Improving Flexibility in a Real-Time Fieldbus Network

Rosa Pasadas, Luís Almeida, J. Alberto Fonseca

***Resumo* - O esquema de arbitragem tipicamente usado nos barramentos de campo "tempo-real" baseia-se num *scheduler* do tipo *off-line* que gera uma tabela estática e cíclica contendo a atribuição de tempo de barramento às transacções associadas a variáveis do processo. Esta abordagem, por exemplo usada no barramento de campo FIP (Factory Instrumentation Protocol) é altamente inflexível no sentido de que qualquer alteração ao sistema, tal como adicionar um sensor, obriga à interrupção do funcionamento do barramento.**

Neste artigo propomos a utilização de um *scheduler* do tipo planeamento para resolver esta inflexibilidade. Este scheduler representa uma situação de compromisso entre entre as vantagens e desvantagens do escalonamento (*scheduling*) dinâmico e estático típicos.

É, também, apresentada uma condição suficiente de escalonabilidade (*schedulability*) que implica um custo de desempenho mínimo (*overhead*) e que, por essa razão, é adequada para a análise *on-line*.

A possibilidade de utilização do *scheduler* de planeamento no contexto do barramento FIP também é descrita obtendo--se, desse modo, um barramento compatível *tipo-FIP*.

***Abstract* - A typical approach to real-time fieldbus arbitration is the use of an off-line scheduler that generates a cyclic static table containing the allocation of bus-time-slots to the transaction of process control variables. This approach, used in the FIP fieldbus (Factory Instrumentation Protocol), is highly inflexible in the sense that any system changes, such as adding a sensor, requires the interruption of the fieldbus operation.**

In this article we propose the use of a planning scheduler to overcome such inflexibility. This scheduler compromises between the advantages and disadvantages of typical dynamic and static scheduling.

A sufficient schedulability condition is also presented that incurs minimal run-time overhead and, therefore, is suited to on-line analysis.

The possibility of using the planning scheduler within the FIP context is also described resulting in a compatible FIP-like fieldbus.

I. INTRODUCTION

The increasing demand for higher production volumes and lower production costs has been pushing industry towards an ever increasing automation level. This fact has led to a higher complexity of the typical industrial plant including more and more equipment such as sensors, actuators and controllers. Due to processing power, reliability, flexibility and modularity requirements, industrial systems have become distributed, making use of intelligent equipment spread over the factory plant and interconnected by means of an industrial communications network. The typically large number of equipment interconnected led such networks towards a bus configuration in order to reduce cabling cost and increase modularity and ease of maintenance. These networks, known as fieldbuses, convey data which are used within control loops and thus, are under precise timing constraints. Therefore, a network protocol is required which uses a real-time deterministic access arbitration scheme allowing to know in advance whether the conveyed data will meet its timing constraints [1,2]. If these constraints are not satisfied severe consequences to human lives, equipment and/or environment may happen.

This problem has been solved by the FIP fieldbus (Factory Instrumentation Protocol) using the producer--distributor-consumer(s) model [1,2,3,4,5]. A centralised bus arbitrator uses a cyclic static schedule to initiate periodic transactions[1] for the production and broadcasting of process variables[2] (e.g. sensor readouts or controller outputs). Such schedule is produced off-line and must be loaded into the local memory of the bus arbitrator prior to the start of system operation.

The FIP protocol suffers from the most common disadvantage of static scheduling: operational inflexibility. In order to improve the use of FIP-like buses in dynamic industrial environments, different scheduling schemes must be used that allow on-line system changes [2].

In this paper we propose the use of a planning scheduler to overcome the referred inflexibility. We will show that this scheduler is a good compromise between static and dynamic approaches.

[1] Actually, the FIP protocol also allows a traffic of non-time-constrained sporadic transactions. However, such traffic is isolated from the time-constrained periodic traffic and is not relevant for the real-time behaviour of the fieldbus. Therefore, such sporadic traffic is not mentioned throughout this article.

[2] Whenever the term "variable" is used we refer to variables which value has to travel over the fieldbus, only.

II. ADVANTAGES AND DRAWBACKS OF THE FIP PROTOCOL

As was briefly mentioned in the previous section the FIP fieldbus follows the producer-distributor-consumer(s) model. The protocol uses a time-division multiplexing scheme (TDM) [6] to allocate non-preemptable bus-time-slots to transactions which include the whole production / distribution / consumption cycle of each variable. The allocation is controlled by a centralised arbitrator (the Bus Arbitrator - BA) according to a cyclic static schedule (or schedule table) [2,3,4].

Prior to system operation, a network configuration program builds the schedule according to the rate-monotonic criterium (shortest period first) with phase control for load balancing. The schedule table is then loaded into the BA local memory. At run-time the BA reads this table in a serial order and thus knows which variable shall be broadcasted at each instant (figure 1).

The schedule table has a finite length because all variables are periodic and thus the overall communication pattern will also be periodic. The static schedule produced off-line describes the bus allocation for exactly one of these periods called a macro-cycle (Mc). The size of the Mc is the least common multiple of all the variables' periods (LCM(P_i)).

The system also uses a finite time resolution for the periodic sampling of variables. The minimum time allowed for variables' periods is called an elementary cycle (Ec) and it has a fixed duration defined off-line at configuration time. Each Ec contains several bus-time-slots that can be used for different transactions. The Mc contains an integer number of Ec's.

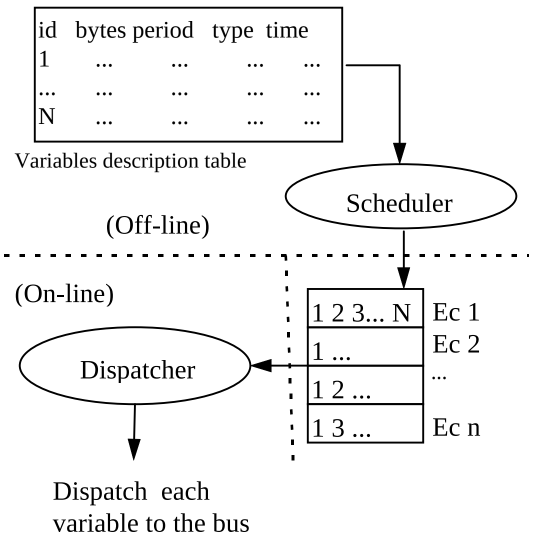

Figure 1 - The FIP static approach[3].

[3] Throughout this article we use the expression "dispatch a variable" meaning the allocation of a bus-time-slot to a transaction for production/distribution/consumption of that variable.

Some of the functional advantages and drawbacks of using FIP come from the fact that the protocol uses an off-line static scheduler [2]. Some of the most important advantages are:

- **Full determinism**. The off-line scheduler uses the characteristics of the variables to produce a periodic schedule (macro-cycle) that fully describes all the future activity of the bus. Whenever it is possible to build such a schedule so that all variables' time constraints are met then the schedulability of the variable set is guaranteed for all the system operation.
- **Low run-time overhead**. The production of the static schedule is, normally, computationally intensive. However, this is performed only once and off-line, before system operation is started. After the schedule is built and loaded into the BA's memory the on-line dispatching of variables is very fast and causes very low overhead. This makes it possible to use higher transmission speeds over the bus.

On the other hand the most prominent drawbacks are [2]:

- **Operational inflexibility**. Perhaps the major drawback presented by the FIP fieldbus is the typical inflexibility of the static schedules. In fact, whenever a change in the variable set is required (e.g. when a new sensor is added) the whole system must be halted and the new variable set must be rescheduled. Then, the new schedule must be loaded into the BA and only then the system may resume its activity.
- **Potentially huge schedule size**. Typical industrial plants may easily include over hundread equipments connected to the fieldbus. If the system operation requires the periodic sampling of over hundread variables, some with sampling periods that can be long and relatively primes, then, the least common multiple of the periods will be a huge number. And so will the size of the schedule (macro-cycle) requiring very large amounts of local memory in the BA to hold it.

III. USING A DYNAMIC SCHEDULER

One of the possibilities to improve the flexibility of the FIP protocol is to use a dynamic scheduler [2]. Such a scheduler would determine only the next variable to be dispatched, searching the whole variable set with a given criterium (figure 2). And it would need to be invoked upon the termination of any transaction. However, between two consecutive invocations, any changes demanded by a dynamic environment could be introduced in the variable set and immediately taken into account by the scheduler.

Also, this scheduler would require only the amount of memory to hold the variables description table, normally smaller than the macro-cycle schedule table.

On the other hand, the disadvantages are the considerable run-time overhead and the incapability of guaranteeing the future schedulability of the variable set. This sort of schedulers must be complemented with a schedulability analyser that must be invoked whenever there is a change in the variable set. Only the use of the analyser allows for a guaranteed timely operation [6, 7, 8].

It is easily seen that the off-line static scheduler and the on-line dynamic scheduler are rather symmetrical. However, in between there is a vast area for schedulers able to reach a good compromise. Such is the case of what we call the planning scheduler.

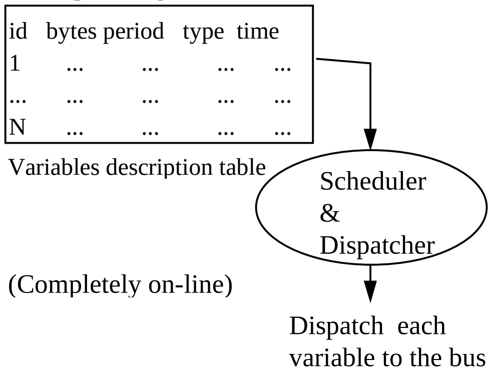

Figure 2 - The dynamic scheduler approach.

VI. The planning scheduler

Let us suppose that we have an on-line dynamic scheduler. Now, suppose that each time the scheduler is invoked, instead of determining only which variable will be dispatched next, it produces a static schedule for a fixed period of time called a plan. Now, it is only necessary to invoke the scheduler every such period of time to have the plan updated. We call this the planning scheduler because it dynamically creates fixed duration plans independently of the variables' periods (figure 3).

The generation of a plan is achieved using a given criterium (heuristic) applied over the variable set. When using a static priority criterium, such as the rate monotonic (RM), it is possible to attain a higher computational benefit when comparing with the dynamic scheduler. Notice that in the dynamic approach the scheduler is invoked once upon the termination of any transaction. Within a fixed time frame of duration W, being P_i the period of variable *i* among a set of N variables, the scheduler is invoked at most S times given by the following expression:

$$S = \sum_{i=1}^{N} \left\lceil \frac{W}{P_i} \right\rceil \qquad (1)$$

In each invocation the scheduler has to perform a search over the set of N variables. The total number of operations performed by the scheduler during the period W will be of the order of N*S.

When the planning scheduler is used with a plan of duration W it is invoked only once during that period. For each variable the scheduler allocates the required bus-time-slots up to the end of the period W. The total number of operations required is now of the order of S.

The main characteristics of the dynamic planning scheduler are the following:

- **Higher flexibility compared to the static approach.** Since the plans are redone every fixed period of time it is possible to introduce changes to the variable set from one plan to the next.
- **Lower run-time overhead compared to the dynamic approach.** The generation of the next plan and the dispatching of the present plan are overlapped. This fact allows to spread the overhead introduced by the scheduler activity over the duration of the plan. Concerning the dispatching of the variables, it is similar to the one performed in FIP and incurs very low overhead.
- **Bounded memory requirements**. The amount of memory required for the operation of the planning scheduler is bounded due to the fixed duration nature of the plans. Also, in systems where there is a reasonable number of variables with relatively prime and long periods the size of the plan may be significantly smaller than the FIP's schedule size (macro-cycle).
- **Limited schedulability guarantee**. The planning scheduler has a full knowledge of the bus activity during each plan, only. However, like the dynamic schedulers it cannot guarantee the schedulability of the variable set for the future beyond the next plan. Therefore, it also requires an on-line schedulability analyser that must be invoked everytime there is a change in the variable set.

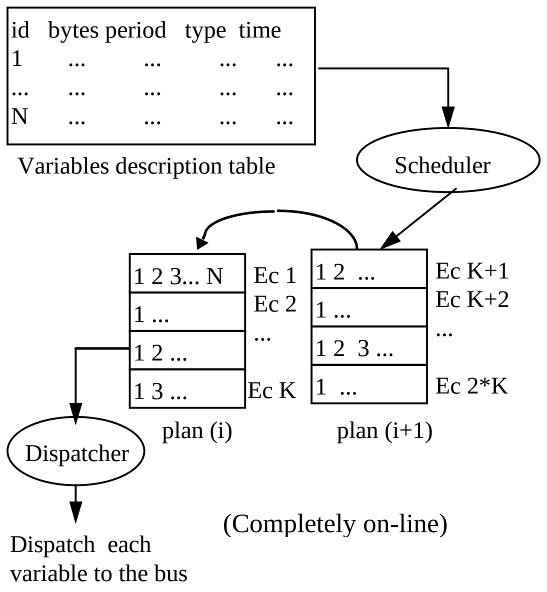

Figure 3 - The planning scheduler

Notice that the properties of the planning scheduler depend on the plan duration (W). If W$\rightarrow$0 this scheduler behaves like the dynamic one. If, on the contrary, W is increased to LCM(P_i) then the plan contains a full macro-cycle of bus operation and there is no need for further updates because all plans will be identical until there is a change in the variable set. In other words, it approaches the behaviour of the cyclic static scheduler. Table 1 summarizes the properties of these 3 types of schedulers in what concerns operational flexibility, run-time overhead, schedule size and schedulability guarantee.

Scheduler	**Static**	**Dynamic**	**Planning**
operational flexibility	low	high	reasonable*
run-time overhead	low	considerable	reasonably low*
schedule size	potencially huge	negligible	bounded*
schedula-bility	guaranteed	not guaranteed	guaranteed each plan

* Depends on the plan duration

Table 1 - Comparison among the 3 mentioned types of schedulers.

V. THE SCHEDULABILITY ANALYSIS

The unpredictability of meeting all future time constraints presented by the dynamic and planning schedulers requires the use of a schedulability analyser. The function of the analyser is to determine if a given variable set can be successfully scheduled indefinitely. In dynamic systems this analysis has to be performed everytime there is a change in the variable set, i.e., on-line. Therefore, the computational cost incured by the analyser must be as low as possible.

At this point it is useful to recall that we are concerned with the scheduling of non-interruptible, independent, periodic transactions over a single bus. This problem is equivalent to the well studied non-preemptive scheduling of periodic and independent tasks over a single CPU [2].

Some of the common methods to analyse the non-preemptive schedulability imply the actual generation of the full schedule. As examples there are the search methods, such as the branch-and-bound and heuristic search [9, 10], or the decomposition methods [10]. However, both are computationally intensive and are very dependent on the particular task/transaction phasing. For this reason we are interested in a simpler analyser based on the verification of a single schedulability suficient condition [8]. Using just a sufficient but not necessary condition will, eventually, decrease the efficiency of the analyser. Nevertheless, due to the very low overhead incurred by the use of such conditions, their applicability to on-line analysis is very common [7, 8]. The drawback of its use is a normally lower utilization. Yet, this might be a reasonable trade-off if the schedulability guarantee in a dynamic environment is a *must*.

The analyser required by the planning or dynamic schedulers is only dependent on the criterium used and on eventual particularities of the scheduler implementation. In the analysis that follows we will see that the use of inserted idle time within each elementary cycle together with the rate-monotonic criterium will allow us to apply Liu and Layland's schedulability condition for normal rate-monotonic preemptive scheduling [12] with a minor modification.

Let us now constrain all variables' periods as well as the duration of the plans so that they will be multiples of a minimum time unit called an elementary-cycle (Ec).

Then, let us see how bus-time-slots are allocated within an Ec. Starting with the variable that has the shortest period (rate monotonic) the scheduler allocates, up to the end of the plan, one bus-time-slot in a number of Ec's according to the variable's period and initial phase. During this process it might happen that the Ec where a transaction should be placed has not enough time left. In this situation we say that the Ec has not enough time capacity and the transaction is delayed ("carried") to the next Ec. If the next Ec also has not enough time capacity this process is repeated until an Ec with enough capacity is found.

Under this scheduling implementation two consequences become evident (figure 4):

- **A bounded waste of bus-time in overloaded Ec's.** Since the duration of the transactions that fitted within that Ec is less or equal the Ec duration (E) there might be a small amount of idle time at the end which will be wasted.
- **Avoidance of potencial preemption instants.** Since it is not allowed that a transaction crosses any Ec boundary all possible instants of preemption (were it allowed) are eliminated. Thus, it becomes irrelevant whether the scheduler is preemptive.

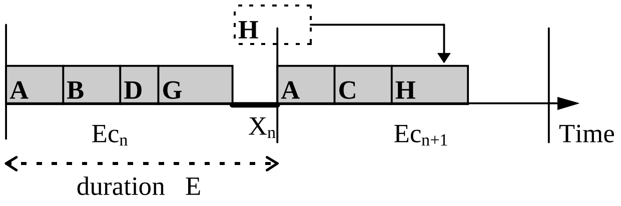

Figure 4 - Delaying a transaction due to Ec time-capacity overflow.

Both consequences result in the schedulability condition (2) which can be calculated on-the-fly incurring minimal overhead. Being N the number of variables in the set, P_i the period of variable i and C_i the duration of the corresponding transaction yields:

$$\textbf{\textit{If}} \quad U = \sum_{i=1}^{N} \left[\frac{C_i}{P_i} \right] < N\left(2^{\frac{1}{N}} - 1\right) \times \frac{E - X'}{E} \qquad (2)$$

=> *the set of N variables is schedulable with any phasing*

The meaning of E and X' are: E=duration of Ec and X' = max (X_n) according to figure 4.

To prove this condition we must first recall that, at the end of each Ec there might be some inserted idle time in the bus period traffic due to the delay of transactions which do not fit within the Ec. In a simplistic worst-case approach the maximum idle time (X') that can be inserted in any Ec_n is equal to the duration of the longest transaction (C_i).

$$X' = \max_{allEc_n}\left(X_n\right) = \max_{i=1,N}\left(C_i\right)$$

Still in a very pessimist approach we can consider that all Ec's will be overloaded and thus, the insertion of idle time (X_i) will happen with a period of E. This results in a pessimistic higher bound to the consequent wasted utilization of bus time, with a value of X'/E[4]. Although pessimist this simple approach is sufficient for our purpose of demonstrating condition (2).

Now, let us apply the folowing transformation to the initial variable set (V) but keeping its phasing:

$$V \mapsto V' \equiv \left\{ v'_i\left(P'_i, C'_i\right) : \quad (3)\right.$$
$$\left. P'_i = P_i \wedge C'_i = C_i \times \frac{E}{E - X'} \right\}$$

We can hypothetically schedule the new set V' according to the rate monotonic and preemptively. In this case we can use the schedulability condition of Liu and Layland and say that the schedulability of set V' for any phasing will be guaranteed if total bus utilization is less than $N(2^{1/N}-1)$.

Notice that the termination of any transaction in set V' with the normal RM scheduler will always be later than the termination of the corresponding transaction in set V with the proposed scheduler.

Therefore, the schedulability of set V' under normal RM (*) implies the schedulability of set V under the proposed scheduler (**). This proves condition (2):

$$U' = U \times \frac{E}{E - X'} < N\left(2^{\frac{1}{N}} - 1\right) \qquad (*)$$
$$\Leftrightarrow$$
$$U < N\left(2^{\frac{1}{N}} - 1\right) \times \frac{E - X'}{E} \qquad (**)$$

Nevertheless, notice that this bound is lower than the $N(2^{1/N}-1)$ bound and depends on the duration of the longest transaction required by any variable in the set. Further refinements can be done to slightly improve this bound using a deeper analysis of the inserted idle time caused by Ec overloading.

If the longest transaction is 10% of the duration of the Ec and N is large, then the schedulability bound will lower to aprox. 63% maximum utilization. Notice that, as the maximum transaction duration decreases, the schedulability bound will approach the bound for rate-monotonic preemptive scheduling as expected [2].

Such a low bound may not be too much of a problem if we recall that unused bus utilization can still be used for non-real-time sporadic traffic. This is done in the FIP fieldbus and it is common pratice in many other real-time systems.

[4] However, notice that it is impossible to attain permanent Ec overloading. This would imply the non-schedulability of the variable set. Also, the inserted idle time in an overloaded Ec will normally be less than max(Ci).

VI. FIP COMPATIBILITY

The planning scheduler presented in this paper can easily be used in the FIP fieldbus requiring changes at the BA level, only. The result would be a FIP-like fieldbus that keeps most of the operational properties of FIP such as the communication model, efficiency and robustness of the protocol, same level of fault-tolerance, etc..

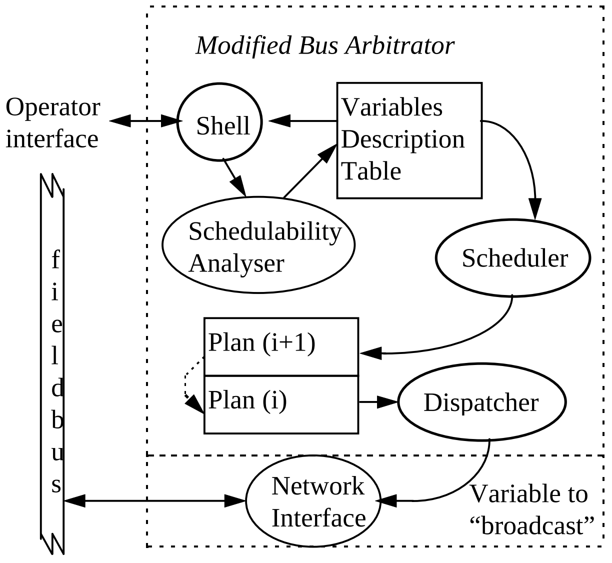

Figure 5 - The modified Bus Arbitrator

The modified BA is presented in figure 5. It contains the scheduler and dispatcher, the schedulability analyser and a shell with an operator interface. Through this shell the operator can monitor the variables' status and add or remove variables as required by system changes.

VII. IMPLEMENTATION AND TESTS

An experimental setup was laid to test the functionality of the proposed scheduler. A low cost, low throughput interconnection bus was made joining in parallel 3 RS-485 serial ports. Three PCs were used as nodes, one of which was fully dedicated to implement the bus arbitrator.

The duration of each elementary cycle (Ec) was 54.9 ms and the plan had a duration of 5 Ec's i.e., 274.5 ms.

The transmission rate of the serial ports was 9600 baud with 1 start and 1 stop bits.

Two different types of frames were used in the communication protocol:

- The *ID frame*, 6 bytes long, containing the identification of the variable to be produced in that bus-time-slot.
- The *DATA frame*, 4 bytes long plus data size, containing the actual variable's value.

The duration of the corresponding transaction for each variable was calculated adding the time to transmit one ID frame, the time to transmit the respective DATA frame plus a certain amount of time for the producer/consumer(s) to identify the variable associated with this transaction and take the appropriate actions. This last amount of time was bounded to 1ms.

A set of 5 variables was used as described in table 2.

Id	**Period** (# of Ec's)	**Size** (bytes)	**Trans.** (ms)*
A	1	4	15.6
B	3	4	15.6
C	4	4	15.6
D	4	4	15.6
E	4	4	15.6

* Duration of the corresponding transaction.

Table 2 - Variables description table

For this variable set the macro-cycle schedule has a length of 12 Ec's. One possible configuration of such a schedule considering the worst-case phasing (when all variables become "ready" for transmission at the same instant in time) is depicted in figure 6. This is not a pratical example simply because in certain Ec's the variables' transaction times overflow the Ec duration.

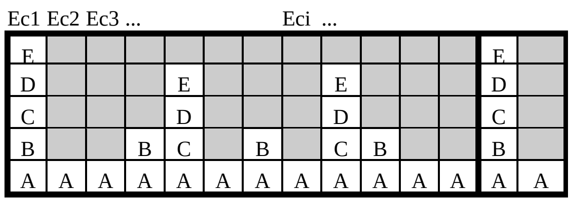

Figure 6 - A macro-cycle schedule

Figure 7 shows the first 2 consecutive plans built by the planning scheduler. Notice that in this case the scheduler takes into account the finite time capacity of each Ec. Thus the variables that do not fit within a given Ec are successively carried to the next until an Ec is found with enough capacity. This search is performed with rate monotonic priority.

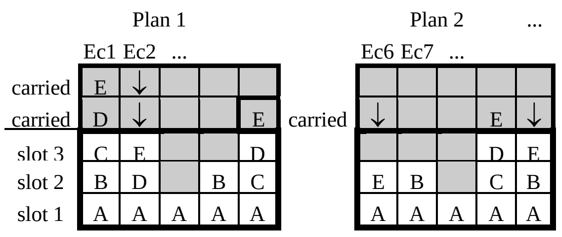

Figure 7 - First 2 plans (only 3 variables fit within each Elementary cycle - Ec).

In this example, the bus utilization required by this set of variables is 59.2%. The number of transactions that fit within an Ec is 3 with a waste of 8.1ms in overloaded cycles (note that all variables are of the same size). In a worst-case approach this corresponds to a wasted utilization of 14.7%. Applying the Rate Monotonic schedulability condition with N=5 (74.3% utilization) over the non-wasted part of the Ec's (85.3%) results in an utilization threshold for guaranteed schedulability of 63.3%.

Since current utilization is less than the threshold, this variable set is schedulable under rate monotonic sequencing, with any phasing among the variables.

VIII. CONCLUSIONS

In this paper we demonstrated that in real-time fieldbus networks relying on off-line static scheduling, such as the FIP fieldbus, the use of a planning scheduler in the Bus Arbitrator can improve the network operational flexibility while keeping most of its functional characteristics. In such a 'FIP-like' fieldbus it is possible to dynamically reconfigure the system, namely add or remove equipment, such as sensors or controllers, without needing to halt system operation.

Such improvement is achieved at the cost, essentially, of lower bus utilization for guaranteed schedulability. To minimize this negative impact work is being carried out in two directions:

- the precise quantification of the overheads incured by the planning scheduler. This quantification is very important to determine the maximum throughput that the modified arbitrator can sustain.
- the improvement of the schedulability analyser efficiency in order to raise the schedulability threshold. This will allow higher bus utilization while keeping guaranteed schedulability.

REFERENCES

[1] J. P. Thomesse, "Les Réseaux Temps-Réel", Ecole d'été: Reseaux de communication et techniques formelles. Ecole Nacional Superior de Telecommunication - Paris, Septembre 1994.

[2] C. Cardeira, Z. Mammeri, "A schedulability analysis of tasks and network traffic in distributed real-time systems", *Measurement, The Journal of the International Measurement Conference IMEKO*, Elsevier, 15(2): 71-83, May 1995.

[3] P. Leterrier, “The FIP Protocol”, WorldFip Europe, 2-4 Rue de Bône, 92160 Antony - France 1992.

[4] C. Cardeira, J.P. Thomesse, “O Bus de Campo FIP”, *Electricidade - Revista de Engenharia Electrotécnica e Electrónica, de Comunicações e Gestão*, 38(307): 6-12, Jan. 1994.

[5] F. Pasadas, C. Cardeira, “Real-Time Protocols for Industrial Local Area Networks”, in Proceedings of the *Network of Excellence in Intelligent Control and Integrated Manufacturing Systems*, Cascais, Portugal, June 1995.

[6] S. C. Cheng, J.A. Stankovic, K. Ramamritham, “Scheduling Algorithms for Hard Real-Time Systems - A Brief Survey”, ,in *Hard Real-Time Systems Tutorial*, IEEE Computer Society Press, 1988.

[7] J.A. Stankovic, “Implications of Classical Scheduling Results for Real-Time Systems”, *IEEE Computer*, Vol. 28, N. 6, June 1995.

[8] M. Kein etal. “A Practitioner’s Handbook for Real-Time Analysis: Guide to Rate-Monotonic Analysis for Real-Time Systems”, *Kluwer Academic Publishers*, The Netherlands, 1993.

[9] K. R. Baker. “Introduction to Sequencing and Scheduling”, *John Wiley & Sons*, 1974.

[10] X. Yuan, M. Saksena, A. Agrawala, “A Decomposition Approach to Non-Preemptive Real-Time Scheduling”, *Journal of Real-Time Systems*, Vol. 6, pag. 7-35, 1994.

[11] K. W. Tindell, A. Burns, A. J. Welling, “An Extendible Approach for Analysing Fixed Priority Hard Real-Time Tasks”, *Journal of Real-Time Systems*, Vol.6, N.2 ,pag. 133-151 ,1994.

[12] C. L. Liu, J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment”, *Journal of ACM*, 20 (1), pag. 46-61, 1973.