On-Line Reconfiguration for a Type of Networked Control System using Genetic Algorithms

BENÍTEZ-PÉREZ H.*+, SAAVEDRA-HERNÁNDEZ H.+, and ORTEGA-ARJONA J. L.**
**Departamento de Matemáticas, Facultad de Ciencias, UNAM, Ciudad Universitaria, CP. 04510, México City, México.
*+Departamento de Ingeniería de Sistemas Computacionales y Automatización, IIMAS, UNAM, Apdo. Postal 20-726., Admón. No. 20, Del. A. Obregón, México D. F., CP. 01000, MÉXICO.
Fax: ++52 55 5616 01 76, Tel: (*) ++52 55 5622 36 39
Email: (*) hector@uxdea4.iimas.unam.mx

Abstract: Nowadays reconfiguration becomes a crucial issue for maintainability and safety requirements. For instance, fault tolerance can be performed by physical redundancy; besides, an efficient way to perform this task is by choosing those spare fault-free redundant components. This goal can be achieved by the use of reconfiguration. Therefore, this strategy needs to be study in terms of its consequences. Specifically, in terms of a distributed system (for the case of this paper), online reconfiguration tends to be quite useful based on the modularity and reusability of certain system components. This paper presents a proposal for online reconfiguration from the perspective of a real-time distributed system and the related case study. The goal is to propose a way to achieve reconfiguration without paying a safety cost from the point of view of case study and communication network. This approach is based on the use of two genetic algorithms in order to search possible strategies for online reconfiguration. This work has been implemented under RT CORBA to get a useful strategy for distributed system management.

Keywords: Genetic Algorithms, Online Reconfiguration, Network Control

1. Introduction

Nowadays online reconfiguration becomes an issue for safety requirement specially for networked control where one of the key elements is to keep the system working even in hazard situations. To pursue on-line reconfiguration, this strategy needs to be focused into the type of system to be reconfigured and the cost to be paid as result of this action. In that respect on-line reconfiguration is pursued in terms of a computer communication system where physical redundancy is allowed and the primary cost it is related to inherent time delays within communication performance. Specifically reconfiguration is related to keep functionality [1] during absence of computer components (where a component is an autonomous device capable to establish communication between similar units). Two actions can be possible either by silent nodes or replaced nodes. Both schemes can be performed by the use of a scheduling algorithm [2] producing predictive configurations where every task reach its communication deadlines.

Online reconfiguration produce an effect into the system, specially, a distributed system. This effect is mainly (but not always) present as time delay effect during the communication accomplishment. Furthermore, this paper is focused into networked control system where loss of time deadline may produce unstable results during transitions.

Strategies for managing time delay within control laws have been studied. For instance [3] proposes the use of a time delay scheme integrated to a reconfigurable control strategy based upon a stochastic methodology. [4] proposes a reconfiguration strategy based upon a performance measure from a parameter estimation procedure. Another strategy has been proposed by [5] where time delays are used as uncertainties, modifying pole placement of a robust control law. [6] present an interesting view of fault tolerant control approach related to time delay coupling. Reconfigurable control has been studied from the point of view of structural modification since fault appearance as presented by [7]. From this point of view, reconfigurable control performs a combined modification of system structure as studied by [8] and [9]. The objective of this paper is to allow safe online reconfiguration based on previous knowledge of already known valid plans, evaluated by planning scheduler algorithm, and two genetic algorithms taking into account networked control performance as shown in Fig. 1.1.

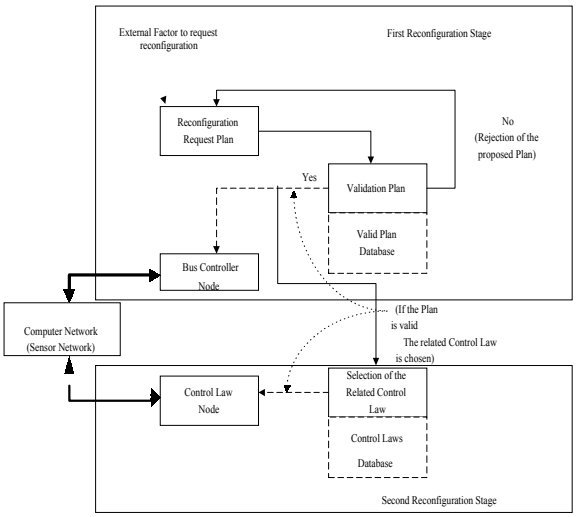

Fig. 1.1 General structure of Reconfigurable System over a Computer Network

This system has been accomplished over a computer network with the peculiarity of using CORBA . This is known as TAO [10] where one of the main characteristics is the capability to communicate objects and components over the network managing which channels need to be open o re-open during communication.

This paper is divided in 6 sections. First section is current introduction, second section describes a general background of genetic and scheduling algorithms. Third section is a review of proposed algorithm. Fourth section gives an idea of implemented algorithm using a case study. Related results from this implementation are shown in section five. Finally some concluding remarks are given in section 6.

2. Background

As mention in first section several techniques need to be used such as genetic algorithms, planning scheduler and implemented over an instance of CORBA. A brief explanation is given for these issues. First Genetic algorithms (GA) are optimization techniques based upon a heuristic approach. These algorithms [11] are heuristic searching methods based on nature evolution. These algorithms establish an analogy between a group of solutions from one problem and the group of individuals from a basic population by codifying the information of each solution as a chromosome. A fitness function evaluates the chromosomes, which is known as adapting function and it is based on an objective of the problem. In a similar fashion, a selection mechanism is introduced where the chromosomes with the best evaluation should be chosen to reproduce more often than the worst cases.

These algorithms integrate two main ideas; simple representations of problem solutions by the use of bit chaining and simple transformations, in order to get a better solution. To implement this approach several elements need to be specified:

- Chromosome Representation
- Initial Population
- Evaluation Metric
- Selection Criteria
- Mutation Operation

The genetic operations are based on the type of representation. Initial population tends to be heuristically generated. The use of this strategy is to optimize a population over a define objective. In here, this objective is defined as the schedulability capacities from distributed systems and the stability analysis from case study. From schedulability analysis several algorithms can be used, such as rate monotonic (RT), earliest deadline first (EDF), flexible time triggered (FTT) [12], and least slack time (LST). The difference between them is marked by the way tasks are ordered. It depends on the application which method for ordering tasks is the most suitable for a particular example. Those algorithms already mentioned are divided into two categories as static and dynamic schedulers. The main difference is that the static scheduler defines during the off-line process the allocation of task, whereas the dynamic scheduler allocates tasks based on current conditions considering a time slot. For instance, consider three tasks with the next characteristics (Table 2.1 and Figure 2.1). Under the EDF algorithm, if a task changes its deadline at Δt, it would have a higher priority than those tasks already defined (Table 2.2).

Table 2.1. Tasks used to exemplify EDF the algorithm

	Consumption Time (C)	Periodic Time (P)	Deadline (D)	Priority
(T_1)	C_1	P_1	D_1	Pr_2
(T_2)	C_2	P_2	D_2	Pr_3
(T_3)	C_3	P_3	D_3	Pr_1

From Table 2.1, task 3 has the smallest slack time (t_{s3}); therefore, it has the highest priority Pr_1. Thereafter, task 1 has the next highest priority and the last task has the lowest priority Pr_3.

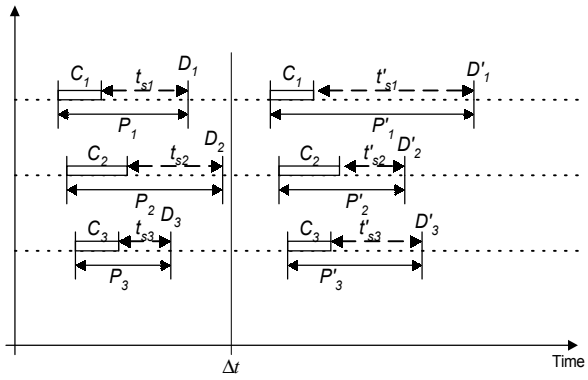

Fig. 2.1. Time graph related to Table 2.1

According to Figure 2.1, there are two scenarios for these three tasks. First, task 1 has slack time ts_1, task 2 has slack time ts_2, and task 3 has slack time ts_3, which gives the highest priority to task 3. The second scenario presents a different priority conformation according to slack time modifications.

Table 2.2. New priority order after at reorganization

	Consumption Time (C)	Periodic Time (P)	Deadline (D)	Priority
(T_1)	C_1	P'_1	D'_1	Pr_3
(T_2)	C_2	P'_2	D'_2	Pr_1
(T_3)	C_3	P'_3	D'_3	Pr_2

For the case of deadline modification, as displayed in Figure 2.1 priorities are modified as shown in Table 2.2 where task 2 has the smallest slack time (t_{s2}); therefore it has the highest priority Pr_1, task 3 has the next highest priority, and the last task has the lowest priority Pr_3.

For real-time purposes, it is best to use static schedulers because of its deterministic behavior. Recently, quasi-dynamic scheduling algorithms have been defined to give certain flexibility to the static communication approach. An example of this sort of algorithm is the planning scheduler [13]. The planning scheduler is a pseudo-dynamic scheduler, in the sense that it presents some dynamic properties but is not fully dynamic. The underlying idea is to use the present knowledge about the system (in particular, the variable set) to plan the system activity for a certain time window in the future.

The scheduler must be invoked once in each plan to build a static schedule that will describe the bus allocation for the next plan. The potential benefit of the planning scheduler in terms of run-time overhead is revealed by the following reasoning. Within a fixed time window of duration P_i, such as the period of variable i among a set of N variables, there are at most S transactions

$$S = \sum_{i=1}^{N} \left(\left\lceil \frac{w}{P_i} \right\rceil + 1 \right) \tag{2.1}$$

Among these algorithms exists the Maximum Urgency First (MUF) algorithm which is quite flexible as required in this paper. This scheduler is managed by properties where tasks with biggest priorities are executed. It uses rate monotonic predictability and the flexibility of dynamic strategies such as EDF. MUF dispatches dynamic and static priorities using the following properties:

Critical Level: Tasks with high critical level are assigned with highest priority levels in a static fashion (These can not be modified during online operation).

Dynamic subpriority: this property is evaluated instantly as function of laxity from current task.

Static subpriority. This priority is used to organize tasks who have the same critical level and dynamic sub-priority. Where static sub-priority has less precedence than the other two. The priority assignment is case study based for the three cases.

Finally this approximation is implemented over a Plattform known as CORBA. This specification focused into object management (Vinoski, 1997) where there is a middle software face that allows communication between entities working at different computing machines.

CORBA applications are integrated from objects who define interfaces using IDL. A specific implementation of CORBA is TAO (The Ace ORB) developed by [10]. It has as main characteristic the implementation on Real Time by using a common event channel between objects. TAO is integrated as a group of libraries known as adaptive communication environment. Classical implementation of TAO is shown in Fig. 2.3.

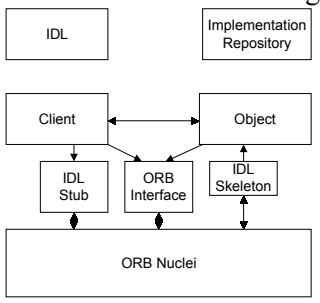

Fig. 2.3. TAO configuration

3. Proposed Algorithm

This proposal is based on management of quality of service from task performance. It takes into account the needs of each task by deploy system resources according to priority levels. These are assigned from case study definition and are taken to modify

operating systems priorities following MUF algorithm. The reconfiguration process is divided in two parts (Fig. 3.1), first genetic algorithm is used to determine those configuration tasks that satisfy the evaluation requirements such as efficiency and stability from case study.

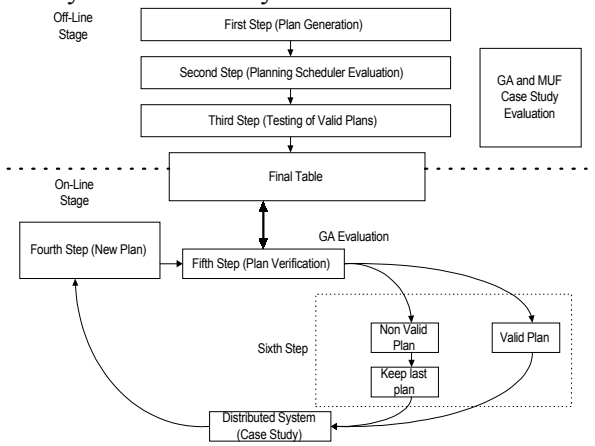

Fig. 3.1 Modified Planning Scheduler

This stage is known as offline evaluation because knowledge search database is integrated from case study response according to a GA evaluation of possible task configurations.

Second stage is known as online configuration bearing in mind, real-time performance, during this stage the online process, the MUF algorithm, the event channel, the second GA and the reconfiguration algorithm perform their tasks in order to adjust task consumption time and reconfiguration according to system behaviour.

During offline performance the genetic algorithm search from an initial population, which plan is valid from the perspective of individual consumption time of each task and the response of the whole group of tasks with respect to case study. If a plan (an individual from initial population) performs a very poor response with respect to any of these two mentioned metrics, fitness value is very low. Therefore GA would try to modify this individual in order to produce a better population.

After several iterations this algorithm produce a population according to those restrictions from plan feasibility and system performance. During online stage every time a plan proposal is presented to perform reconfiguration, this is evaluated through the GA by searching the best candidate similar to that proposed if the reconfiguration is possible. The scheduler values are reported to the event channel and the reconfiguration algorithm. This last algorithm distributes the data to the involved tasks and reconfiguration takes place. As result from this search if one plan is chosen, computer network reconfiguration takes place; otherwise, computer network keeps the same configuration.

During online stage, second genetic algorithm performs optimization over a bounded population using case study error response to be able to determine system performance and be able to chose a suitable plan based on the conditions from reconfiguration. In terms of a time diagram this values takes place every specific time window as shown in Fig. 3.2.

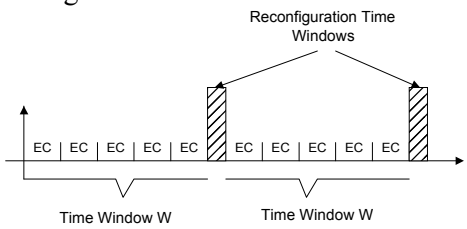

Fig. 3.2 Time Window Managing

If plan candidate is valid, this is distributed to every task during a particular time window. If this plan is not valid, then, current plan is kept for next time window *W*.

During second stage if one of the valid plans is selected, then, the related control law is performed as well.

It is essential to remember that reconfiguration and plan distribution takes place between time windows *W* as shown in Fig. 3.2. In this case, reconfiguration is allowed just during a fixed period of time. Initially, the modification of scheduler strategy is based upon local faults of peripheral elements. Although these topics are of interest, this work is focused on the dynamical modification due to scheduler adjustments rather than the causes of on-line reconfiguration. The AGs use a binary encoded population with chromosomes holding 3 integer variables for each task represented in a scheduling plan. Every chromosome represents a plan that can hold any number or tasks. The AGs are generational, use elitism, roulette wheel selection, a crossover probability of 0.8 and a mutation probability of 0.2. In the offline case the population has 300 individuals and is evolved at a maximum of 500 generations. For the online case is used an initial population with only 10 individuals evolved for 2 generations every time the system tries to find a new solution. This small population is used to minimize the time needed to process an entire generation and to not interfere with the running tasks. The fitness function uses the MUF algorithm and the steady state response to assign fitness to every individual. If the tasks can be accommodated into a feasible plan and the steady state is minimized the fitness is higher.

4. Case Study

The use of inverted pendulum is followed in this work to pursue this approximation. Case study is related to a classic nonlinear benchmark named inverted pendulum (Fig. 4.1). This model it is a two dimension problem with the following dynamics.

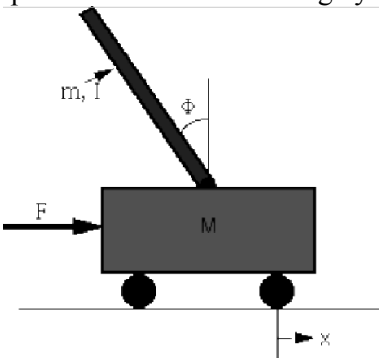

Fig. 4.1 Classic Inverted Pendulum

Where
M= Mass of the car
m= mass of the pendulum
b= Friction from the car
I = Pendulum inertia
g = gravity
u = Applied force
x = position of the car
ϕ = Angle of the pendulum with respect to the car
The transfer function is deployed next:

$$\frac{\phi(s)}{u(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b\left(I+mI^2\right)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}}$$

This system is integrated with 9 components. There is a basic constraint which is the basic communication system (sensing, controller, actuator) Fig. 4.2. The system is implemented in 2.4 GHz PCs using Linux Mandrake 9.1 and TAO 1.4.2 in a 100 Mbps Ethernet local network.

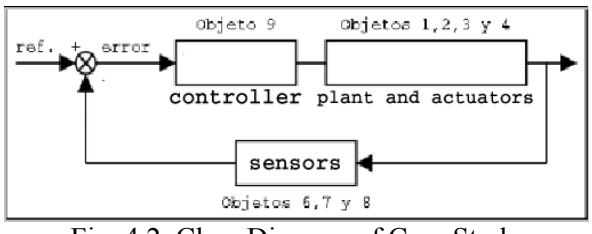

Fig. 4.2. Class Diagram of Case Study

Where classical control strategy like pole placement is pursued.

5. Results

From this implementation two main characteristics need to be mentioned in order to be optimized through GA's. The schedulability of certain group of tasks and the error response from case study.
Second characteristic optimize error in steady state conditions where error less than 1% is highly valued. Fig. 5.1 shows this proposed response.

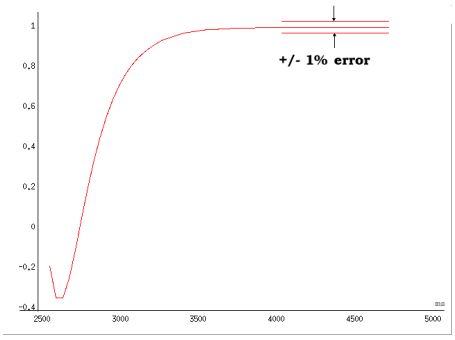

Fig. 5.1. Error Performance

First characteristic tend to optimize wasted time where plans with 80% of databus traffic and high computer power are highly valued with respect to plans with less than 60% from databus traffic. Execution task without reconfiguration using the distributed system is shown in Fig 5.2

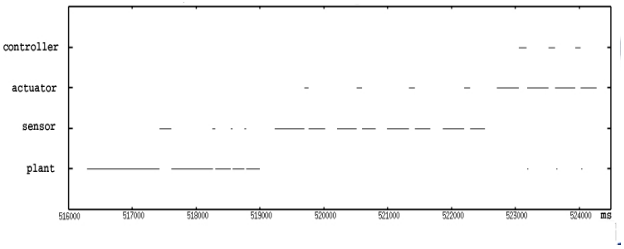

Fig. 5.2. Classical Execution During Non-Reconfiugrable Tasks Perfomance

Based on the optimization procedure shown in section 3 over computer performance and stability tasks reconfiguration is performed over specific time windows as shown in Fig 5.3.

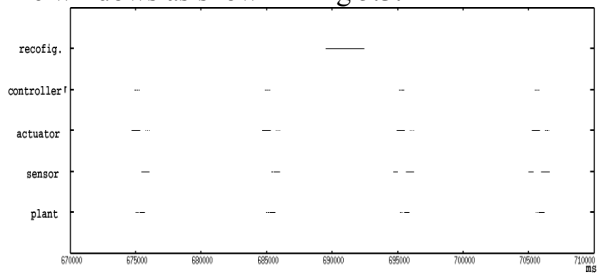

Fig. 5.3. Online Reconfiguration Performance

From the evaluation of reconfiguration, 99.9% percent of selected plans are valid, where 99.8% of these plans are distributed on time.
The measures show that the system is stable even when the reconfiguration takes place. The time needed to reconfigure the system is about 4 ms with tasks around 10 and 20 ms being interrupted. The

system starts the reconfiguration process every 2 seconds but the reconfiguration only takes place if a better solution is found. For a total of 100 experiments of reconfiguration the next results are shown in Fig. 5.4.

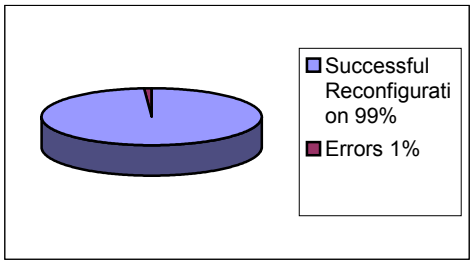

Fig. 5.4 Percentage of success in reconfiguration

6. Conclusions

Present work has shown an approach to implement online reconfiguration as an optimization problem from the use of genetic algorithms. This novel approximation allows periodic reconfiguration with the basic advantage of predictive results over consecutive time window where reconfiguration will take place. Moreover, structural and dynamic modifications can take place over safety basis of pervious known performance. The use of middleware allows a feasible implementation using online reconfiguration due to communication facilities and scheduling integration. Although, system dynamics can be very fast (around 1ms in sampling period) where the system response is around 20ms due to inherent Ethernet response which is the main disadvantage over the implementation. One drawback of this system is when reconfiguration take place which is a periodic task. This can be overcome from GA's implementation and safety requirement measures. Further work is pursued over system mobility when hazard situation occurs considering different measures.

Acknowledgments

The authors would like to thank PAPIIT-UNAM (Num. 106100 and 105303) Mexico.

References

[1] Thompson, H., Chipperfield, A., Fleming P., and Legge, C.; "Distributed aero-engine Control Systems Architecture Selection using multi-objective Optimisation"; *Control Engineering Practice,* Vol. 7, 1999, pp. 655-664.

[2] Cheng, A.; "Real-Time Systems: Scheduling, Analysis and Verification"; *Wiley-Interscience*, 2002.

[3] Nilsson, J.; "Real-Time Control Systems with Delays"; *PhD. Thesise, Department of Automatic Control, Lund Institute of Technology, Sweden*, 1998.

[4] Wu N.; "Reliability of Reconfigurable Control Systems: A Fuzzy Set Theoretic Perspective"; *Proceedings of the 36th Conference on Decision & Control, IEEE, TP15 5:10*, 1997, pp. 3352-3356, San-Diego, USA.

[5] Jiang J., and Zhao Q.; "Reconfigurable Control Based on Imprecise Fault Identification"; *Proocedings of the American Control Conference, IEEE*, 1999, pp. 114-118, San Diego, June.

[6] Izadi-Zamanabadi R. and Blanke M.; "A Ship Propulsion System as a Benchmark for Fault-Tolerant Control"; *Control Engineering Practice*, Vol. 7, 1999, pp. 227-239.

[7] Blanke, M., Kinnaert, M., Lunze, J., and Staroswiecki, M.; "Diagnosis and Fault Tolerant Control"; *Springer*, 2003.

[8] Benítez-Pérez, H., and García-Nocetti, F.; "Reconfigurable Distributed Control"; *Springer-Verlag*, 2005.

[9] Thompson, H.;"Wireless and Internet Communications Technologies for monitoring and Control"*; Control Engineering Practice*, vol. 12, 2004, pp. 781-791.

[10] Schmidt, D. C., D. L. Levine and S. Mungee; "The Design of the TAO Real-Time Object Request Broker"; *Computer communications*, Elsevier Science*, Vol. 21, No. 4, 1998.*

[11] Kuri, M., and Galaviz-Casas J.; "Algoritmos Genéticos"; *Instituto Politécnico Nacional, Universidad Nacional Autónoma de México, Fondo de Cultura Económica*, México 2002.

[12] Almeida L., Pedreiras P., and Fonseca J. A.; "The FTT-CAN Protocol: Why and How"; *IEEE Transactions on Industrial Electronics*, Vol. 49, No. 6, 2002, pp. 1189-1201.

[13] Almeida, L., Pasadas, R., and Fonseca, J.A.; "Using a Planning Scheduler to Improve the Flexibility of Real-Time Fieldbus Networks"; *Control Engineering Practice*, Vol. 7, 1999, pp. 101-108.