PERGAMON

Control Engineering Practice 7 (1999) 101–108

Using a planning scheduler to improve the flexibility of real-time fieldbus networks

L. Almeida*, R. Pasadas[1], J.A. Fonseca

Departamento de Electrónica e Telecomunicações,Universidade de Aveiro, P-3810 Aveiro, Portugal

Received 5 December 1997; accepted 20 August 1998

Abstract

A typical approach to real-time fieldbus arbitration is to use 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, e.g. as used in the Factory Instrumentation Protocol fieldbus, is rather inflexible in the sense that any system changes, such as the addition of a sensor, requires an interruption of the fieldbus operation. In this paper the use of a planning scheduler is proposed to overcome such inflexibility. This scheduler compromises between the advantages and disadvantages of typical dynamic and static scheduling. A sufficient schedulability condition is also derived, in order to overcome the typical inability of dynamic (or even planning) schedulers to guarantee schedulability for long-term system operation. The evaluation of this condition incurs very small run time overhead, and can therefore be used with advantage in a fieldbus system that relies on the planning scheduler. An experimental test is described, to illustrate how the planning scheduler works.

Keywords: Real-time communication; real-time systems; scheduling algorithms; factory automation; fieldbus

1. Introduction

The increasing demand for higher production volumes and lower production costs has been pushing industry towards an ever-increasing level of automation. 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 the requirements for processing power, reliability, flexibility and modularity, industrial systems have become distributed, making use of intelligent equipment, spread across the factory plant and interconnected by means of an industrial communications network. The typically large number of interconnected items of equipment has led such networks towards a serial-bus configuration in order to reduce cabling cost, and to increase their modularity and ease of maintenance (Jordan, 1995).

Therefore, these networks, known as fieldbuses, were initially seen as just a simplification of the wiring. However, the concept of a fieldbus has evolved to the level of a communication system that is capable of delivering services, with a given quality of service, that allow its use at different levels within a computer integrated manufacturing (CIM) architecture. This corresponds to a view of the fieldbus as a fundamental system component that facilitates the design of distributed real-time systems and applications (Thomesse, 1997).

To respond adequately to the communication needs of distributed real-time or control applications, where the data to be conveyed are under precise timing constraints, the fieldbus must deliver time-constrained communication services, and use protocols that are capable of managing such constraints (Thomesse, 1997).

Hence, a network protocol is required which uses a real-time deterministic access-arbitration scheme, allowing users to know in advance whether the conveyed data will meet its timing constraints (Cardeira and Mammeri, 1995; Stankovic, 1996). If these constraints are not satisfied, severe consequences for human lives, equipment and/or the environment may happen, as in any hard-real-time system.

One possible solution to this problem uses the producer–distributor–consumer(s) model (Thomesse, 1994;

*Corresponding author. E-mail: lda@ua.pt.

[1]This work was partially supported by the following grant from the Portuguese Government JNICT-PRAXIS XXI/BM/6615/95.

0967-0661/99/$ – see front matter
PII: S0967-0661(98)00143-9

Cardeira and Mammeri, 1995; Thomesse, 1997). An example of a system that uses such model is the Factory Instrumentation Protocol (FIP) fieldbus (Leterrier, 1992). Therefore, throughout this paper the expression *FIP-like fieldbus* will be used to refer to a fieldbus based on that model.

In FIP-like fieldbuses all transactions are initiated by a centralised bus arbitrator (BA), i.e. the distributor, according to a given schedule. Due to run-time overhead considerations, such a schedule is normally static, and would be built up off-line. This is the case for the FIP fieldbus itself, which concerns the traffic of periodic variables. The schedule must then be loaded into the BA's local memory, prior to the start of system operation.

Nevertheless, this scheduling approach suffers from the most common disadvantage of static scheduling: operational inflexibility. In order to improve the use of FIP-like fieldbuses in dynamic industrial environments, different scheduling schemes that allow on-line system changes must be used (Fig. 1) (Cardeira and Mammeri, 1995; Stankovic, 1996). This capacity to accommodate dynamic changes is becoming more and more important as the concept of flexible manufacturing spreads across industry, and also in response to the need to shorten setup and maintenance down-periods in industrial systems (Jordan, 1995; Stankovic, 1996).

This paper proposes the use of a planning scheduler to overcome the inflexibility referred to above. It will be shown that this scheduler provides a good compromise between the static and dynamic approaches to real-time transaction scheduling, allowing users to profit from the benefits of both. The capacity to accommodate system dynamic changes while maintaining a relatively low run-time overhead is the most important characteristic of such a scheduler.

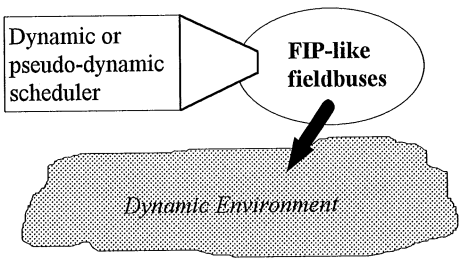

Fig. 1. Using FIP-like fieldbuses within dynamic environments where system changes are allowed on-line requires a dynamic or pseudo-dynamic scheduler instead of an off-line, static one.

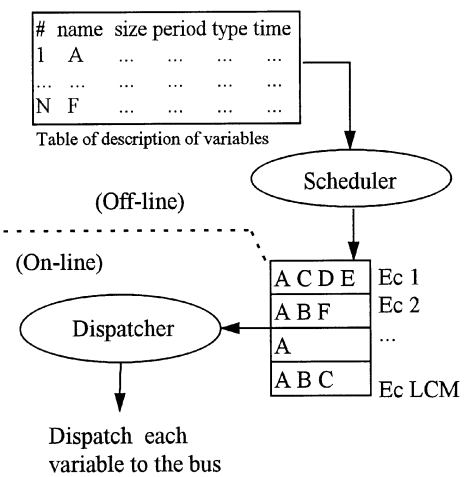

Fig. 2. Static scheduling approach. The expression *dispatch a variable* refers to the actual processing of the respective transaction on the bus.

2. Advantages and drawbacks of static scheduling

In FIP-like fieldbuses relying on static scheduling, the set of distributed variables to be scanned and broadcast must be known *a priori*. Normally this refers to periodic variables only, although sporadic ones can be admitted using appropriate techniques (Sprunt et al., 1989; Sha and Goodenough, 1990). Based on the sampling periods and data length of such variables, a network configuration program uses an off-line scheduler to create a static cyclic schedule table that describes the order in which the respective transactions must be initiated.

Such a schedule table is loaded into the BA's memory prior to system operation. At run time, a dispatcher just reads circularly from the schedule table the identification of the next variable to scan, and initiates the relevant transaction (Fig. 2). Each transaction is non-preemptable, and contains the whole distribution cycle of a variable, i.e. the transmission by the arbitrator of a frame with the identification of the variable to be scanned, and the transmission by the respective producer of the actual variable value.

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 the periods of all the variables (LCM(P_i)).

These systems also use a finite time resolution for the periodic sampling of variables. The minimum time allowed for the periods of the variables 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, which can be used for different transactions. The Mc contains an integer number of Ec's.

Some of the functional advantages of using FIP-like fieldbuses that rely on off-line static scheduling are listed below (Cardeira and Mammeri, 1995):

- *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 where all variables' time constraints are met, then the schedulability of the variable set is guaranteed for the system's entire 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 has been 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 main drawbacks are:

- *Operational inflexibility*. Perhaps the major drawback presented by the use of static scheduling in FIP-like fieldbuses is the typical inflexibility of such schedules. In fact, whenever a change in the variable set is required (e.g. when a new sensor is added, or a scanning period is modified) 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 can the system resume its activity.
- *Potentially huge schedule size*. Typical industrial plants may easily include over a hundred items of equipment, connected to the fieldbus. If the system operation requires the periodic sampling of over a hundred variables, some with sampling periods that can be long and relatively prime, then the least common multiple of the periods will be a huge number. So will the size of the schedule (macro-cycle), requiring very large amounts of the BA's local memory to hold it.

3. Using a dynamic scheduler

One of the possible ways to improve the flexibility of FIP-like fieldbuses is to use dynamic scheduling (Cardeira and Mammeri, 1995). Such a scheduler would determine only the next variable to be dispatched, scanning the whole variable set with a given criterion (Fig. 3). Also, it would need to be invoked before starting 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 required to hold the variables description table. This is 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 scheduler must be complemented with a schedulability analyser, which must be invoked whenever there is a change in the variable set. Only the use of the analyser allows for a guaranteed timely operation (Cheng et al., 1988; Klein et al., 1993; Stankovic 1995).

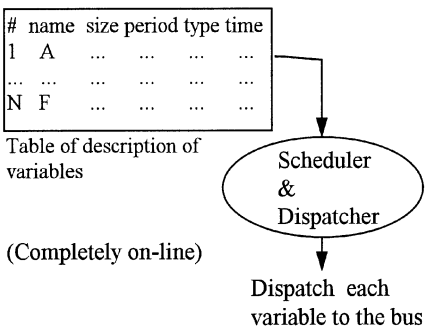

Fig. 3. The dynamic scheduling approach.

It is easily seen that the off-line static scheduler and the on-line dynamic scheduler are rather symmetrical. The planning scheduler introduced in the next section represents a good compromise between two such extremes (Pasadas et al., 1997).

4. The planning scheduler

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 into the future. Such a time window is fixed, and independent of the periods of the variables, and is called a plan.

The scheduler must, then, be invoked once in each plan to build a static schedule that will describe the bus allocation for the next plan (Fig. 4).

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 W, P_i being the period of variable i among a set of N variables, there are at most S transactions as given by expression (1). The time complexity of the calculation of S is $O(N)$:

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

If a dynamic scheduler is used, then it is invoked immediately upon the termination of any transaction, to determine the next transaction to be initiated. 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 $N*S$, with a time complexity of $O(N^2)$.

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

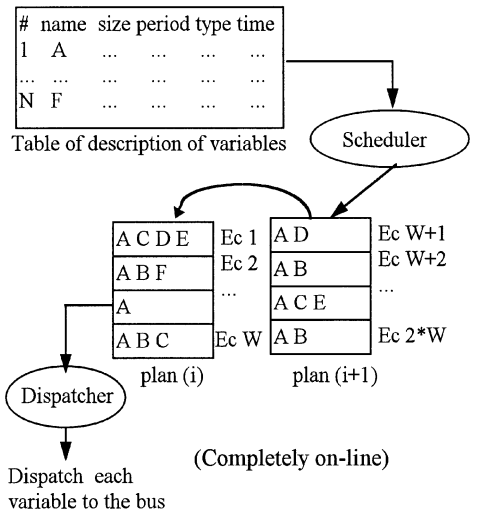

Fig. 4. Using the planning scheduler.

bus-time slots, one for each transaction, up to the end of the period W. The total number of operations required is now of the order of S, although in certain circumstances it may be higher (such as when many variables always occur simultaneously).

The main characteristics of the planning scheduler are the following:

- *Higher flexibility compared to the static approach.* Since the plans are re-done for 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 one to spread the overhead introduced by the scheduler activity, over the duration of the plan. The dispatching of the variables is similar to that performed in FIP, and incurs a 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. 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 every time there is a change in the variable set.

Notice that the properties of the planning scheduler depend on the plan duration (W). If $W \to 0$ 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 update because all the 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 summarises the properties of these three types of schedulers in terms of operational flexibility, run time overhead, schedule size and schedulability guarantee.

Table 1
Comparison between the three mentioned types of scheduler

Scheduler	Static	Dynamic	Planning
Operational flexibility	Low	High	Medium[a]
Run time overhead	Low	High	Medium[a]
Schedule size	Potentially large	Small	Bounded[a]
Schedulability	Guaranteed	Not guaranteed	Guaranteed each plan

[a]Depends on the plan duration.

5. 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 whether a given variable set can be successfully scheduled indefinitely. In dynamic systems this analysis has to be performed every time there is a change in the variable set, i.e., on-line. Therefore, the computational cost incurred by the analyser must be as low as possible, particularly when low-processing-power microcontrollers are to be used.

The scheduling problem under consideration in this paper is the scheduling of non-interruptible, independent, periodic transactions over a single bus. This problem is equivalent to the non-preemptive scheduling of periodic and independent tasks over a single CPU (Cardeira and Mammeri, 1995).

For a non-preemptive scheduling problem such as this one, some common methods to analyse schedulability are: completion times test (Klein et al., 1993), branch-and-bound search (Baker, 1974), heuristic search (Zhao et al., 1987) and decomposition (Yuan et al., 1994). However, even the fastest of these methods requires a reasonable amount of processing power from the CPU.

To really attain a fast schedulability assessment it is also possible to use a simple *sufficient* condition. However, the use of a sufficient but not necessary condition may decrease the efficiency of the analyser. Certain sets of variables may not meet the schedulability condition, and

yet be schedulable. Nevertheless, due to the very low overhead incurred by the use of such conditions, their applicability to on-line analysis is very common (Stankovic, 1995; Klein et al., 1993). The drawback to its use is normally lower utilization. However, this might be a reasonable trade-off if the schedulability guarantee in a dynamic environment is essential.

Perhaps the best-known sufficient schedulability condition is the one derived by Liu and Layland (1973) for Rate-Monotonic Scheduling. Such a condition establishes a lower utilization bound, under which the schedulability of a given task set is guaranteed for any phasing. The assessment of this condition can be performed on the fly, since it contains only one simple expression.

However, the work of Liu and Layland has been applied to preemptive scheduling, only. Since preemptivity is not allowed in the scheduling problem under consideration, Liu and Layland's condition must be properly adapted (Almeida and Fonseca, 1997).

In order to achieve such an adaptation, some implementation details have to be characterized. Firstly, as mentioned in Section 2, all the variables' periods are multiples of a minimum time unit called an elementary cycle. During an Ec several transactions may be initiated. Secondly, to reduce the jitter of the variables with the shortest periods, a technique called *idle time insertion* is used, as explained later in this section.

Using the planning scheduler, where the duration of the plan also contains an integer number of Ec's, the allocation of bus-time slots is performed as depicted in Fig. 5. Starting with the variable that has the shortest period (rate-monotonic ordering) the scheduler allocates, up to the end of each 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. Then, the Ec is said to be *overloaded*, and the transaction is delayed to the next Ec. If the next Ec is also overloaded, this process is repeated until an Ec with enough free time is found.

Delaying transactions from one Ec to another causes the following two consequences (Fig. 6):

- *A bounded waste of bus-time in an overloaded Ec* (*idle time insertion*). Since the duration of the transactions that fitted within that Ec is less than or equal to the Ec's duration (E) there might be a small amount of time at the end, which will be wasted (X).
- *Avoidance of potential preemption instants.* Since no transaction is allowed to cross any Ec boundary, all possible instances of preemption (were it allowed) are eliminated. Thus, it becomes irrelevant whether or not the scheduler is preemptive.

Both consequences lead directly to the following schedulability condition. If N is the number of variables

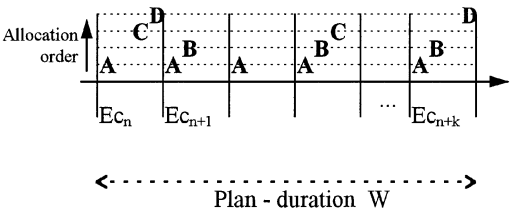

Fig. 5. Allocating bus-time slots in the plan schedule. Letters *A*, *B*, *C* and *D* refer to variables to be scanned.

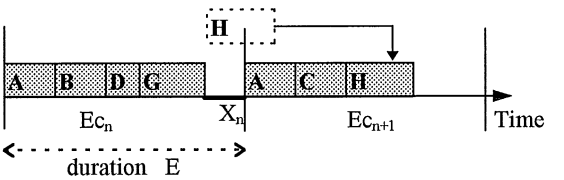

Fig. 6. Inserting idle-time to delay a transaction due to Ec overloading. Letters *A*, *B*, *C*, *D*, *G* and *H* refer to variables being broadcast.

in the set, P_i the period of variable i, and C_i the duration of the corresponding transaction, then condition (2) results. Notice that E equals the duration of any Ec, and X' is the maximum inserted idle-time, $\max(X_n)$ (see Fig. 6). If

$$U = \sum_{i=1}^{N} \frac{C_i}{P_i} < N(2^{1/N} - 1) \times \frac{E - X'}{E} \tag{2}$$

$\Rightarrow$ the set of N variables is schedulable with any phasing.

When $\max(X_n)$ is not easily determined it is possible to use its worst-case upper bound as

$$X' = \max_{allEc_n}(X_n) \leqslant \max_{i=1,N}(C_i) \tag{3}$$

The proof of condition (2) can be easily found, starting from the former consequence of delaying transactions from one Ec to the next: the waste of some bus-time. In a pessimistic approach it can be considered that all Ecs will be overloaded, and thus that the waste of bus-time (X_i) will happen with a period of E. This results in a higher bound to the wasted utilization of bus time, with a value of X'/E. Such a bound is pessimistic because it is impossible to attain permanent Ec overloading as this would imply the non-schedulability of the variable set. Also, the inserted idle-time (X_i) in an overloaded Ec will normally be less than $\max(C_i)$. Although pessimistic, this simple approach is sufficient for the purpose of demonstrating condition (2).

Now, the transformation in expression (4) can be applied to the initial variable set (V) while keeping its phasing, resulting in a new variable set (V'):

$$\begin{aligned} &V \equiv \{v_i(P_i, C_i)\} \\ &V \mapsto V' \equiv \left\{ v_i'(P_i', C_i'):\ P_i' = P_i \wedge C_i' = C_i \times \frac{E}{E - X'} \right\} \end{aligned} \tag{4}$$

Hypothetically, the new set V' can be scheduled according to the rate monotonic (RM) criterion, and preemptively. In this case the schedulability condition of Liu and Layland (1973) can be used, and 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 planning scheduler.

Therefore, the schedulability of set V' under normal RM (5) implies the schedulability of set V under the planning scheduler (6), thus proving condition (2):

$$U' = \sum_{i=1}^{N} \frac{C'_i}{P'_i} = U \times \frac{E}{E - X'} < N(2^{1/N} - 1) \quad (5)$$

$$\Downarrow$$

$$U < N(2^{1/N} - 1) \times \frac{E - X'}{E} \quad (6)$$

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 made to improve this bound slightly, using a deeper analysis of the wasted 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 be lower, from near 69% (the bound for normal RM scheduling with large N) to approximately 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 (Cardeira and Mammeri, 1995).

Such a low bound may not represent a severe problem, since the remaining bus utilisation can still be used for non-real-time sporadic traffic. This is done in the FIP fieldbus, and it is common practice in many other real-time systems.

Also, many practical situations use relatively short transactions (2–16 bytes of data). In this case, the longest transaction will probably be relatively small compared to the Ec duration (less than 10%), and thus the utilization bound will further approach the bound for rate-monotonic preemptive scheduling.

As an example, for sets with a large number of variables (large N), if the FIP fieldbus is used with a transmission rate of 1 Mb/s, a turn-around time of 20 μs and an elementary cycle of 5 ms, limiting each transaction to carry at most 32 bytes (duration = 418 μs) reduces the bus utilization bound to 64% for guaranteed schedulability. If the data size were limited to 8 bytes, which is enough for many practical situations, the longest transaction would take 221 μs, and the utilization bound would just decrease from 69 to 67%.

Even in a very unfavourable situation, using an RS485 bus with a transmission rate of 9600 baud, a turn-around time of 2 ms and an elementary cycle of 54.9 ms, with the data size limited to 4 bytes (longest transaction taking about 16 ms), the utilization bound for guaranteed schedulability drops from 69 to 50%. Although this is a relatively low value for bus utilization, it is still acceptable for many practical situations.

6. Implementation and tests

An experimental setup was built to test the functionality of the proposed scheduler. The physical layer was based on a low-cost RS-485 bus, operating at a low transmission rate of 9600 baud. Three PCs were used as nodes, one of which was fully dedicated to implement the bus arbitrator (BA). The BA implementation is depicted in Fig. 7.

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

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 value.

The duration of the corresponding transaction for each variable was calculated by adding the time to transmit one ID frame, the time to transmit the respective DATA frame, plus twice the turnaround time, which in this system was set to 1 ms.

A set of five variables was used, as described in Table 2. For this variable set the macro-cycle schedule has

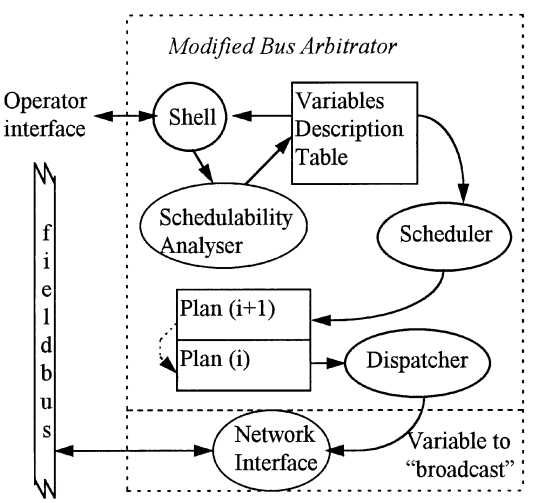

Fig. 7. The modified Bus Arbitrator.

Table 2
Table of descriptions of variable

Id	Period (No. of Ec's)	Size (bytes)	Trans. (ms)[a]
A	1	4	16.6
B	3	4	16.6
C	4	4	16.6
D	4	4	16.6
E	4	4	16.6

[a]Duration of the corresponding transaction.

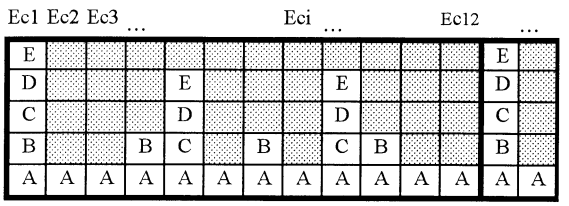

Fig. 8. The macro-cycle schedule.

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 Fig. 8. Notice that some Ec's (e.g. Ec_1 or Ec_5) are overloaded because the duration of the transactions overflows the Ec duration.

Fig. 9 shows the first two consecutive plans built by the planning scheduler. Notice that in this case the scheduler takes into account the finite duration 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 time left. This search is performed with rate monotonic priority.

In this example, the bus utilization required by this set of variables is 63.0%. The number of transactions that fit within an Ec is 3, with a waste of 5.1 ms in overloaded cycles (note that all variables are of the same size). In a worst-case approach, this corresponds to a wasted utilization of 9.3%. Applying the rate monotonic schedulability condition with $N = 5$ (74.3% utilization) over the non-wasted part of the Ec's (90.7%) results in a utilization threshold for guaranteed schedulability of 67.4%.

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

7. Main results

In Almeida et al. (1997) the authors have characterized the average-case behaviour of the planning scheduler. A theoretical model for the run-time overhead has been derived, and practical experiments have been carried out.

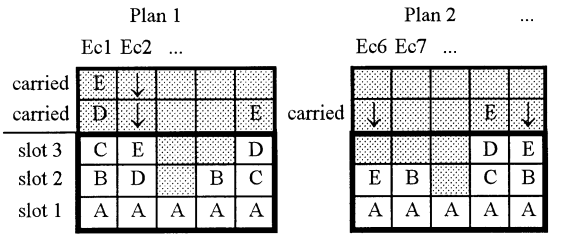

Fig. 9. First two plans (only three transactions fit within each Ec).

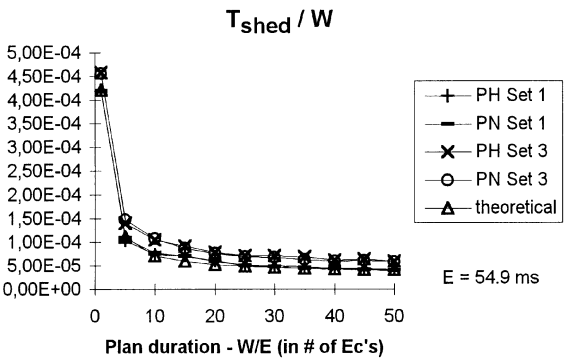

Fig. 10. The ratio scheduling time over plan duration for different durations.

One of the main results is the quantification of the reduction in run-time overhead obtained by using the planning instead of the dynamic scheduler. In fact, for four different sets of variables, described in the work referred to above, varying the plan duration from 1 to 20 Ec's allows a reduction of one order of magnitude in the ratio of the time taken by the scheduler over the plan duration (Fig. 10). This ratio is a measure of the run time overhead caused by the use of the planning scheduler.

Another important result is the practical determination of the time complexity of the scheduler, using sets with N variables and increasing N up to the situation in which at least one transaction misses its deadline. The observed time complexity of the scheduler is $O(N^2)$ in a worst-case situation, when all variables have identical periods and are in phase. If the variables have periods and phases such that their transmitting instants are not always coincident, then the time complexity can be reduced down to O(N). This value is in accordance with the reasoning explained in Section 4. The variation in time complexity is also explained by the model developed in the work referred to above.

8. Conclusion

This paper demonstrates that in real-time FIP-like fieldbuses relying on off-line cyclic static scheduling, the use of a planning scheduler in the Bus Arbitrator can

improve the operational flexibility of the network, while maintaining a relatively low run-time overhead. In such a fieldbus it is possible to reconfigure the system dynamically, namely to add or remove equipment, such as sensors or controllers, or to change the properties of existing entities such as the sampling period of variables, without needing to halt system operation.

In order to guarantee the schedulability of the periodic bus transactions dynamically, an on-line analyser based on the bus utilization has been developed. Such analyser uses a sufficient condition based on that of Liu and Layland (1973) for normal rate-monotonic scheduling (RMS). The resulting utilization bound for guaranteed schedulability is slightly lower than that for RMS.

The average-case behaviour of the planning scheduler has been characterized (Almeida et al., 1997), and the authors are currently developing experiments to test a model for the worst-case behaviour.

The authors are also involved in the development of a real-time distributed system based on a FIP-like fieldbus, with the planning scheduler, but using nodes with low-processing-power microcontrollers, such as the 8051. This will allow a more accurate testing of the planning scheduler with real workloads.

References

Almeida, L., & Fonseca, J.A. (1997). Schedulability analysis in a real-time fieldbus network. *Proc. SICICA* '97 (*Int. Symp. on Intelligent Components and Instruments for Control Applications*), Annecy, France.

Almeida, L. et al. (1997). Using the planning scheduler in real-time fieldbuses: theoretical model for run-time overhead. *Proc. of WFCS* '97 (*IEEE Int. Workshop on Factory Communication Systems*), Barcelona, Spain.

Baker, K.R. (1974). *Introduction to sequencing and scheduling*. New York: Wiley.

Cardeira, C., & Mammeri, Z. (1995). 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.

Cheng, S.C., Stankovic, J.A., & Ramamritham, K. (1988). Scheduling algorithms for hard real-time systems – a brief survey. In *Hard real-time systems tutorial*. Silver spring, MD: *IEEE Computer Society Press*.

Jordan, J.R. (1995). *Serial networked field instrumentation*. England: Wiley.

Klein, M. et al. (1993). *A practitioner's handbook for real-time analysis: guide to rate-monotonic analysis for real-time systems*. The Netherlands: Kluwer Academic Publishers.

Leterrier, P. (1992). *The FIP Protocol*. WorldFip Europe, 2–4 Rue de Bône, 92160 Antony, France.

Liu, C.L., & Layland, J.W. (1973). Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM*, *20* (1), 46–61.

Pasadas, R., Almeida, L., & Fonseca, J.A. (1997). A proposal to improve flexibility in real-time fieldbus networks. *Proc. SICICA*' 97 (*Int. Symp. on Intelligent Components and Instruments for Control Applications*), Annecy, France.

Sha, L., & Goodenough, J.B. (1990). Real-time scheduling and Ada. *IEEE Comput. 23*, (4).

Sprunt B. et. al. (1989). Aperiodic task scheduling for hard-real-time systems. *J. Real-Time Systems*, *1*, 27–60.

Stankovic, J.A. (1995). Implications of classical scheduling results for real-time systems. *IEEE Comput. 28* (6).

Stankovic, J.A. (1996). Strategic directions in real-time and embedded systems. *ACM Computing Surveys*, *28* (4), 751–763.

Thomesse, J.P. (1994). Les réseaux temps-réel. *Ecole d'été: Reseaux de communication et techniques formelles*. Ecole Nacional Superior de Telecommunication, Paris.

Thomesse, J.P. (1997). The fieldbuses. *Proc. SICICA* '97 (*Int. Symp. on Intelligent Components and Instruments for Control Applications*), Annecy, France.

Yuan, X., Saksena, M., & Agrawala, A. (1994). A decomposition approach to non-preemptive real-time scheduling. *J. Real-Time Systems*, *6*, 7–35.

Zhao, W., et al. (1987). Preemptive scheduling under time and resource constraints. *IEEE Trans. Comput*., *36* (8).