SETTING TARGET ROTATION TIME IN PROFIBUS BASED REAL-TIME DISTRIBUTED APPLICATIONS[1]

Eduardo Tovar[‡], Francisco Vasques[†]

[‡] *Polytechnic Institute of Porto, ISEP, Rua de São Tomé, 4200 Porto, Portugal,*
Tel.: +351.2.8340500, Fax.: +351.2.821159, E-mail: emt@dei.isep.ipp.pt
[†] *University of Porto, FEUP, Rua dos Bragas, 4099 Porto Codex, Portugal*
Tel.: +351.2.2041774, Fax.: +351.2.2059278, E-mail: vasques@fe.up.pt

Abstract: In this paper, we analyse the ability of Profibus fieldbus to cope with the real-time requirements of a Distributed Computer Control System (DCCS), where messages associated to discrete events must be made available within a maximum bound time. Our methodology is based on the knowledge of real-time traffic characteristics, setting the network parameters in order to cope with timing requirements. Since non-real-time traffic characteristics are usually unknown at the design stage, we consider an operational profile where, constraining non-real-time traffic at the application level, we assure that real-time requirements are met. *Copyright © 1998 IFAC*

Keywords: Fieldbus Networks, Real-time Communication, Distributed Computer Control Systems.

1. INTRODUCTION

Within industrial communication systems, fieldbus networks are specially devoted for the interconnection of process controllers, sensors and actuators, at the lower level of the factory automation hierarchy.

Among other characteristics, these hierarchical levels have dissimilar message flows. It is possible to classify (Prince, and Soloman, 1981) such flows, carried by the communication systems, according to:

- the required response time, that is, how quickly messages must be transferred;
- their length, that is, the amount of information to be transferred;
- the required reliability, which means, for instance, the importance of error-free or guaranteed delivery;
- the message rate, in other words, how frequently an application task sends a particular type of message, for instance, from a sensor to the process controller.

In a rough way, one can say that time constraints are more stringent as we go down in the automation hierarchy. In the context of this paper, we consider time constraints or deadlines, as the *maximum delay* between sending a request and receiving the related response at the application level. In other words, we are emphasising the association of *deadlines* to messages cycles (request followed by response at the application level).

The message cycle delay is made up of multiple factors, such as transmission time (frame length / transmission rate), protocol processing time and propagation, access or queuing delays. As we are dealing with real-time communication across a shared transmission medium, the most relevant factors to our analysis are the access and queuing

[1] This work was partially supported by FLAD under the project SISTER 471/97 and by ISEP under the project REMETER.

delays, which heavily depend on the Medium Access Control mechanism.

Different approaches for the Medium Access Control mechanism have been adopted by fieldbus communication systems. As relevant examples, we can mention the timed token protocol in Profibus (EN 50170, 1996b), the centralised polling in FIP (EN 50170, 1996c), the virtual token passing in P-NET (EN 50170, 1996a) and the CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) in CAN (SAE J1583, 1992).

Recently, several studies on the ability of fieldbus networks to cope with real-time requirements have been presented, such as (Tindell, *et al*., 1994; Zuberi and Shin, 1997) on CAN, (Pedro and Burns, 1997; Vasques 1996) on FIP, (Tovar and Vasques, 1998a) on P-NET and finally (Tovar and Vasques, 1998b) on Profibus.

In this paper, a methodology to guarantee the real-time requirements of Profibus based distributed applications is described. The methodology is based on the knowledge of the real-time traffic requirements, allocating enough bandwidth to each network node in order to cope with it. However, since non-real-time traffic requirements are usually not known at the design stage, we consider an operational profile where, by constraining non-real-time traffic, we ensure that real-time requirements are met.

2. ESSENTIALS ON PROFIBUS PROTOCOL

2.1. *General Characteristics*

The Profibus MAC includes a token passing procedure used by master stations to communicate between each other, and a master-slave procedure used to communicate with slave stations (or generally to communicate with stations not holding the token). Figure 1 illustrates this hybrid-operating mode.

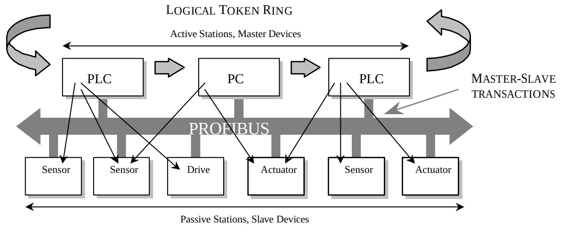
LOGICAL TOKEN RING
Active Stations, Master Devices
PLC
PC
PLC
MASTER-SLAVE TRANSACTIONS
PROFIBUS
Sensor
Sensor
Drive
Actuator
Sensor
Actuator
Passive Stations, Slave Devices

Fig. 1. Profibus MAC Hybrid Operating Mode.

The MAC protocol, implemented at the layer 2 of the OSI reference model, is called Field bus Data Link (FDL). In addition to controlling the token cycle time and the bus access, the FDL is also responsible for the provision of data transmission services for the FDL user (user of the link layer, e. g. the application layer). Profibus supports four basic non-cyclic data transmission services: Send Data with No acknowledge (SDN); Send Data with Acknowledge (SDA); Request Data with Reply (RDR) and Send and Request Data (SRD).

The SDN is an unacknowledged service mainly used for broadcasts from an active station to all the other bus stations. Conversely, all the other services are based on a real dual relationship between the initiator (master station holding the token) and the responder (passive or active station not holding the token). Another important characteristic of these services is that they must be immediately answered, with an acknowledge or a response. This feature, which is particularly important for the real-time bus operation, is also called “immediate-response”.

In addition to these non-cyclical services, industrial applications often require the use of cyclical transmission methods. A centrally controlled polling method (cyclical polling) is a suitable transmission method to scan basic field devices, such as sensors or actuators. Profibus enables a polling list to be created in the FDL layer, and can thus carry out a cyclical polling based on the non-cyclical RDR and SDR services.

An important Profibus concept is the message cycle. A message cycle consists of a master station’s action frame (request or send/request frame) and the associated acknowledgement or response frame. User data may be transmitted in the action frame (send) as well as in the response frame (response).

All stations, except the token holder (initiator) shall in general monitor all requests. The acknowledgement or response shall arrive within a predefined time, the slot time, otherwise the initiator repeats the request. The initiator shall not issue a retry or a new request before the expiration of a waiting period (Idle Time).

All the real-time properties of the Profibus protocol presented in this paper are based on the knowledge of the messages cycle time length. This time includes the time needed to issue the action frame and receive the associated response and also should include possible message retries.

2.2. *Behaviour of the Access Control*

After receiving the token, the measurement of the token rotation time begins. This measurement expires at the next token arrival and results in the real token rotation time (T_{RR}). T_{RR} is of significance for carrying out non-high priority message cycles. In order to keep the system reaction time, the token target rotation time (T_{TR}) must be set at the start-up.

Independently of the real token rotation time (T_{RR}), each master station may always execute one high priority message cycle per token arrival. In order to perform non-high priority message cycles, T_{RR} must be lower than T_{TR} at the execution instant, otherwise the station retains non-high priority message cycles and only transmits them at the following token arrival. Once a message cycle is started it is always completed, including any required retries, even if T_{RR} reaches or exceeds the value of T_{TR} during the execution.

Apart from distinguishing high and low priority message cycles, the Profibus MAC differentiates three subtypes of low priority message cycles: poll list, non-cyclic low priority (application layer and remote management services) and GAP list message cycles. The GAP is the address range between two consecutive master stations addresses. Each master must periodically check the GAP addresses for supporting dynamic changes in the logical ring.

As a basis, when holding the token, a master station will successively handle:

1. high priority non-cyclic message cycles;

2.1 poll list message cycles;

2.2 low priority non-cyclic message cycles;

2.3 GAP list management (logical ring maintenance).

The Profibus standard specifies that these three subtypes underlain with the following rules. After receiving the token, the poll list is handled after all high priority messages have been carried out. If the poll cycle is completed within T_{TH}, the requested low priority messages are then carried out and a new poll cycle will start at the next token arrival with available T_{TH}. If a poll cycle takes several token visits, the poll list is processed in segments, without inserting requested low priority messages. Low priority message cycles are carried out at the end of a complete poll cycle. At most one GAP address is checked per token visit, interleaved with poll cycles and non-cyclical low priority messages.

Figure 2 synthesises the Profibus MAC priority mechanism, where *pl_len* stands for the poll list length.

3. TIMING ANALYSIS ASSUMPTIONS

In this section, we provide the basis for the proposed Profibus real-time analysis.

3.1. Network and Message Models

We consider a bus topology with n master stations. A special frame (the token) circulates around the logical ring formed by the masters (from node k to nodes k + 1, k + 2, … until node n, then to nodes 1, 2, …). We denote the logical ring latency (token walk time, including node latency delay, media propagation delay, etc) as τ.

Message cycles generated at run-time may be classified as either high priority or low priority messages. To each k master node we assume that there are $nh^{(k)}$ high priority and $nl^{(k)}$ low priority messages streams. A message stream corresponds to a sequence of message cycles related with, for example, the reading of a process variable.

We denote the i^{th} high priority message stream associated to a master node k as $Sh_i^{(k)}$. Similarly low priority message streams are denoted as $Sl_j^{(k)}$.

A high priority message stream Sh_i is characterised as $Sh_I = (Ch_i, Dh_i)$. Ch_i is the maximum amount of time required to transmit a message in stream i. In Profibus this time should include the message cycle duration and also all possible messages retries. Dh_i is the messages relative deadline, which is the maximum amount of time that may elapse between a message arrival and the completion of its transmission. We consider that, in the worst case, the deadline can be seen as the minimum inter-arrival time between two consecutive messages in the same stream.

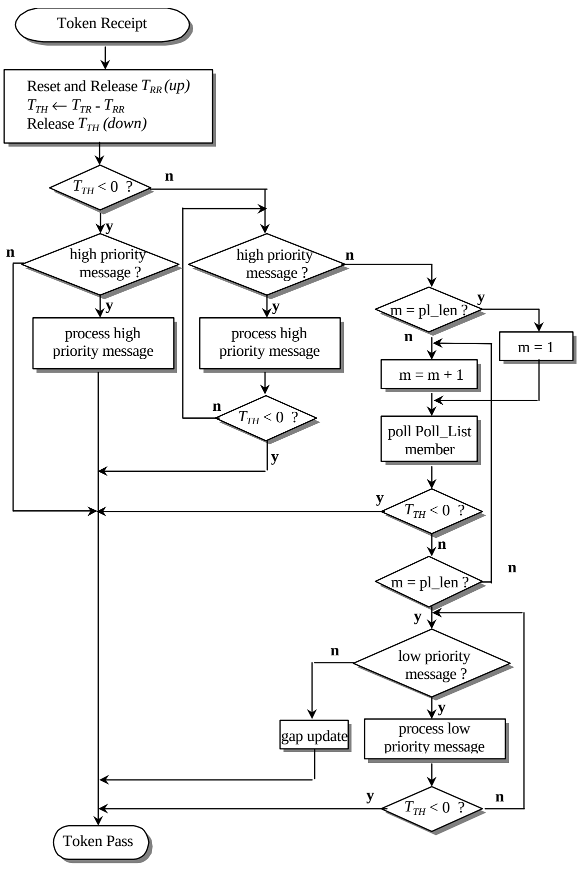

Fig. 2. Profibus MAC Priority Mechanisms.

The following high priority message stream notation will be used:

$$Sh_i^{(k)} = \left(Ch_i^{(k)}, Dh_i^{(k)}\right) \quad (1)$$

As for low priority message streams, we assume that only one message stream per station *k*, grouping all non-real-time traffic issued by the station, is characterised as $Sl^{(k)} = (Cl^{(k)}, nlp^{(k)})$. $Cl^{(k)}$ is the maximum amount of time required to transmit low priority messages in station *k*. $nlp^{(k)}$ is the maximum number of non-high priority message cycles that a station *k* is allowed to perform at each token visit.

The following low priority message stream notation will be used:

$$Sl^{(k)} = \left(Cl^{(k)}, nlp^{(k)}\right) \quad (2)$$

3.2. Timing Analysis Approach

Real-time approaches for timed token based protocols, such as for FDDI (Agrawal, *et al.*, 1994; Zheng and Shin, 1995) or for IEEE802.4 Token Bus (Montuschi, *et al.*, 1992), rely on the possibility of allocating specific bandwidth for real-time traffic. This means that a minimum amount of time is always available, at each token visit, to transmit real-time messages. The above-referred solutions formulate allocation schemes according to real-time message requirements. Conversely, in the Profibus protocol it is not possible to allocate real-time bandwidth to stations.

As a consequence, and considering that real-time traffic is supported by high priority message cycles, the non-real-time traffic may drastically affect Profibus real-time capabilities. In fact, if a station receives an early token (T_{TR} -T_{RR} > 0) and uses the available time to transmit non-high-priority message cycles, the subsequent stations may receive a late token (see Profibus token passing algorithm in §2.2).

The proposed Profibus timing analysis is based on a *constrained low priority traffic profile* where, by controlling low priority traffic, for instance at the application level, it is possible to guarantee the high priority traffic requirements. This analysis provides a pre-run-time schedulability condition, which is the basis for setting the Target Rotation Time (T_{TR}) parameter.

Another approach, based on an *unconstrained low priority traffic profile*, that is considering that in the worst case only one high priority message cycle is executed per token arrival, has also been proposed in (Tovar and Vasques, 1998b). Such kind of approach, leading to smaller T_{TR} parameters, is intended to support message streams with shorter deadlines. However, reducing the T_{TR} parameter value will also reduce the network ability to support low-priority traffic, which may lead to an undesirable operation mode on DCCS applications.

4. PROFIBUS TIMING ANALYSIS

The proposed Profibus timing analysis is based on the following assumption:

- at each token arrival, any station must be able to execute, at least, all pending high-priority message cycles.

The proposed analysis is based on the evaluation of a Deadline Constraint and the associated maximum bound of the T_{TR} parameter. In this context, the Deadline Constraint is defined as the condition that must be satisfied in order to guarantee that all real-time messages are transmitted before their deadlines, i.e., before the end of the minimum inter arrival time between two consecutive message arrivals.

4.1. Deadline Constraint

Considering that all real-time pending messages are to be sent at each token arrival, a deadline constraint may be defined as:

$$\min_i \left\{ Dh_i^{(k)} \right\} \geq Tcycle^{(k)}, \forall_k \quad (3)$$

where $Tcycle^{(k)}$ stands for the maximum elapsed time between two consecutive token arrivals.

Considering that each station is able to transmit all the high-priority pending traffic, $Tcycle^{(k)}$ is bounded by:

$$Tcycle \leq \sum_{k=1}^{n} \sum_{i=1}^{nh^{(k)}} Ch_i^{(k)} + Cl \times \sum_{k=1}^{n} nlp^{(k)} + t \quad (4)$$

which is equal for all the stations since it corresponds to the maximum allowed amount of traffic transferred by all the stations.

Expressions (3) and (4) can be re-written as:

$$\min_{i,k} \left\{ Dh_i^{(k)} \right\} \geq \sum_{k=1}^{n} \sum_{i=1}^{nh^{(k)}} Ch_i^{(k)} + Cl \times \sum_{k=1}^{n} nlp^{(k)} + t \quad (5)$$

which means that *Tcycle* must be smaller than the smallest real-time message deadline.

4.2. Setting the Target Rotation Time ($\mathrm{T_{TR}}$*)*

Considering that when the token arrives at a station *k*, this station must still have enough time to transfer, at least, its real-time traffic (as at the token arrival, the token holding time is T_{TR}-T_{RR}), a minimum bound for T_{TR} is:

$$T_{TR} \geq Tcycle + \max_{k=1..n}\left\{\sum_{i=1}^{nh^{(k)}} Ch_i^{(k)}\right\} \qquad (6)$$

since *Tcycle* is a maximum bound of T_{RR}. We may now rewrite (6) as,

$$Tcycle \leq T_{TR} - \max_{k=1..n}\left\{\sum_{i=1}^{nh^{(k)}} Ch_i^{(k)}\right\} \qquad (7)$$

which is clearly shown at figure 3 (a scenario with $nh^{(1)}$=2, $nh^{(2)}$=2, $nh^{(3)}$=2, $nlp^{(1)}$=3, $nlp^{(2)}$=1 and $nlp^{(3)}$=4).

Combining (4) and (6), a minimum bound for T_{TR} may be given by,

$$T_{TR} \geq \sum_{k=1}^{n}\sum_{i=1}^{nh^{(k)}} Ch_i^{(k)} + Cl \times \sum_{k=1}^{n} nlp^{(k)} + t + \max_{k=1..n}\left\{\sum_{i=1}^{nh^{(k)}} Ch_i^{(k)}\right\} \qquad (8)$$

These two expressions (5) and (8) are the basis of the proposed methodology to set the Target Rotation Time (T_{TR}) in Profibus based applications, in order to guarantee its real-time requirements.

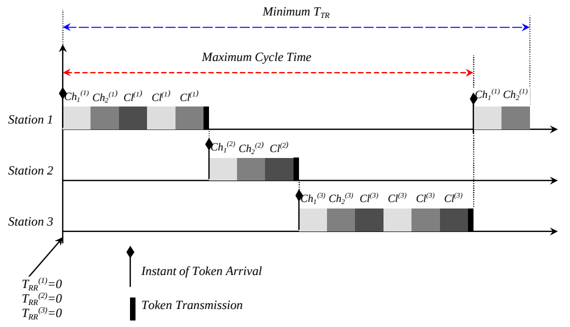

Fig. 3. Determining the PROFIBUS T_{TR} Parameter.

5. SOME IMPLEMENTATION CONSIDERATIONS

In this section, we give some guidelines concerning implementation issues. We propose two different alternatives:

1. The first one, based on a Profibus protocol modification, intends to control the number of transferred low priority messages at the MAC level.
2. The second one, based on the application level control of low priority services, such as application layer non-cyclical low priority services, remote management services and a specific remote management service (Live List service).

5.1. Controlling Low Priority Traffic at the MAC Level

We define at each station *k* the maximum number of low priority messages to be transferred ($nlp^{(k)}$), per token arrival. The low priority traffic is then controlled by means of a low priority messages counter (*nlp_c*). Figure 4 illustrates the proposed protocol modification. Please refer to figure 2 for the original protocol, where *p_len* stands for the poll list length.

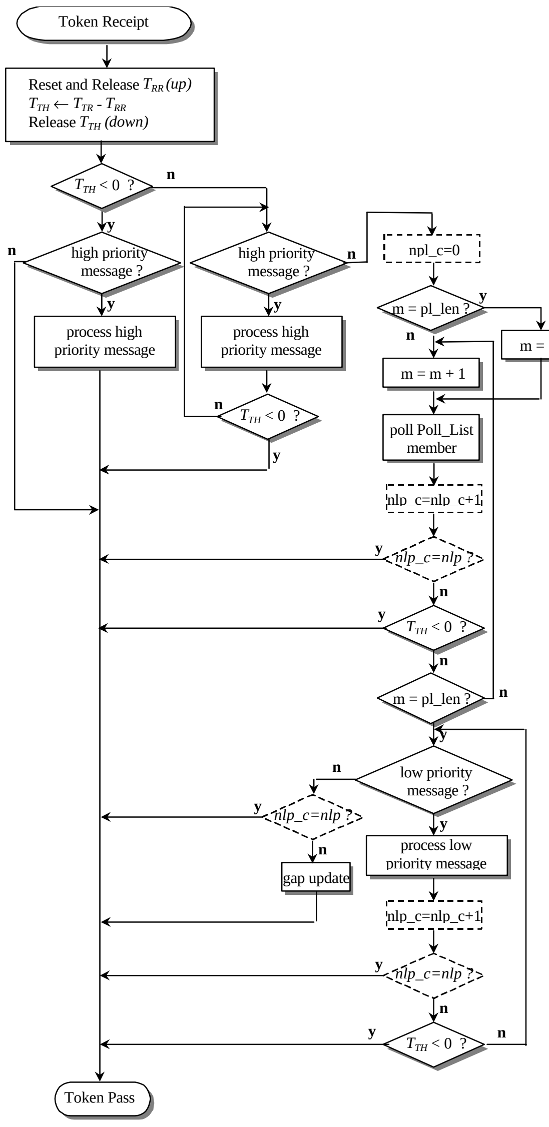

Fig. 4. Proposed Modifications of the Profibus MAC Priority Mechanisms, at the MAC level.

5.2. Controlling Low Priority Traffic at the Application Layer Level

Concerning the user explicitly generated traffic, we opt for not supporting the live list management service. The Live List service requests the FDL status of all stations (masters and slaves) and thus will generate multiple frames in the network. If, in the worst case, every master station requests a live list, the expression for *Tcycle* should then be written as follows:

$$T_{cycle} \le \sum_{k=1}^{n} \sum_{i=1}^{nh^{(k)}} Ch_i^{(k)} + \sum_{k=1}^{n} \left(nlp^{(k)} \times Cl^{(k)} \right) + t + n \times \sum_{k=1}^{n+s} C_{live} \quad (9)$$

where C_{live} stands for a request status message cycle length and $n+s$ corresponds to the sum of masters and slave stations.

Concerning low priority non-cyclical services, the application process software must be able to accept a T_{cycle} parameter, in order to control the number of low-priority messages ($nlp^{(k)}$) generated in the station.

The expression for *Tcycle* must include the influence of the GAP updating. We denote C_{gap} as the length of a GAP maintenance message cycle. Expression (10) includes such influence, considering that, in the worst case, each station generates one GAP maintenance message at each token arrival.

In order to support Poll List, an additional term must be added. Notice that one should only consider small length Poll Lists. In fact, as the user is not able to control the Poll List messages schedule, the whole list length must be included in the T_{cycle} evaluation.

$$T_{cycle} \le \sum_{k=1}^{n} \sum_{i=1}^{nh^{(k)}} Ch_i^{(k)} + \sum_{k=1}^{n} \left(nlp^{(k)} \times Cl^{(k)} \right) + \\ + t + n \times C_{gap} + \sum_{k=1}^{n} C_{poll}^{(k)} \quad (10)$$

where $C_{poll}^{(k)}$ stands for the station k Poll List length.

6. CONCLUSIONS

In this paper we have provided a comprehensive study on how to use Profibus networks to support real-time communication.

We have derived an operational profile within which real-time behaviour is guaranteed constraining low priority traffic at the application layer.

The major contribution is to provide a methodology on how to set the Target Rotation Time parameter.

REFERENCES

Agrawal, G., B. Chen, W. Zhao and S. Davari (1994). Guaranteeing Synchronous Message Deadline with the Timed Token Medium Access Control Protocol. In: *IEEE Transactions on Computers*, **Vol. 43**, **No. 3**, pp. 327-339.

EN 50170 (1996a). General Purpose Field Communication System. **Vol. 1/3** (P-NET), CENELEC, Brussels.

EN 50170 (1996b). General Purpose Field Communication System. **Vol. 2/3** (PROFIBUS), CENELEC, Brussels.

EN 50170 (1996c). General Purpose Field Communication System. **Vol. 3/3** (FIP), CENELEC, Brussels.

Montuschi, P., L. Ciminiera and A. Valenzano (1992). Time Characteristics of IEE802.4 Token Bus Protocol. In: *IEE Proceedings*, **Vol. 139**, **No. 1**, pp. 81-87.

Pedro, P. and A. Burns (1997). Worst Case Response Time Analysis of Hard Real-Time Sporadic Traffic in FIP Networks. In: *Proceedings of 9th Euromicro Workshop on Real-time Systems*, IEEE Press, pp. 3-10.

Prince, S. M. and M. S. Soloman (1981). Communication Requirements of a Distributed Computer Control System. *IEE Proceedings*, **Vol. 128**, **No. 1**, pp. 21-34.

SAE J1583 (1992). Controller Area Network CAN, an In-Vehicle Serial Communication Protocol. SAE-Handbook, pp. 20341-20355.

Tindell, K., H. Hansson and A. Wellings (1994). Analysing Real-Time Communications: Controller Area Network (CAN). In: *Proceedings of the IEEE Real Time Systems Symposium (RTSS'94)*, IEEE Press, pp. 259-263.

Tovar, E, and F. Vasques (1998a). Enhancing P-NET Real-Time Properties Using Priority Queuing Mechanisms. In: *Proceedings of the 4th IEEE Real-Time Technologies and Applications Symposium, WIP Session, available as Technical Report from Boston University, Computer Science Department BUCS-TR-98-013*, pp. 27-30.

Tovar, E, and F. Vasques (1998b). Guaranteeing Real-Time Message Deadlines in Profibus Networks. In: *Proceedings of the 10th Euromicro Workshop on Real-time Systems*, IEEE Press, pp. 79-86.

Vasques, F. (1996). Sur L'Intégration de Mécanismes d'Ordonnancement et de Communication dans la Sous-Couche MAC de Réseaux Locaux Temps-Réel. *PhD Thesis, available as technical Report LAAS Nº 96229*, Toulouse, France.

Zheng, Q. and K. G. Shin (1995). Synchronous Bandwidth Allocation in FDDI Networks. In: *IEEE Transactions on Parallel and Distributed Systems*, **Vol. 6**, **No. 12**, pp. 1332-1338.

Zuberi, K. M. and K. G. Shin (1997). Scheduling Messages on Controller Area Network for Real-Time CIM Applications. In: *IEEE Transactions on Robotics and Automation.Proceedings*, **Vol. 15**, **No. 2**, pp. 310-314.