_l^j}$ the message to be exchanged between τ_k^i and τ_l^j which are assigned to dev_i and dev_j respectively ($i, j \in [1..n]$). According to [39], a message is segmented into multiple frames F. Each message is segmented into f frames. In this work, f is assumed to be equal to 1. Each periodic message $m_{\tau_k^i,\tau_l^j}$ is characterized according to [46] by (i) Period $T_m(\tau_k^i, \tau_l^j)$: The regular inter-arrival time, (ii) Worst case transmission time (WCTT) $S_m(\tau_k^i, \tau_l^j)$: The spent time to transmit the message, (iii) Deadline $D_m(\tau_k^i, \tau_l^j)$: The absolute deadline which is equal to $D_m(\tau_k^i, \tau_l^j) = (D_j^l - C_j^l)$, (iv) Size $Z_m(\tau_k^i, \tau_l^j)$: The size of the message, and (v) Priority $Pr_m(\tau_k^i, \tau_l^j)$, such that the highest static priority is equal to 1. Each aperiodic message $m_{\tau_a^i,\tau_b^j}$ arrives according to the Poisson process based on a poisson distribution with a rate $\lambda_r^{i,a}$. We consider also that its WCTT is exponentially distributed with the same mean value $\frac{1}{\lambda_C^{i,a}}$.

Let Ψ_{mp}^b (resp, Ψ_{ma}^b) be the set of exchanged periodic (resp, aperiodic) messages on RCB before Re^i. The total utilization in RCB before Re^i is equal to the sum of $U_{C,b}^{pm}$ and $U_{C,b}^{am}$ with (i) $U_{C,b}^{pm} = \sum_{m_{\tau_k^i,\tau_l^j} \in \Psi_{mp}^b}^{1..mp_b} \frac{S_m(\tau_k^i,\tau_l^j)}{T_m(\tau_k^i,\tau_l^j)}$ where mp_b is the number of periodic messages in Ψ_{mp}^b before Re^i, (ii) $U_{C,b}^{am} = \sum_{m_{\tau_k^i,\tau_l^j} \in \Psi_{ma}^b}^{1..map_b} \frac{\lambda_r^{i,k}}{\lambda_C^{i,k}}$, where map_b is the number of aperiodic messages in Ψ_{ma}^b before Re^i. Note that EDF is used in theory to check the CAN feasibility and FIFO is used in a real simulation to schedule the messages. We suppose that CAN is preemptive to verify the feasibility in CAN, i.e, we must ensure $U_{C,b} <= 1$, i.e., in theory we use EDF to verify the feasibility (off-line). Thus, We consider the CAN as a virtual processor and the messages as virtual tasks in order to verify the related real-time constraints. The proposed solution can be adapted to TTCAN and FLEXRay. However, we have chosen CAN which is low-cost and well applied to various industrial applications compared with TTCAN and FlexRay [45].

IV. NEW METHODOLOGY FOR RECONFIGURABLE CONTROL SYSTEMS

We propose a new methodology Cyna-RCS to be dynamically applied to the feasible reconfigurations of distributed control systems. Cyna-RCS is a methodology to be applied automatically on different facets after any reconfiguration scenario to guarantee: (1) The coherence between devices, (2) The feasibility of tasks in each device, and (3) The correct generation and feasibility of updated frames on the network. In this work, we consider three forms of predicted reconfigurations to be locally applied at run-time: (i) *Architectural Reconfiguration* allowing the addition/removal of tasks to/from devices, (ii) *Scheduling Reconfiguration* allowing the modification of the composition of tasks without modifying their architecture, and (iii) *Data Reconfiguration* allowing the modification of values of data while keeping the same architecture and scheduling. The idea of the current paper is as follows: A device dev_i ($i \in [1..n]$) wants to apply a deep reconfiguration Re^i that changes the architecture (addition/removal of tasks), the scheduling (composition of tasks) and the values of some data. This reconfiguration should be applied with the agreement of remote devices in order to allow a coherent and correct distributed behavior of the whole system. Once the coordination is guaranteed, the feasibility of the OS/RCB should be verified and the feasible messages in RCB should be constructed. We note that the feasibility of the system depends on the coherent distributed behaviors of devices, their real-time feasibility under energy constraints, and the real-time feasibility of messages on the network. Before we present the steps of this methodology, let us describe the proposed multi-agent based architecture.

A. MULTI-AGENT BASED ARCHITECTURE

Figure 1 presents the architecture of the defined agents that manage the coordination as well as the schedulability analysis in the networked devices.

The following multi-agent architecture for reconfigurable networked control systems: (i) Coordination between devices: (i.a) Request Agent: A request agent RA_i handles the local software reconfigurations in each dev_i ($i \in [1..n]$), (i.b) Coordination Agent: For each device dev_i ($i \in [1..n]$), a coordination agent CA_i is proposed to coordinate with remote controllers to look for the application of the desired reconfigurations from RA_i. CA_i sends a request to the remote coordinator agents. These coordinators analyze this request with their request agents to possibly decide if it is acceptable or not. If all the remote coordinators of the different remote devices agree to the request of CA_i, then this reconfiguration is applied since the coherence is guaranteed. (ii) OS/RCB Feasibility and Frame-Packing: (ii.a) Ag_M: denotes a master agent that (1) Controls the evolution of the whole system's environment before applying software-hardware reconfigurations, (2) Checks the feasibility of CAN bus after any reconfiguration scenario and (3) Handles the reconfiguration of the frame-packing, (ii.b) Ag_i ($i \in [1..n]$):

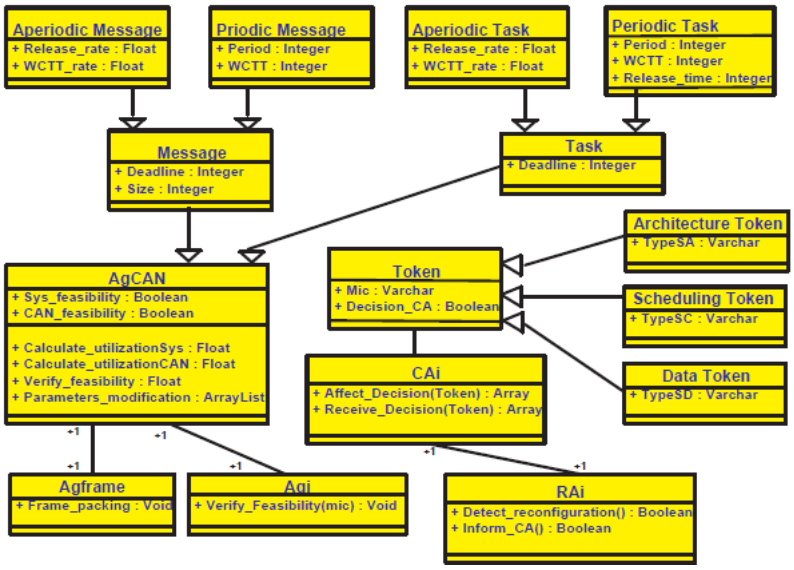

FIGURE 1. Multi-agent architecture of RCS.

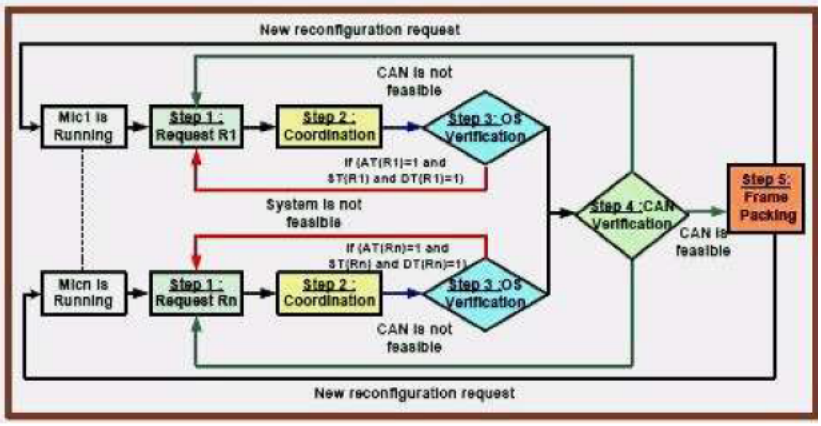

FIGURE 2. Cyna-RCS methodology.

denotes the slave agent that controls dev_i to check if any local reconfiguration scenario increases the energy or triggers some local tasks to violate the corresponding deadlines. We note that this work is not interested in the time delays related to the coordination process since the main goal is to guarantee the feasibility of the networked devices after any reconfiguration scenario. Note that the entity that is related to tokens will be described in Section IV.B.1.

B. METHODOLOGY

A dynamic methodology Cyna-RCS is proposed to be performed automatically when concurrent distributed reconfiguration scenarios are applied to add/remove/update tasks in controllers and messages on the bus. This methodology is applied step by step: (i) To perform a required coordination between the distributed devices for their coherence after any reconfiguration, (ii) To arrange the feasibility of tasks in each controller, (iii) To apply a required frame packing that updates the exchanged messages, and (iv) Finally the last step arranges the exchanged messages on the bus. This methodology is original and complete since it guarantees a feasible distributed architecture after any reconfiguration. We propose the following steps that encode this methodology: (i) Step 1: Dynamic application of reconfiguration scenarios allowing the addition/removal/update of tasks/messages in the different devices and also in RCB, (ii) Step 2: Coordination between devices to select a list of scenarios among concurrent ones, (iii) Step 3: Feasibility analysis of real-time tasks on each reconfigured device, (iv) Step 4: Feasibility analysis on RCB, and (v) Step 5: Reconfiguration of the frame-packing in each device dev_i ($i \in [1..n]$) if RCB is feasible. Figure 2 summarizes the whole contribution of this current research by showing the different states of the system and the proposed strategy to resolve the destabilization that can happen after applying any reconfiguration scenario. We present in particular the

different steps to be applied in any device after any reconfiguration scenario.

1) RECONFIGURATION AND COORDINATION

In order to coordinate between the different devices dev_i ($i \in [1..n]$), we define three types of tokens: (i) Architecture Token, (ii) Scheduling Token, and (iii) Data Token. A reconfiguration scenario is applied when a coordination agent CA_i ($i \in [1..n]$) sends first an Architecture Token in order to have an authorization from remote devices. This token has a priority, defined manually according to user requirements, and can be rejected from the network if another concurrent Architecture Token with a higher priority is sent. CA_i awaits the authorization from all remote devices before considering effectively this reconfiguration. The architecture of the different distributed tasks is fixed at this level in the different devices. CA_i sends now a new Scheduling Token in order to define the scheduling of the local tasks and also in the rest of devices. When an agreement is received, CA_i sends then Data Token in order to ask the application of reconfigurations allowing the modification of data. This step-by-step reconfiguration is useful since it is not important to apply architectural, scheduling and data reconfigurations at same time.

We formalize the proposed tokens as follows: (i) Architecture Token AT: Defined by an array of size $n + 2$ such that (i.a) Column 1: Identifier of a new software architecture after adding/removing tasks, (i.b) Column 2: Priority of this token, (i.c) Columns 3..n+2: Decisions of CA_i ($i \in [1..n]$) (1 or 0). After a reconfiguration request, each CA_i will answer by 1 if the reconfiguration is authorized, 0 if rejected. (ii) Scheduling Token ST: Defined by an array of size $n + 2$ such that (ii.a) Column 1: Identifier of a new scheduling strategy after adding/removing tasks to/from devices, (ii.b) Column 2: Priority of this token, (ii.c) Columns 3..n+2: Decisions of CA_i ($i \in [1..n]$) (1 or 0). (iii) Data Token DT: Defined by an array of size $n + 2$ such that (iii.a) Column 1: New values of data after adding/removing tasks to/from devices, (iii.b) Column 2: Priority of this token, (iii.c) Columns 3..n+2: Decisions of CA_i ($i \in [1..n]$) (1 or 0).

We propose a protocol for coherent distributed reconfigurations of networked devices. This protocol is based on the following three parts: (i) Part 1: RA_i of a particular device dev_i ($i \in [1..n]$) detects a reconfiguration request, (ii) Part 2: Coordination between RA_i and CA_i: CA_i receives this request and checks if the related devices agree this request, (iii) Part 3: Communication between CA_i and the remote devices.The reconfiguration can be halted when Architecture, Scheduling, Data Tokens are not authorized. The goal of this protocol is to have useful and optimal distributed reconfigurations: If the devices do not agree a suggested new architecture, then it is not important to go to scheduling and so on. Algorithm 1 is developed to show the behavior of RA_i ($i \in [1..n]$) when it sends a request to CA_i. Algorithm 2 shows the behavior of a remote CA_j when it receives this request. This algorithm is applied at all the different levels of reconfiguration.

Algorithm 1 Communication Protocol

Variables:
Input: Reconfiguration request (AT, ST, DT) from RA_i.
int CA_i; // Coordination Agent of dev_i($i \in [1..n]$)
int RA_i; // Request Agent of dev_i

1: **if** (RA_i sends an Architecture Request ($AT(RA_i)$)) **then**
2: **if** (answer(CA_i)==1) **then**
3: Invoke Algorithm 2; // Receives all the answers from the remote devices in the architectural level.
4: **end if**
5: **else**
6: **break**;
7: **end if**
8: **if** (RA_i sends a Composition or Scheduling Request ($ST(RA_i)$)) **then**
9: **if** (answer(CA_i)==1) **then**
10: Invoke Algorithm 2; // Receives all the answers from the remote devices in the composition level.
11: **end if**
12: **else**
13: **break**;
14: **end if**
15: **if** (RA_i sends a Data Request ($DT(RA_i)$)) **then**
16: **if** (answer(CA_i)==1) **then**
17: Invoke Algorithm 2; // Receives all the answers from the remote devices in the data level.
18: **end if**
19: **else**
20: **break**;
21: **end if**
Output: Reconfiguration request of RA_i treated by CA_i.

Algorithm 2 is invoked in Algorithm 1 for each reconfiguration level. Algorithms 1 and 2 are with complexity $O(n^2)$.

A new parameter Thd^r is defined in order to guarantee a safe behavior. If the coordination time exceeds this parameter, then all the reconfiguration requests are rejected and we keep the same configuration. Also, another new parameter Pr_{reconf} where all reconfiguration scenarios are received at any time but only is treated at $t = K * Pr_{reconf}$, with $K \in IN$. Thanks to this solution, the system is stable during Pr_{reconf}, which is required to support the related services. It is worth mentioning that in this work we are not interested in the causes of reconfiguration scenarios, i.e., when they occur; what exactly happens; how the system is brought back to stability. The main goal of this research is to study the schedulability analysis of new and old tasks and the related messages.

2) FEASIBILITY ANALYSIS OF DEVICES

Once the coordination is finished, we move to the next step dealing with the OS feasibility in each reconfigured device dev_i ($i \in [1..n]$). We propose different possible solutions to

Algorithm 2 Communication Between Coordination Agents

Variables:
int CA_j; // Coordination Agent of $dev_j (j \in [1..n])$.
int RA_j; // Request Agent of dev_j.
int l=0; // Counter of devices.
boolean RR; // reconfiguration decision for each level
Input: Table of integers: $Mat_{decision}$
Table containing the decision of devices for each level.

1:
2: **for** each remote CA_j **do**
3: CA_j gives its answer in $Mat_{decision}$;
4: **end for**
5: **for** $(l = 1; l \leq size(Mat_{decision}); l++)$ **do**
6: **if** $Mat_{decision}[l] == 0$ **then**
7: Message="Reconfiguration is not authorized";
8: RR =0;
9: **break**;
10: **end if**
11: **end for**
12: Message="Reconfiguration is authorized";
13: RR =1;
Output: $Mat_{decision}$ treated by remote devices.

re-obtain the feasibility of the system after any reconfiguration scenario. The goal of this study is to just define these solutions where only one can be applied at run-time.

a: SOLUTION 1. MODIFICATION OF TASK PARAMETERS

The utilization must be changed in order to guarantee a stable energy consumption. A technical solution is proposed by Ag_M to modify the parameters of all the old and new tasks after the reconfiguration scenario. In this work, we suggest the modification of periods or WCETs as a solution to meet all the assumed constraints after any scenario [42]. In order to minimize the device utilization, the new constant period of tasks T^i_Δ assigned to dev_i $(i \in [1..n])$ is calculated as follows

$$T^i_\Delta = \left\lceil \frac{\sum_{k=1}^{nt^i_a} C^i_k + B^i_k}{U^i_{pb}} \right\rceil \tag{3}$$

where nt^i_a is the number of periodic tasks of Ω^i_{pa} in dev_i $(i \in [1..n])$ after Re^i. Note that a new value P_{max} is defined to adjust the quality of control. If the newest value of periods exceeds P_{max}, then the obtained value will not be considered and another solution will be applied.

After modification of periods, the new device utilization of periodic tasks U^i_{pa} is represented by

$$U^i_{pa} = \sum_{k=1}^{nt^i_a} \frac{C^i_k + B^i_k}{T^i_\Delta} \tag{4}$$

We formulate the new power consumption of the periodic tasks as follows

$$P^i_{pa} \propto (U^i_{pa})^2 \tag{5}$$

The new constant worst case execution time C^i_Δ of tasks assigned to any device dev_i $(i \in [1..n])$ is expressed by

$$C^i_\Delta = \begin{cases} \left\lfloor \dfrac{U^i_{pb} - \sum_{k=1}^{nt^i_a} \frac{B^i_k}{T^i_k}}{\sum_{k=1}^{nt^i_a} \frac{1}{T^i_k}} \right\rfloor \\ 1, \quad if \ \dfrac{U^i_{pb} - \sum_{k=1}^{nt^i_a} \frac{B^i_k}{T^i_k}}{\sum_{k=1}^{nt^i_a} \frac{1}{T^i_k}} \leq 0 \end{cases} \tag{6}$$

After the modification of WCETs, the new device utilization of periodic tasks U^i_{pa} is given by

$$U^i_{pa} = \sum_{k=1}^{nt^i_a} \frac{C^i_\Delta + B^i_k}{T^i_k} \tag{7}$$

The new power consumption is as follows

$$P^i_{pa} \propto (U^i_{pa})^2 \tag{8}$$

We prove in [30] that the constant rate $\lambda^{i,k}_C$ of each set of aperiodic tasks Ω^i_{aa} is modified to be:

$$\lambda^{i,k}_C = \lambda^{i,k}_C \times \frac{U^i_{aa}}{U^i_{ab}} \tag{9}$$

The new device utilization of aperiodic tasks is calculated by

$$U^i_{aa} = \sum_{k=1}^{mt^i_a} \frac{\lambda^{i,k}_r}{\lambda^{i,k}_C} \tag{10}$$

where mt^i_a is the cardinality of Ω^i_{aa} after Re^i. We note finally that the modification of these parameters can impact the blocking factors of tasks. In this case, we propose to recompute these factors and analyze the feasibility of the corresponding device after any reconfiguration scenario.

b: SOLUTION 2. BIN-PACKING RELOCATION

Bin-packing algorithm [47] can be used in this work to minimize the utilization by the relocation of tasks according to different conditions that will be mentioned below. Bin-packing is characterized by five heuristics: First Fit Decreasing (FFD), Best Fit Decreasing (BFD), Worst Fit Decreasing (WFD), Next Fit Decreasing (NFD) and Random Fit Decreasing (RFD). Based on them, Ag_M can reconfigure the system on two levels to be temporally feasible with low-power consumption. Note that we cannot relocate the aperiodic tasks since they arrive with two rates. Two levels must be done for periodic tasks:

- Level 1: We relocate them by using one of the proposed Bin-packing algorithms into activated devices,

- Level 2: We modify their parameters by following Solution 1 once tasks are re-located by applying Bin-packing.

To apply Bin-packing, we should order tasks in an ascending order and devices in a descending one according to their utilization. In order to re-locate task τ_h^k to dev_i ($i \in [1..n]$), two conditions should be satisfied:

- Condition 1: $dev_i \in Inc_{\tau_h^k}$,
- Condition 2: $U^i \leq \lceil U_{\tilde{a}} \rceil$.

where $Inc_{\tau_h^k}$ is the set of devices that can execute τ_h^k and $U_{\tilde{a}}$ is the utilization average of all devices.

c: SOLUTION 3. REMOVAL OF TASKS

As a third solution, Ag_M proposes the removal of some tasks according to their priorities. Each slave agent Ag_i decides which task can be removed. Before any removal, Ag_i should verify if the dependent tasks are with a lower priority to remove them too. Otherwise, it is not acceptable to remove this task.

3) CAN FEASIBILITY AND FRAME-PACKING

Once we finish the coordination and verification of devices, Step 4 deals with verification of RCB feasibility. We define new software solutions to re-obtain required RCB temporal correctness after any reconfiguration. The goal of this study is to just define these solutions where only one can be applied at run-time. In this case, we propose the modification of parameters of messages or the *(m,k)*−firm-based degraded reconfiguration [12] if some deadlines can be allowed to be violated. We can remove some messages according to their priorities. In this section, we start by describing the modification of temporal parameters of messages.

a: SOLUTION 1. MODIFICATION OF MESSAGE PARAMETERS

In order to minimize the RCB utilization, we propose to modify the periods or WCTTs of messages. The new constant period $T_{m,\Delta}$ and WCTT $S_{m,\Delta}$ of messages is calculated as follows

$$T_{m,\Delta} = \left\lceil \frac{\sum_{m_{\tau_k^i,\tau_l^j} \in \Psi_{mp}^a}^{1..mp_a} S_m(\tau_k^i, \tau_l^j)}{U_{C,b}^{pm}} \right\rceil \tag{11}$$

$$S_{m,\Delta} = \begin{cases} 1, & if\ 0 < \dfrac{U_{C,b}^{pm}}{\sum_{m_{\tau_k^i,\tau_l^j} \in \Psi_{mp}^a}^{1..mp_a} \frac{1}{T_m(\tau_k^i,\tau_l^j)}} \leq 1 \\ \left\lfloor \dfrac{U_{C,b}^{pm}}{\sum_{m_{\tau_k^i,\tau_l^j} \in \Psi_{mp}^a}^{1..mp_a} \frac{1}{T_m(\tau_k^i,\tau_l^j)}} \right\rfloor, & \\ \quad if\ \dfrac{U_{C,b}^{pm}}{\sum_{m_{\tau_k^i,\tau_l^j} \in \Psi_{mp}^a}^{1..mp_a} \frac{1}{T_m(\tau_k^i,\tau_l^j)}} > 1 \end{cases} \tag{12}$$

For the aperiodic messages, we modify their $\lambda_c^{i,k}$ as follows

$$\lambda_C^{i,k} = \lambda_C^{i,k} \times \frac{U_{C,a}^{am}}{U_{C,b}^{am}}, \quad \forall k \in [0..mt_a^i] \tag{13}$$

More details are available in [29] which focuses on the partial problem of real-time scheduling of messages on a CAN bus. The current paper resolves the global problem since the feasibility of messages depends on that of coherent tasks.

b: SOLUTION 2. (m,k)−FIRM MODEL

The *(m,k)*−firm model [12] is a useful solution to find a feasible schedule after applying a reconfiguration scenario. We choose to apply the *(m,k)*−firm model as a solution to optimize **RCB** that runs in a degraded mode. In the hyper period $[0, LCM]$, where LCM represents the lowest common multiple of all periodic messages, we consider that if we have m among k consecutive periodic messages that satisfy their real-time constraints, then we can consider that this system is feasible in a degraded mode. The verification will be done in the worst case. The scheduling will be based on EDF.

c: SOLUTION 3. MESSAGE REMOVAL

As a third solution, the **RCB** agent Ag_M proposes the removal of some messages according to their priorities. After any removal of unimportant messages, we should remove also their related tasks. The removal of the useless messages can minimize both the bus and system utilizations since the tasks that exchange the removed messages will be removed automatically.

d: SOLUTION 4. FRAME-PACKING UNDER BANDWIDTH MINIMIZATION

After the addition of new messages, we need to construct the frames in Step 5. The frame-packing problem is NP-hard [40]. To resolve this problem, we use the Bin-packing algorithm to reduce the utilization of the bandwidth. We suppose that the messages are the items and the frames are the bins. In what follows, Algorithm 3 represents the proposed strategy to assign messages into frames. Each slave agent Ag_i ($i \in [1..n]$) starts by ordering messages and frames in descending and ascending orders, respectively. If the size of the current frame can support the added message, then it is uploaded. Otherwise, it is assigned to the next frame. In the worst case, if there is no frame that can support this message, then we create a new one and add it to the list of frames. We propose also to merge the frames if their size is inferior or equal to the standard size of the frames. This algorithm is invoked every time a message is added.

We note that the complexity of Algorithm 3 is $O(n^2)$. Moreover, although Cyna-RCS is a useful run-time methodology, the calculation time remains a problem especially for complex systems with intensive concurrent critical reconfiguration scenarios. This problem is controllable when (i) we assign priorities to reconfiguration scenarios such that only a subset is treated among all input reconfigurations at a particular time, (ii) we decompose the reconfigurations into three

Algorithm 3 Frame-Packing Under Bandwidth Minimization

Variables:
Input: Integer $k(dev_i)$ number of frames in a particular microcontroller dev_i ;
Input: List $frame_{Listper}(dev_i)$: List of the periodic frames sent by a dev_i ;
Input: List $frame_{Listape}(dev_i)$: List of the aperiodic frames sent by a dev_i;
Sort the messages in a descending order;
Sort the frames in an ascending order;

for each periodic message i **do**
  for each periodic frame j **do**
    if size(i) < size(j) **then**
      Put i into j
    end if
    if (no frame j can support message i) **then**
      Create a new frame z in $frame_{Listper}$;
      Put z into j;
    end if
  end for
end for
for each aperiodic message i **do**
  for each aperiodic frame j **do**
    if size(i) < size(j) **then**
      Put message i into frame j;
    end if
    if (no frame j can support message i) **then**
      Create a new frame z in $frame_{Listaper}$;
      Put z into j;
    end if
  end for
end for
Refinement of the frames:
for $n = 1; n < k; n++$ **do**
  for $m = n + 1; m <= k; m++$ **do**
    if size($frame_{List}[n]$) + $size(frame_{List}[m])$ $\leq$ $size(standardframe)$ **then**
      Fusion($frame_{List}[n]$), $size(frame_{List}[m])$;
    end if
  end for
end for
Output: Frames constructed and satisfy related real-time properties.

forms: Architecture, scheduling and data such that it is not useful to arrange the scheduling and data of reconfigurable tasks if the coordinator agents do not agree the distributed architectures on devices, and (iii) We consider that real-time constraints cannot be satisfied by tasks or messages during the reconfiguration process of networked devices.

V. CYNA-RCS CORRECTNESS

We prove the correctness of the proposed methodology Cyna-RCS with the following theorem.

Theorem 1: Let *Sys* be a distributed control system according to the current paper assumptions, i.e., reconfigurable real-time dependent-independent and periodic-aperiodic distributed tasks that exchange periodic-aperiodic messages on the considered network. If the system is automatically reconfigured at run-time according to the methodology Cyna-RCS, then it satisfies all its real-time and energy constraints.

Part1 (Real-Time Guarantees of Cyna-RCS): Let *Sys* be a feasible system distributed on n devices such that each one dev_i contains nt_b^i periodic and mt_b^i aperiodic tasks before a reconfiguration scenario. Let $mp_b + map_b$ be the number of messages to be exchanged on the bus before this scenario. We suppose that new tasks and messages are added such that each device contains nt_a^i periodic and mt_a^i aperiodic tasks ($\chi_i = nt_a^i + mt_a^i$). Let $\xi = mp_a + map_a$ be the number of messages to be exchanged on the bus after the reconfiguration scenario. Let us suppose that *Sys* becomes unfeasible in spite of Cyna-RCS is applied dynamically. Therefore, there exists τ_j^i ($i \in [1..n]$, $j \in [1..\chi_i]$) that does not meet its deadline, or $m_{\tau_k^i,\tau_l^j}$ ($k \in [1..\xi]$) that does not meet its deadline, which contradicts the solutions given by Cyna-RCS. In fact, equations (1), (4), (7), (9), (10), (11) as well as the rest of Cyna-RCS solutions guarantee that: (i) for any τ_j^i ($i \in [1..n]$ and $j \in [1..\chi_i]$), τ_j^i meets its deadlines, and (ii) for any m_k ($k \in [1..\xi]$), $m_{\tau_k^i,\tau_l^j}$ meets its deadlines.

Part2 (Energy Guarantees of Cyna-RCS): Let us consider that the energy consumption is increased although Cyna-RCS is applied after such a scenario. Let dev_i ($i \in [1..n]$) be a device of *Sys* such that the energy consumption is increased after the reconfiguration scenario Re^i. According to equations (3), (6) and the rest of Cyna-RCS solutions we have: for any dev_i ($i \in [1..n]$), $P_{pa}^i \leq P_{pb}^i$, which contradicts the solutions given by Cyna-RCS. In conclusion, *Sys* is always feasible and meets its energy constraints whatever the methodology Cyna-RCS is automatically applied after any reconfiguration scenario.

VI. IMPLEMENTATION

We propose an original algorithm based on the studies in [29], [30], and [38] to form a complete methodology allowing feasible reconfigurable distributed control systems composed of periodic and aperiodic tasks. Before describing the algorithm, we present the different used functions: (i) Send-approval-power(Ag_i,Ag_M): When the power consumption is less than 1 after a reconfiguration, Ag_i sends an approval message to Ag_M, (ii) Send-alert-power(Ag_i,Ag_M): When the power consumption is greater than 1 after a reconfiguration, Ag_i sends an approval message to Ag_M, (iii) Evaluate-power-consumption(Ag_i): After the application of the solutions, Ag_M calculates the difference between the power consumption before and after reconfiguration Re^i, (iv) Manage-removal(Ag_i): Each agent Ag_i must update the memory after the removal of tasks from dev_i, (v) App-sol1-Task(): Modification of periods or WCETs and $\lambda_C^{i,k}$ of OS tasks, (vi) App-sol2-Task(): Relocation by using Bin-packing, (vii) App-sol3-Task(): Tasks removal, (viii) App-sol1-Message():

Modification of periods or WCTTs of messages, (ix) App-sol2-Message(): Application of the *(m,k)*−firm model, and (x) App-sol3-Message(): Removal of messages. Algorithm 4 is proposed to control the power consumption, to manage any reconfiguration, and to handle the frame-packing on CAN. It is with complexity $O(n^2)$. After the addition of tasks and messages, it reads their parameters and invokes Algorithm 1 to coordinate between the different devices. If the utilization is bigger than 1, then the user suggests one of the proposed solutions for the system feasibility. We denote by Sol_t and Sol_M the solution that can be chosen by the user to be applied for the OS and CAN feasibility.

After that, it verifies the feasibility on CAN. Finally, it constructs dynamically the frame-packing by invoking Algorithm 3.

VII. CASE STUDY AND SIMULATION

We present a formal case study that deals with a distributed reconfigurable control system. The goal of this study is to run the proposed methodology with a simple example of pre-computed reconfiguration scenarios, before applying different simulations for the evaluation of its performance. The simulation results are generated by a developed tool named also Cyna-RCS.[2] These results highlight the performance of the proposed solutions. To handle run-time reconfiguration scenarios, a middleware is proposed to arrange the new lists of tasks and messages to be executed after each scenario[3] (Figure 3). The middleware based on agents and deployed on each device listens to input reconfigurations, arranges the coordination with remote controllers, arranges the parameters of tasks and controls also the traffic on bus. Note that all the tasks are compiled off-line and any reconfiguration scenario is predicted before the system cold start.

To present an application of Cyna-RCS, we propose a networked control system composed of four reconfigurable devices dev_1, dev_2, dev_3 and dev_4 that run distributed real-time tasks exchanging messages on RCB. dev_i ($i \in [1..4]$) activates the blue led when it requires a well-defined reconfiguration, and activates the yellow led when the related devices give their authorization. Table 1 lists the initial parameters before the reconfiguration scenario of the different devices. We suppose initially that tasks τ^1_{p1} and τ^1_{p2} in dev_1 (τ^2_{p3} and τ^2_{p4} in dev_2, respectively) share a software resource Res_1 (Res_2, respectively). The blocking times of all these tasks are equal to 1. Described in Table 2, we consider that aperiodic tasks τ^1_{a9}, τ^1_{a10} and τ^2_{a11} arrive with $\lambda^{i,k}_r = 0.09$ and $\lambda^{i,k}_C = 0.3$. We suppose that the periodic and aperiodic tasks of the system exchange messages on RCB presented in Tables 3 and 4. Let us consider that the aperiodic messages arrive with the same rate as the aperiodic tasks. The initial utilization of dev_1, dev_2, dev_3 and dev_4 is presented in Table 7. The OS feasibility is proven since the utilization of each device is less than 1. We show in Table 15 the initial utilization of RCB that affirms the initial feasibility of the CAN utilization.

Tables 5 and 6 present the parameters of the added tasks at run-time after a reconfiguration scenario. We suppose that τ^1_{p12} (τ^1_{p24}, respectively) share a software resource Res_3 (Res_4, respectively) with τ^1_{p21} (τ^1_{p27}, respectively). In this case, $B^1_{p12} = 2$, $\tau^1_{p21} = 2$, $\tau^1_{p24} = 1$ and $\tau^1_{p27} = 2$. We suppose that the added tasks exchange periodic and aperiodic messages presented in Tables 8 and 9. The utilization in dev_1, dev_2, dev_3

Algorithm 4 Reconfigurable RCB

```
Input: Reconfiguration requests in devices.

for each reconfiguration scenario do
    Invoke Algorithm 1;
end for
for each device do
    Compute the utilization U_a^i;
    if U_a^i ≤ 1 then
        Send-approval-power(Ag_i,Ag_M);
    else
        Send-alert-power(Ag_i,Ag_M);
        if (Sol_t == 1) then
            Call(App-sol1-Task());
        else
            if (Sol_t == 2) then
                Call(App-sol2-Task()) ;
            else
                Call(App-sol3-Task());
            end if
        end if
        Evaluate-power-consumption(Ag_M);
        Calculate the new utilization U_a^i after each solution;
        Manage-removal(Ag_i);
    end if
end for
Calculate U_{C,a};
if U_{C,a} ≥ 1 then
    if (Sol_M == 1) then
        Call(App-sol1-Message());
    else
        if (Sol_M == 2) then
            Call(App-sol2-Message());
        else
            Call(App-sol3-Message());
        end if
    end if
end if
Invoke Algorithm 3;
Output: Coherent and feasible behaviors in devices and RCB.
```

[2]For more details, the reader can contact Cynapsys company (http://www.cynapsys.de/en/).

[3]The middleware is tested in the projects of Cynapsys Corporation. For confidentiality, the reader can have a direct contact with Cynapsys for more explanation.

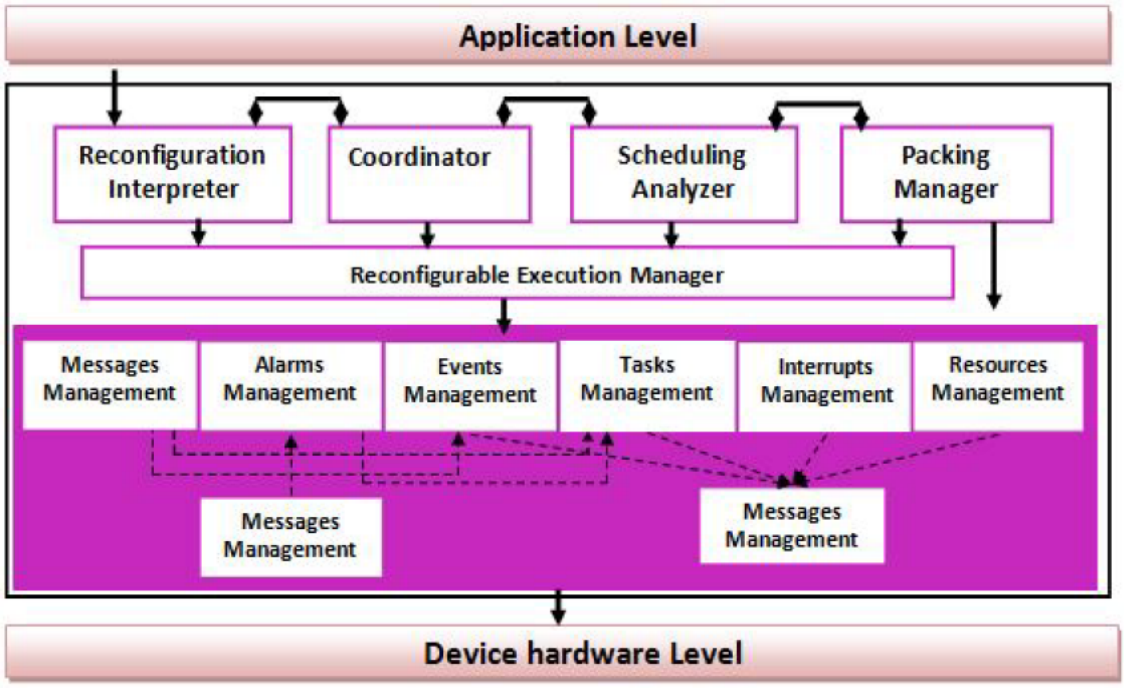

FIGURE 3. Middleware architecture.

TABLE 1. Characteristics of initial periodic tasks.

Tasks	C_j^i	T_j^i	dev_i
τ_{p1}^1	3	30	dev_1
τ_{p2}^1	2	20	dev_1
τ_{p3}^2	2	30	dev_2
τ_{p4}^2	4	20	dev_2
τ_{p5}^3	2	30	dev_3
τ_{p6}^3	5	20	dev_3
τ_{p7}^4	6	30	dev_4
τ_{p8}^4	6	20	dev_4

TABLE 2. Characteristics of initial aperiodic tasks.

Tasks	C_j^i	D_j^i	dev_i
τ_{a9}^1	2	30	dev_1
τ_{a10}^1	2	20	dev_1
τ_{a11}^2	8	20	dev_2

and dev_4 becomes higher than 1 after this reconfiguration as described in Table 7, i.e., the feasibility condition is not respected. Moreover, the CAN utilization given in Table 10 after the reconfiguration of RCB becomes higher than one. The bus becomes also unfeasible.

A. COORDINATION

The methodology Cyna-RCS is automatically applied since real-time properties are not satisfied in all the devices and *RCB*. The second step of Cyna-RCS deals with a required coordination between devices for their coherence

TABLE 3. Characteristics of the initial periodic messages.

Messages	S_m	T_m	Z_m
$m(\tau_{p1}^1, \tau_{p3}^2)$	1	30	10
$m(\tau_{p1}^1, \tau_{p5}^3)$	1	30	10
$m(\tau_{p2}^1, \tau_{p4}^2)$	1	30	12
$m(\tau_{p4}^2, \tau_{p6}^3)$	1	20	15
$m(\tau_{p4}^2, \tau_{p8}^4)$	1	20	14
$m(\tau_{p5}^3, \tau_{p7}^4)$	1	30	15

TABLE 4. Characteristics of the initial aperiodic messages.

Messages	$\lambda_c^{i,k}$	$\lambda_r^{i,k}$	Z_m
$m(\tau_{a9}^1, \tau_{a11}^2)$	0, 3	0, 09	16

after reconfiguration. We present in Table 11 the exchanged architecture tokens on RCB where each device marks one if it agrees the requested reconfiguration. The first token (line 1) is sent from dev_1 to propose the addition of τ_{p12}^1 and τ_{a27}^1. In this case, dev_2 agrees and adds τ_{p23}^2 that exchanges $m(\tau_{p12}^1, \tau_{p23}^2)$ with τ_{p12}^1, and dev_4 agrees and adds τ_{a26}^4 that exchanges $m(\tau_{a27}^1, \tau_{a26}^4)$ with τ_{a27}^1. The second token (line 2) is sent from dev_2 to propose the addition of τ_{p13}^2, τ_{p14}^2, and τ_{p16}^2. In this case, dev_4 agrees to add τ_{p15}^4 that exchanges $m(\tau_{p13}^2, \tau_{p15}^4)$ and $m(\tau_{p14}^2, \tau_{p15}^4)$ with τ_{p13}^2 and τ_{p14}^2. dev_3 agrees also to add τ_{p18}^3 that exchanges $m(\tau_{p16}^2, \tau_{p18}^3)$ with τ_{p16}^2. The composition and data tokens are represented by Tables 12 and 13. All the devices agree first with the different

TABLE 5. Characteristics of added periodic tasks.

Tasks	C_j^i	T_j^i	Inc
τ_{p12}^1	3	30	dev_1
τ_{p13}^2	5	20	dev_2
τ_{p14}^2	2	20	dev_2
τ_{p15}^4	4	20	dev_4
τ_{p16}^2	3	30	dev_2
τ_{p17}^3	5	25	dev_3
τ_{p18}^3	6	30	dev_3
τ_{p19}^2	6	20	dev_2
τ_{p20}^3	7	35	dev_3
τ_{p21}^1	3	15	dev_1
τ_{p22}^1	1	10	dev_1
τ_{p23}^2	9	30	dev_2
τ_{p24}^1	3	40	dev_1

TABLE 6. Characteristics of added aperiodic tasks.

Tasks	C_j^i	D_j^i	Inc
τ_{a25}^3	2	30	dev_3
τ_{a26}^4	1	20	dev_4
τ_{a27}^1	2	20	dev_1
τ_{a28}^2	4	15	dev_2

TABLE 7. Utilization of devices before and after the reconfiguration.

	dev_1	dev_2	dev_3	dev_4
U_b^i	0.58	0.64	0.39	0.58
U_a^i	1.59	2.2	1.38	1.13

TABLE 8. Characteristics of the added periodic messages.

Messages	S_m	T_m	Z_m
$m(\tau_{p13}^2, \tau_{p15}^4)$	1	10	12
$m(\tau_{p14}^1, \tau_{p15}^4)$	2	20	14
$m(\tau_{p16}^2, \tau_{p18}^3)$	1	10	10
$m(\tau_{p12}^1, \tau_{p23}^2)$	3	30	13

TABLE 9. Characteristics of the added aperiodic messages.

Messages	S_m	D_m	Z_m
$m(\tau_{a27}^1, \tau_{a26}^4)$	2	20	15

TABLE 10. CAN before and after the addition of the messages.

	CAN utilization
$U_{C,b}$	0.53
$U_{C,a}$ Without Cyna-RCS	1.27

proposed schedules then they agree with the newest proposed values of periods for the given schedule in dev_1 and dev_2.

B. FEASIBILITY ANALYSIS

After this coordination between devices, the next step deals with the feasibility analysis to allow the required satisfaction of real-time and energy constraints after reconfiguration. The proposed equations (1), (4) and (7) are applied to modify the

TABLE 11. Exchanged tokens between devices for coordination: Architectural level.

Sender	Tasks to add	Priority	dev_1	dev_2	dev_3	dev_4
dev_1	$\tau_{p12}^1, \tau_{a27}^1$	1	1	1	1	1
dev_2	$\tau_{p13}^2, \tau_{p14}^2, \tau_{p16}^2$	2	1	1	1	1

TABLE 12. Exchanged tokens between devices for coordination: Composition level.

Sender	Proposed schedule	Priority	dev_1	dev_2	dev_3	dev_4
dev_1	$\tau_{p12}^1, \tau_{p21}^1, \tau_{p22}^1$ and τ_{p24}^1	1	1	1	1	1
dev_2	$\tau_{p13}^2, \tau_{p14}^2, \tau_{p16}^2$ and τ_{p13}^2	2	1	1	1	1

TABLE 13. Exchanged tokens between devices for coordination: Data level.

Sender	Data	Priority	dev_1	dev_2	dev_3	dev_4
dev_1	$T = 20$	1	1	1	1	1
dev_2	$T = 30$	2	1	1	1	1

TABLE 14. Utilization of devices after the reconfiguration.

	dev_1	dev_2	dev_3	dev_4
T_Δ	75	112	77	33
C_Δ	1	1	1	1
λ_C	0.3	0.3	0.3	0.3
U_a^i with T_Δ	0.58	0.64	0.39	0.58
U_a^i with C_Δ	0.65	0.88	0.68	0.56

TABLE 15. CAN before and after the reconfiguration.

	CAN utilization
$U_{C,a}$ With Cyna-RCS after the periods modification	0.53
$U_{C,a}$ With Cyna-RCS after the WCTTs modification	0.54

periods and WCET of tasks in order to have the utilization less than one in each device. Table 14 presents the new constant values of periods and WCET of tasks in dev_1, dev_2 and dev_3. The new utilization is less than one and each device becomes again feasible. Equations (9) and (10) are also applied to calculate new constant values for the periods and WCTT of the exchanged messages. According to Table 15, the RCB utilization becomes again less than one and the messages are feasible.

We illustrate the packing problem by applying Algorithm 3 and choosing FFD to the considered running example. First of all, we order the periodic and aperiodic frames in an ascending order and the messages in a descending one. We consider four frames to be exchanged on the bus. Table 16 presents the assignment of messages to the frames F_1, F_2, F_3 and F_4. The size of the frame F_1 is 40 and contains the messages $m(\tau_{p1}^1, \tau_{p3}^2)$, $m(\tau_{p1}^1, \tau_{p5}^3)$ and $m(\tau_{p5}^3, \tau_{p7}^4)$ that have the same period value. Equations (3) and (6) are applied to estimate the new energy consumed by devices and also RCB. Table 17 presents the new values of energy consumption which is decreased thanks to Cyna-RCS. We present in Table 17

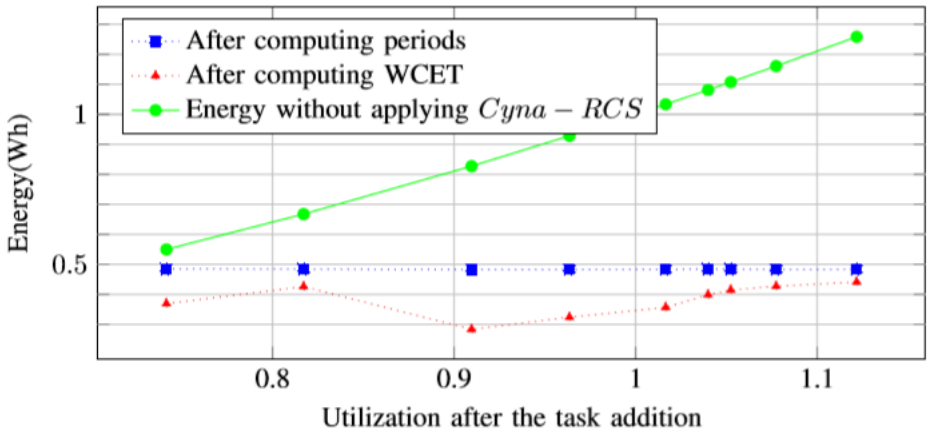

FIGURE 4. Energy evolution after the parameters modification in devices.

TABLE 16. Available space in each frame.

Frames	$Space(bits)$	$Period$	Messages
F_1	40	30	$m(\tau^1_{p1}, \tau^2_{p3})$, $m(\tau^1_{p1}, \tau^3_{p5})$,$m(\tau^3_{p5}, \tau^4_{p7})$
F_2	70	20	$m(\tau^1_{p2}, \tau^2_{p4})$, $m(\tau^2_{p4}, \tau^3_{p6})$,$m(\tau^2_{p4}, \tau^4_{p8})$, $m(\tau^2_{p14}, \tau^4_{p15})$,$m(\tau^2_{p13}, \tau^4_{p15})$
F_3	30	30	$m(\tau^2_{p16}, \tau^3_{p18})$, $m(\tau^1_{p12}, \tau^2_{p23})$
F_4	40	40	$m(\tau^1_{a9}, \tau^2_{a11})$, $m(\tau^1_{a27}, \tau^4_{a26})$

TABLE 17. Power consumption of devices before and after the reconfiguration.

	dev_1	dev_2	dev_3	dev_4
P^i_a without Cyna-RCS	2.53	4.84	1.9	1.28
P^i_a with Cyna-RCS	0.336	0.4	0.15	0.336
Gain	86	91	93	82

the global energy 82% and 93% thanks to the Cyna-RCS methodology.

C. EVALUATION OF PERFORMANCE

We have developed a C++ simulator which is based on the real-time simulator cheddar.[4] First, it reads the initial parameters of periodic/aperiodic tasks/messages. Then, it computes the initial utilizations of each device as well as that of CAN. After that, it reads the parameters of the added tasks/messages and applies the coordination protocol. Once the tasks/messages are added, Cyna-RCS will be applied to ensure a feasible system thanks to equations (1), (4), (7), (9), (10) and (11). Finally, the frame-packing will be handled and the energy consumption will be calculated thanks to equation (6).

In order to evaluate the performance of the paper's contribution, we consider a system implemented initially by 20 OS periodic and 10 OS aperiodic tasks and assumed to be feasible. Initially, we assume that our system is feasible. We consider two types of reconfigurations: (i) A sequence of software reconfigurations that gradually increase the number of tasks to be 80 periodic and 20 aperiodic tasks, (ii) A sequence of hardware reconfigurations that increase in step by step the number of devices to be 20. The goal of this experimental part is to explain the different steps of the proposed methodology. We aim to consider pre-defined reconfigurations and to compute off-line (and can be on-line) the periods and frequencies of processors after any reconfiguration scenario to meet real-time and energy constraints.

We start first by applying the coordination protocol between the different devices. Once the reconfigurations are authorized, we verify the first solution which is the period/WCET modification. In order to compare the proposed WCET and period oriented solutions, we present in Figure 4 the energy evolution after adding tasks to three devices when both of them are applied. The new values of energy are given thanks to equation 6. The dotted curve with squares represents the power consumption after the periods modification. The dotted curve with triangles represents the power consumption after applying the WCETs modification. To highlight the importance of the proposed solution, the dotted curve is added. The power consumption decreases after the parameters modification is applied. Note that the power consumption is further reduced when the WCETs modification is applied. This is logical since changing WCET means the modification of the processor speed which is proportional to the consumed energy. When we compare the obtained results with [42], we deduce that the given results after the periods modification is similar to [42]. But, our work shows better results than those in [42] if we modify the WCETs. Also, we have compared our contribution with [19]. The energy found is between 0 and 45 % for the periods modification however in our work the energy is stable after we apply this solution.

[4]F. Singhoff, J. Legrand, L. Nana, L. Marce, "Cheddar: A flexible real time scheduling framework," *Proc. Int. ACM SIGAda Conf.*, pp. 1-8, 2004.

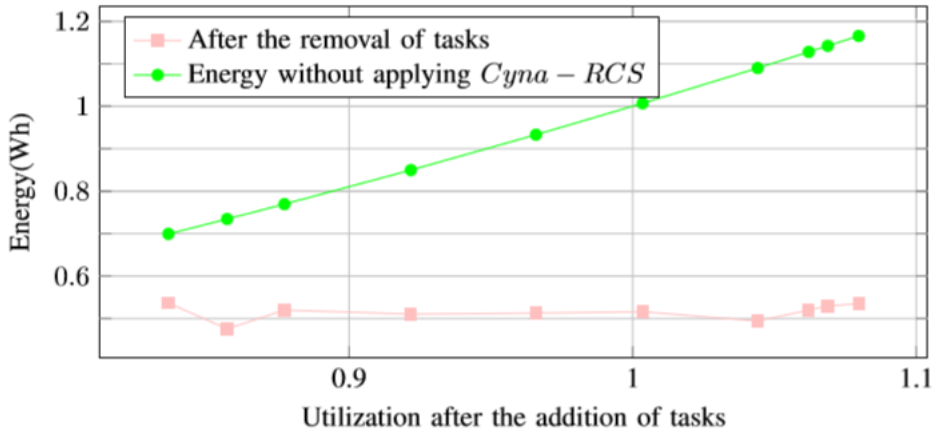

FIGURE 5. Energy evolution after task's removal.

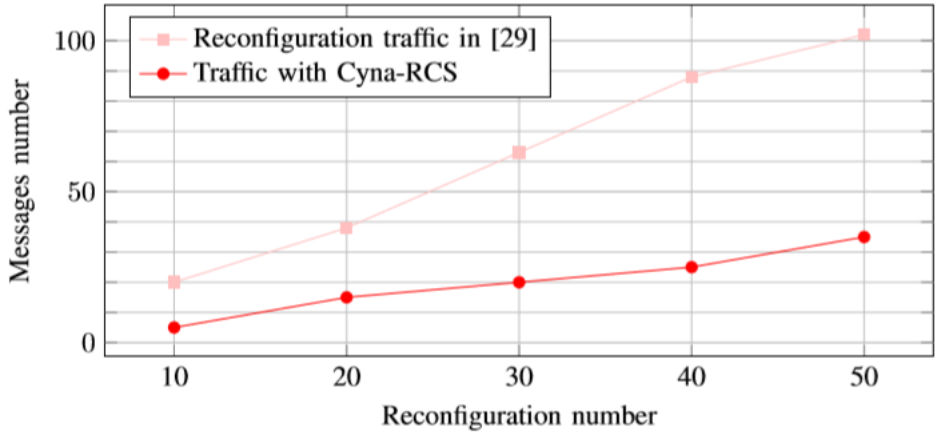

FIGURE 6. Evolution of the number of exchanged messages in (i) Multi-agent architecture of Cyna-RCS, (ii) Centralized agent-based architecture in [35].

We present in Figure 5 the energy evolution after the removal of the unimportant tasks (tasks with the lowest priority) from the different devices. The dotted curve with squares represents the power consumption when Cyna-RCS is applied. The energy is between 0.4757 and 0.5365. Thus, we can ignore the variations. The dotted curve represents the energy when no solution is applied after the tasks addition. As we can see, this solution can stabilize/minimize the power consumption in the whole system. In terms of energy, the obtained results in our work are better than the given results in the related publications. For example in [23], the energy is between 10% and 30% after applying the pipeline. But in this current research, energy saving is 76% in average.

We present in Figure 7 the energy evolution when the number of exchanged messages is increased step by step from 10 to 100 after applying a sequence of reconfiguration scenarios. The dotted curves with squares and triangles represent the power consumption after the application of Cyna-RCS by modifying the periods and the WCTTs of the messages, respectively. The dotted curve represents the energy if we do not apply Cyna-RCS. We remark that more the CAN utilization increases, the latter will be reduced if we apply both of the proposed solutions. Thus, the power consumption is minimized.

In order to be more effective, we compare our protocol with [35]. Therefore, we present in Figure 6 the comparison between the current work and that reported in [35] which is based on a unique controller to manage distributed reconfiguration scenarios. By using the token oriented solution in Cyna-RCS, the number of exchanged messages for coordination between devices is lower than in that reported in [35]. In fact, the centralized control of reconfigurations increases the point to point requests whereas the use of architecture, scheduling and data tokens reduces the reconfiguration traffic on RCB.

Figures 9 and 8 present the evolution of the energy consumption when the number of tasks and devices is increased. We remark that the energy remains nearly constant by applying Cyna-RCS when the number of tasks from 10 reaches

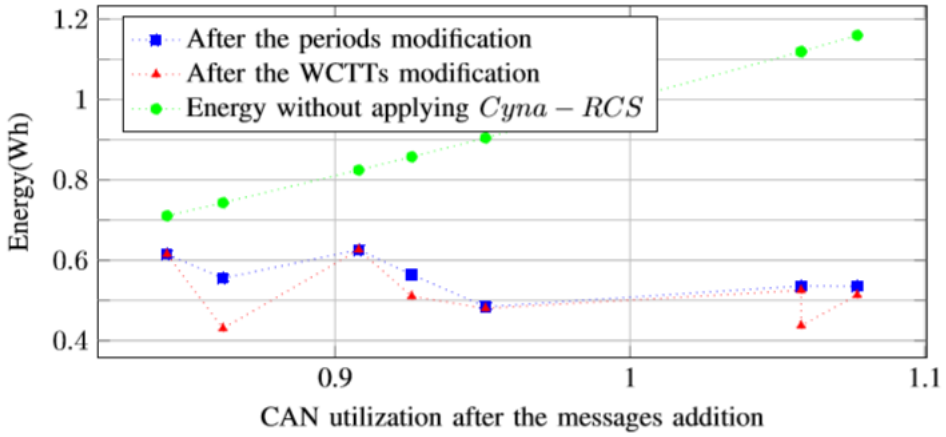

FIGURE 7. Energy evolution after adding messages and the obtained gain after computing constant WCTT in RCB.

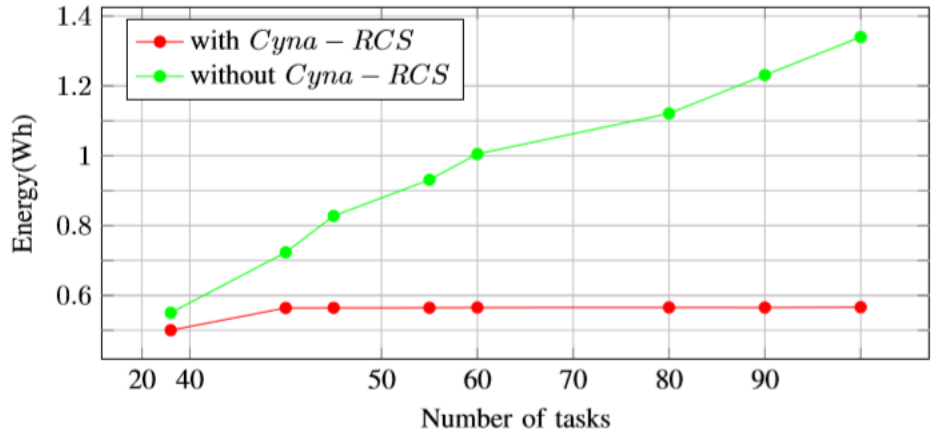

FIGURE 8. Evolution of energy when the number of tasks are increased thanks to Cyna-RCS.

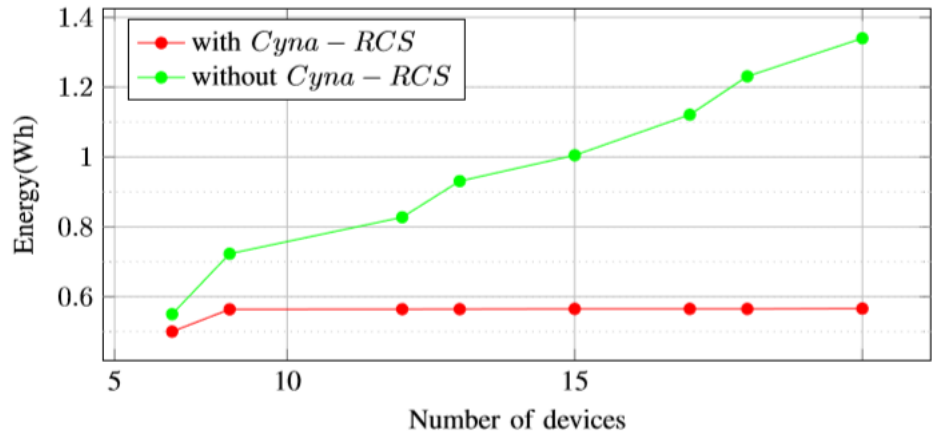

FIGURE 9. Evolution of energy when the number of devices are increased thanks to *Cyna – RCS*.

90 and the number of devices from 4 to 15. This is logical since the proposed solutions allow to have $P_a^i \leqslant P_a^i$.

Table 18 compares the proposed approach with several related works in the literature. The current paper deals with the global feasibility since the correctness of tasks in devices depends on that of messages on the network. Indeed, the problem considered in the current paper is different from those in previous papers since we deal with the feasibility

TABLE 18. Comparative study.

Research	Architecture	Strategy	power decrease	Remarks
[42]	Agent based architecture for real-time embedded systems.	Users should choose one of the proposed solutions by the agent in order to guarantee the system feasibility under real-time and energy constraints.	Energy is minimized when the WCETs are modified after any reconfiguration scenario. Energy is stabilized when the periods are modified after any reconfiguration scenario	+ Studies periodic/probabilistic tasks. + Efficient intelligent agent. - Monoprocessor system.
[23] [1]	Wireless Sensor Nodes.	Follows a specific set of steps to respond more effectively to reconfiguration requests in terms of energy, memory and real-time constraints.	Energy is between 10% and 30% after applying the pipeline.	+ Gives a decision making mechanism for any reconfigurable real-time system. + Covers many axes: energy and memory optimization, system checking and stability. - Minimizing the power consumption is based only on the number of removed tasks.
[19] [5]	Distributed architecture with three agent types.	Resizes geographically the zones in a reconfigurable wireless sensor network.	The use of the resizing protocol is more beneficial than the use of the mobility protocol to gain more in terms of the energy consumption. Energy between 0 and 45% for the periods modification.	+ Combination of resizing and mobility forms. + Executes more reconfiguration tasks on a flexible node. - The functional safety of reconfigurable systems is not considered.
[2]	MPSoC hardware architecture.	Proposes a new R-codesign which is based on new modeling and partitioning techniques for reconfigurable embedded systems.	not mentioned.	+ Modeling and partitioning probabilistic real-time systems having multiple reconfiguration scenarios. + Communication costs will be reduced. - Considers just hardware tasks.
[3]	Multi-agent architecture with a global agent and four local agents	deals with the adaptive scheduling of real-time DAG tasks with energy harvesting.	Not mentioned.	+ Efficient and complete reconfiguration. - Does not treat the resource sharing. - Considers just periodic tasks.
$Cyna - RCS$	Distributed architecture with four types of agents: two agents for the coordination, one for the system feasibility and one for the frame-packing.	Applies a coordination protocol and applies one of the proposed solutions to ensure the system feasibility under low-power constraints.	Energy saving is 76% in average.	+ Deals with Periodic/probabilistic tasks/messages. + Develops a new protocol for coordination between microcontrollers. + Offers diverse services: real-time feasibility, coherent networked, low power optimization., management of dependent tasks sharing resources. - The QoS and development cost are not considered.

of distributed reconfigurable real-time periodic/aperiodic tasks that exchange messages under energy constraints. By comparing our work with the existing methods above [1]–[3], [19], [23], [42], we believe that our contribution is original since these related works do not consider the same assumptions of this work.

VIII. CONCLUSION

We propose in this paper a run-time automatic global approach called Cyna-RCS for reconfigurable CAN that links several flexible devices. In this approach, different solutions are used to guarantee a feasible system under real-time, low-power and networking constraints.

After applying a reconfiguration scenario which is assumed to be any modification of the software as well as the hardware architecture, the power consumption can increase and some real-time constraints may be not satisfied. Moreover, CAN cannot support the added messages to be exchanged between tasks.

A multi-agent-architecture following the master/slave model is defined where we can apply technical solutions to meet all fixed constraints. This architecture is based on a token ring protocol for an optimal coordination between devices. The different steps of our methodology are explained in the experimentation part to verify the respect of real-time constraints and evaluate the energy minimisation. This protocol is applied in step-by-step to control the complexity of the problem under consideration. Our methodology is certain since it can be applied to any system with the predefined assumptions. To the best of our knowledge, this paper is original since no one in all related works and in all our previous works deals with the automatic adaptation of reconfigurable distributed systems where the coherence between devices, the feasibility of tasks in each one, the frame packing and the feasibility on the network are treated together.

We plan in a future work to deal with a real case study. Also, other constraints such as memory, fault tolerance, quality of service that should be satisfied by this kind of systems. Moreover, we will work on optimizing the calculability of the proposed solutions where we will consider the performance of this methodology.

ACKNOWLEDGMENT

The authors would like to thank for collaboration: Systemscontrol Lab at Xidian University in China, PASRI (http://www.pasri.tn/), Cynapsys Corporation (http://www.cynapsys.de/), Walid Bouzayen, Abdelkader Gharbi, Mohamed Aymen Jouini, Hanen Ben Naceur, Feriel Gammoudi at Faculty of Science of Tunis (http://www.fst.rnu.tn/), University Tunis El Manar in Tunisia (http://www.utm.rnu.tn/utm/fr/index.php).

REFERENCES

[1] M. Gasmi, O. Mosbahi, M. Khalgui, L. Gomes, and Z. Li, "Performance optimization of reconfigurable real-time wireless sensor networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2018.2824900.

[2] I. Ghribi, R. Ben Abdallah, M. Khalgui, Z. Li, K. Alnowibet, and M. Platzner, "R-codesign: Codesign methodology for real-time reconfigurable embedded systems under energy constraints," *IEEE Access*, vol. 6, pp. 14078–14092, 2018.

[3] W. Housseyni, O. Mosbahi, M. Khalgui, Z. Li, and L. Yin, "Multiagent architecture for distributed adaptive scheduling of reconfigurable real-time tasks with energy harvesting constraints," *IEEE Access*, vol. 6, pp. 2068–2084, 2017.

[4] A. Ben Ahmed, O. Mosbahi, M. Khalgui, and Z. Li, "Toward a new methodology for an efficient test of reconfigurable hardware systems," *IEEE Trans. Autom. Sci. Eng.*, to be published, doi: 10.1109/TASE.2018.2822050.

[5] 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.

[6] W. Lakhdhar, R. Mzid, M. Khalgui, Z. Li, G. Frey, and A. Al-Ahmari, "Multiobjective optimization approach for a portable development of reconfigurable real-time systems: From specification to implementation," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2017.2781460.

[7] Y. Chen, Z. Li, K. Barkaoui, N. Wu, and M. Zhou, "Compact supervisory control of discrete event systems by Petri nets with data inhibitor arcs," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 2, pp. 364–379, Feb. 2017.

[8] L. Bai, N. Wu, Z. Li, and M. Zhou, "Optimal one-wafer cyclic scheduling and buffer space configuration for single-arm multicluster tools with linear topology," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 10, pp. 1456–1467, Oct. 2016.

[9] G. M. Mancuso, E. Bini, and G. Pannocchia, "Optimal priority assignment to control tasks," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 5, pp. 161–178, Nov. 2014.

[10] J. Real and A. Crespo, "Mode change protocols for real-time systems: A survey and a new proposal," *Real-Time Syst.*, vol. 26, no. 2, pp. 161–197, 2004.

[11] A. Burns and R. I. Davis, "Mixed criticality systems—A review," Dept. Comput. Sci., Univ. York, York, U.K., Tech. Rep., 2016.

[12] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines," *IEEE Trans. Comput.*, vol. 44, no. 12, pp. 1443–1451, Dec. 1995.

[13] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Trans. Comput.*, vol. 39, no. 9, pp. 1175–1185, Sep. 1990.

[14] D. Grunwald, C. B. Morrey, III, P. Levis, M. Neufeld, and K. I. Farkas, "Policies for dynamic clock scheduling," in *Proc. 4th Symp. Operating Syst. Design Implement.*, San Diego, CA, USA, 2000, pp. 73–86.

[15] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proc. 1st USENIX Conf. Oper. Syst. Design Implement.*, 1995, pp. 13–23.

[16] S. Saha and B. Ravindran, "An experimental evaluation of real-time DVFS scheduling algorithms," in *Proc. 5th Annu. Int. Syst. Storage Conf.*, 2012, Art. no. 7, doi: 10.1145/2367589.2367604.2012.

[17] T. Baker, "Stack-based scheduling of realtime processes," *J. Real-Time Syst.*, vol. 3, no. 1, pp. 67–99, 1991.

[18] U. N. Bhat, *An Introduction to Queueing Theory: Modeling and Analysis in Applications*. Cambridge, MA, USA: Birkhäuser, 2015.

[19] H. Grichi, O. Mosbahi, M. Khalgui, and Z. Li, "New power-oriented methodology for dynamic resizing and mobility of reconfigurable wireless sensor networks," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 7, pp. 1120–1130, Jul. 2018, doi: 10.1109/TSMC.2016.2645401.

[20] 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.

[21] F. Yang, N. Wu, Y. Qiao, M. Zhou, and Z. Li, "Scheduling of single-arm cluster tools for an atomic layer deposition process with residency time constraints," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 502–516, Mar. 2017.

[22] J. W. M. Pang and F. Tobagi, "Throughput analysis of a timer controlled token passing protocol under heavy load," *IEEE Trans. Commun.*, vol. 37, no. 7, pp. 694–702, Jul. 1989.

[23] M. Gasmi, O. Mosbahi, M. Khalgui, L. Gomes, and Z. Li, "R-node: New pipelined approach for an effective reconfigurable wireless sensor node," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 6, pp. 892–905, Jun. 2018.

[24] X. Xie, D. Yue, and S. Hu, "Fault estimation observer design of discrete-time nonlinear systems via a joint real-time scheduling law," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1451–1463, Jul. 2017.

[25] X. Wang, Z. Li, and W. M. Wonham, "Optimal priority-free conditionally-preemptive real-time scheduling of periodic tasks based on DES supervisory control," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1082–1098, Jul. 2017.

[26] M. Salehi and A. Ejlali, "A hardware platform for evaluating low-energy multiprocessor embedded systems based on COTS devices," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1262–1269, Feb. 2015.

[27] A. Lekidis, M. Bozga, and S. Bensalem, "Model-based validation of CANopen systems," in *Proc. 10th IEEE World Conf. Factory Commun. Syst.*, Toulouse, France, May 2014, pp. 1–10.

[28] A. Marino and J. Schmalzel, "Controller area network for in-vehicle law enforcement applications," in *Proc. IEEE Sensors Appl. Symp.*, San Diego, CA, USA, Feb. 2007, pp. 1–5.

[29] I. Khemaissia, O. Mosbahi, and M. Khalgui, "Reconfigurable CAN in real-time embedded platforms," in *Proc. 11th Int. Conf. Inform. Control, Automat. Robot.*, Vienna, Austria, Sep. 2014, pp. 355–362.
[30] I. Khemaissia, O. Mosbahi, M. Khalgui, and W. Bouzayen, "New reconfigurable middleware for feasible adaptive RT-Linux," in *Proc. 4th Int. Conf. Pervasive Embedded Comput. Commun. Syst.*, Lisbon, Portugal, 2014, pp. 158–167.
[31] M. Khalgui and H. M. Hanisch, "Reconfiguration protocol for multi-agent control software architectures," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 41, no. 1, pp. 70–80, Jan. 2011.
[32] M. O. Ben Salem, O. Mosbahi, M. Khalgui, Z. Jlalia, G. Frey, and M. Smida, "BROMETH: Methodology to design safe reconfigurable medical robotic systems," *Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 13, p. e1786, Sep. 2017, doi: 10.1002/rcs.1786.
[33] T. Yokoyama, G. Zeng, H. Tomiyama, and H. Takada, "Static task scheduling algorithms based on greedy heuristics for battery-powered DVS systems," *IEICE Trans. Inf. Syst.*, vol. E93.D, no. 8, pp. 2737–2746, Oct. 2010.
[34] 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, Feb. 1973.
[35] J. Zhang, M. Khalgui, Z. Li, G. Frey, O. Mosbahi, and H. Ben Salah, "Reconfigurable coordination of distributed discrete event control systems," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 1, pp. 323–330, Jan. 2015.
[36] A. Burns, M. Gutiérrez, M. A. Rivas, and M. G. Harbour, "A deadline-floor inheritance protocol for EDF scheduled embedded real-time systems with resource sharing," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1241–1253, May 2015.
[37] 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, Feb. 2016.
[38] I. Khemaissia, O. Mosbahi, and M. Khalgui, "New multi-token based protocol for flexible networked microcontrollers," in *Proc. 9th Conf. Softw. Eng. Appl.*, Vienna, Austria, Aug. 2014, pp. 464–469.
[39] M. B. N. Shah, A. R. Husain, and A. S. A. Dahalan, "An analysis of CAN-based steer-by-wire system performance in vehicle," in *Proc. IEEE Int. Conf. Control Syst., Comput. Eng.*, Penang, Malaysia, Nov. 2013, pp. 350–355.
[40] K. Sandstrom, C. Norstom, and M. Ahlmark, "Frame packing in real-time communication," in *Proc. 7th Conf. Real-Time Comput. Syst. Appl.*, Cheju Island, South Korea, 2000, pp. 399–403.
[41] Y. Chen, Z. Li, A. Al-Ahmari, N. Wu, and T. Qu, "Deadlock recovery for flexible manufacturing systems modeled with Petri nets," *Inf. Sci.*, vol. 381, pp. 290–303, Mar. 2017.
[42] 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.
[43] M. Kang, K. Park, and M. K. Jeong, "Frame packing for minimizing the bandwidth consumption of the FlexRay static segment," *IEEE Trans. Ind. Electron.*, vol. 60, no. 9, pp. 4001–4008, Sep. 2013.
[44] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in *Proc. 36th Annu. ACM/IEEE Design Automat. Conf.*, New Orleans, LA, USA, Jun. 1999, pp. 134–139.
[45] S. C. Talbot and S. Ren, "Comparision of FieldBus systems CAN, TTCAN, FlexRay and LIN in passenger vehicles," in *Proc. 29th IEEE Int. Conf. Distrib. Comput. Syst. Workshops*, Jun. 2009, pp. 26–31.
[46] B. D. Bui, R. Pellizzoni, and M. Caccamo, "Real-time scheduling of concurrent transactions in multidomain ring buses," *IEEE Trans. Comput.*, vol. 61, no. 9, pp. 1311–1324, Sep. 2012.
[47] S. Albers and M. Mitzenmacher, "Average-case analyses of first fit and random fit bin packing," in *Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms*, Philadelphia, PA, USA, 1998, pp. 290–299.
[48] M. Uzam, Z. Li, G. Gelen, and R. S. Zakariyya, "A divide-and-conquer-method for the synthesis of liveness enforcing supervisors for flexible manufacturing systems," *J. Intell. Manuf.*, vol. 27, no. 5, pp. 1111–1129, Oct. 2016.
[49] Y. Hou, N. Wu, M. Zhou, and Z. Li, "Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 517–530, Mar. 2017.
[50] X. Y. Cong, M. P. Fanti, A. M. Mangini, and Z. W. Li, "Decentralized diagnosis by Petri nets and integer linear programming," *IEEE Trans. Syst., Man, Cybern. Syst.*, to be published, doi: 10.1109/TSMC.2017.2726108.
[51] H. M. Zhang, L. Feng, N. Q. Wu, and Z. W. Li, "Integration of learning-based testing and supervisory control for requirements conformance of black-box reactive systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 2–15, Jan. 2018.

IMEN KHEMAISSIA received the B.S. degree in computer science from Tunis University, Tunisia, in 2010, the M.S. degree in computer science from the Faculty of Sciences of Tunis in 2012, and the Ph.D. degree from Tunis El Manar University in 2016. She was with the National Institute of Applied Sciences and Technologies, University of Carthage, Tunisia, where she was involved in research on reconfigurable software and hardware architectures. She is currently a Researcher with Complex Systems Lab, Jinan University, China, an Assistant Professor with King Khalid University, Saudi Arabia, and a Member of Computing Lab on Industrial Systems, University of Carthage. Her current research interests include developing complex reconfigurable software and hardware architectures under different constraints (i.e., real time, energy, memory, QoS, and so on).

OLFA MOSBAHI received the B.S., M.S., and Habilitation Diploma degrees in computer science from Tunis El Manar University, Tunisia, in 1999, 2002, and 2018, respectively, and the Ph.D. degree from the French Polytechnic Institute of Lorraine, France, in 2008. She was with INRIA, where she was involved in research in computer science. She was a part time Researcher with INRIA and a temporary Lecturer with Nancy 2 University, France. She was a Researcher with Martin Luther University of Halle-Wittenberg, Germany. She is currently a member of Complex Systems Lab, Jinan University, China, and 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. She 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 from the Martin Luther University of Halle-Wittenberg, Halle, Germany, in 2012, with a Humboldt Grant. He was a Researcher in computer science with the Institut National de Recherche en Informatique et Automatique, Lorraine, France, ITIA-CNR, Milan, Italy, the Systems Control Laboratory, Xidian University, Xi'an, China, and KACST, Riyadh, Saudi Arabia. He was a Collaborator with the SEG Research Group, University of Patras, Patras, Greece, the Director of RECS Project at O3NEIDA, Canada, the Director of RES Project at Synesis Consortium, Lomazzo, Italy, the Manager of Cyna-RCS Project at Cynapsys Consortium, France, and the Director of BROS and RWiN Projects at 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 member of many TPC of 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 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 as well as many international conferences. He was listed in the book *Who's Who in the World* (Marquis, 27th edition, 2010).

TING QU received the B.Eng. and M.Phil. degrees from the School of Mechanical Engineering, Xi'an Jiaotong University, China, and the Ph.D. degree from the Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong. He was a Post-Doctoral Research Fellow and a Research Assistant Professor with HKU. In 2010, he was appointed as a Full Professor and the Department Head of industrial engineering with the Guangdong University of Technology, China. He currently serves as the Dean of the School of Electrical Engineering and the Deputy Dean of the School of Intelligence Science and Engineering. In 2016, he moved to Jinan University. His research interests include IoT-based smart manufacturing systems, logistics and supply chain management, and industrial product/production service systems. He has undertaken over 20 research projects funded by government and industry and published over 150 technical papers in these areas, half of which have appeared in reputable journals. He serves as a Regional/Area Editor, an Associate Editor, and an Editorial Board/Member of a number of national and international journals.