A Proposal for On-Line Reconfiguration based upon a Modification of Planning Scheduler and Fuzzy Logic Control Law Response

Benítez-Pérez H.*, García-Zavala A.**, F. García-Nocetti, ***.

Departamento de Ingeniería de Sistemas Computacionales y Automatización, IIMAS, UNAM, Apdo. Postal 20-726. Del. A.Obregón, México D.F., 01000, México.
Fax: ++52 5616 01 76, Tel: (*) ++52 5622 36 39, (***) ++52 5622 35 69
Email: (*) **hector@uxdea4.iimas.unam.mx** (contact author)
(**) **agarciaz@yahoo.com**
(***) **fabian@uxdea4.iimas.unam.mx**

Abstract. Nowadays on-line reconfiguration for computer networks is pursued as an alternative approach to keep performance levels when a mal function is presented in the system. In this case, reconfiguration is proposed in three stages. Firstly, computer network presents a degradation in time communication due to the appearance of certain local faults. Secondly, based upon this scenario a strategy for on-line reconfiguration is pursued in order to cover faults where new time delays appear between elements. These delays modify the behaviour of the dynamical response of the system. During third stage, the control law needs to be modified in terms of current time delays. Therefore, in this paper, on-line system reconfiguration as multivariable and multi-stage problem is pursued based upon a quasi-dynamic scheduler that takes into account those predetermined time delays and the related control law. Control law reconfiguration is pursued as soon as structural computer network reconfiguration is taken place by using current system performance.

1. Introduction

Nowadays, on-line reconfiguration is an open field for several applications such as computer network based systems and safety critical systems. The complexity of on-line reconfiguration modifies several conditions within the application like communication performance and system behaviour [1]. Moreover, on-line reconfiguration can be reached by the use of several strategies like research operations [2] or scheduling algorithms [3]. In this work, the authors follow second strategy because it presents a feasible technique in order to keep real-time requirements which are necessary for safety critical systems. As mention before, different variables need to be measured in order to perform on-line reconfiguration for a safety critical system, then, a scheduling algorithm cannot reach this goal by its own because it does not consider system performance. It is necessary to take into account several measures such as, the planning analysis, the square error of the application response and the degradation of the system behaviour

The scope of this work is related to safety critical systems response during on-line reconfiguration, where an ad-hoc procedure is presented in order to get on-line reconfiguration. This paper is focused into the definition of a method based upon two algorithms. First algorithm is the planning scheduler that is used to define which plans are valid during an off-line stage. This algorithm takes into account the performance from the related control law (second algorithm) of each plan under the related structural conditions. The problem is to overcome performance degradation from the control law when local time delays appear due to structural reconfiguration. The solution stated above is the goal of this work (the proposal of two algorithms). The goal of this paper is to present an approach for on-line reconfiguration for a safety critical computer network system based upon two algorithms, one for structural reconfiguration (scheduler) and another for system dynamics reconfiguration (reconfigurable control law).

A similar strategy is presented by [4] where an interesting analysis is proposed based upon a trade-off between schedulability and real-time control performance. Alternative strategies have been pursued such as that presented by [5] where a complete framework is reviewed for the design and analysis of distributed real-time control systems. Moreover, [6] have proposed an interesting overview of how time delays related to communication systems are integrated to the control law. An alternative strategy has been presented by [7] where a foundation for optimal controller design is defined for multiple time delays. These delays are caused by a distributed communication system. The result defines a very interesting structure for the same scope addressed in this paper, nevertheless, it presents the constraint of a complete observable system where not always is possible. Moreover, time delays are considered constant where as in here, these are considered time variable based upon scheduling algorithm. This method is focused in a combination of two issues. On one hand, the reconfiguration of a computer network due to certain exogenous demands named as structural reconfiguration. On the other hand, control law reconfiguration as a result of the same exogenous demands named as a dynamic reconfiguration.

This procedure, as first step, proposes a reconfiguration plan. If this is validated (second step) from a comparison procedure explained in a latter section, bus controller takes the correspondent actions to further develop structural reconfiguration over the computer network. At the same time, if the selected plan allows reconfiguration, the bus controller node sends a message to control law node in order to select the related control law (third step). When this last action takes place, control law node gets synchronized to bus controller node to perform both reconfigurations. It is important to define that both databases are determined during offline process. Therefore, two stages are needed, off-line and on-line. During off-line performance a scheduling algorithm tests a group of plans in order to validate some of them. Afterwards (but still in off-line stage), these plans are tested as separate scenarios into the computer network dynamical system with a predefined control law who takes into account the related time delays inherent to current plan. If the response is satisfactory both, the tested plan and the control law are saved into the respective databases. The scheduling algorithm used is the planning scheduler [8] for planning analysis during off-line performance and for scheduling construction during on-line stage. For online stage, the request for reconfiguration from an exogenous agent is carried out. As soon as this requisition is dispatched, this plan is verified within the bus controller node

and the control law node where both (the selected plan and the related control law) are delivered to the system. The verification procedure is based upon a simple comparison of the proposed plan and the valid plan database. This database is to be referred as table subsequently. For on-line stage, planning scheduler is used just to build tasks distribution.
In this paper, a case study has been used; this is based upon a ball and beam example [9]. Third section gives a review of this case study.
This computer network system has been implemented on RTLinux [10] and case study has been simulated in MATLAB 5.3 [11]. This paper has been divided in six sections. First section is current introduction. Second section is planning scheduler revision. Third section is the modification proposal of this last scheduling algorithm. Fourth section presents case study and dynamic reconfiguration. Fifth section presents some preliminary results. Finally, concluding remarks are presented in section sixth.

2. Planning Scheduling Review

This scheduler has been proposed by [8]. It is composed of several components such as tasks, main plan consumption time and elementary cycles. Each task is defined by a local consumed time (c) and local period (T). The main plan consumption time (W) is divided into several elementary cycles (EC) where each EC is divided into local time windows named as esp_i. These last divisions result into a more efficient time managing based upon a preemptive strategy. This proposal (planning scheduler) divides a time window into a more complex time division to that presented by rate monotonic [12].
This planning scheduler is based upon eqns 1, 2 and 3 where U is the total consumption time with respect to related periods. N is the total number of tasks, X is the maximum wasted time between time windows $EC's$.

$$U = \sum_{i=1}^{N} \frac{c_i}{T_i} \tag{1}$$

$$U = \sum_{i=1}^{N} \frac{c_i}{T_i} \langle N\left(2^{1/N} - 1\right) * \frac{E - X}{E} \tag{2}$$

$$X = \quad (X) \leq \quad (C) \tag{3}$$

In this case, time performance is increased in comparison to Rate Monotonic due to re-order of useless time spaces. It presents the advantage of a possible dynamical modification every time window W who is defined as the time window where a very task is executed at least one time. This characteristic makes the system pseudo dynamic in terms of reconfiguration. This algorithm (planning scheduler) is enhanced in order to incorporate new measures such as system performance. As explained in first section, these measurements are taking into account during off-line performance in order to define a suitable control law for those valid plans. This implementation is further reviewed in next section.

3.- The Proposed Method

Having reviewed the planning scheduler, this paper proposes a modification based upon the increment of case study efficiency taking into account dynamic system performance.

This procedure is divided in two main stages (Fig. 3.1). Firstly, the off-line stage (First Step) is performed by the use of several combinations of *c's* (consumed time by local task) and *T's* (Periodicity related to local task) from the total number of tasks. The list of combinations is conformed in a classification who is named table and it is conformed of a fixed number of lists that are tested by the planning scheduler in order to select those who are valid (Second Step). This new group is tested in case study simulation considering a suitable control law (Third Step). This step generates a smaller group of valid lists with suitable control laws. In this step, two groups are formed, the valid plans and the valid control laws. Both have a one to one relation. Both groups integrate the valid response of case study for different dynamical configuration scenarios. This is named as final table.

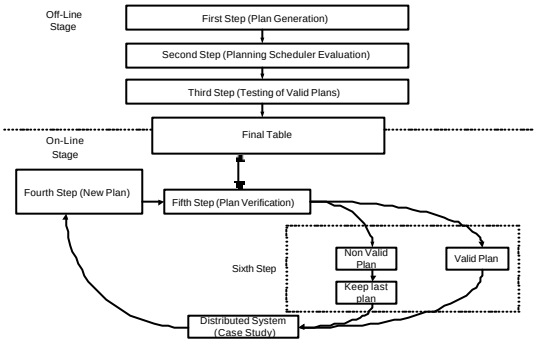

Fig. 3.1. Modified Planning Scheduler

Second stage represents the on-line performance. Firstly, an external element proposes a plan candidate (Fourth Step). This is verified by a simple comparison against those plans presented in final table (Fifth Step).

This comparison is based on an inner product between current proposed plan and those valid plans. If the maximum resultant value from all products is bigger than a determined threshold, current plan is declared as valid. The related control is selected as well.

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

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*. 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.

4.- Case Study

Having explained current approach, a case study is reviewed in order to perform case based evaluation. This case study represents a ball and beam with different optical sensors and two actuators [13]. The linearised mathematical model of the ball and beam is next:

$$A(z^{-1})y(t) = z^{-d}B(z^{-1})u(t-1) + c(z^{-1})e(t) \quad (4)$$
$$B = 0 + 0.0013\,z^{-1} + 0.0016\,z^{-2}$$
$$C = 1 + 1.5977\,z^{-1} + 0.8258\,z^{-2}$$
$$A = 1 + 2.018z^{-1} + 1.032z^{-2}$$

This model is separated in different modules such as peripheral elements and the control law strategy. Sensor and actuator dynamics are neglected for clarity purposes. This implementation is based upon RT-Linux module. This approach has been followed due to synchronization requirements.

Having shown the main structure of this case study, modelling dynamics are taking into account based upon eqn. 4 where sensors and actuators are considered to be linear. In Fig 4.1 it is shown the actual strategy for reconfigurable control.

In this case, the proposed method is performing *on-line stage*, meaning that on-line reconfiguration takes place based upon previous *off-line stage* depuration and current plan comparison. During this *on-line stage* Modified Planning Scheduler module takes into account an external act in order to perform reconfiguration. This external act is based upon local actuators and sensors behaviour in terms of local faults who are not studied in this work.

This Modified Planning Scheduler module is based upon an inner product between current proposed plan and already defined plans (Defined within Final Table). From the result of this operation if one of the plans produces a result bigger than a defined threshold reconfiguration takes place based upon proposed plan (from external act) and the related control law.

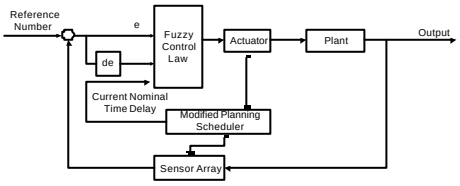

Fig. 4.1. Fuzzy Control Law

Fuzzy control has been chosen rather than gain-scheduler controller and smith's predictor because it has a smooth transition between scenarios. Furthermore, the chosen operating points are the reference elements of proposed fuzzy control. Thus,

any degradation from time delays would degrade control law but the plant keeps a stable response. Time delay degradation is bounded from communication protocol as explained by [14].

Current approach follows Mamdani strategy rather than Takagi Sugeno (TKS) proposal. Further on TKS is focused into future work pursued as the integration of time delays into subsequent part of fuzzy rules.

The actual structure of this controller for fault free scenario is proposed in Fig. 4.2. This is based upon [15]. Membership functions are gaussian bells, where *e* variable has six membership functions (PB, PM, PS, NS, NM, NB), *de* has 6 membership functions (PB, PM, PS, NS, NM, NB). The output variable has eight membership functions (PB, PM, PS, PZ, NZ, NS, NM, NB). Additional variable named Current Nominal Time Delay (CNTD) has three membership functions (N, Z, P). Stability issue is not pursued in this paper. The interested reader may consult [16].

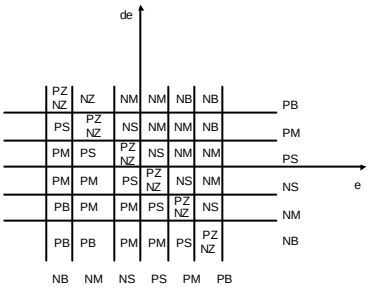

Fig. 4.2. Classical Structure for Fuzzy Control Law

This implementation is a common approach for fuzzy control. For the case of second and third scenarios, Fig. 4.3 shows actual implementation.

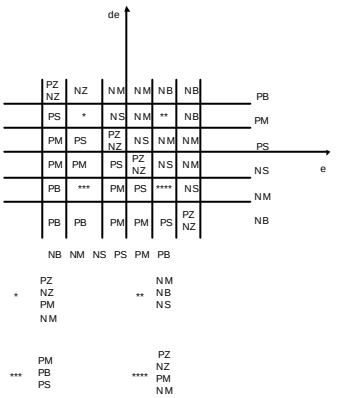

Fig. 4.3. Modification for Fuzzy Control Law

Fig. 4.3 (Fault Scenario II) shows different possibilities at the same condition. This case is proposed due to the possible situation that may be presented at next stage. This

is at 100 percent time delay. For instance, condition *de* is NM and *e* is PM has a result NZ, PZ for fault free scenario (Fig. 4.2). However, Fig. 4.3 presents same scenario with four possible solutions NM, PM, NZ and PZ. This is the result of considering where *e* and *de* suppose to be with 100 percent delayed. In this case every new state in terms of fuzzy control is considered equally possible.

Both control laws have been established firstly from try and error approach, afterwards, the use of a classical cluster technique such as fuzzy C-Means is used in order to validate both control laws [17]. The results are similar to those presented in Figs. 4.2 and 4.3.

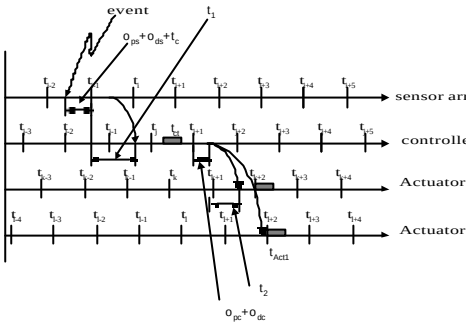

Fig. 4.4. Scheduler for Fault Free Scenario

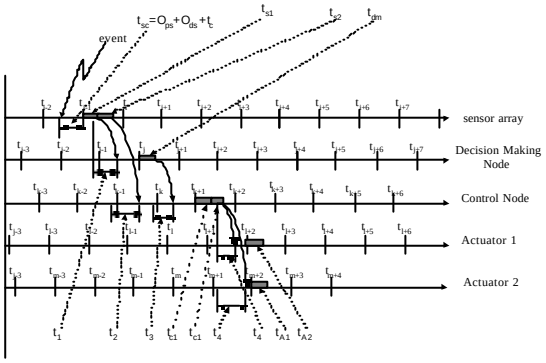

Fig. 4.5. Scheduler for Fault Scenario

Having defined the structure of control reconfiguration, it is necessary to determine those time delays who are the result of system reconfiguration and inter-node communication. These results from each node are transmitted as part of the information flow of control system (sensor-control-actuator). Related to time delays, control node may produces either t_{ff} (time spent for fault free scenario) or t_{fsI} (time spent for fault scenario I) based upon the stage of peripheral elements, furthermore, it gets an estimation of time spent by actuator node and its communication ( ). Having obtained these sources of time delay, control node produces a global time delay .

This value is composed of time spent by sensor, communication time spent between sensor and control, time spent by control node. Δt_e has three different values as shown in eqns. 6, 8 and 10. These values depend on the current scenario. This Δt_e value is considered as an extra input for controller [9]. First scenario is named as fault free scenario. Fig. 4.4 presents a result of time performance. Total time spent during this scenario is 11.5 milliseconds according to table 1.a and eqn. 5.

Var	Name	Time Consume (micro seconds)
c	Communication	450
b	Blocking	50
i	Interference	0
t_c	Capture sensor information	100
O_{ps}	Overhead time from pre-processing sensor information	3000
O_{ds}	Overhead time from post-processing sensor information	3000
t_1	Communication time from sensor node to control node	575
O_{pc}	Overhead of Pre-processing Information from control node	1000
O_{dc}	Overhead of Process Information from control node	1000
t_{ct}	Control Process Time	250
t_2	Communication time from control node to actuator node	575
t_{A1}	Processing time from actuator 1 and 2	2000

Table 1.a Time variable from Fault Free Scenario

Var	Name	Time Consume
O_{ps}	Overhead time from pre-processing sensor information	1000
O_{ds}	Overhead time from post-processing sensor information	1000
t_c	Capture sensor information	1000
c	Communication	450
b	Blocking	50
i	Interference	0
t_1	Communication time from sensor node to decision making module	575
t_2	Communication time from sensor node to control node	575
t_{s1}	Processing time before sending information	2000
t_{s2}	Processing time before sending information	2000
O_{dm}	Overhead of Pre-processing Information	3000
O_{pm}	Overhead of Process Information	3000
t_{dm}	Sending information from Decision Making to Controller	1000
$t_{c1} = t_{c2}$	Processing time from control node	1000
t_3	Communication time from Decision making node to control node	575
t_4	Communication time from control node to actuator node	575
t_{A1}	Processing time from actuator 1 and 2	2000

Table 1.b. Time variables from Fault Scenario

$$t_{ff} = t_1 + o_{ps} + o_{ds} + t_{ct} + o_{pc} + o_{dc} + t_2 + \ldots \quad (5)$$

Where t_{ff} is the total time spent during fault free scenario. This time is a measure related to scheduling scheme shown in Fig. 4.5.

Global time delay (Δt_*) is defined from the occurrence of an event until the information reaches control node. Following eqn. 5 actuator its time consumption and time communication are estimated from previous event. Eqn 6 shows this result.

$$t_{ff} - t_{A1} = \Delta t_{ff} \quad (6)$$

Where t_{ff} represents global time spent, t_{A1} represents time delay spent by actuator at fault free scenario and Δt_{ff} represents time delay at fault free scenario. In nominal conditions Δt_{ff} value is zero. For fault scenario I, see Fig. 4.10 (Table 1.b), the summation of this graph is as follows

$$t_{fsI} = t_{dm} + t_{s2} + t_{s1} + t_2 + t_3 + t_{c2} + t_4 + t_{c1} + t_{A1} + t_1 + t_{sc} \quad (7)$$

This case presents another time delay result due to the appearance of an extra element identified as decision maker module. New communication transactions between sensor and control nodes appear due to this extra element. As a result of this interaction an extra time delay is sum as shown in eqn. 7. As soon as last time delay from actuator node t_{A1} is estimated from previous scenario. Final result is equal to equation 8.

$$t_{fsI} - t_{A1} = \Delta t_{fsI} \quad (8)$$

This time delay represents how long control action is taken to be ready before actuator node acts upon the plant. In nominal conditions this value represents 20- 40% from worst case scenario.

For second fault scenario shown in Fig, 4.10. A similar situation of former case is exposed due to appearance of extra elements. Eqn. 9 shows total time consumed in this scenario.

$$t_{fsII} = t_{dm} + t_{s2} + t_{s1} + t_2 + t_3 + t_{c2} + t_4 + t_{c1} + t_{A1} + t_1 + t_{sc} \quad (9)$$

This third scenario is shown as

$$t_{fsII} - t_{A1} = \Delta t_{fsII} \quad (10)$$

Although t_{fsII} and t_{fsI} are similar in nominal terms, it is expected to be modified due to fault conditions. Nevertheless, the differences between scenarios are not explored in this paper. As result of these three scenarios three time delays are obtained. For the case of this simulation, CANbus standard is used to establish the communication between elements and clock synchronisation is time stamping over each communication process. The implementation of this scheduler as well as the case study is based upon State-Flow toolbox from [11].

5.- Preliminary Results

The evaluation of the system consists of three scenarios during on-line stage. First scenario is a fault free scenario where the response of the plant is ideal. Second scenario is a fault scenario based upon the loss of "smart" sensor output. Finally a

catastrophic condition is presented, then the control law no longer uses the current sensor output.

For these three scenarios the current input of the plant is a pulse train whose indicates the current position of the ball. The output of the plant represents the force applied to the beam. In order to switch to last two scenarios a fault is applied to one of the sensors. The fault is noise that modifies the output of the sensor that measures current position of the ball.

Fig. 5.1 shows the response of the sensor, the plant and the controller during a fault free scenario. Fig. 5.2 shows the response of the system when a fault is presented at 2000 seconds. In here, the control law is second fuzzy logic control (Fig. 4.6) where the input of the faulty sensor is still considered. This fault is active during 500 seconds.

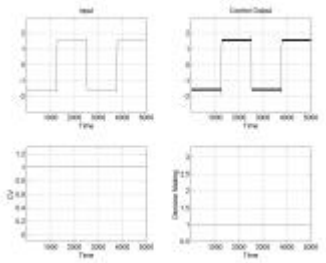

Fig. 5.1. Fault Free Scenario

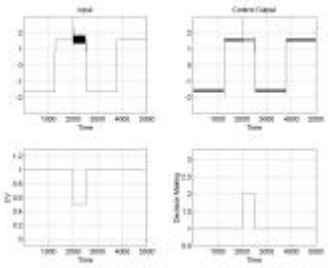

Fig. 5.2. Fault Scenario Starting at 2000 Seconds

As mention in third section the comparison between proposed plan and Final Table is performed by an inner product between them. From the result of this operation a vector is obtained, the maximum element from this vector is considered the winner. If this value is bigger than a defined threshold the proposed plan is taken into account for reconfiguration. The related to control law is switched as well. As depicted in Fig. 5.3 the number of accepted plans is presented taking into those selected with no adequate response from structural reconfiguration. For instance, some tasks would not have enough time to be sampled and executed. This result is presented as the percentage of the adequate use of structural reconfiguration during on-line stage. In

this case, current control law is modified according to time delays status. Case study produces an error due to structural and control reconfiguration. This is evaluated in order to declare a valid plan or not.

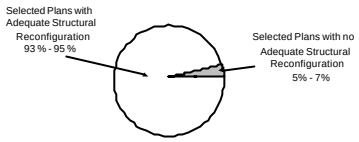

Fig. 5.3. Percentage of Selected Valid Plans for Structural Reconfiguration

Having defined the percentage related to those adequate plans during structural reconfiguration, this is taking as 100 % and is evaluated in terms of control law performance. The results are presented in Fig. 5.4. In here, 97% of the valid plans have a valid response in terms of the mean square error response from the dynamic response of case study.

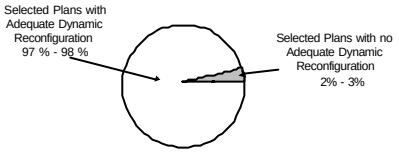

Fig. 5.4. Percentage of Selected Valid Plans for Control Law Reconfiguration

As preliminary conclusion it can stated that on-line reconfiguration will not always be successful by just taking into account the isolate response of control law. It is necessary to take into account the transitions from one configuration to another.
On-line reconfiguration is pursued during a fixed time window named *W* equal to 200 ms. This is presented as a drawback, however, it does keep a safety standard in terms on time dependability.

6.- Conclusions

Present approach has shown how on-line reconfiguration can be pursued based upon dynamic system performance defined from time delays appearance. In order to define these time delays, it is necessary to determine those scenarios where reconfiguration would take place. This selection of suitable scenarios is an *off-line stage* where planning scheduler selects those suitable scenarios and the related control laws. During *on-line* procedure, a simple comparison between a proposed plan and the already selected plan allows on-line reconfiguration. The related control is dispatched at the same time when the selected plan is send to the rest of the elements in the computer network.
Although, this approach is based upon two separate problems, it presents an ad hoc view of how control performance needs to be taken into account in order to develop

on-line system reconfiguration based upon a quasi-dynamic scheduler algorithm. Further work is required in terms of a more precise comparison between current proposed plan and Final Table. In this case, two different approaches may be pursued, the use of Neural Networks for pattern classification and genetic algorithms for table optimisation.

Acknowledgements:
The authors would like to thank the financial support of UNAM-PAPIIT (IN106100 and IN105303) Mexico in connection with this work.

References

1. Cervin A., Eker J., Bernhardsson B., Arzen, K.; "Feedback-Feedforward Scheduling of Control Tasks"; Real-Time Systems, Kluwer Academic Publishers, Vol. 23, No. 25-53, 2002.
2. Cheng A.; "Real-Time Systems"; Wiley Interscience, USA, 2003.
3. Liu J.; "Real-Time Systems"; Ed. Prentice Hall, 2000.
4. Seto D., Lehoczky J., Sha, L., and Shin, K.; "Trade-Off Analysis of Real-Time Control Performance and Schedulability"; Real-Time Systems, Vol. 21, pp. 199-217, 2001.
5. Törngren, M., and Redell, O.; "A Modelling Framework to support the Design and Analysis of Distributed Real-Time Control Systems"; Microprocessors and Microsystems, vol. 24, pp. 81-93, 2000.
6. Cervin A., Henriksson, D., Lincoln B., Eker, J., and Arzen K.; "How Does Control Timming Affect Performance"; IEEE Control Systems Magazine, Vol. 23, No. 3, pp. 16-30, 2003.
7. Lian F. Moyne J. and Tilbury D. ; "Optimal Controller Design and Evaluation for a Class of Networked Control Systems with Distributed Constant Delays"; American Control Conference, pp. 3009-3014, May 2002a.
8. 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, pp. 101-108, Pergamon,1999.
9. Benítez-Pérez H. and García-Nocetti F.; "Switching Fuzzy Logic Control for a Reconfigurable System considering Communication Time Delays"; Proceedings, CD-ROM, European Control Conference; ECC03, UK, September 2003b.
10. Ripoll, J.; "Tutorial of Real-Time Linux"; http://bernia.upv.es/~iripoll/rt-linux/rtlinux-tutorial/index.html, 2001.
11. Mathworks (1998). *MATLAB User's Guide*, MATLAB.
12. Liu L., and Layland L.; "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment" Journal of ACM., Vol. 20, pp. 46-61, 1973.
13. Benítez-Pérez, H., and García Nocetti, F.; "Reconfigurable Distributed Control using Smart Peripheral Elements ", Control Engineering Practice, vol. 11, No. 9, pp. 975-988, 2003a.
14. Lian F. Moyne J. and Tilbury D. ; "Network Design Consideration for Distributed Control Systems"; IEEE Transactions on Control Systems Technology, Vol. 10, No. 2, pp. 297-307, March 2002b.
15. Driankov, D., Hellendoorn, H., Reinfrank, M.; " An Introduction to Fuzzy Logic Control"; Springer-Verlag, 1994.
16. Nguyen H., Prasad N., Walker C., and Walker E.; "Fuzzy and Neural Control"; Ed. CRC, 2003.
17. Höppnner F., Klawonn F., Kruse R., and Runkler T.; "Fuzzy Cluster Analysis"; Ed. John Wiley, 2000.