Feasible Automatic Reconfigurations of Real-Time Tasks

Hamza Gharsellaoui[1], Mohamed Khalgui[2,3], Samir Ben Ahmed[1]
[1]National Institute of Applied Sciences and Technologies, INSAT, Tunisia
[2]Martin Luther University, Germany
[3]Xidjian University, China

Abstract

The paper deals with Reconfigurable Uniprocessor Real-Time Systems to be classically implemented by different tasks that we suppose independent, synchronous and periodic. Two forms of automatic reconfigurations are assumed to be applied at run-time: Addition-Remove of tasks or just modifications of their temporal parameters: WCET and/or Periods. We define an Intelligent Agent that automatically checks the system's feasibility after any reconfiguration scenario to verify if all tasks meet the required deadlines. It provides otherwise for users new temporal parameters of tasks when deadlines are violated. The tool $RT - Reconfiguration$ is developed to support the useful agent's services.

1 Introduction

Nowadays in Industry, new generations of embedded control systems are addressing new criteria as flexibility and agility. To reduce their cost, these systems should be changed and adapted to their environment without disturbances. Several interesting academic and industrial research works have been made last years to develop reconfigurable systems [1]. We distinguish in these works two reconfiguration policies: static and dynamic reconfigurations where static reconfigurations are applied off-line to apply changes before the system cold start [2], whereas dynamic reconfigurations are applied dynamically at runtime. Two cases exist in the last policy: manual reconfigurations applied by users [3] and automatic reconfigurations applied by Intelligent Agents [4, 5]. We are interested in this paper in automatic reconfigurations of real-time embedded control systems that should meet deadlines defined in user requirements [6]. These systems are implemented by sets of tasks that we assume independent, periodic and synchronous (e.g. they are simultaneously activated at t = 0 time units). We assume also that the deadline of each task is equal to the corresponding period. According to [7], we characterize each task by a period to be denoted by T equal to the deadline denoted by D and by a Worst Case Execution Time denoted by C. We define an automatic reconfiguration as any operation allowing additions-removes or updates of tasks at run-time. Therefore the system's implementation is dynamically changed and should meet all considered deadlines of the current combination of tasks. Nevertheless, when a reconfiguration is applied, the deadlines of new and old tasks can be violated. We define an agent-based architecture that checks the system's evolution and defines useful solutions for users when deadlines are not satisfied after each reconfiguration scenario. Two cases of suggestions are possible to be provided by the agent: modification of periods (equal to deadlines) or modification of worst case execution times of tasks. The users should choose one of these solutions to re-obtain the system's feasibility. A tool $RT - Reconfiguration$ is developed and tested in our laboratory to support the agent's services. We present related works in Section 2 and some real-time concepts in Section 3. The agent-based architecture is proposed in Section 4 before presented analysis in Section 5. We conclude the paper with conclusions and presentation of future works in Section 6.

2 Related Works

We analyze previous related works dealing with static, manual and automatic reconfigurations of systems, before we present thereafter the real-time scheduling.

2.1 Reconfiguration of Systems

Nowadays, rich research works have been proposed to develop reconfigurable systems. The authors propose in [2] reusable tasks to implement a broad range of systems where each task is statically reconfigured without any reprogramming. This is accomplished by updating the supporting data structure, i.e. a state transition table, whereas the executable code remains unchanged and may be stored in permanent memory. The state transition table consists of multiple-output binary decision diagrams that represent the next-state mappings of various states and the associated control actions. The authors propose in [3] a complete methodology based on the human intervention to dynamically reconfigure tasks. They present in addition an interesting experimentation showing the dynamic change by users of tasks without disturbing the whole system. The authors use in [8] Real-time-UML as a meta-model between design models of tasks and their implementation models to support dynamic user-based reconfigurations of control systems. The authors propose in [9] an agent-based reconfiguration approach to save the whole system when faults occur at run-time. Finally the authors propose in [5] an ontology-based agent to perform system's reconfigurations that adapt changes in requirements and also in environment. They are interested to study reconfigurations of control systems when hardware faults occur at run-time. We are interested in this paper in feasible automatic reconfigurations of real-time systems where additions and removes of real-time tasks are applied at run-time. We define an agent-based architecture to propose new parameters for tasks when deadlines are violated in order to re-obtain the system's feasibility.

2.2 Real-Time Scheduling

Real-time scheduling has been extensively studied in the last thirty years [6]. Several Feasibility Conditions (FC) for the dimensioning of a real-time system are defined to enable a designer to grant that timeliness constraints associated to an application are always met for all possible configurations. Different classes of scheduling algorithm are followed nowadays: (i) Clock-driven: primarily used for hard real-time systems where all properties of all jobs are known at design time. (ii) Weighted round-robin: primarily used for scheduling a real-time traffic in high-speed, (iii) Priority-driven: primarily used for more dynamic real-time systems with a mixture of time-based and event-based activities. Among all priority-driven policies, Earliest Deadline First (EDF) or Least Time to Go is a dynamic scheduling algorithm used in real-time operating systems. It places processes in a priority queue. Whenever a scheduling event occurs (task finishes, new task released, etc.) the queue will be searched for the process closest to its deadline. This process is the next to be scheduled for execution. EDF is an optimal scheduling algorithm on preemptive uniprocessors in the following sense: if a collection of independent periodic jobs characterized by arrival times equal to zero and by deadlines equal to corresponding periods, can be scheduled by a particular algorithm such that all deadlines are satisfied, then EDF is able to schedule this collection of jobs.

3 Known Concepts in The EDF Theory

We present the following well-known concepts in the theory of real-time scheduling [7]:

- A periodic task i $(C_i; T_i; D_i)$ is an infinite collection of jobs that have their request times constrained by a regular inter-arrival time T_i, a Worst Case Execution Time (WCET) C_i and a relative deadline D_i,
- A real-time scheduling problem is said feasible if there is at least one scheduling policy able to meet the deadlines of all the considered tasks,
- A set of tasks is schedulable with a given scheduling policy if and only if no jobs of this set miss their deadlines,
- A task is valid with a given scheduling policy if and only if no jobs of this task miss their deadlines,
- An idle time t of a processor is defined as a time where no tasks released before time t are pending at time t. An interval of successive idle times is classically called an idle period,
- A busy period is defined as a time interval $[a, b)$ such that there is no idle time in $[a, b)$ (the processor is fully busy) and such that both a and b are idle times,
- In the case of independent, periodic and synchronous tasks (e.g. simultaneously activated at $t = 0$), the verification of the system's schedulability is possible to be done in a hyper period $[0, lcm]$ where lcm is the Least Common Multiple [7],
- $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$ is the processor utilization factor. In the case of synchronous, independent and periodic tasks such that their deadlines are equal to their periods, $U \leq 1$ is a necessary and sufficient condition for the EDF-based scheduling of real time tasks.

4 Agent-based architecture for Reconfigurable Embedded Control Systems

We define in this section an agent-based architecture for reconfigurable real-time embedded systems that should classically meet different deadlines defined in user requirements. The agent controls the system's evolution and provides solutions for users when deadlines are violated after any reconfiguration scenario. Let Sys be such system to be classically composed of a set of real-time tasks that support different functionalities. We mean in this paper by an automatic reconfiguration any operation that adds, removes or also updates tasks at run-time. Automatic updates of tasks mean modifications of their temporal parameters e.g. Periods and Deadlines, or modifications of their Worst Case Execution Times. Let S_{Sys} be the set of all tasks that can possibly implement the system, and let us denote by $Current_{Sys}(t)$ the set of tasks implementing the system Sys at t time units. These tasks should meet all deadlines defined in user requirements. In this case, we note that $Feasibility(Current_{Sys}(t)) \equiv True$.

Running Example1: Let us suppose a first real-time embedded system $Sys1$ to be initially implemented by 5 characterized tasks in Figure 1. These tasks are feasible because the processor utilization factor $U = 0.94 < 1$. We suppose that a reconfiguration scenario is applied at $t1$ time units to add five new tasks $6, 7, 8, 9, 10$. The new processor utilization becomes $U = 1.85 > 1$ time units. Therefore the system is unfeasible $Feasibility(Current_{Sys1}(t1)) \equiv False$.

Sys	C	T	D
1	2	7	7
2	1	8	8
3	2	9	9
4	3	13	13
5	1	12	12
6	4	15	15
7	2	13	13
8	1	10	10
9	3	14	14
10	2	11	11

Figure 1. Implementation of the $Sys1$ by Real-Time Tasks

Sys	C	T	D
1	35	150	150
2	25	130	130
3	30	140	140
4	22	125	125
5	32	180	180
6	40	100	100
7	25	90	90
8	30	120	120
9	36	180	180
10	28	165	165

Figure 2. Implementation of the $Sys2$ by Real-Time Tasks

Running Example2: Let us suppose a second real-time embedded system $Sys2$ to be initially implemented by 5 characterized tasks in Figure 2. These tasks are feasible because the processor utilization factor $U = 0.99 < 1$. We suppose that a reconfiguration scenario is applied at $t2$ time units to remove the tasks $3, 4, 5$ and to add five new tasks $6, 7, 8, 9, 10$. The new processor utilization becomes $U = 1.72 > 1$ time units. Therefore the system is unfeasible $Feasibility(Current_{Sys2}(t2)) \equiv False$.

4.1 Agent's Principal

Under different conditions, the embedded system Sys is reconfigurable at run-time by changing its implementation to delete old or to add new tasks. The proposed agent should check after any reconfiguration scenario the system's feasibility that can be affected when tasks violate deadlines. In this case, the agent should propose useful solutions for users to re-obtain the system's feasibility. Two cases of solutions exist:

- the agent proposes a new period (e.g. equal to the corresponding deadline) for particular new and old tasks that compose $Current_{Sys}(t)$,
- or it proposes a new Worst Case Execution Time (e.g. WCET) for the same tasks.

4.2 Formalization

By considering synchronous real-time tasks, the schedulability analysis should be done in the Hyper-Period hp = [0,lcm], where lcm is the well-known Least Common Multiple [6]. Let n be the number of tasks in $Current_{Sys}(t)$. The reonfiguration of the system Sys means the modification of its implementation that will be as follows at t time units:

$$Current_{Sys}(t) = \xi_{new} \cup \xi_{old}$$

Where ξ_{old} is a subset of old tasks which are not affected by the reconfiguration scenario (e.g. they implement the system before the time t), and ξ_{new} a subset of new tasks in the system. We assume that an updated task is considered as a new one at t time units. By considering a feasible system Sys before the application of the reconfiguration scenario, each one of the tasks of ξ_{old} is feasible, e.g. the execution of each instance is finished before the corresponding deadline:

$$\forall i \in \xi_{old}, \forall j \in IN \text{ (e.g. instance } j \text{ of task } i\text{) such that } j * T_i \leq t$$

$$E_{i,j} \leq D_{i,j}$$

Where $E_{i,j}$ is the time when the execution of the instance j is finished. When the reconfiguration scenario is applied at time t, two cases exist:

- If tasks of $Current_{Sys}(t) = \xi_{new} \cup \xi_{old}$ are feasible, then no reaction should be done by the agent,
- Otherwise, the agent should provide different solutions for users in order to re-obtain the system's feasibility. We define the following steps allowing such services:
  - **First Step:** the agent tries to modify the periods (equal to deadlines) and execution times of tasks belonging to ξ_{new} in order to meet all deadlines that correspond to tasks of $Current_{Sys}(t)$,
  - **Iterative Second Step:** the agent tries to consider old tasks of ξ_{old} as new tasks to be introduced in ξ_{new}. A computation of periods, deadlines and WCET of these tasks with the new tasks is applied.

Let n_1 and n_2 be the number of tasks respectively in ξ_{old} and ξ_{new}. By assuming an unfeasible system at t time units, the following formula is satisfied:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} > 1$$

We propose to add the tasks of ξ_{old} to a linked list L_{old} that we sort in the decreased order of the processor times. Let j be the first j tasks of L_{old}.

- **Approach. For each** $j \in [0, n_1]$,

  Add the first j tasks of L_{old} to ξ_{new}. Two different solutions exist for the feasibility of the system:

  Comments. If $j = 0$, Then no tasks to be add to ξ_{new}.

  - **Solution 1: Modification of Periods**

  The agent proceeds as a first solution to change the periods of tasks of ξ_{new}. To obtain a feasible system, the following formula should be satisfied;

$$\sum_{i=1}^{n_1-j} \frac{Ci}{Ti} + \sum_{i=n_1-j+1}^{n_2+j} \frac{C_i}{T_i+\theta_i} = 1$$

$\longrightarrow \sum_{i=n_1-j+1}^{n_2+j}, \frac{C_i}{T_i+\theta_i} = 1 - \sum_{i=1}^{n_1-j}, \frac{C_i}{T_i}$

Let the constant β_j be $T_i + \theta_i$,

$\longrightarrow \frac{1}{\beta_j}\sum_{i=n_1-j+1}^{n_2+j} C_i = 1 - \sum_{i=1}^{n_1-j} \frac{C_i}{T_i}$

$\longrightarrow \beta_j = \lceil \frac{\sum_{i=n_1-j+1}^{n_2+j}, C_i}{1-\sum_{i=1}^{n_1-j} \frac{C_i}{T_i}} \rceil = constante$

The new period of tasks of ξ_{new} is therefore deduced from β_j.

  - **Solution 2: Modification of Worst Case Execution Times**

  The agent proceeds now as a second solution to modify the Worst case Execution Times of tasks of ξ_{new}. To obtain a feasible system, the following formula should be satisfied:

$$\sum_{i=1}^{n_1-j} \frac{C_i}{T_i} + \sum_{i=n_1-j+1}^{n_2+j} \frac{C_i+\alpha_i}{T_i} = 1$$

$\longrightarrow \sum_{i=n_1-j+1}^{n_2+j} \frac{C_i+\alpha_i}{Ti} = 1 - \sum_{i=1}^{n_1-j} \frac{C_i}{T_i}$

$\longrightarrow \sum_{i=n_1-j+1}^{n_2+j} \frac{\alpha_i}{Ti} = 1 - \sum_{i=1}^{n_1-j} \frac{C_i}{T_i} - \sum_{i=n_1-j+1}^{n_2+j} \frac{C_i}{T_i}$

$\longrightarrow \sum_{i=n_1-j+1}^{n_2+j} \frac{\alpha_i}{T_i} = 1 - \sum_{i=1}^{n_2+j} \frac{C_i}{T_i}$

Let γ_j be the following constant: $\gamma_j = \alpha_i = Constant$,

$\longrightarrow \gamma_j = \lceil \frac{1-\sum_{i=1}^{n_2+j} \frac{C_i}{T_i}}{\sum_{i=n_1-j+1}^{n_2+j} \frac{1}{T_i}} \rceil = constante$

The new WCET of tasks of ξ_{new} is therefore deduced from γ_j.

4.3 Algorithm

We present the algorithm of the agent to help users and to allow correct parametrizations of feasible tasks.

```
Algorithm Reconfigure($Task\_List$ $L_{old}$, $New\_Tasks$ $\xi_{new}$, $int$ $n_1$)
  • Check_Feasibility($L_{old}$, $\xi_{new}$);
  • For $j$ from 0 to $n_1$ step by step,
      – $\beta_j \leftarrow$ Compute_Beta($L_{old}$, $\xi_{new}$, $j$);
      – $\gamma_j \leftarrow$ Compute_Gamma($L_{old}$, $\xi_{new}$, $j$);
      – Display_Parameters($L_{old}$, $\xi_{new}$, $\beta_j$, $\gamma_j$);
End of Algorithm.
```

We note that the complexity of the proposed algorithm is $O(n^2)$. We implemented a prototype $RT - Reconfiguation$ to give helps and indications for users when automatic reconfigurations of real-time embedded systems are applied at run-time. The simulator Cheddar [10] is used for the schedulability analysis of the considered tasks.

Running Example1. In the first example of $Sys1$, the agent should react to propose useful solutions for readers in order to re-obtain the system's feasibility. We note that:

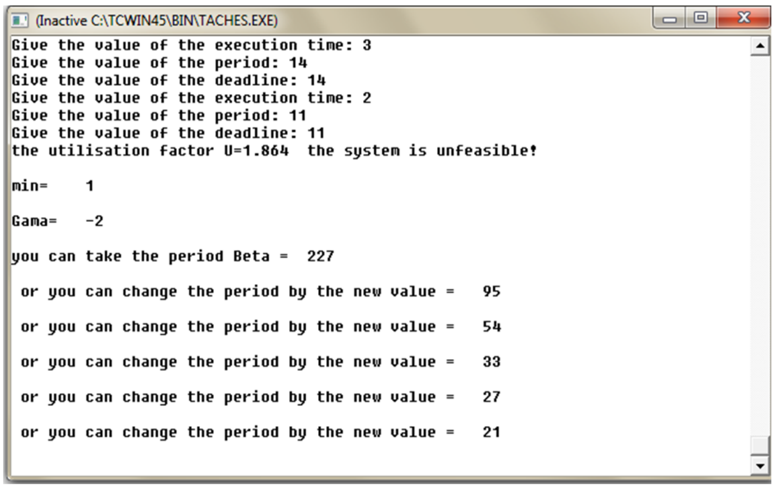
$$\xi_{new} = \{6, 7, 8, 9, 10\}$$

$$L_{old} = \{1, 4, 3, 2, 5\}$$

The agent computes the constant values β_j and γ_j ($j \in [0, 5]$) as follows:

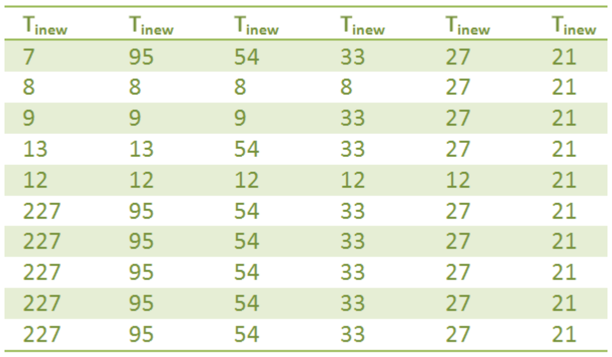
- β_0 = 227 time units where $L_{old} = \{1, 4, 3, 2, 5\}$ and $\xi_{new} = \{6, 7, 8, 9, 10\}$,
- β_1 = 95 time units where $L_{old} = \{4, 3, 2, 5\}$ and $\xi_{new} = \{1, 6, 7, 8, 9, 10\}$,
- β_2 = 54 time units where $L_{old} = \{3, 2, 5\}$ and $\xi_{new} = \{4, 1, 6, 7, 8, 9, 10\}$,
- β_3 = 33 time units where $L_{old} = \{2, 5\}$ and $\xi_{new} = \{3, 4, 1, 6, 7, 8, 9, 10\}$,
- β_4 = 27 time units where $L_{old} = \{5\}$ and $\xi_{new} = \{2, 3, 4, 1, 6, 7, 8, 9, 10\}$,
- β_5 = 21 time units where $L_{old} = \emptyset$ and $\xi_{new} = \{5, 2, 3, 4, 1, 6, 7, 8, 9, 10\}$,
- $\gamma = -2$ higher that the minimal value of WCET of tasks belonging to ξ_{new}. Therefore, the agent deduces that modifications of execution times cannot solve the problem.

We present in Figure 5 the new temporal parameters of the considered tasks. The user can choose one of these solutions to have a feasible system. We present in Figure 4 the results given by the developed tool $RT - Reconfiguration$ that provides new solutions by using



```
(Inactive C:\TCWIN45\BIN\TACHES.EXE)
Give the value of the execution time: 3
Give the value of the period: 14
Give the value of the deadline: 14
Give the value of the execution time: 2
Give the value of the period: 11
Give the value of the deadline: 11
the utilisation factor U=1.864  the system is unfeasible!

min=    1

Gama=   -2

you can take the period Beta =  227

 or you can change the period by the new value =   95

 or you can change the period by the new value =   54

 or you can change the period by the new value =   33

 or you can change the period by the new value =   27

 or you can change the period by the new value =   21
```

Figure 4. Presentation of results described by the developed tool $RT - Configuration$



T_{inew}	T_{inew}	T_{inew}	T_{inew}	T_{inew}	T_{inew}
7	95	54	33	27	21
8	8	8	8	27	21
9	9	9	33	27	21
13	13	54	33	27	21
12	12	12	12	12	21
227	95	54	33	27	21
227	95	54	33	27	21
227	95	54	33	27	21
227	95	54	33	27	21
227	95	54	33	27	21

Figure 3. New temporal configuration of $Sys1$

the simulator Cheddar in order to re-obtain a feasible system $Sys1$.

Running Example2. In the second example of $Sys2$ where a reconfiguration scenario is applied to delete the tasks $3, 4, 5$, the agent should react to propose useful solutions for readers to re-obtain the system's feasibility. We note that:

$$\xi_{new} = \{6, 7, 8, 9, 10\}$$

$$L_{old} = \{2, 1\}$$

The agent computes the constant values β_j and γ_j ($j \in [0, 2]$) as follows:

- β_0 = 277 time units where $L_{old} = \{1, 2\}$ and $\xi_{new} = \{6, 7, 8, 9, 10\}$,
- β_1 = 240 time units where $L_{old} = \{1\}$ and $\xi_{new} = \{2, 6, 7, 8, 9, 10\}$,
- β_2 = 219 time units where $L_{old} = \emptyset$ and $\xi_{new} = \{1, 2, 6, 7, 8, 9, 10\}$,
- $WCET = 7$ for each task of $\xi_{new} = \{6, 7, 8, 9, 10\}$,
- $WCET = 6$ for each task of $\xi_{new} = \{2, 6, 7, 8, 9, 10\}$,
- $WCET = 5$ for each task of $\xi_{new} = \{1, 2, 6, 7, 8, 9, 10\}$ (Figure 6),

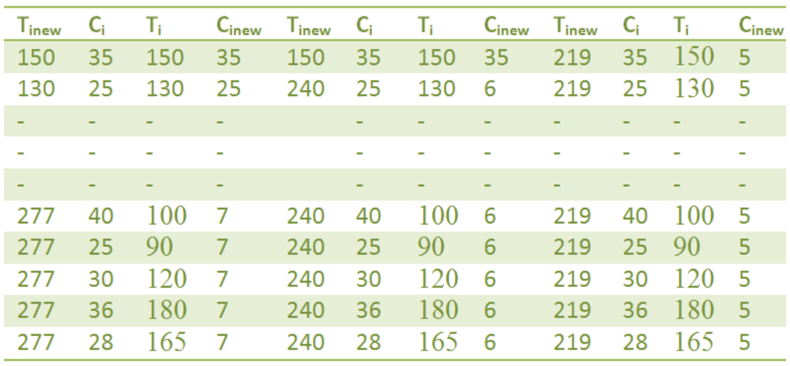
We present in Figure 5 the new temporal parameters of the considered tasks. The user can choose one of these solutions to have a feasible system.

5 Analysis of Results

We evaluate the proposed approach by considering the systems $Sys1$ and $Sys2$ defined in the running example. By applying the first solution, we find that the period of tasks of ξ_{new} is decreased each time an old task is removed from L_{old} (Figure 7). This reduction can allow more reactive and also feasible systems. This advantage can be important in many cases where critical control tasks should be intensively executed in small periods of time. We want also to compare the proposed solutions for the same system $Sys2$. By applying the first solution, the utilization factor U is equal to 1 in the three different scenarios generated by the tool $RT - Reconfiguration$, whereas it takes lower values when we apply the second solution dealing with modifications of WCET. Let $Area$ be the area of the rectangle delimited by the first curve that corresponds to the first solution, and let $WCET(t)$ be the simulation curve corresponding to modifications of execution times for feasible reconfigurations of real-time systems. The performance of Solution 2 is as follows:

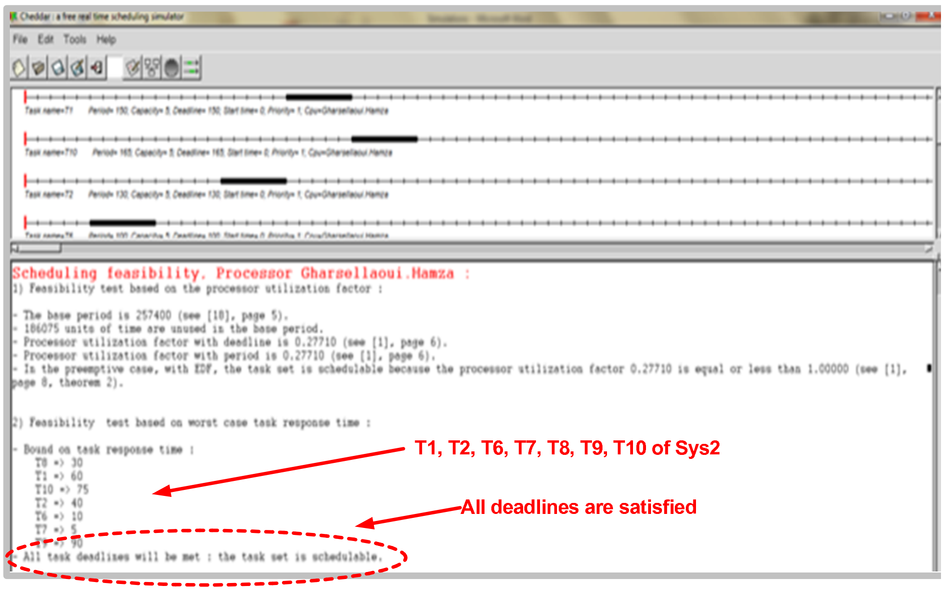
$$performance_{Solution2} = (\int_{Scenario1,Scenario2,Scenario3} WCET(t).dt)/Area = 0.6139$$

Therefore, the second solution is 39% more efficient than the first that deals with modifications of periods.



T_{inew}	C_i	T_i	C_{inew}	T_{inew}	C_i	T_i	C_{inew}	T_{inew}	C_i	T_i	C_{inew}
150	35	150	35	150	35	150	35	219	35	150	5
130	25	130	25	240	25	130	6	219	25	130	5
-	-	-	-		-	-	-	-	-	-	-
-	-	-	-		-	-	-	-	-	-	-
-	-	-	-		-	-	-	-	-	-	-
277	40	100	7	240	40	100	6	219	40	100	5
277	25	90	7	240	25	90	6	219	25	90	5
277	30	120	7	240	30	120	6	219	30	120	5
277	36	180	7	240	36	180	6	219	36	180	5
277	28	165	7	240	28	165	6	219	28	165	5

Figure 5. New temporal configuration of $Sys2$


Cheddar : a free real time scheduling simulator
File Edit Tools Help
T1, T2, T6, T7, T8, T9, T10 of Sys2
All deadlines are satisfied

```
Task name=T1    Period= 150; Capacity= 5; Deadline= 150; Start time= 0; Priority= 1; Cpu=Gharsellaoui.Hamza
Task name=T10   Period= 165; Capacity= 5; Deadline= 165; Start time= 0; Priority= 1; Cpu=Gharsellaoui.Hamza
Task name=T2    Period= 130; Capacity= 5; Deadline= 130; Start time= 0; Priority= 1; Cpu=Gharsellaoui.Hamza
Task name=T8    Period= 100; Capacity= 5; Deadline= 100; Start time= 0; Priority= 1; Cpu=Gharsellaoui.Hamza

Scheduling feasibility, Processor Gharsellaoui.Hamza :
1) Feasibility test based on the processor utilization factor :

- The base period is 257400 (see [18], page 5).
- 186075 units of time are unused in the base period.
- Processor utilization factor with deadline is 0.27710 (see [1], page 6).
- Processor utilization factor with period is 0.27710 (see [1], page 6).
- In the preemptive case, with EDF, the task set is schedulable because the processor utilization factor 0.27710 is equal or less than 1.00000 (see [1],
page 8, theorem 2).

2) Feasibility  test based on worst case task response time :

- Bound on task response time :
   T8 => 30
   T1 => 60
   T10 => 75
   T2 => 40
   T6 => 10
   T7 => 5
   T9 => 90
- All task deadlines will be met : the task set is schedulable.
```

Figure 6. A possible implementation of $Sys2$

6 Conclusion

The paper deals with reconfigurable systems to be implemented by different tasks that should meet real-time constraints. We assume in this work independent, periodic and synchronous tasks. We define a reconfiguration scenario as a dynamic operation allowing additions, removes or updates of tasks at run-time. When a reconfiguration scenario is automatically applied, the system can become unfeasible because some tasks violate corresponding deadlines. We define an agent-based architecture where an agent is proposed to provide new parameters of unfeasible tasks in order to re-obtain the system's feasibility. We plan in future works to study reconfigurations of asynchronous tasks to be released in different times, and we plan also to study reconfigurations of distributed real-time tasks.

References

[1] A.-L. Gehin and M. Staroswiecki, "Reconfiguration Analysis Using Generic Component Models", in *IEEE Transactions on Systems, Machine and Cybernetics, Vol.38, N.3*, 2008.

[2] C. Angelov, K. Sierszecki, and N. Marian, "Design models for reusable and reconfigurable state machines", in *L.T. Yang and All (Eds): EUC 2005, LNCS 3824, pp:152-163. International Federation for Information Processing.*, 2005.

[3] M. N. Rooker, C. Sunder, T. Strasser, A. Zoitl, O. Hummer, and G. Ebenhofer, "Zero Downtime Reconfiguration of Distributed Automation Systems : The εCEDAC Approach", in *Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, 2007. Springer-Verlag.

[4] M. Khalgui, "NCES-based modelling and CTL-based verification of reconfigurable embedded control systems", in *Computers in Industry. 61(3)*, 2010.

[5] Y. Al-Safi and V. Vyatkin, "An ontology-based reconfiguration agent for intelligent mechatronic systems", in *Third*

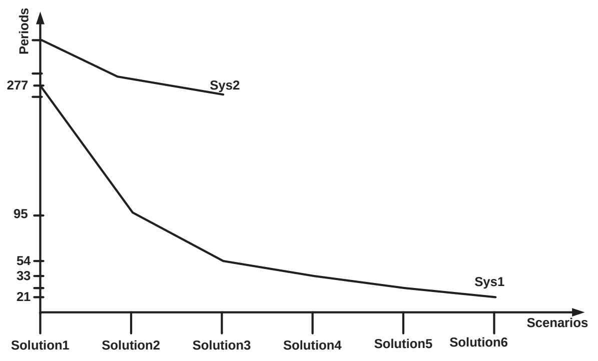


Figure 7. Decrease of new periods in $Sys1$ and $Sys2$ by applying the proposed algorithm

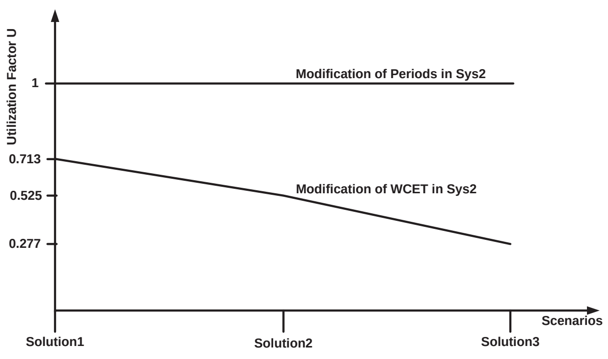


Figure 8. Comparison between the proposed Solution1 and Solution2 for feasible reconfiguration of $Sys2$

International Conference on Industrial Applications of Holonic and Multi-Agent Systems, 2007. Springer-Verlag.

[6] S. Baruah and J. Goossens, "Scheduling Real-time Tasks: Algorithms and Complexity", in *In Handbook of Scheduling: Algorithms, Models, and Performance Analysis, Joseph Y-T Leung (ed). Chapman Hall/ CRC Press*, 2004.

[7] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment", in *Journal of the ACM, 20(1):46-61*, 1973.

[8] K. Thramboulidis, G. Doukas, and A. Frantzis, "Towards an implementation model for FB-based reconfigurable distributed control applications,", in *Proceedings of 7th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 193-200*, 2004.

[9] R. W. Brennan, M. Fletcher, and D. H. Norrie, "A holonic approach to reconfiguring real-time distributed control systems", in *Book: Multi-Agent Systems and Applications: MASA'01*, 2001. Springer-Verlag.

[10] L. N. L. M. F. Singhoff, J. Legrand, "Cheddar : a Flexible Real Time Scheduling Framework", in *ACM SIGAda Ada Letters, volume 24, number 4, pages 1-8. Edited by ACM Press, ISSN:1094-3641*, 2004.