Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 00 (2013) 000–000

Procedia Technology

www.elsevier.com/locate/procedia

The 7th International Conference Interdisciplinarity in Engineering (INTER-ENG 2013)

Scheduling optimizations for real time embedded systems

Lajos Losonczi[a, b]*, László F. Márton[b], Tihamér S. Brassai[b]

[a] *Lambda Communications Ltd, str.Avram Iancu, nr.37, Tîrgu Mureș 540089, Romania*
[b] *Sapientia - University of Transylvania, Șos. Sighișoarei 1.C, Tîrgu Mureș, O.P.9 CP4, Romania, NSRG - Neural Systems Research Group*

Abstract

In this paper assessments are made on planning embedded inhomogeneous applications, composed of both time controlled processes and event controlled processes. Particularities of the optimization of inhomogeneous process scheduling are presented based on original optimization strategies and algorithms. The planning optimization and the strategies optimization flowchart of the distributed application is presented, then the flowchart of the optimized planning algorithm of multi-cluster distributed systems is described. Also the optimization of communication is discussed. A multi-cluster distributed application model is considered, and the most important stages of planning optimization are followed.

Selection and peer-review under responsibility of Department of Industrial Engineering and Management, Department of Electrical and Computer Engineering, Faculty of Engineering, "Petru Maior" University of Tîrgu Mureș.

Keywords: Embedded System, Scheduling Techniques, Distributed Control, Task Management;

1. Introduction

Distributed electronic systems are the rule rather than the exception nowadays. The more widespread they are, the more difficult is their design and analysis. It is very difficult to specify what the aim of a distributed system is and it is even more difficult to verify if a distributed application complies with the respective specifications. For

* Corresponding author. Tel.: +40-722-352178; fax: +40-265-211361.
E-mail address: lajos@lambda.ro, lajos.losonczi@ms.sapientia.ro

Selection and peer-review under responsibility of [Organizer Name].

each embedded, distributed application the optimal strategy design must be chosen, depending on the particular process that must be scheduled.

A scheduling mechanism must coordinate the execution of a set of processes, with certain temporal and resource constraints, as can be seen in Fig. 1. If the system is multiprocessor the scheduling has a local and a global aspect as well.

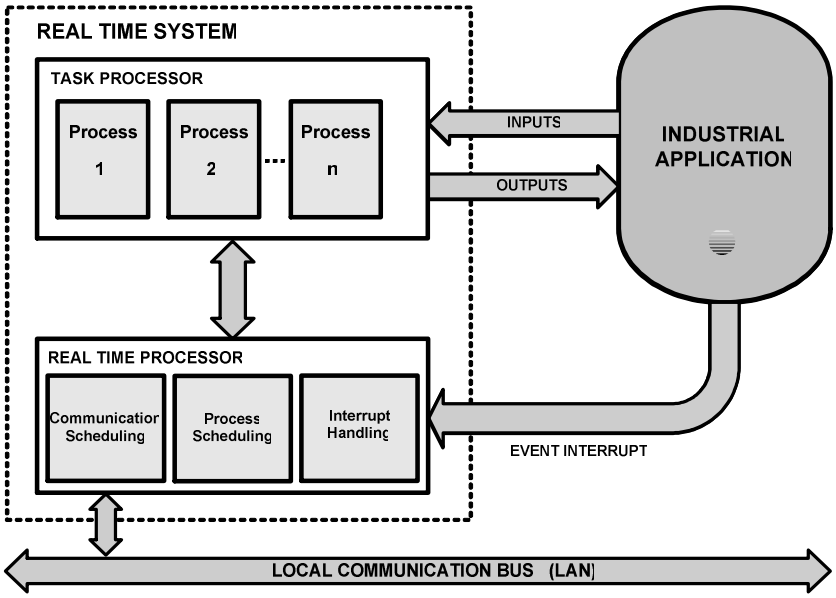

Fig. 1. Real time system structure

The first refers to the time management and the second one refers to the placement of the (dynamic) processes on the available processors, so it is a distributed management.

Respecting the time restrictions in a real time application shall be provided by the planning techniques of the execution processes.

The scheduling of the processes in distributed systems can be defined as allocating the execution time to the processors in such a way that the execution time is minimized, the CPU utilization is maximized, and the balancing processes are optimized.

The real-time planner is a program unit that controls the execution, interruption and the end of program modules based on a pre-established schedule algorithm in order to meet the imposed time restrictions. The planning strategy decides on the election of the process to be executed.

In distributed computer systems, the scheduling process has two basic functions:

- Optimization (maximization) of CPU time usage. If the rate of processor utilization is defined as the ratio between the active and inactive period, this value must be between 65% - 95% for a good project.
- Minimizing the response time at the execution of a process. If we define response time as the time between calling and ending of process execution, in a good project this value should not exceed an average of more than 1-10% during the effective execution of the process.

2. Distributed, embedded, multi-cluster electronic systems

Multi-cluster systems are composed of both functionally and temporally tightly coupled clusters formed by internal nodes (Fig. 2.). The link between clusters is done by crossing nodes (gateways). Nevertheless, the coupling between clusters remains much weaker than the coupling between nodes within a cluster.

The advantage of using multi-cluster systems is to ease the planning of processes for complex distributed systems, respectively to limit the communication bandwidth between nodes.

To obtain a feasible plan to meet all time constraints, even in the worst case of a sophisticated application, several planning ways need to be used together.

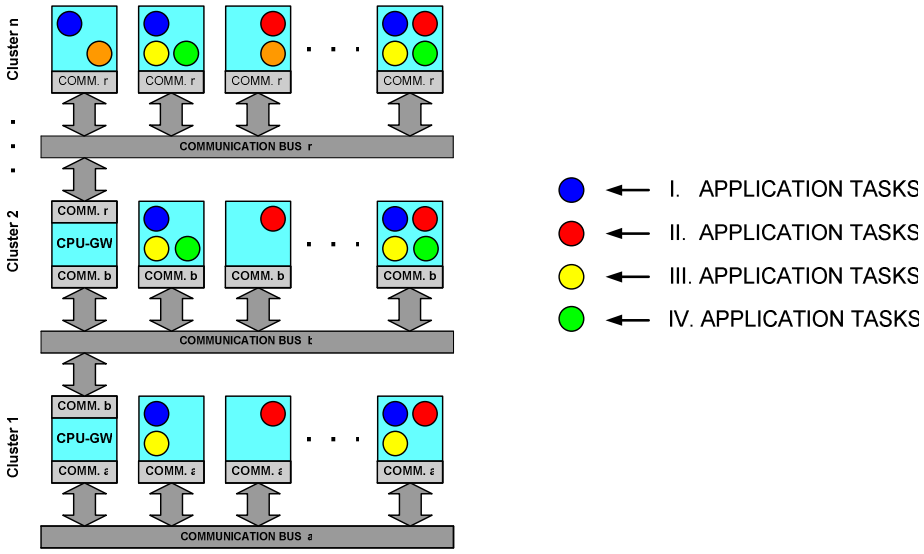

Fig. 2. Distributed multi-cluster embedded system

This trend is motivated by the possibility of reducing the number of used nodes, and hence the costs, as well as by the need for processes to be run on nodes physically close to sensors and actuators.

Moreover, different applications can not only be distributed on different nodes and network nodes (clusters), but the running processes can exchange information between networks through gateway nodes.

3. Scheduling optimization

In case of multi-cluster embedded applications which require large amounts of computation and memory, optimization of architectures, scheduling algorithms and communication protocols are imposed, by looking for the best compromise between reducing the required hardware resources and software implementation, and ultimately increasing the application functionality.

The optimization problem can be formulated as follows: we have an application A, represented by a set of graphs Gi, respectively the system architecture that consists of N distributed processing nodes on which the Pi application processes are running. The following steps should be followed:

- finding the planning strategies for processes -Si (Strategy)
- distribution of the processes on the processing nodes - Mi (Mapping)
- determining the communication bus configuration - Bi (Bus)
- establishing the planning table for the controlled processes
- setting priorities for processes controlled by events

Design optimization strategy involves the following steps:

- deciding on the initial configuration of:
  - communication Bus Bo,
  - scheduling strategy So,
  - process distribution Mo,

  after which the application is scheduled using a suitable scheduling algorithm.

- if the application is planned, the operation is completed, otherwise it continues with an iterative method of improving the process distribution and of the planning strategies.
- if the application is not yet planned, an optimization algorithm is called for in order to improve the communication infrastructure.
- if the application remains unplanned, it is considered that it cannot be implemented with the available system resources and the solution is to upgrade the hardware.

The generic flowchart of the optimization strategy of the designing the application is presented in Fig. 3.

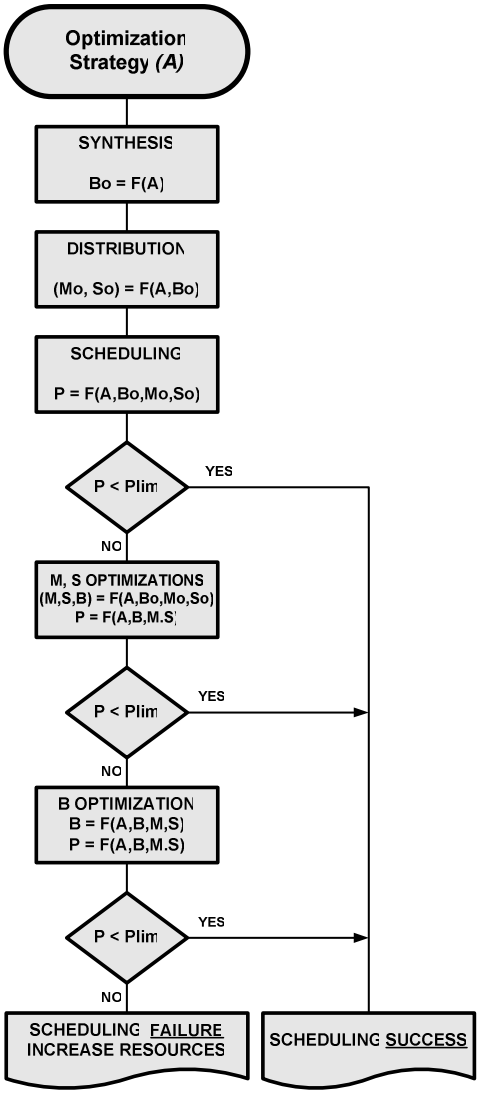

Fig. 3. Optimization strategy for a distributed embedded application

In the case of multi-cluster distributed applications, the application planning has some special aspects. Due to inhomogeneity of message transmission between different cluster nodes, instead of analyzing the delays between the time of occurrence and starting time of processes, for defining and optimizing the scheduling the process phase (offset) is used, namely the nearest possible time moment to start the processes.

The optimized processes scheduling algorithm for distributed multi-cluster systems is presented in Fig. 5. A critical section of the multi-cluster distributed applications is the transmission of messages from a process running

on a node in a cluster triggered by time to a process on a node in a cluster triggered by events.

It is therefore useful to determine the worst case delay in queue $W_m(q)$ of the message m, depending on the number of occupied periods under examination q.

In the case of the transmission of a message from a node located in a cluster triggered by events to a node in a cluster triggered by time, we can write [5]:

$$W_m(q) = w_m(q) - q \cdot T_m \tag{1}$$

where $w_m(q)$ is the length of the period occupied by level i, measured from the moment qT_m [5]:

$$w_m(q) = B_m + \sum_{\forall m_j \in hp(m)} \left\lceil \frac{w_m(q) + J_j}{T_j} \right\rceil \cdot C_j \tag{2}$$

where $hp(m)$ is the set of messages m with a higher priority than the message m_j.

The conclusion to be drawn from the above formula is that message m has to wait, in the worst case, for the longest message with lower priority that has just started to be transmitted (B_m), which is also increased by the length of higher priority messages staying in the queue and must be sent before message m.

Duration B_m, namely the time it takes for the longest lower priority message that has just begun to be transmitted is calculated taking into account the set of lower priority messages $lp(m)$:

$$B_m = \max_{\forall m_k \in lp(m)} \{C_k\} \tag{3}$$

In the case of multi-cluster systems, $l_p(m)$ and $h_p(m)$ must include the messages generated at transfer level node NG, when messages are transferred from the cluster triggered by time to the cluster triggered by events.

Based on the considerations made, the relationship for message queue dimensioning in the worst case can be written, for the messages sent from the cluster triggered by time to the cluster triggered by event [5]:

$$s_o = \max_{\forall m} \left(s_m + \sum_{\forall m_j \in hp(m)} \left\lceil \frac{w_m(q) + J_j}{T_j} \right\rceil \cdot C_j \right) \tag{4}$$

where s_m and s_j are the size of the message m respectively m_j

It has to be noted that the above relations remain valid also if the messages are transmitted from one node to another node, if both nodes are in the same cluster triggered by events.

In the case of the transmission of a message from a node in a cluster triggered by time to a node in a cluster triggered by event, the delay in the transmission queue depends on the number of messages in front of message m in the queue, the dimension of the slot SG assigned to node NG for communication on bus, and the repetition period of slot SG in bus cycle T_C.

For length of the busy period of level m started at a moment qT_m, we can write [5]:

$$w_m(q) = B_m + \left\lfloor \frac{(q+1) \cdot S_m + I_m(w_m(q))}{SG} \right\rfloor \cdot T_C \tag{5}$$

where I_m is the total size of the messages in the queue before the message m. For messages $m_j \in hp(m)$ before the message m, with higher priority than the message m, we can write [5]:

$$I_f(w_m(q)) = \sum_{\forall m_j \in hp(m)} \left\lceil \frac{w_m(q) + J_j}{T_j} \right\rceil \cdot S_j \tag{6}$$

where J_j is the time of establishment (jitter) in the worst case of the process sending the message, and B_m is the time interval in which m cannot be transmitted as SG slot has not started yet. In the worst case it has to wait a full bus cycle for SG slot.

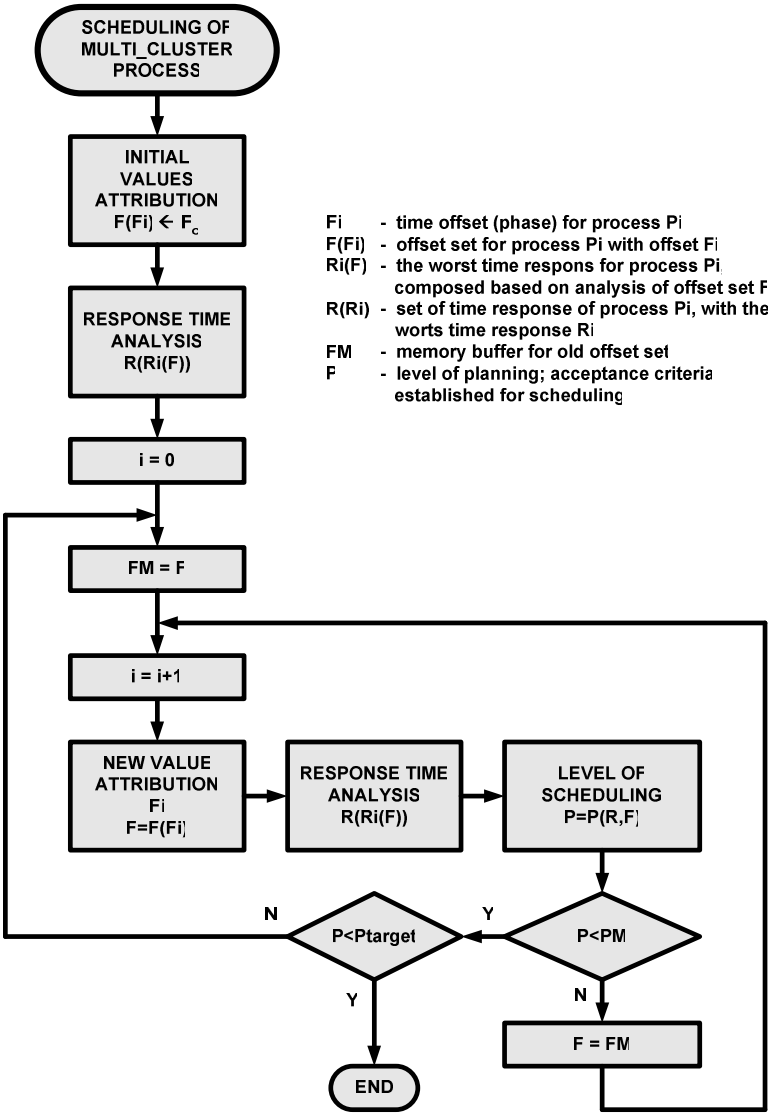

Fig. 4. Scheduling optimization for distributed multi-cluster embedded systems

Based on the considerations made, in the worst case scenario the queue size relationship for message sent from a cluster triggered by event to a cluster triggered by time can be written:

$$s_o = \max_{\forall m} (S_m + I_m) \tag{7}$$

Optimization strategy of communication in multi-cluster systems is reduced to the execution of the following steps:

- checking the possibility of scheduling of application A,

$$r_{G_j} \leq D_{G_j}, \forall G_j \in A \tag{8}$$

- determining the overall size S_T of the queue:

$$s_T = s_o^{PST \to PDE} + S_o^{PDE \to PST} + \sum_{\forall N_{i \in PST}} s_o^{PDE \to PDE} \tag{9}$$

- minimizing the function S_T.

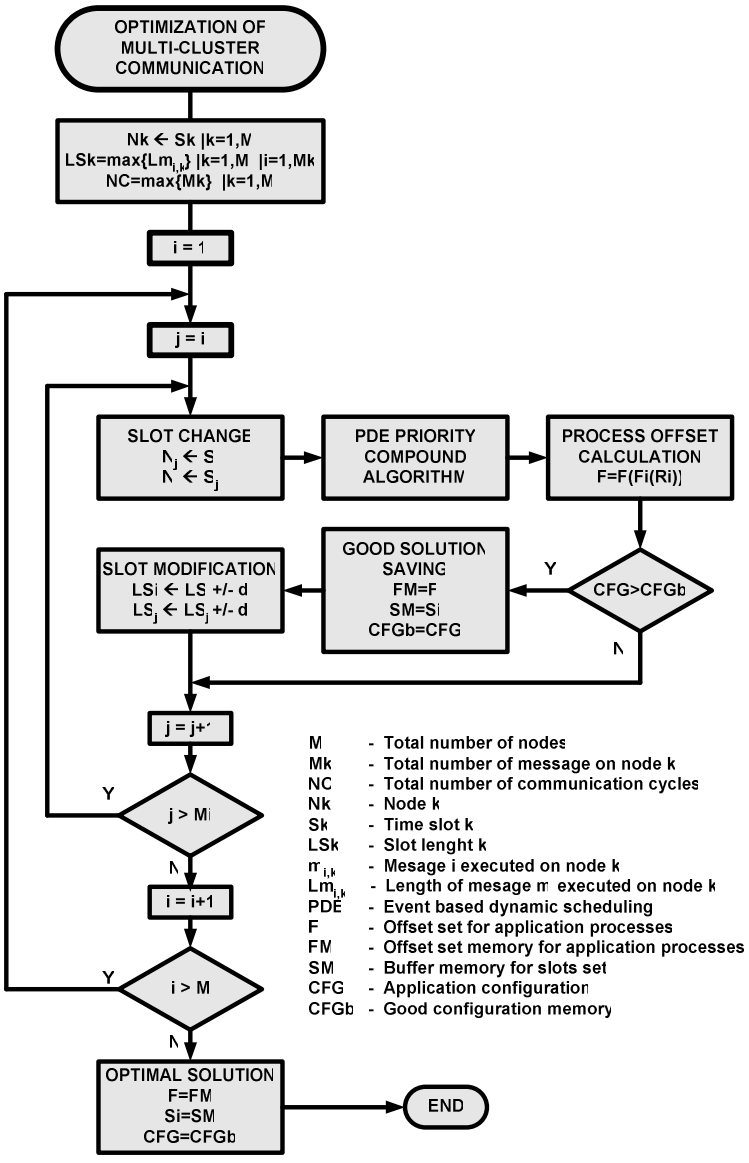

Fig. 5. Optimization of distributed multi-cluster embedded system communication

The generic optimization algorithm of the multi-cluster distributed communication systems is presented in Fig. 5. Next to the procedures used by the algorithm shown in Fig. 5, some heuristic methods may also be tried that lead to the optimization of application resources.

The following four methods are the most important:

- Moving the processes and messages to nodes within the cluster triggered by time, based on offset values and response times.
- Interchange of priorities of two messages sent on the channel targeted by time
- Increase or decrease the duration of extreme slots (the longest decreased / the shortest increased)
- Change the sequence of slots to circular within a time oriented bus cycle

4. Conclusions

Most real-time embedded systems for acquisition, monitoring and control of industrial processes are distributed heterogeneous systems.

Predictability requires us to plan processes statically, cyclically, time synchronized, but instead real life is full of activities to be executed only when an event occurs, asynchronous, with irregular appearance, sometimes unpredictable and random.

Appreciations of inhomogeneous applications are made in the sense that both are composed of time-controlled processes and by event controlled processes, namely communication between processes takes place both guided by time and guided by events

In the case of multi-cluster distributed applications, planning of the application has some special aspects.

Due to inhomogeneity of transmission of messages between nodes of different clusters, instead of analyzing the delays between the time of occurrence and the starting time of the process, for determining and optimizing the scheduling, the process phase (offset) is used, the possible nearest time moment to start the processes.

A critical section of multi-cluster distributed applications is the transmission of messages from a node in a cluster triggered by time to a node in a cluster triggered by events.

It is therefore useful to determine the worst case delay in queue, and to optimize the communication based on computing relationships and the state diagram presented.

Acknowledgements

This work is part of the project funded by the Romanian National Authority for Scientific Research, grant No. 347/23.08.2011.

References

[1] Kopetz H, &all. Time-triggered versus event-triggered systems. Proc. IWOS: Springer Verlag Berlin; 1992. vol. 563, p. 87-101

[2] Izosimov V, &all. Design Optimization of Time- and Cost-Constrained Fault-Tolerant Distributed Embedded Systems. IEEE Design, Automation and Test; 2005.

[3] Eles P, & all. Scheduling of conditional process graphs for the synthesis of embedded systems. Design Automation and Test: Conf. EU; 1998.

[4] Coulouris G, Dollimore J, Kindberg J. Distributed systems - Concepts and Design. Addison Wesley: ISBN13: 9780321263544; 2005.

[5] Pop P, & all. Analysis and Synthesis of Distributed Real Time Embedded Systems. Univ. course: Ed. Linkoping University; 2006.

[6] Losonczi L, Marton L.F. Scheduling techniques for hard real-time embedded systems. MACRo 2011 Conference: Tg.Mures; 8-9 April, 2011.