New Solutions for AI-Based Adaptive System Under Real-Time and Low-Memory Constraints

Walid Bouzayen*†, Hamza Gharsellaoui*‡, Mohamed Khalgui*§
*LISI, INSAT Institute, Carthage University, Tunisia.
†FST, Tunis El Manar University, Tunisia.
‡ENI-Carthage, Carthage University, Tunisia.
§SystemsControl Lab, Xidian University, China.
Email: {walid.bouzayen, gharsellaoui.hamza, khalgui.mohamed}@gmail.com

***Abstract*—This paper deals with a new approach for real-time feasibility of reconfigurable embedded systems based on Artificial Intelligence (AI) with low power and memory consumption. Reconfigurable Real-Time Embedded Control Systems (RRTECS) update their control behaviour based on a set of functional conditions or critical situation to adapt the system to environment uncertainties and to optimize the hardware resources governance. Using AI features, a set of production rules, governs the cited reconfiguration scenarios and make the ability of autonomous and adaptive control behaviour. Complex and concurrent reconfigurations scenarios decrease the performance of process control where inference response time increases due to big rule base and Quality of Service (QoS) halts due to rule's lacks. In this context, we propose an architecture with an adaptive rule base for the reconfiguration of intelligence process. Also, we propose to study the real-time scheduling and feasibility of the proposed approach. A strategy for safety system in critical battery charge and a paging algorithm for memory overload based on real-time on-line memory compression are proposed in this research work. A Multi-Agent System (MAS) is used as an implementation of the proposed contribution with experimental studies and comparative works.**

***Index Terms*—Adaptive Control, Real-Time Feasibility, Dynamic Software Reconfiguration, Memory Overload**

Regular Research Paper

I. Introduction

With the grow of the Internet of Things (IoT), autonomous and adaptive control become a big field of research. Embedded hardware infrastrcutre and various software architecture are enhancing many implementation in manufacturing, agriculture, health and social areas. Using different sensors interrupts and heterogeneous controllers implementation, the dynamic software reconfiguration technology is implemented to get adaptive, feasible and economic control behaviour in embedded systems. Indeed, those systems are battery-powered and have limited resources in memory and processor frequency. For this, economic control strategy is used to minimize the processor utilisation and the memory allocation by deactivating unused controllers based on system execution context. Feasible real-time constraints are a major field of research to guarantee the response time of process control. Finally, correct behaviour is the semantic to satisfy functional description of services by avoiding contradictory in controllers outputs. In this context, Weiser [1] cited the important axes in pervasive computing where adaptive and autonomous system should guarantee human invisibility and system context awareness. Authors of our community used the terms of dynamic reconfiguration [11], [4] as a reference for adaptation where a system is composed of a set of controllers with a set of configurations to describe the functional behaviour of the system. Changing from one configuration to another describes the controllers reconfiguration process where activating, deactivating and updating tasks is a dynamic solution to manage the running tasks as a strategy to optimize resources allocation and context awareness for environment changes. Using AI features, inference rules are implemented in a Rule-Based System (RBS) for controllers reconfiguration to get adaptive and autonomous system behaviour. Indeed, RBS uses sensors interruptions to decide which sets of tasks should run once a moment. Complex RRTECS based AI solution may have huge number of rules which affect the RBS response time and the memory allocation. To face this problem, our previous work [18] proposes a new strategy described as the reconfiguration of intelligence where the rule base is dynamically reconfigured at run-time. This solution is based on a set of contextual parameters dealing with a set of localizations and an agenda of dates. Nevertheless, real-time scheduling is not studied above where activating new set of rules is a probabilistic task and a reconfiguration of controllers may hold-up real-time feasibility. Also with battery-powered system and low size memory constraints, critical battery charge and memory surcharge should be faced with new strategy for adaptive RRTECS based AI-solution. We propose new solutions for intelligence adaptation with real-time guarantee for the described above situation and context awareness.

In this research work, an RRTECS is made by a set of dynamic running tasks, a dynamic reconfigurable rule base and an inference engine. The real-time tasks are split in two types; the Static Controller Tasks (SCTs) which are static and don't accept the features of reconfigurations. The second type of tasks are the Behavioural Controller Tasks (BCTs) which their states/parameters (activation/deactivation, parameters modification) are managed by inference rules residing in the Effective Rule Base (ERB). We discuss the whole system architecture on a mono-processor with a study of the real-time scheduling of the different tasks and the feasibility of the system after

each reconfiguration scenario. A new factor is proposed as the intelligence impact of each task to describe its consistency in the process control over the environment changes. Based on this factor, a new strategy is presented in this work for real-time reconfiguration and for low power management. Also, we propose to use the features of memory on-line compression [19] as a solution for memory overload using virtual memory support. We propose a new approach for the memory paging management as a composition of Least Recently Used (LRU) [20] and a new dedicated page replacement algorithm using the intelligence impact of the tasks.

The remainder of the paper is organized as follows. Section 2 presents a brief background to our work presented in [18] and an overview of the application of real-time AI systems and the dynamic reconfiguration technology. Section 3 describes the intelligent system architecture and the system formalization. Section 4 presents the new strategy for real-time AI reconfiguration with low-power consumption. In Section 5, we present the proposed MAS architecture. Section 6 presents a case study and a set of simulations with comparative works and finally Section 7 concludes the paper.

II. Background and State of the Art

This research work is based on the work reported in [18] where we have proposed a MAS architecture for the intelligence adaptation of RRTECS.

Indeed, the rule base is split into an Effective Rule Base (ERB) and the Generic Meta Base (GMB) to optimize the inference time and the memory consumption of the RBS. Also, we have proposed the IF as the intelligence QoS of the system which is presented in the next equations as :

$$(IF): \begin{cases} IF(O,T) & = & \frac{\sum_{i=1}^{N} IC(SC(i))}{N} \\ IF(E,P) & = & \frac{\sum_{i=1}^{S} IH(SM(i),P)}{S} \\ IF(T) & = & IF(E,P)+IF(O,T) \end{cases} \quad (1)$$

Where the *SC* is a set of the installed BCTs in the system and *IC* is a boolean value to describe the status of each BCT in the system. If a rule exists in the effective base where its RHS is equal to *SC(i)* then the value is 1 else it is equal to 0. The same thing for the IH function which describes the behaviour of the system against the accepted sensors measurements *SM(i)* after each polling period P. The solution process is implemented with a MAS where we proposed an agent to evaluate the sensors measurements, then it holds only the values for the current dimensions of system context execution; the environment, the date and the controlled physical object. Another agent to hold the status of the physical objects, by activating/deactivating a physical object, a reconfiguration of controller is executed. Finally, the reconfiguration of intelligence agent which helps the system to adapt the rule base in function of the IF variation.

We propose to extend the proposed architecture to real-time, memory overload and low-power consumption constraints.

A. Artificial Intelligence

AI is based on logical language instruction with a rule base and an inference engine which provides the right action/decision to run. Data representation and inference process are the axes to distinguish one technique to other. RBS is the most used implementation of the AI in industry [2]. The production rules have LHS describing the conditions to run this specific rule and they have a RHS which is the action to fire by the inference engine. Many rule languages and inference engines are used to implement RBS. Most cited in literature are, Prolog, CLIPS [3].

B. Embedded Artificial Intelligence

Many works aim to implement AI techniques on an embedded platform. Nalepa and Ziecik in [6], developed an Embedded Prolog Platform for embedded hardware. The authors use XTT format for the data representation of the rule base and a supervisor to control the device interrupt and using the Linux run-time. The work reported in [5], developed a universal XML Scheme for knowledge base and it proposes two approaches for hardware assistance and acceleration with the utilization of programmable hardware. The work described in [7] deals with the implementation of a microcontroller based rule inference system for smart home on an Arduino platform. Davis et al. in [8] present a taxonomy and a survey in the application of AI techniques for adaptive RTECS. The paper describes the behaviour of different AI methods with a set of areas from distributed, real-time, embedded, robust and autodidactic. Authors show in this work that a variety of AI techniques may be applied to adaptive RTECS. Unlike this survey didn't propose an implementation of adaptive AI techniques in RTECS with a low memory constraint and power consumption. We propose a MAS architecture with a CLIPS implementation to discuss the features of AI-based RRTECS.

C. Reconfiguration Technology in Embedded Real-Time Systems

In literature, some works deal with the hardware reconfiguration to perform multiple hardware behaviours using programmable hardware [10]. Other works deal with the technology of software reconfiguration to get adaptive behaviour with the system context execution. Wang et al. in [4] used the updating of tasks parameters for probabilistic and periodic tasks when they developed a new approach for the real-time scheduling of the embedded system under study by defining virtual processors. This research work deals with the reconfiguration of periodic and aperiodic tasks which are triggered by simple events. Nevertheless, no one in our community has address the problem of the reconfiguration of intelligent tasks which are triggered by a rule engine. In this context, our paper's contribution deals with the reconfiguration of BCTs.

Dealing with memory overload, authors of our community has developed some techniques as paging based on virtual memory support. The authors in [19] propose to implement swapping in RAM where they used LRU algorithm [20] for page replacement. Indeed, the optimization in Yang's work [19] is when a page fault is handled by the system, a compression of least recently used page is swapped out into a

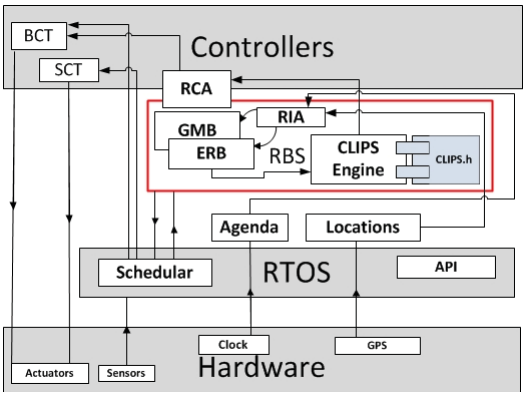

Fig. 1: System Architecture.

compressed area of the main memory. They propose to split the main memory into two areas; uncompressed and compressed one. We propose to use this features with a new proposition for page replacement algorithm based on system intelligence QoS.

III. Formalization

RRTECS is defined in this paper as follow Sys={β, $\gamma(t)$, F} where β is the set of tasks having two types β_{st} which are static tasks without reconfigurations features and β_{dy} which are dynamic reconfigurable tasks. $\gamma(t)$ are set of production rules in ERB for managing the controllers reconfiguration using the inference engine F. Indeed, a set of functional conditions dealing with sensors measurements trigger a reconfiguration of controller by activating/deactivating or/and modifying a set of tasks. We denote n the cardinality of β_{st}, m the cardinality of β_{dy} and finally b for $\gamma(t)$. Let S denotes the set of system sensors; $S = \{S_1, S_2, .., S_v\}$ then the inference process is defined as follows :

$$F : S_k.value \rightarrow \{\gamma_j \in \gamma(t)\} \tag{2}$$

$$\gamma_j : EC_j \rightarrow \{\{\beta^j_{dy}, j \in [1,m]\}, \{\beta^j_{dy}.P, j \in [1,m]\}\} \tag{3}$$

$$EC_j = S_k.value \quad oprel \quad cte | EC_j \tag{4}$$

$$k = 1..v \tag{5}$$

$$oprel = < | > | => | =< | = | != \tag{6}$$

The system architecture is shown in Fig. 1 where BCTs represent the dynamic reconfigurable tasks and SCTs the non-reconfigurable ones. The RBS offers the features of controllers reconfiguration and the RoI layer presents the features of intelligence reconfiguration. This later service proposed in previous work [18] is keeping a dynamic reconfigurable rule base. Indeed, a set of activation condition $AC = \{AC_1..AC_h\}$ are triggers for adapting ERB to new environment requirements based on context-awareness. We propose an agenda of dates $\Lambda = \{\lambda_1, .., \lambda_t\}$ and a set of locations described by $\Xi = \{\xi_1, .., \xi_l\}$ as contextual activation conditions to reconfigure ERB $\gamma(t)$ using GMB depicted in the formalization as γ.

$$AC_i \rightarrow \gamma(t) \quad Op \quad \{\gamma_j, \gamma_j \in \gamma\} \tag{7}$$

$$Op = + \mid - \tag{8}$$

$$AC_i = \lambda_i \mid \xi_i \tag{9}$$

The Op field in the equation above presents the add/delete operation into/from ERB and AC_i is a value from $\Lambda \cup \Xi$.

A task β_i is defined as a periodic real-time task with an arrival time R_i, a worst case execution time C_i, a deadline D_i and a period P_i. The scheduling feasibility with the EDF algorithm is done with the processor utilisation that should be lower than 1 [12].

$$U = (\sum_{q=1}^{n+m} \frac{C_q}{P_q}) < 1 \tag{10}$$

Based on the work reported in [15] the duration of an idle period before each arrival time R_q is calculated as follows:

$$\begin{cases} Idle_0 = 0 \\ Idle_q = Sup(0, R_q - \sum_{j=1}^{n+m} \lceil \frac{R_j}{P_j} \rceil C_j - \sum_{k=1}^{q-1} Idle_k) \\ q = 1, 2, .., p \\ p = N - n + m + 1 \end{cases} \tag{11}$$

Where R_q is the arrival time and P_q is the period of a task β^q_{st} or β^q_{dy} and N denotes the total number of distinct requests that occur within $[0, P[$ [15]. RBS can be implemented as a periodic task when the strategy of polling is used to read sensors measurements. Nevertheless, this strategy may run the RBS when unchanged sensors measurements for two successive polling periods also real-time environment awareness and decision making in critical systems may be failed because of standby period. For this, we propose to use the interrupt strategy where changing values on sensor's measurements is handled directly by the RBS with the field of defined values. This later strategy makes RBS a probabilistic task depending on sensor's interrupts and the inferring time. By applying the M/M/1 model [16] for probabilistic tasks, we affect to RBS two rates with the following assumptions; the execution time is following an exponential distribution $\frac{1}{\lambda c}$ and its arrival time is following a Poisson process with the rate λr [4]. The feasibility of the RBS inferring for M_i requests that occur within $[0, P[$ is guaranteed by the following equation :

$$U^{pro} = \sum_{k=1}^{M} \frac{\lambda r_k}{\lambda c_k} < 1 \tag{12}$$

We assume that initially the system is feasible and each request to allocate RBS execution in an idle period is accepted. The power consumption of Sys depends on the processor utilisation [17] with the following equations:

$$E = k * U^2 \tag{13}$$

A critical remaining battery energy is defined as E_{Cr}, the system should guarantee the power saving with $E < E_{Cr}$. The memory consumption is guaranteed by this assumption $R(t) < H$ where the size of the used memory, $R(t)$, should be lower than the real size of the physical memory denoted H.

A. Motivation and Contribution

The reconfiguration process is managing the rules inference and the tasks using respectively a set of contextual and functional conditions.

A heavy reconfiguration scenarios of adding a huge number of rules/tasks increase the power and memory consumption. Also, real-time constraints may be violated with the non-feasibility of equations 6 and 7. For this, we aim to adapt the system using parameter modification based on a new factor which is the intelligence impact of each task I_i. This new parameter is calculated dynamically to describe the environment awareness of the system. Indeed, tasks are reconfigured using a set of rules based on functional conditions with the field of sensor's measurements. The intelligence impact of a task T_i is the sum of the whole time computation between its activation duration and its deactivation one at run-time.

$$I_i(t) = (t + R_{i,j}^{Act}) * Run_i + \sum_{j=1}^{p_i} (AT_i^j - DT_i^j) \quad (14)$$

$$AT_i^j = R_{i,j}^{Deact} - R_{i,j}^{Act} \quad (15)$$

$$DT_i^j = R_{i,j-1}^{Deact} - R_{i,j}^{Act} \quad (16)$$

$$R_{i,0}^{Deact} = 0 \quad (17)$$

Where Run_i is a boolean function describing the status of a task T_i; if it is already running $Run_i = 1$ otherwise $Run_i = 0$. $R_{i,j}^{Act}$ is the date of the j^{th} activation for a dynamic task T_i and $R_{i,j}^{Deact}$ is the deactivation date one. We suppose that the system is initialized at a date $t_0 = 0$ and $R_{i,0}^{Deact} = t_0 = 0$ for each task T_i of $\beta_{dy} \cup \beta_{st}$. I_i is a dynamic parameter depending on the predecessor values of p_i previous activation.

This new parameter describes the importance of a task and its consistency on the process control over the environment changes. Previous works [4], [9] propose the modification of parameters for all tasks when real-time constraints are broken. This solution supposes that all tasks are flexible and may accept the fact of parameter modification. Nevertheless, in process control there is static tasks, i.e. non-reconfigurable, as some type of operating system tasks (e.g., the scheduler, the memory management..). Also, parameters modification should be controlled by user predefined maximum values.

IV. New Strategy for Real-Time Reconfiguration with low power and Memory Consumption

In this section, we propose a new strategy with parameter's modification for both real-time feasibility and low power and memory consumption.

A. New Strategy with Parameter Modification for Real-Time Feasibility

Each periodic task in β_{Dy} has P_{Max} parameter as a predefined maximum value for period modification. We propose to update tasks parameters one by one until the system became feasible.

In fact, the proposed solution changes the period of β_{Dy}^i having the less intelligence impact I_i with its maximum period P_{Max}^i. Then, we verify the system feasibility, if the system is feasible the modification process is accomplished else we continue the same strategy with β_{Dy}^j where I_j succeed I_i in the queue of intelligence impact. We repeat this strategy until the system became feasible or all the queue is queried. If the system steals non feasible we proceed in this case to delete tasks based on the intelligence impact where the lowest task in this queue is deleted. In Fig. 2, we present the algorithm of period modification. First, we calculate the intelligence impact of activated β_{Dy} tasks then we sort them in a decreasing queue in function of the calculated parameter. Then, we modify the task's periods and we investigate in each iteration the processor utilisation factor until the system became feasible or all the queue is queried.

```
Data: β,i=0,m
Result: U
begin
    foreach β^i_Dy ∈ β do
        Compute I_i;
    Sort(β_Dy, I);
    repeat
        i++;
        if β^i_Dy.P ≠ β^i_Dy.P_Max then
            β^i_Dy.P = β^i_Dy.P_Max;
            Compute U;
    until U ⩽ 1 || i=m;
    if U < 1 then
        Return(U);
    else
        Return(0);
```

Fig. 2: Algorithm of Period Modification.

TABLE I: Comparative Works.

Complexity	Works in [9]	Works in [4]	Proposed Solution
Worst Case	o(n^2)	o(n^2)	o(n^2)
Best Case	o(n^2)	o(n^2)	o(n+ n*log(n))

In table 1, our solution has the same complexity in the worst case with previous related works [9], [4], but in the best case our algorithm can close the modification process with a complexity of o(n). Indeed, our algorithm uses a sorting algorithm where its complexity is of o(n*logn) and the modification process including the computing of the novel processor's utilisation value can run for only one iteration which equal to o(n) and the best case complexity is equal to $o(n + n * log(n)) \simeq o(n)$. The related works propose to update the parameters of all tasks which results always to a complexity of o(n^2).

B. New Solution for Critical Battery Charge and Memory Overload

In a battery-powered system, critical battery status should be faced by a low-power consumption strategy. For this, we propose to update the parameter of tasks when $E \leqslant E_{Cr}$ to save the process control. Indeed, each task from the set β_{dy}

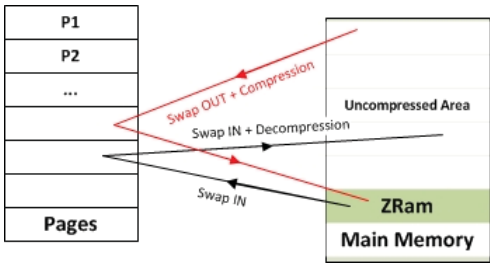

Fig. 3: Memory Swapping.

consumes energy E_{Pr} with the following formulas where our hypothesis in this work is that $k = 1$:

$$E^i_{Pr} = (\frac{C_i}{P_i})^2 \tag{18}$$

We propose to combine the power consumption and the intelligence impact parameters to sort the set of running tasks in a decreasing queue. Indeed, in our research thematic we assume that the system is sensible to intelligence QoS. For this, the reconfiguration of tasks should follow a dynamic priority in parameter modification where tasks having higher consumption regardless to their intelligence impact should be on the top of the reconfigured tasks.

$$E^{Intl}_{i,Pr} = \frac{E^i_{Pr}}{I_i(t)} \ \forall \ I_i(t) > 0 \tag{19}$$

$$E^{Intl}_{i,Pr} = E^i_{Pr} * |I_i(t)| \ \forall \ I_i(t) < 0 \tag{20}$$

$$E^{Intl}_{i,Pr} = E^i_{Pr} \ \forall \ I_i(t) = 0 \tag{21}$$

When $E_{Pr} \leqslant E_{Cr}$, we propose to sort the tasks using their impact $E^{Intl}_{i,Pr}$ in a decreasing queue. Then, we apply the same strategy in the algorithm of real-time parameter modification in Fig. 2. The system now has a definition for power consumption with intelligence QoS guarantee based on processor utilization. To face memory overload described by the formulas $R(t) < H$, a basic solution is to delete tasks based on some factor as works in [9]. Deleting tasks may affect process control where all the variable's values will be lost. For this, we aim to use paging techniques as a strategy for the memory management in RRTECS. The page replacement algorithm in our proposed solution is based on the intelligence impact where the pages of the task T_i having the less I_i will be swapped out.

Saving the process control is a major goal in RRTECS, for this we propose to manage the memory with a strategy of on-line memory compression, where a block of ram is dedicated to overload issues by a field of a compressed swap memory *zram* [22]. The hypothesis is the fact of managing the main memory where we suppose that the system is feasible after testing with $H_f < H$, a dedicated swap device is implemented inside the main memory having size of $H_{Swap} = H - H_f$ as shown in Fig. 3.

When $H(t) > H_f$, the swap memory is activated and contrariwise to the work in [19], we propose to use less intelligence impact of tasks to decide whether the pages to swap out. Beside using virtual memory support and compression

```
Data: β,PTE, I
1  begin
2      foreach β_i ∈ β do
3          foreach Page_j ∈ PTE do
4              Assign(Page_i^j, I_j);
5      Sort(PTE, I);
6      i=0;
7      if GetEventActive(Swap, Event) = 1 then
8          repeat
9              CompressAndSwapOut(Page_i);
10             i++;
11         until H(t) < H_f;
12     else
13         Run(LRU, PTE);
```

Fig. 4: Page Replacement Algorithm.

algorithm, the real-time constraints may be broken for this, the compression is activated only when overload is detected. The proposed strategy is based on two hypothesis; (i) First Case: memory overload caused by new tasks/rules activation, (ii) Second Case: memory overload handled at run-time in a current configuration. Dealing with the first case, the pages of the task having the less intelligence impact will be swapped out in a compressed format to the compression area *zram*. When the arrival time of a compressed task is reclaimed by the processor, the task having the less intelligence impact in the uncompressed area will be compressed and swapped out to *zram* and the arrived task is decompressed and swapped into the uncompressed area. Indeed, the page table is at runtime dynamically reconfigured to add/delete logical address for activated/deactivated tasks after each reconfiguration scenarios and the memory amount of the novel configuration is calculated to handle page faults before that occurs, if $H(t) > H_f$ the proposed strategy is executed. The second case where the system is feasible, nevertheless and without a reconfiguration scenarios, the memory leak is detected, in this case we propose to use LRU as mentioned in [19]. Indeed, the page fault may be caused for a size of one or two mapped pages, it seems be better to swap out only unused pages to satisfy reliability in the system. Nevertheless and as mentioned above, the cost of swapping and compression in time execution may violate the real-time constraints, for this we propose the solution presented above for real-time feasibility. The page replacement algorithm is depicted in Fig. 4 where swapping out the pages is defined in function of the event causing the memory overload if it is equal to 1 then it describes the first case as presented in the previous specification. Otherwise, the event is presenting the second case where the memory is overloaded at run-time in a current configuration.

V. Architecture and Implementation of the Proposed Solution

We propose a MAS for real-time feasibility and low power and memory consumption of the adaptive RRTECS. Three agents are proposed as follow :
(i) **RRT** : Reconfigurable Real-Time agent,

(ii) **RPM** : Reconfigurable Power Management agent,
(iii) **RMM** : Reconfigurable Memory Management agent.

RRT is supervising the scheduler feasibility using the equations 6, 7. When the real-time constraints are violated, RRT uses the proposed queue named Reconfigurable Queue for Real-Time feasibility (RQRT) where tasks are sorted using their intelligence impact. Based on RQRT, RRT applies the algorithm presented above in Fig. 2.

RPM is managing the battery life by the field of a trigger on its energy consumption. The same strategy with RRT is proposed in RPM by using a second queue, the Reconfigurable Queue for Battery Management (RQBM) to maximize the battery life as depicted in equations 19, 20 and 21. RMM supervises the memory consumption where it calculates the amount of memory and the free memory after each reconfiguration and it activates the swap block to real-time compression using the LZ4 algorithm [21] and based on RQRT. RMM also updates the page table to add and to map logical address for newly activated tasks/rules and to delete one for the deactivated services.

TABLE II: Agents Priorities.

Agents	Priorities
RRT	1
RPM	2
RMM	3
RIA	4
RCA	5

All those agents and their interactions are shown in Fig. 5. We propose a set of priorities for the proposed solution as depicted in table 2 where the new agents presented in this work are highly important from RIA and RCA presented in [18].

The RBS implementation is based on Clips.h [13] by defining a set of UserFunctions for building the BCTs.

Running Example 1. *EnvDefineFunction2(environment, "BCT-Implementation", 'v', PTIEF BCTImpl, "BCTImpl", "00");*

```
void BCTImpl( void *theEnv)
{
printf("BCT is running");
}
```

As mentioned in the example above, the EnvDefineFunction2 is used to implement the BCTs,

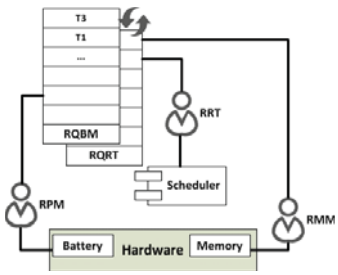

Fig. 5: Agent's Implementation.

then in the clips rules definition we can introduce a rule in the rule base by the following syntax :

(defrule LHS => (BCT-Implementation))

Where the LHS is a set of conditions to run this specific rule. The whole rule base is generated in a file ERB.clp and this file is introduced in the proposed middleware for controller reconfiguration. By examining the CLIPS documentation [13], [14], we find that it provides the possibility to define multiple environments in which a variety of RBS can be loaded and executed. Indeed, we use the following routines CreateEnvironment(), EnvLoad() and EnvRun() from clips.h to run the RBS as a middleware in the main system implementation.

VI. Experimentation and Case Study

In this section, we propose to evaluate our proposed contribution by a simulation of a case study presented as a set of tasks and rules describing the adaptive intelligence behaviour of the RRTECS. We suppose that the system is running with

TABLE III: Tasks Description.

Task	WCET	P	D	I	P_{Max}
T1	3	10	5	9	18
T2	2	8	3	10	10
T3	5	15	6	5	28
T4	3	8	7	14	10
T5	10	12	11	-2	30

TABLE IV: Rules of RCA for the RRTECS.

Rule ID	Condition	System Configuration
R1	$Temperature < 10$	T1,T2,T3
R2	$Temperature = 10$	T1,T2,T3,T4
R3	$Temperature > 10$	T1,T2,T3,T5

the configuration presented in rule R1 as mentioned in table 4 at a moment t_1. Using the parameters of tasks depicted in table 3, we investigate the real-time feasibility using equation 6, U=0,89. Now, the temperature is changed to the value of 10 then RCA runs to reconfigure the system by adding the task T4 by the field of R2. The system became non-feasible where $U = 1,25 > 1$. Using the intelligence impact, RRT runs to sort the tasks in the RQRT. After two iterations, RRT changes successively the period of T3 with its P_{Max} 28 and T1 with 18, the system became feasible where U=0,97. In the same way, RCA runs R3 and the system became non-feasible with U=1,43 and by changing the period of T5 with its P_{Max} = 30, U became equal to 0,93 and the system is feasible.

After a set of simulations, we compare our solution to the related works of [9] and [4] to enhance our community with the cost of our contribution in time computation and power consumption. As shown in Fig. 6, our solution with the blue line is less time computation than the works of Khemaissia [9] with the brown line and the work of Wang [4] with the red line. Indeed, our solution is in its worst case equal to the related works of [9], [4]. Also with our solution, the system consumes less power consumption by decreasing the cost of the reconfiguration process for critical battery charge

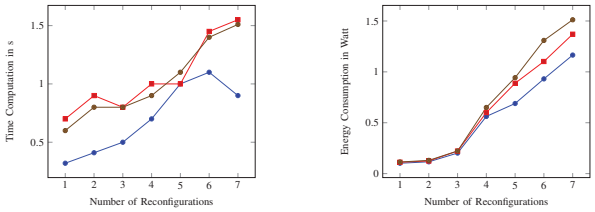

Fig. 6: Comparative Works in Time Computation and Energy Consumption.

using RPM agent and the proposed algorithm for parameters modification as depicted in Fig. 6.

VII. Conclusion

In this paper, we present new strategy for the real-time feasibility of an adaptive system with low power and memory consumption. A new parameter is presented as the intelligence impact of each task in an autonomous and adaptive real-time control system. Based on this factor, we propose the use of queues to manage the reconfiguration process by the field of parameter modification. Our contribution is less expensive in time and power consumption regardless to previous related works. In the future, we aim to adapt the proposed contribution in a real case study and to expand it to a distributed platform.

References

[1] M. Weiser, "The computer for the 21st century", ACM SIGMOBILE Mob. Comput. Commun. Rev., Vol. 3, Issue 3, pp. 3-11, July 1999.

[2] R. Davis and J. J. King, "The Origisn of Rule-Based Systems in AI", reprinted as Ch. 2 of Rule-Based Expert Systems: Edited by Bruce G. Buchanan and Edward H. Shortliffe, 1984.

[3] S.J. Russel and P. Norvig, "Artificial Intelligence: A Modern Approach", Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2, chpt. 2, 2003.

[4] X. Wang, I. Khemaissia, M. Khalgui, ZW. Li, O. Mosbahi, MC. Zhou, "Dynamic Low-Power Reconfiguration of Real-Time Systems With Periodic and Probabilistic Tasks", IEEE Trans. Aut. Sci. and Eng., Vol. 12, Issue 1, pp. 258-271, 2015.

[5] M. Pohronská, "Implementing Embedded Expert Systems via Programmable Hardware", Inf. Sci. and Tech. Bulletin of the ACM Slovakia, Vol. 4, Issue 2, pp. 10-19, 2012.

[6] G. J. Nalepa, P. Ziecik, "Integrated Embedded Prolog Platform for Rule-Based Control Systems", In Proceedings of Inter. Conf. on Mixed Design of Integrated Circuits and Systems (MIXDES), pp. 716-721, Poland, 2006.

[7] B.J. Koo, Y.S. Park, S.H. Yang, "Microcontroller Implementation of Rule-based Inference System for Smart Home", Inter. Journal of Smart Home, Vol. 8, Issue 6, pp. 197-204, 2014.

[8] J. Davis, J. Hoffert, and E. Vanlandingham, "A Taxonomy of Artificial Intelligence Approaches for Adaptive Distributed Real-time Embedded Systems", In Proceedings of IEEE Inter. Conf. on Elec. Inf. Tech. EIT'2016, pp. 233-238, USA, May 2016.

[9] I. Khemaissia, O. Mosbahi, M. Khalgui, W. Bouzayen, "New Reconfigurable Middleware for Feasible Adaptive RT-Linux", In Proceedings of Inter. Pervasive and Embedded Computing and Communication Systems (PECCS), pp. 158-167, Portugal, 2014.

[10] G. Estrin, "Reconfigurable computer origins: the UCLA fixed-plus-variable (F+V) structure computer". IEEE Annals of the Hist. of Comp., Vol. 24 , Issue 4, pp. 39, 2002.

[11] M. Khalgui and O. Mosbahi, "Intelligent distributed control systems", Inf. and Softw. Tech. Journal, Vol. 52, issue 12, pp. 1259-1271, 2010.

[12] C. L. Liu and James W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment", Journal of the ACM, Vol. 20, Issue 1, pp. 4661, January 1973.

[13] CLIPS Reference Manual, Vol 2., "Advanced Programming Guide", Version 6.30, Preface pp. 16, pp.203-209, March 17th 2015.

[14] CLIPS Reference Manual, Vol 1., "Basic Programming Guide", Version 6.30, March 17th 2015.

[15] H. Chetto and M. Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm", IEEE Trans. on Softw. Eng., Vol. 15, Issue 10, pp. 1261-1269, 1989.

[16] U. N. Bhat, "An Introduction to Queueing Theory". Cambridge, MA, USA: Birkhuser, 2008.

[17] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-timesystems", In the 36th Proceedings of Design Automation Conference, pp. 134-139, 1999.

[18] W. Bouzayen, H. Gharsellaoui, M. Khalgui, "New Solutions for the Intelligence Adaptation of Reconfigurable Embedded Systems" In Proceedings of the 30th European Simulation and Modeling Conference (ESM), Spain, pp. 233-239, October 2016.

[19] L. Yang, R. P. Dick, H. Lekatsas, "Online memory compression for embedded systems", ACM Trans. Embed. Comput. Syst., Vol. 9, No. 3, pp. 27:1-27:30, 2010.

[20] L.A. Belady, "A study of replacement algorithms for a virtual-storage computer", IBM Systems Journal, Vol. 5, N.2, pp. 78-101, 1966.

[21] Y. Collet, "LZ4 Block Format Description", [OnLine] available at : "http://lz4.github.io/lz4/lz4_Block_format.html", 2015.

[22] N. Gupta, "zram: Compressed RAM based block devices", [OnLine] available at : "https://www.kernel.org/doc/Documentation/blockdev/zram.txt", 2016.