Journal Paper

Multi-agent Adaptive Architecture for Flexible Distributed Real-time Systems

Hamza Chniter
Yonghui Li
Mohamed Khalgui
Anis Koubâa*
Zhiwu Li
Fethi Jarray

*CISTER Research Centre
CISTER-TR-180404

2018
Multi-agent Adaptive Architecture for Flexible Distributed Real-time Systems

Hamza Chniter, Yonghui Li, Mohamed Khalgui, Anis Koubâa*, Zhiwu Li, Fethi Jarray

*CISTER Research Centre
Polytechnic Institute of Porto (ISEP-IPP)
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
Portugal
Tel.: +351.22.8340509, Fax: +351.22.8321159
E-mail: aska@isep.ipp.pt
http://www.cister.isep.ipp.pt

Abstract

Recent critical embedded systems become more and more complex and usually react to their environment that requires to amend their behaviors by applying run-time reconfiguration scenarios. A system is defined in this paper as a set of networked devices where each of which has its own OS (Operating System), a processor to execute related periodic software tasks, and a local battery. A reconfiguration is any operation allowing the addition-removal-update of tasks to adapt the device and the whole system to its environment. It may be a reaction to a fault or even optimization of the system functional behavior. Nevertheless, such a scenario can cause the violation of real-time or energy constraints, which is considered a critical run-time problem. We propose a multi-agent adaptive architecture to handle dynamic reconfigurations and ensure the correct execution of the concurrent real-time distributed tasks under energy constraints. The proposed architecture integrates a centralized scheduler agent (ScA) which is the common decision making element for the scheduling problem. It is able to carry out the required run-time solutions based on operation research techniques and mathematical tools for the system’s feasibility. This architecture assigns also a reconfiguration agent (RAp) to each device to control and handle the local reconfiguration scenarios under the instructions of ScA. A token-based protocol is defined in this case for the coordination between the different distributed agents in order to guarantee the whole system’s feasibility under energy constraints.
Multi-agent Adaptive Architecture for Flexible Distributed Real-time Systems

HAMZA CHNITER\(^1,2\), YUTING LI\(^3\), MOHAMED KHALGUI\(^1,2\), ANIS KOUBAA\(^4\), ZHIWU LI\(^5,7\), FETHI JARRA\(^6\)

\(^1\)School of Electrical and Information Engineering, JiaoTong University (Zhuhai Campus), Zhuhai 519070, China
\(^2\)National Institute of Applied Sciences and Technology (INSA), University of Carthage, Tunis 1080, Tunisia
\(^3\)School of Microelectronics, Xidian University, Xi’an 710071, China
\(^4\)Prince Sultan University, Saudi Arabia/Gaitech Robotics, China/ CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal
\(^5\)School of Microelectronics, Xidian University, Xi’an 710071, China

Corresponding author: M. Khalgui (e-mail: khalgui.mohamed@gmail.com) and Z. Li (e-mail: zhwli@xidian.edu.cn).

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

ABSTRACT
Recent critical embedded systems become more and more complex and usually react to their environment that requires to amend their behaviors by applying run-time reconfiguration scenarios. A system is defined in this paper as a set of networked devices where each of which has its own OS (Operating System), a processor to execute related periodic software tasks, and a local battery. A reconfiguration is any operation allowing the addition-removal-update of tasks to adapt the device and the whole system to its environment. It may be a reaction to a fault or even optimization of the system functional behavior. Nevertheless, such a scenario can cause the violation of real-time or energy constraints, which is considered a critical run-time problem. We propose a multi-agent adaptive architecture to handle dynamic reconfigurations and ensure the correct execution of the concurrent real-time distributed tasks under energy constraints. The proposed architecture integrates a centralized scheduler agent (ScA) which is the common decision making element for the scheduling problem. It is able to carry out the required run-time solutions based on operation research techniques and mathematical tools for the system’s feasibility. This architecture assigns also a reconfiguration agent (RA\(_p\)) to each device \(p\) to control and handle the local reconfiguration scenarios under the instructions of ScA. A token-based protocol is defined in this case for the coordination between the different distributed agents in order to guarantee the whole system’s feasibility under energy constraints.

INDEX TERMS
Embedded system, Integer programming, Low power consumption, Multi-agent architecture, Multi-processor reconfiguration, Real-time scheduling.

NOMENCLATURE

\(T\): Set of periodic tasks
\(n\): Number of tasks in the system
\(T_i\): \(i\)-th periodic task, \(i = 1 \ldots n\)
\(m\): Number of processors in a system
\(nbf\): Available scaling factors for each processor
\(\tau_i\): Release time of task \(T_i\)
\(d_i\): Absolute deadline of task \(T_i\)
\(C_{kip}\): Effective computational time of task \(T_i\)
\(U_{kip}\): Utilization factor of task \(T_i\) on processor \(P_p\)
\(t_{kip}\): Start time of task \(T_i\) on processor \(P_p\)
\(U_{tot}\): Total utilization in all processors
\(U_{max}\): Largest utilization of any task in any processor
\(C\): Constant related to the processor type

\(V_{np}\): Normalized voltage of processor \(P_p\)
\(F_{np}\): Normalized frequency of processor \(P_p\)
\(F_{ip}\): Frequency of processor \(P_p\) when executing task \(T_i\)
\(V_{ip}\): Voltage of processor \(P_p\) when executing task \(T_i\)
\(P_p\): \(p\)-th processor
\(\eta_k\): \(k\)-th available scaling factor of a processor
\(P\): Power consumption
\(C_{nip}\): Computational time of task \(T_i\) at normalized processor frequency
\(X_{ipk}\): Variable describing the assignment of task \(T_i\) to processor \(P_p\) with scaling factor \(k\)

CM: Content message
I. INTRODUCTION

A n embedded system (ES) is a device dedicated to specific functions. It includes hardware and software parts which are designed to operate without human intervention. They run usually under real-time constraints that determine their reliability and accuracy [1]. A real-time embedded system is any system whose correctness depends on both functional and temporal aspects [2], [3].

Embedded real-time systems can be integrated into mobile equipments such as automobiles, airplanes, and smartphones which allow them to perform time sensitive and specific applications under functional and extra-functional constraints [2], [4]. Reconfigurable real-time systems aim to combine the benefits of flexibility by using a distributed architecture [5]–[7]. Many of these systems are often safety-critical and may be characterized by diverse degrees of timeliness constraints [8]. A reconfiguration scenario is technically defined as any operation allowing the addition-removal-update of OS software tasks to adapt the system architecture to its environment [9], [10]. The reconfiguration may be a reaction to a fault or even optimization of the system behavior. Nevertheless, it can cause the violation of real-time or energy constraints which is a critical run-time problem.

The real-time reconfigurable embedded systems require new flexible and adaptive solutions for a true real-time schedule under power constraints [11]–[13]. However, the development and design of a scheduling architecture with high quality for real-time environment is difficult and complex. The recent advanced technologies that enable communication and coordination in a computing system provide the perfect way to implement real-time based software solutions.

This paper deals with a scheduling problem of real-time tasks in distributed architectures supporting the dynamic voltage and frequency scaling (DVFS) capabilities [2]. The DVFS technique is considered in the modeling phase to dynamically supervise the supply voltage and clock frequency of the processors. These processors support different power consumption profiles and processing speeds. Depending on the system workload, the proposed model tries to scale up or down the supply voltage and clock frequency in order to save power when keeping all operational constraints of the system.

In the current study, we are interested in the control of reconfigurable discrete-event systems. Their evolution is governed by the occurrence of asynchronous discrete events. We consider a system composed of a set of networked devices with limited hardware resources [14]. Each device has its own OS, a processor to execute the local tasks and a battery as a local power source. The considered tasks are non-preemptive, synchronous and independent [15]. The processors are assumed to be uniform with DVFS capabilities. Each task is characterized by a period, a deadline, a first release date, and a normalized duration since the actual duration depends on the voltage scaling factor. In this work, we consider a non-preemptive scheduling since we use limited hardware resources. In fact, the context switching during preemption may cause a delay and subsequently an additional calculation time. Therefore, the non-preemption allows to deploy non-expensive and non-heavy platforms in terms of calculations.

The challenge is to achieve a real-time distributed system which is reliable and accurate [16], [17]. Nevertheless, the difficulty lies in the development and integration of an adaptive distributed architecture supporting these characteristics. Many problems raise in this case such as how to manage and control the feasibility in each device to ensure the functional aspects and how to guarantee a better coherence through a required coordination between the different devices. The main purpose of setting up such an architecture is to ensure the feasibility in the various devices that make up the system. Each device can undergo a reconfiguration scenario that may affect the overall system feasibility. The charge of executing the tasks can also exceed the CPU capacity. Each device has its appropriate battery which may not have the necessary load to operate the processor. Since the system is composed of multiple devices, a synchronization must be established through the communication between them.

The communication between the different components in the proposed architecture is based on a set of self-ruling intelligent software entities called agents [5], [18]. This architecture presents the way in which the roles and the relationships between agents are defined. An agent can take the role of controlling, reasoning and making decision according to the environment reaction. Moreover, the agents can cooperate and coordinate by communication.

This architecture integrates two active agents which are: Reconfiguration agent (RA_p) assigned to Device_p (p = 1...m) and scheduler agent (ScA). These intelligent agents cooperate and communicate to ensure the operating con-
straints and to control the stability of the system. The data transmission between the different devices is ensured by a token that takes the form of a data frame and is circulated continuously between the devices. A device should grab the token and use it as a vehicle for transmitting the information.

The reconfiguration agent (RA_p) is a software component assigned to each device in order to dynamically perform local run-time reconfigurations according to functional or environment requirements. The scheduler agent (ScA) is considered as the common decision making element for the scheduling problem. It has the autonomy to perform its functions which are constructed by a number of behaviors and communications triggered as a set of modules.

The proposed solution is able to operate in dynamic environment in which the system behavior continuously changes. This new solution produces a coherent communication between the different components through the related agents.

All the proposed solutions are compared according to a set of metrics such as energy, execution time, and makespan. Various tests were carried out with different instances. IP and SA are compared to different related solutions such as memetic algorithm (MA) reported in [48], tabu search (TS) and simulated annealing (SA) reported in [47]. The proposed approaches have been compared with the energy-efficient adaptive scheduling algorithm (HVSA) and rolling-horizon (RH-VHVSA) reported in [38]. Furthermore, another comparison is made with the approaches reported in [46] integrating hybrid genetic algorithm (GA), simulated annealing algorithm (SA), and LINGO software.

To evaluate the proposed architecture, we have selected a set of metrics such as the energy consumption, response time and the number of transmitted messages between ScA and RA_p. The comparison is considered in the worst and best cases of the reconfiguration scenarios that may occur in the system.

Experimental results show also the required coordination between the system components through intelligent agents. This is reflected in the number of received error messages compared with the total exchanged messages after any reconfiguration. The performance of this solution is also seen through a gain in terms of execution time of the distributed system compared with the centralized one despite the large number of synchronization messages. The energy gain also approves the performance and efficiency of the proposed solution. The token has allowed to valorize the synchronization between the different components in the system by reducing the total number of exchanged messages. This also ensures the accuracy of the information transmitted by each agent.

The originality of this paper can be summarized as follows:

- The decision making modules of the scheduling agent are implemented by using operation research techniques and mathematical tools. A multi-objective mathematical model is formulated for this reason. The asset of the proposed model consists in the possibility of adaptation by adding constraints and tuning on the objective functions to solve analogous problems. The considered optimization approach seeks to produce a sub-optimal solution by optimizing the calculation time, the makespan and the energy consumption in the whole system.
- An implementation of the proposed solution is devoted to schedule and evaluate the performance of the multi-agent architecture. The implementation is based on Java technology supported by Real-Time Specification for Java (RTSJ).

The remainder of this paper is organized as follows: We discuss in Section 2 the originality of this paper by studying the state of the art. In Section 3, we expose the formalization of the problem, and we present the proposed architecture in Section 4. In Section 5, we detail the integer programming formulation and heuristics for reconfigurable distributed embedded systems. Finally, we present a UML-based approach to explain the communication between the different agents and, we detail the experimental simulations and comparisons to showcase and evaluate the performance of the proposed solution.

II. STATE OF THE ART

Different approaches are proposed under a reconfigurable architecture to resolve the scheduling problem by using the DVFS technique [2]. These combinatorial optimization approaches are based on integer programming and heuristics. The work reported in [19] exposes an integer programming model and a heuristic strategy which try to act to the processor speed during the execution of the OS tasks. A mixed integer program (MIP) is formulated in [20] for the scheduling of concurrent real-time tasks. The work reported in [21] exhibits a real-time scheduling middleware called RT-MED.

In [22], [23], the authors present a multi-agent system that maximizes the power production of local distributed generators and optimizes the power exchange. Here, the authors take care just for the energy constraint to minimize the operational cost.

The work reported in [24] presents a study that describes a real-time middleware for distributed service-based applications. The work reported in [25] presents an approach to time bounded reconfiguration in distributed real-time settings. It exhibits a real-time running algorithm divided into a well-bounded set of reconfiguration phases.

In [26], the authors propose a hybrid model to evaluate the response time of real-time system under non-preemptive fixed priority scheduling and reduced the upper bound of network calculus for a multi-hop network. In [27], the authors...
propose a cloud system for real-time monitoring of multi-drone systems used from the tracking of moving objects. However, reconfiguration was not considered in this system.

The real-time online verification of target system configurations is performed in [28]. A predictable cloud computing system in which some tasks should meet temporal constraints is reported in [29].

The work reported in [30] exposes a real-time distributed system in the domain of avionics by using data distribution services over partitioned and virtualized systems.

The authors in [31] present an adaptive bandwidth based group scheduling mechanism that supports the reconfiguration of components in compositional software architectures. However, the software scheduler shows inefficacy in terms of utilization in a multi-core environment.

The work reported in [32] proposes a multi-agent simulation model which is inspired by the ant colony approach. This research presents some limitations such as the disturbance events which are not considered, and the arrival date which is ignored.

In [33], the authors present a decentralized multi-agent system to perform optimal supply and demand matching of the local resources and flexible appliances.

The authors presents in [34] a real-time implementation of a multi-agent-based game theory reverse auction model for microgrid market operations.

The research works reported in [35], [36] deal with the implementation of distributed controllers on networked cyber-physical systems. They expose self-triggered control where agents communicate and make promises between them about their future states.

The constraint satisfaction problem of efficiently allocating virtual machines (VM) resources to physical machines with the aim of minimizing the energy consumption is addressed in [37].

Each of the related works has benefits and limits. Some related works do not address the context of reconfigurable systems and cannot be generalized as the number of tasks increases. A few of the proposed multi-agent architectures produce too many messages between agents which increases the computational cost. Furthermore, the problem of switching processors between different modes of power consumption and scaling the CPU speed according to the workload in a multi-agent system has not been explicitly addressed. Thus, the works supporting all of these criteria provide specific models that remain linked to an explicit application framework, well-defined input parameters and pre-determined environmental constraints which make its update difficult. Therefore, we try to solve the problems mentioned above by proposing a multi-agent architecture that aims to model the communication between all the system components and determine the operational measures while minimizing the energy consumption. The proposed architecture aims to centralize the decision in order to avoid errors.

III. FORMALIZATION OF THE SYSTEM AND RELATED CONSTRAINTS

We formalize a reconfigurable system composed of real-time tasks under energy constraints to be distributed on networked devices. In this section, we present the notation used for the tasks and energy modeling.

We consider a set of $n$ real-time tasks $T = \{T_1, T_2, \ldots, T_n\}$ to be executed upon a distributed platform. The tasks are periodic, independent and non-preemptive [38]. The deadlines are equal to periods and the multi-processing is authorized. We define a hardware platform to be composed of $m$ identical processors where each one has $nbf$ available scaling factors. Each task $T_i$ can be assigned to at most one processor $P_p \ (p = 1 \ldots m)$ and is characterized by the following parameters [1]: (i) Release time $r_i$, i.e., $T_i$ cannot start before time $r_i$ on each processor, (ii) Absolute deadline $d_i$, i.e., $T_i$ must finish execution before deadline $d_i$, and (iii) Computation time at normalized processor frequency $Cn_{ip}$.

We define for each task $T_i$ its utilization factor on $p$-th processor by $U_{ip} \ (i = 1 \ldots n, p = 1 \ldots m)$. The utilization is the ratio of execution requirement to its period in processor $P_p$, i.e.,

$$U_{ip} = \frac{C_{ip}}{d_i} \quad \text{(1)}$$

We define the total utilization $U_{tot}$ to be the sum of the all tasks utilization in each processor, i.e.,

$$U_{tot} = \sum_{p=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{nbf} U_{ip} \cdot X_{ipk} \quad \text{(2)}$$

where $X_{ipk}$ is a binary variable defined in equation (2).

We also define the maximum utilization $U_{max}$ of the tasks to be the largest utilization of any task in any processor. We respectively denote by $F_{n_p}$ and $V_{n_p}$ the normalized frequency and voltage of processor $P_p \ (p = 1 \ldots m)$. We assume that they are proportional.

We suppose that $T_i$ is executed at frequency $F_{ip}$ and voltage $V_{ip}$ in processor $P_p$. Let $\eta_k$ be the $k$-th available scaling factor of voltage on processor $P_p$. We have

$$V_{ip} = \frac{V_{n_p}}{\eta_k} \quad \text{(3)}$$

Thus, we obtain

$$C_{ip} = C_{n_{ip}} \cdot \eta_k \quad \text{(4)}$$

When the system is running at frequency $F_{n_p}$ and voltage $V_{n_p}$, the power consumption is given by

$$P = C \cdot V_{n_p}^2 \cdot F_{n_p} \quad \text{(5)}$$

where $C$ is a constant that depends on the hardware circuit [1]. The power $P_{ip}$ consumed by task $T_i$ on processor $P_p$ is given by

$$P_{ip} = C \cdot V_{ip}^2 \cdot F_{ip} = C \cdot \frac{V_{n_p} \cdot F_{n_p}}{\eta_k^3} \quad \text{(6)}$$
Consequently, the energy $E_{ip}$ consumed by task $T_i$ on processor $P_p$ is expressed by

$$E_{ip} = P_{ip}.C_{ip} = C.\frac{V_{ip}.f_{ip}.C_{nip}}{\eta_k^2} = R.\frac{C_{nip}}{\eta_k^2}$$ (7)

where constant $R = C.V.n_p.F.n_p$. The total energy consumption in the system is presented by

$$E = \sum_{p=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{nbf} E_{ip}.X_{ipk} = \sum_{p=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{nbf} R.\frac{C_{nip}.X_{ipk}}{\eta_k^2}$$ (8)

where

$$X_{ipk} = \begin{cases} 1 & \text{if task } T_i \text{ is assigned to processor } P_p \\ \text{with scaling factor } k \\ 0 & \text{otherwise} \end{cases}$$ (9)

The feasibility of the system represents the starting point to the scheduling problem. It must fulfill the schedulability analysis of all tasks on the different processors while keeping an eye on the energy reserve for the battery of each device. The schedulability analysis presents a crucial phase in the system on the processors without exceeding their capabilities. In the resolution process, For a distributed platform, it consists in the aptness to plan the execution of the different tasks in the system on the processors without exceeding their capabilities while guaranteeing all deadlines.

According to the research work reported in [39], the utilization guarantee for EDF or any other static-priority multi-processor scheduling algorithm cannot be higher than $\frac{m+1}{2}$ for an $m$-processors platform. The research work reported in [40] proves that it is possible to schedule on $m$ processors, any system of $n$ independent periodic tasks with maximum individual utilization $U_{max}$ and a total utilization $U_{tot} < \frac{nk+1}{k+1}$, where $k = \frac{U_{max}}{m+1}$. When $U_{max} = 1$, the guaranteed utilization bound is $\frac{C_{nip}}{\eta_k^2}$. Since each device has its appropriate battery reserve, each battery will have the corresponding capacity which is presented by $B_{ip}^r (p = 1 \ldots m)$. The energy consumption in each device does not exceed the battery capacity; otherwise the feasibility will be lost and the system will fail.

### IV. MULTI-AGENT ARCHITECTURE FOR DISTRIBUTED RECONFIGURABLE EMBEDDED SYSTEMS

After formalizing the problem by using the integer programming approach, we come to a mathematical model that brings together the objective function and a set of constraints such as schedulability, time and energy cost to pattern the system. The proposed model is responsible for the optimal allocation of all tasks to the processors for execution. It is able to determine for each task the execution speed, start time, finish time and effective execution duration on the target processor.

We are now looking to embed the developed solution in a distributed architecture to implement a functional real-time system. However, the development of such an architecture with a high quality of communication and coordination between all the components is difficult and complex. Indeed, it demands several requests such as the system implementation, validation and optimization.

We present in this section an overview of the proposed multi-agent architecture that meets the real-time standards. This architecture attempts to centralize the decision in order to avoid errors. We present the structure of the overall architecture followed by a description of the modules that ensure the coordination and communication between the different agents across a communication protocol.

This architecture integrates two active agents which are: Scheduler Agent (ScA) and Reconfiguration Agent (RA_p).

The objective of these agents is to ensure the operating constraints and to control the stability of the system. These intelligent agents cooperate and communicate on time to perform this challenge. Fig. 1 describes the interactions of these agents in the proposed architecture and outlines the principle functionalities.

**Reconfiguration Agent (RA_p):** The role of the reconfiguration agent consists in locally applying the addition-removal-update of real-time tasks to adapt the related device and the whole system to its environment. However, these functional reconfiguration scenarios may not respond to the time and power requirements and can push the system to an infeasible state. In this case, the RA_p coordinates and requests a help from the scheduler agent which proposes the required solutions by using advanced operation research techniques. RA_p applies them in the related local device.

**Scheduler Agent (ScA):** It is considered as the common decision making element for the scheduling problem. The scheduler has the autonomy to perform its functions which are constructed by a number of behaviors and communications triggered as a set of modules. The computation is centralized in this agent to avoid any error. The decision making modules of the scheduler are implemented using operation research techniques and mathematical tools. Once a request is received from RA_p, the scheduler triggers proactively the coordination module and the solver which is based on mathematical programming and heuristics. It uses advanced operation research techniques in decision making modules to produce an optimal solution [19], [20], [41]. It is responsible

<table>
<thead>
<tr>
<th>Real-time systems</th>
<th>[1], [2], [3], [8], [9], [10], [20], [21]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reconfigurable systems</td>
<td>[5], [6], [7], [12], [13], [14], [15], [23]</td>
</tr>
<tr>
<td>DVFS capabilities</td>
<td>[2], [3], [13], [14], [15]</td>
</tr>
<tr>
<td>Distributed systems</td>
<td>[4], [5], [8], [9], [12], [18], [19], [22], [25], [28], [29], [30], [31], [38], [39], [40]</td>
</tr>
<tr>
<td>Multi-agent systems</td>
<td>[16], [17], [24], [25], [26], [27], [28]</td>
</tr>
<tr>
<td>Middleware</td>
<td>[18], [21], [15]</td>
</tr>
</tbody>
</table>

**TABLE 1:** Classification of the related works.
for (i) the management of the consumed energy in the system, (ii) ensuring the calculation of the solution, (iii) sending statistics on the current state, (iv) updating a non-feasible solution, and (v) communicating the solution to each RA<sub>p</sub>.

ScA is capable to carry out the necessary calculation to produce a solution which supports the system’s demand to ensure the feasibility. It includes (i) an executive, and (ii) CPLEX equations to deal with calculations. CPLEX<sup>1</sup> is a commercial optimization software package to handle mathematical programming problems [42]. It presents a decision analytical tool that enables the rapid development and deployment of optimization models by using mathematical and constraints programming.

**Token:** The access to different devices is supported by a dedicated token which ensures the regular passage through each one to exchange the messages and the related information. The token is used to minimize the traffic of messages between all entities by avoiding the point-to-point communication. In fact, token ring adapts the unbalanced loads which are common in heavy traffic. However, TDMA (Time Division Multiple Access) adaptation is required to support unbalanced loads which will increase the overhead relative to the token ring.

**Communication Protocol:** The proposed architecture requires a specification of how the responsibilities of the system will be distributed among its agents. Such a specification enhances how the related agents will interact with each other to reach their requested liability. The system is composed of <i>m</i> different networked devices. If a point-to-point communication between the devices is considered, then the number of messages will be large and can produce a congested traffic especially when the number of devices is multiplied [43]. This can slow down the communication and eventually the operating services in the system.

To avoid this problem, we propose a token ring topology where a token is used to minimize the exchanges of messages among all entities. A set of communication rules will be necessary to control the interactions of the different agents and the information exchange. This architecture requires a specific communication protocol that determines the possible relationships among all agents. This protocol assumes that the communication among agents takes the form of messages routed from a particular agent to another. This communication allows to share the status and circulate the information between the agents in order to achieve a solution and facilitate the decision-making. For example, a message from RA<sub>p</sub> to ScA allows to exchange information about the device where the reconfiguration occurs, the processor utilization, the assigned tasks and the new parameters after reconfiguration. After calculating solutions, ScA re-sends the new parameters to be applied in each device to reach a

---

feasible system.

The protocol imposes particular constraints on agent messages to manage the agent communication and negotiation. The constraints specify the set of allowed message types, message contents and the correct order of messages during the conversation between agents. The message structure is specified to abstractly represent the message content. A message is structured with three parts: The sender, the receiver and the content. The message is formulated by $Msg(p \in [1 \ldots m])$, where the sender is $S$, the receiver is $R$, and the content message is $CM$. When a reconfiguration is requested in $Device_p(p \in [1 \ldots m])$, $RA_p$ tries to apply it. If it has a local problem about temporal or power constraints, it sends a message $Msg_k$ to inform $ScA$ in order to resolve the detected problem.

The content message (CM) has two different types. The first is called an information content message when the sender is $RA_p(p \in [1 \ldots m])$ and the receiver is $ScA$. It can be formally presented by $ICM(Device_p, AT_p, Up, Ep)$ which includes the appropriate information (assigned tasks, CPU utilization, energy, etc) of each related $Device_p$ at time $t$. Each $RA_p$ puts the information about the device status in the token to be delivered to $ScA$. Each line in the table corresponds to the information inserted by each $RA_p$. Table 2 exhibits an example of a message instance.

$ScA$ receives $Msg_k$ from $RA_p$ and sends a request to collect the required information from all devices. Once the information is collected, it triggers the calculation modules to make a decision about the current situation. When the solution is ready, $ScA$ delivers the content message (CM) to all devices. The second message is called result content message when the sender is $ScA$. This message can be presented by $RCM(T_i, P_p, Cn_{ip}, t_i, f_{t_i}, d_{i}, nd_i, C_{ip}, \eta_i)$. It includes the resulting solution calculated by the scheduler agent and describes the tasks assignment and the new operational parameters such as the start, finish time and the deadline of each task (Table 3). The reliability of the system requires various entities and programs that must work together to achieve the global feasibility. The whole system should react in such a way that all the entities are closely coordinated. A number of intelligent agents have been applied and developed to achieve the objectives. The purpose of these intelligent agents is to achieve faster decision control. This architecture is able to quickly adapt to system changes. It specifies the distribution of the responsibilities among all agents. This new solution produces a coherent communication between the different components through the related agents. It provides good quality both in terms of execution time and energy consumption.

V. OPTIMIZATION-BASED APPROACH FOR INTELLIGENT SCHEDULER AGENT

We are interested in this section in the approaches handled by the $ScA$ agent. This method generates the functional parameters of the feasible system. The correspondent modules attempt to (i) modify the different processors speeds if necessary, and (ii) adjust the basic settings of the system. Here, we introduce two distinct approaches for the online modification of processor’s speeds and tasks periods: Integer programming (IP) and simulated annealing (SA).

A. INTEGER PROGRAMMING APPROACH

The integer programming model includes task’s parameters, constraints to be met throughout execution and the objective function. A set of modules will be called to support this need such as Change processor speed ($\lambda$), feasibility test ($\lambda$), and Determine start_finish time($\lambda$).

1) Assignment Constraints

The main challenge of the scheduling problems upon distributed platforms is to assign tasks to the processors and determine their execution sequences with the corresponding frequencies [44]. We suppose that each processor has a set of $nbf$ available scaling factors. To model the assignment constraint between the tasks, processors and frequency scaling factors, we propose a binary variable $X = (X_{ipk})$, where $i = 1 \ldots n$, $p = 1 \ldots m$, $k = 1 \ldots nbf$, and

$$X_{ipk} = \begin{cases} 1 & \text{if task } T_i \text{ is allocated to processor } P_p \text{ with the scaling factor } k \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

In the system, each task must be executed with one and only one frequency. This constraint is given by

$$\sum_{k=1}^{nbf} X_{ipk} = 1; \quad i = 1 \ldots n; \quad p = 1 \ldots m. \quad (11)$$

In the same way each task must be allocated to only one processor, which is represented by

$$\sum_{p=1}^{m} X_{ipk} = 1; \quad i = 1 \ldots n; \quad k = 1 \ldots nbf. \quad (12)$$

2) Non-preemption Constraints

In this work, we consider a non-preemptive scheduling policy in order to reduce the number of context switching. Let $t_{ip}$ be the effective starting time of task $T_i$ on processor $P_p$. The starting time of each task must have a positive value, i.e.,

$$t_{ip} \geq 0; \quad i = 1 \ldots n; \quad p = 1 \ldots m. \quad (13)$$

Since the treated scheduling problem is non-preemptive, task $T_j$ cannot be started before $T_i$ ends its execution, which means that the difference between the starting times of $T_i$ and $T_j$ is necessarily greater than the execution time of $T_i$ or in reverse if $T_j$ starts before $T_i$. To ensure a single executed task at any particular time, we should have either $t_{jp} - t_{ip} - C_{ip} \geq 0$ or $t_{ip} - t_{jp} - C_{jp} \geq 0$ for every pair of
tasks \( T_i \) and \( T_j \). We add a binary variable \( \alpha_{ij} \) to ensure both inequalities at the same time, i.e.,

\[
\alpha_{ij} \in \{0, 1\}; \ i \neq j; \ i, j = 1 \ldots n. \quad (14)
\]

\( \alpha_{ij} = 1 \) means that \( T_i \) is executed before \( T_j \). To guarantee the respect of constraints (2.5) and (2.6), the starting time of each task should be less or equal to the big constant \( M \), i.e.,

\[
t_{ip} \leq M; \ i = 1 \ldots n; \ p = 1 \ldots m. \quad (15)
\]

\( \alpha_{ij} = 1 \) means that \( T_j \) is executed before \( T_i \), Such a formula means that, if task \( T_i \) starts before task \( T_j \) on processor \( P_p \), then task \( T_j \) cannot start execution on processor \( P_p \) before task \( T_i \) finishes execution. By adding the assignment constraints, we ensure only one active task at any time on each processor. The related constraints are given by

\[
t_{ip} - t_{jp} \geq C_{n_{ip}} \eta_k X_{ipk} - M \alpha_{ij}; \ i \neq j; \ i = 1 \ldots n; \ p = 1 \ldots m; \ k = 1 \ldots nbf. \quad (16)
\]

\[
t_{jp} - t_{ip} \geq C_{n_{jp}} \eta_k X_{ipk} - M (1 - \alpha_{ij}); \ i \neq j; \ i = 1 \ldots n; \ p = 1 \ldots m; \ k = 1 \ldots nbf; \quad (17)
\]

\[
\eta_k \geq 0; \ p = 1 \ldots m; \ k = 1 \ldots nbf. \quad (18)
\]

3) Temporal Constraints

The temporal constraints are assigned when the accuracy of the system is determined by the dates on which the execution results are available.

1) The deadline of each task should be respected in a way that each starting task should finish its execution without violating its deadline, i.e.,

\[
t_{ip} + C_{n_{ip}} \eta_k X_{ipk} \leq d_i; \ i = 1 \ldots n; \ p = 1 \ldots m; \ k = 1 \ldots nbf. \quad (19)
\]

2) Each task should start execution after its release time, i.e.,

\[
t_{ip} \geq r_i; \ i = 1 \ldots n; \ p = 1 \ldots m. \quad (20)
\]

4) Power Constraints

Embedded systems are increasingly incorporating more functionalities that require significant computing power. However, the operation of such systems relies on batteries. The minimization of the energy consumed by the system becomes a very important criterion. Several solutions based on DVFS technology have been performed. These solutions aim to minimize the system energy consumption by adjusting the working voltages and frequencies of the processor. According to the considered architecture of a system, the energy \( E_{ip} \) consumed by task \( T_i \) on processor \( P_p \) is modeled by

\[
E_{ip} = P_{ip} C_{n_{ip}} = \frac{C_{n_{ip}} V_n F_n}{\eta_k^2}; \ i = 1 \ldots n; \ p = 1 \ldots m; \ k = 1 \ldots nbf. \quad (21)
\]

where constant \( R = C V_n F_n \). Then, the total energy consumption in the system is

\[
E = \sum_{p=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{nbf} E_{ip} X_{ipk} = R \sum_{p=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{nbf} \frac{C_{n_{ip}} X_{ipk}}{\eta_k^2}. \quad (22)
\]

The aim is to fulfill the schedulability analysis of all tasks on the different processors while keeping an eye on the energy reserve for each device battery. Since each device has its appropriate battery reserve, each battery will have the correspondent capacity which is presented by \( B^c_p \). The energy consumption in each \( Device_p \) does not exceed the battery capacity. For this reason, we add the following constraint, i.e.,

\[
R \sum_{p=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{nbf} \frac{C_{n_{ip}} X_{ipk}}{\eta_k^2} \leq B^c_p \quad (23)
\]

In this paper, we suppose that the reconfiguration scenarios are not as frequent as the execution frequency of tasks. Therefore, we are not interested in the energy consumed by each reconfiguration.
5) Problem Modelling

After the modeling of assignment, temporal and power constraints related to the present system, we illustrate the basic linear program Prob. The objective is to minimize the energy consumption while respecting the previous defined constraints. This model Prob can be extended to adjust the periods whenever no solution exists even for the highest voltages. We add the variable \( \theta_i \) to represent the increasing factor of the period of task \( T_i \). To confirm the drawing of the period, we propose the following constraint, i.e.,

\[
\theta_i \geq 1; \quad i = 1 \ldots n.
\]  

(24)

We add the constant \( \lambda \) which represents a trade-off weight between the increasing factor \( \theta_i \) and the energy.

The considered objective function becomes

\[
\text{Minimize} \quad \lambda \sum_{i=1}^{n} \theta_i + R \sum_{p=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{nbf} \frac{Cn_{ip}X_{ipk}}{\eta^2k^2}
\]  

(25)

By incorporating the periods adjustment constraint, we guarantee that each task finishes execution before its related deadline. Constraint (12) becomes

\[
t_{ip} + C n_{ip} \cdot \eta_k \cdot X_{ipk} \leq \theta_i \cdot p_i \quad ; \quad i = 1 \ldots n; \quad p = 1 \ldots m; \quad k = 1 \ldots nbf.
\]  

(26)

To control the adjustment of any period \( p_i \), when the system needs this service, we define two parameters \( p_{\min} \) and \( p_{\max} \) to represent respectively the lower and upper bounds of period \( p_i \).

We add equation (20) in the mathematical program EProb.

\[
p_{\min} \leq \theta_i \cdot p_i \leq p_{\max} \quad ; \quad i = 1 \ldots n.
\]  

(27)

The mathematical model is multi-objective and combines a set of performance measurement criteria such as computational time and makespan in addition to the minimization of the energy consumption in the whole system. The computation based on the mathematical programing allows to produce all the functional parameters to achieve a feasible system. This model allows not only to provide an operational solution but also a sub-optimal one. A strength point which characterizes this model is that it is extensible and adaptable to several ranges of real-time systems.

B. HEURISTIC APPROACH

Since these problems are NP-hard [45], it is a common idea to use a heuristic approach to achieve optimal solutions. The simulated annealing (SA) [19] has been implemented in order to compare it to integer programming. The simulated annealing is based on neighborhood search (Fig. 2). It starts with a random solution to improve it over iterations. Such heuristics always move from a solution to the best neighboring one. In order to escape local minima, SA allows different movements in a controlled manner where in each step it generates a perturbation. If the objective function decreases, then the generated solution is accepted. Otherwise, the new state is accepted with a probability related to the increase. The initial starting temperature and the stop criteria should be ensured.

![Heuristic flowchart](image)

**FIGURE 2: Heuristic flowchart**

**Initial Solution:**

The initial solution can be computed by the following way: Among 100 random combinations, we choose the combination which gives the best objective function as the start point. The selected combination should only respect the deadlines of all tasks.

**Objective Function:**

The objective function consists in minimizing the sum of the total energy consumption, the makespan and the total execution time.

**Neighborhood Structure:**

By defining the neighborhood of a configuration in the set of solutions that can be reached from the current one. In practice, a neighbor solution is built by either swapping the allocation order of two tasks randomly selected, or by acting in the execution frequency of a task randomly chosen.

**Simulated Annealing Parameters:**

The main parameters of the simulated annealing method are the initial temperature, the temperature length, the cooling ratio and the stopping criteria.

- **Initial Temperature:** The temperature parameter plays an important role for accepting or rejecting objective
functions. The initial temperature is fixed to 95. It must be high enough such that the final solution is independent from the starting one. It determines the probability of deterioration.

- Temperature Length: The temperature length (40) is the number of iterations at a given temperature. However the temperature length may vary from temperature to temperature and is important to spend a long time at lower temperatures.

- Rate of Temperature Decrease: For less probability of accepting unfavorable solutions, the temperature should be decreased. The cooling ratio is the rate at which the temperature is reduced. In this paper, it is preferred to be fixed to \( \mu = 0.9 \).

- Stop Criteria: In our simulation, the simulated annealing stops when the minimum value of the temperature reaches (5) or a certain number of iterations has been passed without improvement or when a number of 1000 iterations has been reached.

VI. EXPERIMENTATION

This part is devoted to explain the communication between the different agents by using a UML-based approach. Furthermore, experimental simulations and comparisons are depicted to evaluate the performance of the proposed solution.

A. DESIGN OF THE PROPOSED MULTI-AGENT RECONFIGURABLE ARCHITECTURE

The classes diagram presents the various entities that can act in the execution phase. The “Device” class represents an instance of the devices that make up the system. Each device contains an operating system (Fig. 3) that ensures the management and execution of a set of tasks that represent functional requirements. It includes also a processor to execute the assigned tasks and a battery. Each OS handles the coordination with related RA_P. The agent RA_{P} has a “Listener” that takes the role of an event controller, and its objective is to apply the reconfiguration scenario in the
device.

The “Scheduler” class is composed of a set of modules to recover the functional parameters and find solutions that stabilize the system. These modules are based on the “integer programming” and “heuristic” approaches. The class “Reconfiguration Scenario” presents the reconfiguration that any device can sustain. It includes the following modules: (i) Receive status() to confirm the reception of the status according to the standard required, (ii) Update solutions() to re-calculate a new solution according to the current system situation, (iii) Change processor speed() to change the processor speed and to ensure the execution of all tasks while minimizing the energy consumption, (iv) Adjust periods() to change the basic tasks parameters such as periods in order to provide a new parallelism that guarantees the system feasibility, (v) Feasibility test() to measure the feasibility of the system at run-time, (vi) Display results() to take care of the dynamic display which describes the flow of execution and finds results, (vii) CPLEX Java() to use the modules CPLEX Java responsible to execute mathematical programming models and recovery results, (viii) Calculate new periods() to calculate new tasks parameters after modification, and (ix) Determine Start_finish time() to produce the execution sequence of tasks (start and finish time) for each solution.

Each reconfiguration agent RA

is responsible of applying reconfigurations in each device and some of related implemented modules are: (i) Solution Request(): To request a solution from ScA when a local problem appears, (ii) Apply Reconfiguration(): To apply a reconfiguration scenario in the Device

, and (iii) Receive Solution(): To confirm the reception of the sent solution.

The intermediary through which the different agents communicate is represented by the class “Protocol”. Some of important implemented modules are presented as follow: (i) Fix Sender(): Determines the sender of the related message, (ii) Fix Receiver(): Determines the receiver of the related message, (iii) Resolve Error Message(): Detects and resolves the error messages, and (iv) Ensure Message Receipt(): Ensures the reception of all transmitted messages. The sequence diagram presents an order of events and communications between the agents as shown in Fig. 4. The objective is to achieve the requisite outputs and provide the best solution that maintains the system in correct conditions. The reconfiguration agent (RA) stays available and listening to any reconfiguration that can be raised in the related device. Once it spots a reconfiguration, it sends a request to the scheduler agent (ScA) to solve the problem. ScA examines the system feasibility and try to achieve a solution by using optimization approaches which act on the processor speed. Once the solution is accomplished, RA will apply the new reconfiguration in the related system’s device.

A reconfiguration agent (RA) is assigned to each device to apply required reconfiguration scenarios at run-time and under well-defined conditions described in user requirements to adapt the device and the whole system to its environment.

It is not a reconfigurable agent that undergoes reconfigurations. Indeed, the reconfiguration agent is a software component responsible for dynamically applying reconfigurations in each system’s device. A reconfiguration scenario is any run-time operation allowing the addition-removal or update of system software tasks. When the related temporal and power constraints are not satisfied after any scenario, RA changes the tasks’ WCETs provided by the scheduler agent by dynamically changing the processor speed.

B. APPLICATION: FORMAL CASE STUDY

We present a case study in this section to explain the interactions of the different agents through this architecture. We consider a multi-agent distributed platform composed of three devices and the set of tasks presented in Table 4.

Table 4: Information content of RA

<table>
<thead>
<tr>
<th>Device</th>
<th>Assigned tasks</th>
<th>CPU Utilization</th>
<th>Energy</th>
</tr>
</thead>
</table>
| Device 1 | \{T

, T

, T

\} | 0.815 | 1680 |
| Device 2 | \{T

\} | 0.433 | 936 |
| Device 3 | \{T

\} | 0.32 | 768 |

Table 5 below describes the tasks parameters before applying the reconfiguration scenario.

Table 5: Task parameters before reconfiguration

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Release time</th>
<th>WCET</th>
<th>deadline</th>
<th>period</th>
</tr>
</thead>
</table>
| T

 | 0 | 20 | 70 | 70 |
| T

 | 0 | 22 | 80 | 80 |
| T

 | 0 | 39 | 90 | 90 |
| T

 | 0 | 28 | 110 | 110 |
| T

 | 0 | 32 | 100 | 100 |

According to the feasibility test (Section III), we have $1/U_{\text{max}} = \frac{80}{39} = 2.051$. The system is feasible since the total utilization $U_{\text{tot}} \leq \frac{3(2.307\cdot3+1)}{2.307} = 2.395$. The energy is equal to 338.4 Joules.

We assume a run-time reconfiguration scenario in Device

 to add three new tasks under environment requirements shown in Table 6. Table 7 describes the parameters of the global tasks system after the reconfiguration scenario.

Table 6: Information content transmitted from RA

<table>
<thead>
<tr>
<th>Device</th>
<th>Assigned tasks</th>
<th>CPU Utilization</th>
<th>Energy</th>
</tr>
</thead>
</table>
| Device 1 | \{T

, T

, T

\} | 0.815 | 1680 |
| Device 2 | \{T

, T

, T

\} | 2.474 | 5616 |
| Device 3 | \{T

\} | 0.32 | 768 |

Table 7: Tasks parameters after reconfiguration

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Release time</th>
<th>WCET</th>
<th>deadline</th>
<th>period</th>
</tr>
</thead>
</table>
| T

 | 0 | 20 | 70 | 70 |
| T

 | 0 | 22 | 80 | 80 |
| T

 | 0 | 39 | 90 | 90 |
| T

 | 0 | 28 | 110 | 110 |
| T

 | 0 | 32 | 100 | 100 |

Table 5 below describes the tasks parameters before applying the reconfiguration scenario.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Release time</th>
<th>WCET</th>
<th>deadline</th>
<th>period</th>
</tr>
</thead>
</table>
| T

 | 0 | 20 | 70 | 70 |
| T

 | 0 | 22 | 80 | 80 |
| T

 | 0 | 39 | 90 | 90 |
| T

 | 0 | 28 | 110 | 110 |
| T

 | 0 | 32 | 100 | 100 |

Table 5 describes the parameters of the global tasks system after the reconfiguration scenario.
According to the new system parameters after reconfiguration, ScA analyses the feasibility. The total utilization \( U_{tot} = 3.610 \geq 1.235 + 1.235 + 1 = 2.105 \). The new system becomes infeasible since the timing constraints cannot be met and the energy consumption increases to 8064. The related RA\(_2\) automatically sends an information message to ScA which takes the present format: \( Msg_k( RA_2, ScA, ICM) \). The message ICM is presented by Table 6.

ScA requests the information from all devices where each one puts the corresponding information across the token (Table 4). The token will gather the information and deliver it to ScA which should now ensure the feasibility of all tasks while consuming no more energy. It calls its specific modules based on the optimization approaches to calculate a solution according to the given information.

Once a solution is ready, ScA sends a resulting message to all RA\(_p\). The message contains the following information: \( Msg_k( ScA, RA_2, RCM) \) where the scheduler agent ScA represents the sender, the reconfiguration agent RA\(_2\) represents the receiver and the result content message RCM includes the resulting solution calculated by ScA. The message RCM is described in Table 9. ScA computes for each task the start time, finish time, and the new WCET after changing the scaling factor of the processor speed. This result presents the output of the module which is dedicated to calculate the system feasibility. The new reconfiguration that will be applied in the system as a solution is shown in Table 8.

FIGURE 3: Class diagram.
FIGURE 4: Sequence diagram of a reconfiguration scenario.

TABLE 8: Information content transmitted to RA2

<table>
<thead>
<tr>
<th>Device</th>
<th>Assigned tasks</th>
<th>CPU Utilization</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device1</td>
<td>{T4, T8}</td>
<td>0.86</td>
<td>2188</td>
</tr>
<tr>
<td>Device2</td>
<td>{T1, T2, T7}</td>
<td>0.36</td>
<td>715</td>
</tr>
<tr>
<td>Device3</td>
<td>{T5, T6, T3}</td>
<td>0.74</td>
<td>1588</td>
</tr>
</tbody>
</table>

C. PERFORMANCE EVALUATION

We have used ILOG CPLEX 11.1 solver to execute the integer programming model on a mono-processor core 2 duo, 1.2 Mhz and 1 Giga RAM. CPLEX\(^2\) is a commercial optimization software package to handle mathematical programming problems [42]. It presents a decision analytical tool that enables the rapid development and deployment of optimization models by using mathematical and constraints programming.

In the conducted experimentation, we have randomly generated different task sets with 50 to 400 tasks. In Tables 10 and 11, the first column shows the size of the problem (number of tasks). The sub-column labeled “Time” indicates the computational time in milliseconds for each approach. The sub-column labeled “Energy” gives the total energy consumption. The sub-column labeled “Makespan” gives the maximum execution time from all tasks in the system. The methods faced a common objective and identical constraints. Table 11 shows that the energy consumption of the applied integer program is lower than that of the heuristic. However for the large size instances, the heuristic is still much faster. Moreover, the two approaches guarantee that all constraints are respected.

Figs. 5, 6, 7 and 8 describe the evolution of the energy, execution time and makespan for an instance of 200 tasks executed upon different multi-processor platforms (4CPU, 8CPU, 16CPU). According to the energy consumption, the integer programming (IP) is more effective than simulated annealing (SA) for almost different instances from 50 to 400 tasks and platforms (Figs. 5 and 6). In fact, it allows to more explore the search space and gives a fairly optimal solution. The SA approach allows to achieve sub-optimal results in a reasonable time. It proves its efficiency in terms of computational time and shows a large difference with the

especially large-size task sets and processors (Fig. 7). IP needs more time than SA to produce the desired result. This is because IP represents an exact method which explores all available solutions then selects the optimal one. SA is an approximation method which tries to better explore the research space of solutions and keeps the closest approximation solution to the optimal.

![FIGURE 7: Comparison between SA an IP based on computational time.](image)

About the makespan metric, the two approaches produce close results under a platform of four processors. When we multiply the number of processors, the IP approach approves its ahead versus SA from different instances (Fig. 8). In fact, IP is an exact method which is able to explore more the research space and execute widely the available combinations according to the related constraints. Yet, this can influence on the computing time which can be enough high.

![FIGURE 8: Comparison between SA an IP based on the Makespan.](image)

Figs. 9 and 10 present comparative evaluation of IP and SA based on the number of periods and adjusted frequencies during the execution of the instances. The result still shows...
the advantage of SA over IP.

![Graph of Adjusted frequencies vs Number of processors for SA and IP](image)

**FIGURE 9:** Comparison between SA an IP based on the number of adjusted frequencies.

![Graph of Adjusted periods vs Scenarios of reconfiguration for SA and IP](image)

**FIGURE 10:** Number of adjusted periods after each reconfiguration scenario.

According to the SA complexity (Fig. 11), we show that it offers much better quality to solve the scheduling problem of real-time tasks with a reasonable execution time. One of the algorithm features lies in its adaptation to different constraints such as temporal constraints, precedence constraints and resource-sharing constraints.

We must also admit that the choice of parameters requires certain skills, especially for simulated annealing, where there are more than one parameter to be tuned, namely the number of iterations and the temperature update. It remains difficult to demonstrate SA’s theoretical complexity for this problem.

The proposed solution allows to compute the scaling factors more than the execution sequence of tasks, the start and the finish time of each task. Table 12 and Fig. 12 show that the proposed approaches (IP and Heuristic) produce better results than the works presented in [46] according to the computational time for instances of 10 to 25 tasks with a multi-processor platform composed of 10 and 15 processors. The related approaches do not address also the context of reconfigurable systems. In Table 14, the proposed approaches have been compared with the work in [38] (Table 15) with an instance of 300 tasks to confirm the lower energy consumption. The result shows that IP and SA are more efficient than HVSA and RH-VHVSA and give less energy consumption when the number of processors

![Graph of Experimental complexity of SA](image)

**FIGURE 11:** Experimental complexity of SA.

**TABLE 12:** Comparison between IP, SA and Majazi [46]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>145</td>
<td>80</td>
<td>350</td>
<td>165</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>335</td>
<td>205</td>
<td>400</td>
<td>200</td>
<td>164</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>211</td>
<td>135</td>
<td>416</td>
<td>121</td>
<td>850</td>
</tr>
<tr>
<td>25</td>
<td>15</td>
<td>803</td>
<td>460</td>
<td>900</td>
<td>750</td>
<td>940</td>
</tr>
</tbody>
</table>

![Graph of Comparison between IP, SA and Majazi based on execution time](image)

**FIGURE 12:** Comparison between IP, SA and Majazi [46] based on execution time.
TABLE 13: Comparison between IP, SA, Saracicek [47] and Serafettin [48]

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Processors</th>
<th>IP</th>
<th>SA</th>
<th>MA</th>
<th>TS</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>[48]</td>
<td>[47]</td>
<td>[47]</td>
<td>[47]</td>
<td>[47]</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>221</td>
<td>113</td>
<td>240</td>
<td>170</td>
<td>188</td>
</tr>
<tr>
<td></td>
<td></td>
<td>57.2</td>
<td>67.43</td>
<td>73.05</td>
<td>104.31</td>
<td>97.12</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>266</td>
<td>156</td>
<td>1730</td>
<td>1739</td>
<td>1047</td>
</tr>
<tr>
<td></td>
<td></td>
<td>35.52</td>
<td>51.06</td>
<td>69.46</td>
<td>56.53</td>
<td>72.10</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>512</td>
<td>270</td>
<td>2690</td>
<td>1559</td>
<td>1078</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21.48</td>
<td>23.17</td>
<td>34.62</td>
<td>37.31</td>
<td>59.08</td>
</tr>
</tbody>
</table>

FIGURE 13: Comparison between IP, SA and Chuan [38].

increases (Fig. 13). IP and SA have lost the competition in front of the provided solution in [38] on platforms of 4 and 8 processors in terms of energy consumption. However, their performance is seen when the number of processors becomes more important (e.g. 16 CPUs).

The work in [38] does not deal with the context of reconfigurable systems and cannot be generalized as the number of tasks increases.

Our approaches are also compared with [47] and [48] in term of execution time and makespan (Table 13). The comparison confirms the performance that the current paper gives optimal solutions either in the execution time (Fig. 14) or the makespan (Fig. 15).

Figs. 16 and 17 show some communication statistics and performance of the developed architecture. The response time and error message rate are presented to evaluate the architecture performance.

To comprehensively evaluate the proposed architecture, we have selected a set of metrics such as the energy consumption (Fig. 18), response time (Fig. 19) and the number of transmitted messages between ScA and RAp (Figs. 20 and 21).

The comparison is considered in the worst and best cases of the reconfiguration scenarios that may occur in the system. The worst case is considered when all devices undergo reconfigurations. The other case describes the situation when just one device is affected by the reconfiguration.

D. DISCUSSION AND ORIGINALITY

The quality of the solution generated by the heuristic is worse than that produced by CPLEX because the heuristic provides an approximate solution. However, we notice that the gap or distance to the optimality between the two solutions is low, which proves the performance of this approach.

We note that the approaches (IP and SA) are compared with HVSA, RH-VHVSA, GA, MA, TS, and Lingo reported
TABLE 14: Comparison between IP, SA and Chuan [38]

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Processors</th>
<th>Energy Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>4</td>
<td>1189</td>
</tr>
<tr>
<td>300</td>
<td>8</td>
<td>1489</td>
</tr>
<tr>
<td>300</td>
<td>16</td>
<td>1409</td>
</tr>
</tbody>
</table>

FIGURE 16: Response time evaluation with multiple set of processors.

FIGURE 17: Message error rate.

in [38], [46]–[48] (Table 15). Those works are based on mathematical approaches and heuristics but they are not designed to be applied in reconfigurable systems, which is the case with this work. In addition, apart from the work in [38], they do not take into account the energy consumption constraint which generally requires more computation and thereafter more time to produce the solution compared with the current work which incorporates several assessment metrics such as the energy consumption, the execution time and the makespan.

As far as the related execution platform of those works, it can be seen that the application environment and the
hardware tools are more efficient than those used in the actual work. Despite all this, the proposed solution gives better results which proves its quality and efficiency.

VII. CONCLUSION

This paper deals with a scheduling problem of real-time tasks in distributed architectures supporting the dynamic voltage and frequency scaling. The system composed of a set of networked devices can undergo run-time reconfiguration scenarios which can cause critical problem such as the violation of real-time or energy constraints.

The paper presents a set of combinatorial and optimization solutions which are implemented and integrated to produce a real-time adaptive multi-agent architecture. A multi-objective mathematical model which is extensible and applicable to a wide range of analogous systems is formulated for this reason. The proposed architecture aims to centralize the decision in order to avoid errors and to try to produce suboptimal and flexible solutions for a true real-time schedule under power constraints.

Through this paper, we have made significant progress toward the development of a real-time and reconfigurable architecture with performance criteria. However, the conducted work is only an initial step in this direction. Several major tunings still remain. Some of them are related to the capability of the architecture while others are under real-time constraints. A future study will include more extensions for complementary adaptation, intelligent schedulers, control and reliable coherence in a distributed system.

In this work, we consider a non-preemptive policy which is not time and energy costly especially if we use limited hardware resources. Nevertheless, we will consider preemptive policies in the future work. The considered tasks in this work are periodic. Nevertheless the consideration of an extended model that considers mixed categories of tasks (aperiodic and sporadic) is a necessity. We will consider the dependency constraints between tasks which is frequently present in modern real-time applications.

REFERENCES

**TABLE 15: Comparison with related works**

<table>
<thead>
<tr>
<th>References</th>
<th>Reconfiguration</th>
<th>Architecture</th>
<th>Preemption</th>
<th>Tasks</th>
<th>Approaches</th>
<th>Energy constraints</th>
<th>Execution platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>[38]</td>
<td>No reconfiguration</td>
<td>Multi-processor</td>
<td>Non-preemptive</td>
<td>Aperiodic</td>
<td>Mathematical approaches</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>[46]</td>
<td>No reconfiguration</td>
<td>Multi-processor</td>
<td>Non-preemptive</td>
<td>Periodic</td>
<td>Mathematical approaches</td>
<td>No</td>
<td>PIV, 3.2 GHz, 2GB RAM</td>
</tr>
<tr>
<td>[48]</td>
<td>No reconfiguration</td>
<td>Multi-processor</td>
<td>-</td>
<td>-</td>
<td>Mathematical approaches</td>
<td>No</td>
<td>Intel Core 2 Duo T6400, 2 GHz</td>
</tr>
<tr>
<td>[47]</td>
<td>No reconfiguration</td>
<td>Multi-processor</td>
<td>Periodic</td>
<td>Heuristic approach</td>
<td>No</td>
<td>Intel Due T2300 GHz</td>
<td></td>
</tr>
</tbody>
</table>

This work: Each device has a reconfiguration agent

<table>
<thead>
<tr>
<th>References</th>
<th>Reconfiguration</th>
<th>Architecture</th>
<th>Preemption</th>
<th>Tasks</th>
<th>Approaches</th>
<th>Energy constraints</th>
<th>Execution platform</th>
</tr>
</thead>
</table>