# Genetic Algorithm for Real-Time Scheduling in Distributed Control Systems

Gheorghe Sebestyen
Technical University of Cluj
str. G. Baritiu nr. 26-28, Cluj,
Romania
*gheorghe.sebestyen@cs.utcluj.ro*

Kalman Pusztai
Technical University of Cluj
str. G. Baritiu nr. 26-28, Cluj,
Romania
*kalman.pusztai@cs.utcluj.ro*

Zoltan Puklus
Szechenyi Istvan University
Györ, Egyetem tér 1
Hungary
*puklus@sze.hu*

**Abstract: Real-time scheduling in distributed control systems is a necessary but also challenging task because of the multitude of interconditioning relations and restrictions. Classical exhaustive searching algorithms are prohibitive in the terms of computation time. Therefore in this paper a genetic approach is proposed. At first a transaction-based computation model of a generic distributed control system is built and then genetic operators are adapted to solve the real-time scheduling problem. The proposed scheduling algorithm optimizes the end-to-end worst-case response time for a set of predefined control transactions.**

## I. INTRODUCTION

In most control applications the fulfillment of real-time restrictions is critical for the correct behavior of the controlled system. Real-time scheduling in a single processor system is a well studied and documented theme; there are a number of classical solutions of optimal scheduling algorithms, such as the RM (Rate Monotonic) [4] and its variations (e.g. priority ceiling, deferred server, etc.), the EDF (Earliest Deadline First) [5], the SLF (Smallest Laxity time First) and many others. But optimal real-time scheduling in distributed systems and especially in control systems is still an open problem. This is caused by a sum of factors:

- there is no centralized control of what happens in the distributed devices of the system
- there are a number of interconditioning and restrictive relations (e.g. ordering relations, synchronization, conflicts on concurrent access to common resources, etc.), other than the real-time restrictions, which makes the scheduling problem a NP (non-polynomial) one
- the limited speed of message transmissions and the significant distances between system's resources generates relativistic effects: it is not possible to build an accurate vision about the global state of the distributed system and events happened in different points of the system may be sensed in different order, if they are observed from different points
- it is difficult to combine task scheduling at device level with message scheduling on the network; ordering relations between tasks and messages and inherent delays caused by these relations increase the complexity of the scheduling task
- in control devices extra restrictions may be caused by their limited computing resources (e.g. an intelligent transducer, based on a microcontroller has a very limited memory space and computing speed)
- the great diversity of control devices involved in a control system adds new restrictions, which are difficult to express in a formalized way

A number of generic distributed control system models were proposed as solutions for the real-time scheduling problem. The MARS approach [3], proposed by Kopetz and his team, is based on a pure time-driven solution, in which control tasks and network messages are scheduled off-line, based on their repetition period. The system guarantees the fulfillment of all task and message deadlines as long as the time characteristics don't change during time. The model has pore support for aperiodic or sporadic events. The Spring model [6], is an event-driven approach; tasks and messages are scheduled on-line, as they arrive, based on their deadlines. This model can guarantee deadline fulfillment for a set of tasks and messages with known time parameters and offers a better chance for sporadic event' handling. Other models propose hardware solutions to solve the communication delays and distributed data consistency [6]. These solutions require dedicated system architectures and networks. The scheduling algorithms cannot be generalized for a wider class of control systems.

## II.A TRANSACTION-BASED GENERIC DISTRIBUTED CONTROL SYSTEM

In order to simplify the scheduling problem in a distributed control system a more formalized (abstract) model is proposed. This model takes into account some specific features of control applications. It is of common knowledge that in a control application most tasks are executed in a periodical fashion; the different control functions imply periodic cycles containing data acquisition, processing and control signal generation tasks. These cycles have predefined periods and deadlines. There may be also sporadic events (e.g. alarms) that must be taken into consideration.

Most control functions can be modeled as a linear sequence of steps (e.g. acquisition, processing, control generation) avoiding non-deterministic wait cycles or branches. Some sequences are executed periodically and others are invoked on the occurrence of some sporadic events.

In a distributed architecture network communication and message delays must be taken into consideration. The linear sequence of steps must include transmission times for messages between tasks. The designer must evaluate and guarantee the worst case end-to-end response (reaction) time for the time-critical control sequences.

According to these considerations a transaction-based distributed model was built. A transaction is a linear sequence of tasks and messages that implement a given control function. The whole control application is built as a set of transactions. Some transactions are activated periodically based on their predefined time periods; others are activated by specific events (e.g. alarms from the controlled system or user requests). Tasks contained in a transaction are distributed in automation devices, connected on an industrial network. The "industrial" term suggests a deterministic network protocol, in which there is an explicit way of controlling time restrictions.

Tasks are scheduled locally on every device using an EDF (Earliest Deadline First) algorithm. Preference for this algorithm is based on two aspects: this algorithm equally accepts periodic and sporadic tasks and it offers a better scheduling chance for a given processor load [7]. A similar algorithm is used to order message transmissions. When a given device gets access to the network, it will select from its queue the message with the smallest deadline. During a network cycle a time-slot is reserved for every device connected on the network, when the specific device may initiate data transfers.

The goal of the proposed model is to guarantee end-to-end time restrictions for all the periodic and sporadic control functions contained in a distributed application. The methodology proposed in the next paragraphs, which tries to solve this goal, has two parts:

- an analytical part that evaluates the end-to-end response times of transactions and compares them with their deadlines and
- a reconfiguration part that uses a genetic algorithm to redefine intermediate deadlines in order to obtain a better chance for a feasible scheduling solution

## III.EVALUATION OF WORST-CASE END-TO-END RESPONSE TIME

Evaluation of the worst-case response time is a necessary step in the process of demonstrating the feasibility of a real-time scheduling strategy for a particular system configuration. A designer proves the correctness of its system, from a real-time point of view, by comparing the worst-case response times of all critical control functions (tasks) with their deadlines. If some of the deadlines are not met than the scheduling policy must be changed or the system's components have to be reconfigured (e.g. relax some time restrictions or eliminate some, less critical, functions). A fine tuning approach may be appropriate.

A number of analytical evaluation methods were proposed [4,5,7,9] for different scheduling algorithms, working in a single or multiple processor environment. The most interesting solutions are those described by Tindell et al. [8] for a RM (Rate Monotonic) scheduling algorithm and by Spuri [5] for the EDF (Earliest Deadline First) algorithm. In both cases a "holistic" approach was adopted, which means that the global or end-to-end response time for a sequence of linear tasks and messages was evaluated. The link between the task scheduling and message scheduling processes is made through the concept of "arrival jitter". A task, which sends a message, is causing an arrival jitter (a non-predictable delay) to that message equal with its worst-case execution time. This jitter influences the worst-case delivery time of the message. In a similar way the destination task's arrival jitter is determined by the receiving messages delivery time. An iterative method must be used to evaluate the response times and the delivery times of all tasks and messages involved in the application. A transaction's response time is given by the response time of the last task in the transaction.

In accordance with the proposed transaction-based computation model, in our approach, the evaluation method proposed by Spuri was adopted. Next, the most important steps of this method are reproduced (a more detailed description can be found in [5]). The method is based on the following theorem, which is a consequence of the well known Liu and Layland's theorem on EDF scheduling:

"The worst-case response time of a task i (for the EDF algorithm) is found in a busy period (a period with no idle intervals) in which all other tasks are released at the beginning of the period and then at their maximum rate." (the proof is found in [5]).

For a task i arrived at moment "a" (a is inside of the busy period) the length L(a,i) of the busy period is computed with the following iterative method:

$$\begin{cases} L(a,i)_0 = \sum\limits_{j\neq i \text{ and } Dj<a+Di} C_j & (1) \\ L(a,i)_{m+1} = W_{a,i}(L(a,i)_m) + (1+\lfloor (a+J_i)/T_i \rfloor)C_i & (2) \end{cases}$$

where: $C_j$ - computation time for task j
Dj - the relative deadline for task j
Ti - period of task i
Ji - release jitter of task i
$L(a,i)_m$ - the busy period's length after m iterations
$W_{a,i}$ (t) - the cumulative workload generated until moment t by the higher priority tasks (tasks with $D_j < a + D_i + J_j$)
$(1+ \lfloor (a+J_i)/T_i \rfloor)C_i$ - the workload generated by previous releases of task i

The higher priority workload $W_{a,i}(t)$ can be computed as the sum of all computation times of tasks' instances arrived before time t and which have a higher priority than task i released at moment a. A task j has a higher priority than task i if $D_j<a+D_i+J_j$.

$$Wa,i(t)=\sum_{j\neq i \text{ and } Dj<a+Di+Jj} \min\{\lceil (t+J_j)/T_j \rceil, (1+\lfloor (a+Di+Jj-Dj)/Tj \rfloor)\}C_j \qquad (3)$$

In the previous expression the first term ($\lceil (t+J_j)/Tj_i \rceil$) gives the maximum number of arrivals of task j until the moment t and the second term ($1+\lfloor (a+Di+Jj-Dj)/Tj \rfloor$) gives the maximum number of task j instances that may have a higher priority than task i. The release jitter $J_j$ of task j increases the number of possible releases of task j instances in a given period of time (t or a). This effect was demonstrated in [8].

The previous iterative computation ends when $L_{m+1}=L_m$. The convergence of the computation is assured as long as the processor's overall utilization factor for the given task set is smaller or equal with 1 [5].

$$\sum_{i=1}^{n} Ci/Ti \leq 1 \qquad (4)$$

Based on the busy period's length, the worst-case response time of task i for a release moment "a" can be evaluated with the expression:

$$r_{i,a} = \max\{J_i + C_i, L(a,i) - a\} \qquad (5)$$

The overall worst-case response time $r_i$ for a task i is the maximum between all the worst-case response times $r_{i,a}$ computed for different "a" moments. The significant values of "a" are in the interval [-Ji, L-Ci - Ji], where L is the maximum busy period length of all tasks.

$$r_i = \max_{a \in [-Ji,\, L - Ci - Ji]} \{ r_{i,a} \} \qquad (6)$$

In this way, in every device connected on the network, the worst-case response times for all tasks can be evaluated, as long as the relative deadline $D_i$, the period $T_i$ and the release jitter $J_i$ of every task are known. In a similar way [5] the worst-case delivery time of all messages generated by a device on the network can be evaluated.

But in a distributed system the release jitter of a destination task depends on the worst-case delivery time of the receiving message and vice-versa, the release jitter of a message depends on the response time of the sender task. This problem is solved with an iterative method in which the response times, delivery times and release jitters are computed based on the values generated in a previous step. In the first step the release jitter times are set to their minimum value, which is the sum of the execution and transmission times of all tasks and messages that precede the analyzed item (message or task). The next equation sets shows the iterative computation steps.

$$\begin{cases} R_{1\,(m+1)} = \Re_{EDF}(\, J_{1(m)}) \\ R_{2\,(m+1)} = \Re_{EDF}(\, J_{2(m)}) \\ \quad\cdots\cdots\cdots \\ R_{n\,(m+1)} = \Re_{EDF}(\, J_{n(m)}) \\ R_{net(m+1)} = \Re_{netEDF}(J_{net(m)}) \end{cases} \qquad (7) \qquad \text{and}$$

$$\begin{cases} J_{1(m+1)} = \wp_1(R_{ret(m+1)}) \\ J_{2(m+1)} = \wp_2(R_{ret(m+1)}) \\ \cdots\cdots\cdots \\ J_{n(m+1)} = \wp_n(R_{ret(m+1)}) \\ J_{net(m+1)} = \wp_{net}(R_{1(m+1)}, R_{2(m+1)}, \ldots R_{n(m+1)}) \end{cases}$$

where:

- $R_{i(m)}$ - the response time vector for all tasks contained in device "i" after „m" iterations
- $R_{net(m)}$ - the delivery time vector for all messages sent on the network after „m" iterations
- $J_{i(m)}$ - release jitter vector of all tasks in device "i" after "m" iterations
- $J_{ret(m)}$ - release jitter vector of all messages after "m" iterations
- $\Re_{EDF}$, $\Re_{netEDF}$ - the expressions of the release time and delivery time
- $\wp_{i,\,net}$ – the expression of the release jitter

The iterative process stops when the computed values in two consecutive steps are the same or if a given deadline is not reached.

The other problem concerning the presented method is that in most cases the relative deadlines of individual tasks and messages composing different control sequences (transactions) are not specified; only the end-to-end deadlines of control sequences can be derived from the controlled system's requirements. So the designer has the difficult task of specifying the relative deadlines, based on empiric considerations. These deadlines significantly influence the chances of finding a feasible scheduling solution. A relative deadline for a given item (message or task) is influencing not only the worst-case response time of the transaction in which is included but also the schedulability of other transactions. The implications are complex and therefore an analytical approach is not feasible. For this reason a genetic approach is proposed.

## IV.GENETIC ALGORITHM FOR THE SCHEDULING PROBLEM

The goal of the bellow genetic algorithm is to generate a set of relative deadlines for tasks and messages, contained in distributed transactions, which assures the success of an EDF scheduling strategy.

Solving a given problem with a genetic algorithm implies two important steps: one is the proper selection of the coding scheme and the other is the adaptation of genetic operators to the specific characteristics of the problem.

In the case of the proposed distributed system model, relative deadlines must be coded in chromosomes. A chromosome contains a number of segments (genes) equal with the number of transactions specified in the system. A segment is a linear sequence of numbers that determine the relative deadlines of tasks and messages contained in a transaction. A successful chromosome is an individual that contains a set of deadlines, which assures the fulfillment of end-to-end deadlines for all the existing transactions. Fig. 1 shows a coding example for a system with 4 transactions.

The initial population of chromosomes is generated with a guided algorithm, instead of a pure random one. In many cases a well-generated initial population increases the probability of finding an acceptable result, or reduces the search time. So a relative deadline for item i (task, or message) must be selected from the interval:

$$D_i \in [C_i,\; D_{tr(k)} - \sum_{j \in tr(k) \text{ and } j \neq i} C_j\,] \qquad (8)$$

where: Di - the relativ deadline of item i
tr(k) - transaction k, to which item i is part of
$D_{tr(k)}$ - the end-to-end deadline of transaction k
$C_i$ - the computation time of task i or transmission time of message i

A very small relative deadline, close to the computation or delivery time, reduces the probability of finding a feasible schedule for that item. A bigger deadline reduces the other items' chances of being scheduled. Therefore a fair laxity time algorithm is proposed; according to this algorithm the relative deadline of an item is selected randomly around a value, which is obtained as a sum between its computation/delivery time and an average waiting time (called laxity time). The average waiting time

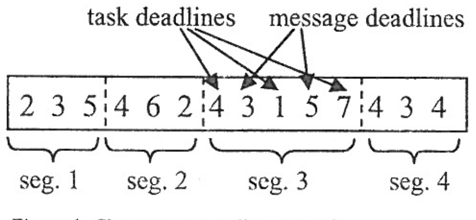

Figure 1 Chromosome coding example

is computed per transaction and is obtained by dividing the spare time in a transaction with the number of its items. So in the initial population a relative deadline is randomly generated around the value:

$$C_i + ( D_{tr(k)} - \sum_{j \in tr(k)} C_j ) / m(k) \qquad (9)$$

where: m(k) is the number of items in transaction k

A fitness function (F(I)) quantifies the scheduling chances for a given deadline set (chromosome). This function takes into account the relative difference between the end-to-end deadline and the worst-case response time of every transaction. A penalty coefficient is used to reduce the survival chances of an individual for which one or more deadlines are not met. The fitness function has the following expression:

$$f(I) = \sum_{k=1}^{N} (D_{tr(k)} - r_{tr(k)})/Dtr(k) * p_{tr(k)} \qquad (10)$$

where: p tr(k) is the penalty coefficient of transaction k and is computed as follows

$$ptr(k) = \begin{cases} 1 \text{ if } (D_{tr(k)} - r_{tr(k)}) \geq 0 \text{ ; no penalty} & (11) \\ \text{penalty if } (D_{tr(k)} - r_{tr(k)}) < 0 \end{cases}$$

The fitness function is used to determine the limited group of individuals that survive unchanged from one generation to the other (based on an elitist strategy), the group that is combined with the crossover operator and also the individuals that are eliminated.

The crossover operator combines segments from two chromosomes to obtain two new individuals. The cross-point is always between two consecutive segments. The experiments were made with one and two cross-points.

The mutation operator is adapted to the specific features of the scheduling problem. The following two rules are used:

- for a segment (gene) corresponding to a transaction that does not fulfill its deadline condition a random item's relative deadline is increased; the new deadline must be in the interval specified at the beginning of this paragraph
- for a segment corresponding to a transaction that fulfill its deadline condition a random item's relative deadline is reduced

The process is performed until a chromosome fulfills all the deadlines, or it may continue until a more reliable solution is found. A solution is considered more reliable if there are greater distances between deadlines and worst-case response times for all transactions. The explanation is that in case of an error more time remains for recovery procedures. The most reliable individual is selected from chromosomes that fulfill all the deadline conditions and which maximizes a performance function P(I).

$$P(I) = \min_{\text{for all } k} \{ D_{tr(k)} - r_{tr(k)} \} \qquad (12)$$

The experiments were made on a hypothetical control system with a variable number of devices (5 to 10) connected on a network and a variable number of transactions (10 to 30). The transactions had a limited number of tasks and messages (3 to 5) in accordance with real case situations. The tasks were randomly allocated to devices, but preserving the condition that for any device the processor utilization factor was smaller than 100%.

The tests showed that in most cases the proposed method generates a final population in which a number of solutions are acceptable from the schedulability point of view. For those cases when no acceptable solution had been generated in a reasonable computation time, probably there was no such solution for that configuration. Unfortunately there was no other available method to analyze the schedulability of the given configuration.

## V. CONCLUSIONS

This paper presents a solution for the real-time scheduling problem in a distributed control system. At first a transaction-based computational model is proposed in which control functions are implemented as linear sequences of tasks and messages. Then an analytical method is given to evaluate the worst-case response time in the case of an EDF scheduling strategy. The evaluation method requires deadlines for the internal items of transactions. These deadlines are determined through a genetic algorithm. The genetic approach reduces the complexity of the search process.

## VI.ACKNOWLEDGEMENTS

The work was possible due to the support given by the Joint Hungarian -- Romanian Intergovernmental S & T Co-operation Program under grant 13/2002 and contract No. 18051/2003 (code RO-9/2002 and HU-17/2002).

## VII. REFERENCES

[1] T.F. Abdelzaher, K.G. Shin, "Optimal combined task and message scheduling in distributed real-time systems", *in Proceedings of the IEEE Real-Time Systems Symposium*, 1995

[2] C.Gonzalez and R. Waiwright, "Dynamic Scheduling of Computer Tasks Using Genetic Algorithms", *Proc. of the first Intl. Conference on Evolutionary Computation*, 1994

[3] H. Kopetz, "Distributed fault-tolerant real-time system. The March approach", *IEEE Micro, 9(1), february*,1989

[4] C.I. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment", *Journal of ACM 20(1)*, 1973

[5] M. Spuri, "Analysis of deadline Scheduling Real-Time Systems", *INRIA, Research report Nr. 2772*, 1996

[6] J. Stankovic, K. Ramamritham, - The Spring Kernel: A New Paradigm for Real-Time Systems, - IEEE Software, vol. 8, no. 3, pp62-67, 1991

[7] J. Stankovic, M. Spuri, K. Ramamritham and G. Buttazzo, "Deadline Scheduling For Real-Time Systems: EDF and Related Algorithms", *Kluwer Academic Publishers, Boston*, 1998

[8] K. Tindell J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems", *Microprocessors and Microprogramming*, no.