Submission to Computer Communications, Elsevier
January 1999

Cycle Time Properties of the PROFIBUS Timed Token Protocol*

Eduardo Tovar[1], Francisco Vasques[2]

[1] Department of Computer Engineering, Polytechnic Institute of Porto, Rua de São Tomé, 4200 Porto, Portugal; Tel.: +351.2.8340500; Fax: +351.2.8321159
emt@dei.isep.ipp.pt
[2] Department of Mechanical Engineering, University of Porto, Rua dos Bragas, 4099 Porto Codex, Portugal
vasques@fe.up.pt

Abstract. A recent trend in distributed computer-controlled systems (DCCS) is to interconnect the distributed elements by means of a multi-point broadcast network. As the network bus is shared between a number of network nodes, there is an access contention, which must be solved by the medium access control (MAC) protocol. Usually, DCCS impose real-time constraints. In essence, by real-time constraints we mean that traffic must be sent and received within a bounded interval, otherwise a timing fault is said to occur. This motivates the use of communication networks with MAC protocols that guarantee bounded access and response times to message requests. PROFIBUS is a fieldbus network with a MAC protocol based on a simplified version of the timed token protocol. In this paper, we analyse PROFIBUS MAC's cycle time properties, as they are crucial to guarantee a real-time behaviour of DCCS supported by this type of fieldbus networks.

Keywords. Real-Time Communication, Timed Token Protocol, Fieldbus Networks, PROFIBUS

1. Introduction

Local area networks (LANs) are becoming increasingly popular in industrial computer--controlled systems. LANs allow field devices like sensors, actuators and controllers to be interconnected at low cost, using less wiring and requiring less maintenance than point-to--point connections [1]. Besides the economical aspects, the use of LANs is also reinforced by the increasing decentralisation of control and measurement tasks, as well as by the use of intelligent microprocessor-controlled devices.

Broadcast LANs aimed at the interconnection of sensors, actuators and controllers are commonly known as fieldbus networks. In the past, the fieldbus scope was dominated by vendor specific solutions, which were mostly restricted to specific application areas. Moreover, concepts behind each proposed network were highly dependent on the manufacturer of the automation system, each one with different technical implementations and also claiming to fulfil different application requirements, or the same requirements with different technical solutions [2]. More recently, standardised fieldbuses supporting the open system concept, thus vendor independent, started to be commonly used. PROcess FIeld BUS

* This work was partially supported by ISEP under project REMETER and by FLAD under project SISTER (471/97).

(PROFIBUS) [3] is one of the most popular fieldbuses, and has recently been granted the status of a real international standard by CENELEC [4].

In this paper we address the ability of PROFIBUS to cope with the real-time requirements of distributed computer-controlled systems (DCCS). In essence, by timing requirements we mean that messages must be sent and received within a bounded interval, otherwise a timing fault is said to occur. That means, for instance, that a control device must be able to read data from a remote sensor within a specified interval, whichever the network load.

The PROFIBUS medium access control (MAC) protocol is based on a token passing procedure, used by master stations to grant the bus access to each one of them, and a master--slave procedure used by master stations to communicate with slave stations. The master--slave interaction is denoted as a message cycle: the master sends a request frame and the addressed slave immediately sends a response frame. If the token holding time for that master has been exceeded or the master has no more pending requests, the token is passed to the subsequent master. Typically, the process relevant devices (sensors and actuators) will be interconnected to the network via a slave network interface, whereas the distributed control algorithms will reside at master stations. Therefore, in PROFIBUS, the end-to-end communication delay [5] for the master-slave transactions (those that typically deal with real-time traffic) is composed of the following four major components:

1. *generation delay*: time elapsed between the release of the sender task and the queuing of the related message request;
2. *queuing delay*: time taken by a message request to access the communication medium after being queued;
3. *transmission delay*: time taken by a message request to be transmitted on the communication medium and processed at the slave side, added with the time taken by the message response to be transmitted back to the master;
4. *delivery delay*: time taken by the master's application task to process a message response.

The generation delay includes the application processing time needed to generate the contents of the message and the time taken to queue the message. This issue has been extensively addressed in the literature related with tasks' worst-case response time analysis in single-processor systems ([6,7] are just two examples).

The queuing delay is a consequence not only from the contention between message requests from the same master but also with message requests from other masters. The impact of the first factor in the overall queuing delay depends on the policy used to queue message requests, while the second factor depends on the behaviour of the token passing procedure. Therefore, the evaluation of the worst-case queuing delay of the message requests is paramount to guarantee the message timing requirements. The aggregate value for the queuing and transmission delays is termed in this paper as the message request response time.

In this paper we address the analysis of the worst-case message response time in PROFIBUS networks. As it will be shown, this analysis depends on the knowledge of the maximum time interval between two consecutive token arrivals to a master. Thus, the study of the cycle time properties of the PROFIBUS timed token protocol is fundamental to the guarantee the message timing requirements in PROFIBUS networks.

The remaining of this paper is organised as follows. In section 2 we give a methodology for the worst-case message response time analysis in PROFIBUS networks. This section presents an improved version of the analysis proposed in [8,9], which assumes the worst-case scenario for the token cycle time. In section 3 we survey some results on the evaluation of token cycle time in several network protocols based on the timed token protocol [10], as it is the case of IEEE802.4 [11] and FDDI [12]. As PROFIBUS uses a simplified version of the

timed token protocol, we discuss the applicability of the surveyed results to the PROFIBUS case. In section 4, we derive an accurate result for the PROFIBUS token cycle time, which is the basis for the evaluation of the worst-case message response time. Finally, in section 5 we show how the methodology and results provided in this paper can be used to set the target token rotation time (T_{TR}) parameter of a PROFIBUS network, in order to guarantee the real-time requirements of a DCCS application.

2. Real-Time Analysis for PROFIBUS Networks

2.1. A Brief Description of the PROFIBUS Timed Token Protocol

In PROFIBUS, messages are classified in the two following categories: high priority messages and low priority (including cyclic, non-cyclic and management) messages. In the proposed methodology, real-time traffic is supported using PROFIBUS high priority messages.

The PROFIBUS medium access control (MAC) protocol is, as previously mentioned, a simplified version of the timed token protocol [10], and behaves as is below explained.

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}), which is a measure of the token cycle time. A target token rotation time (T_{TR}) must be defined in a PROFIBUS network. The value of this parameter is common to all masters, and is used as follows. When a station receives the token, the token holding time (T_{TH}) timer is given the value corresponding to the difference, if positive, between T_{TR} and T_{RR}. If at the arrival, the token is late, that is, the real token rotation time (T_{RR}) was greater than the target rotation time (T_{TR}), the master station may execute, at most, one high priority message cycle. Otherwise, the master station may execute high priority message cycles while $T_{TH} > 0$. T_{TH} is always tested at the beginning of the message cycle execution. This means that once a message cycle is started it is always completed, including any required retries, even if T_{TH} expires during the execution. We denote this occurrence as a T_{TH} overrun. The low priority message cycles are executed if there are no high priority messages pending, and while $T_{TH} > 0$ (also evaluated at the start of the message cycle execution, thus leading to a possible T_{TH} overrun).

Below is a description of the PROFIBUS token passing algorithm:

```
/* initialisation procedure */
At each station k, DO:
T_TH ← 0 ;
T_RR ← 0 ;
Start T_RR ;          /* count-up timer */
/* run-time procedure */
At each station k, at the Token arrival, DO:
T_TH ← T_TR - T_RR ;
T_RR ← 0 ;
Start T_RR ; /* count-up timer */
IF T_TH > 0 THEN
          Start T_TH  /* count-down timer */
ENDIF;
IF waiting High priority messages THEN:
           Execute one High priority message cycle
ENDIF;
```

```
WHILE T_TH > 0 AND pending High priority message cycles DO
Execute High priority message cycles
ENDWHILE;
WHILE T_TH > 0 AND pending Low priority message cycles DO
        Execute Low priority message cycles
ENDWHILE;
Pass the token to station (k + 1) (modulo n);
```

Figure 1 illustrates a scenario where the i^{th} real token rotation time, as seen by master 4 (T^4_{RR}), corresponds to the time of the network token rotation (none of the stations used the token to transmit messages). At that i^{th} token visit, master station 4 uses part of its available token holding time (T^4_{TH}) to transmit two message cycles.

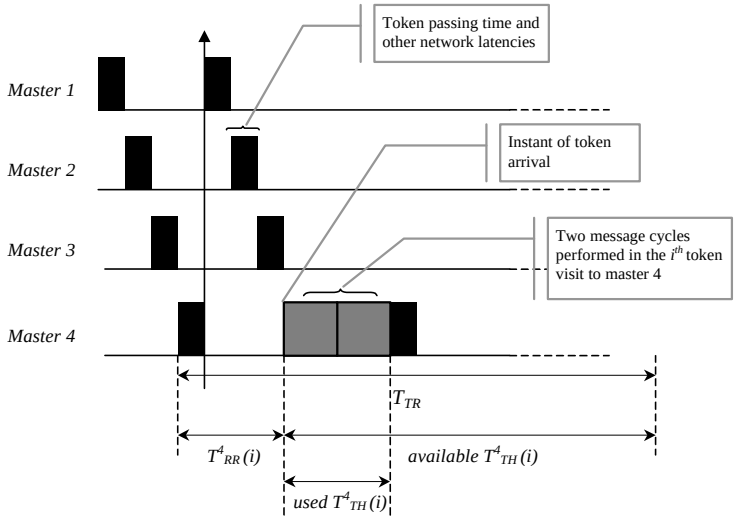

Fig. 1. Example of Token Usage in a Master

In PROFIBUS, a message cycle consists on a master's action frame (request or send/request frame) and the responder's immediate acknowledgement or response frame. User data may be transmitted in the action frame or in the response frame. Note however that in PROFIBUS a master is allowed to send up to a limited number of retries, if the response does not come within a predefined time. Thus, the message cycle time length must also include the time needed to process the allowed retries.

2.2. Worst-Case Response Time for High Priority Messages

We consider a bus topology with n PROFIBUS master stations, where a special frame (the token) circulates around the masters' logical ring. The logical ring latency (token walk time, including node latency delay, media propagation delay, etc.) is denoted as t.

We also consider nh^k high priority message streams in each master k. A message stream corresponds to a temporal sequence of message cycles related, for instance, with the reading of a process sensor or the updating of a process actuator.

We denote the i^{th} (i = 1, 2, …, nh^k) high priority stream associated to a master k as Sh_i^k. A high priority message cycle stream Sh_i^k is characterised as:

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

Ch_i^k is the maximum amount of time required to perform a Sh_i^k message. Dh_i^k is the message cycle relative deadline, which is the maximum admissible end-to-end communication delay. As we do not mean to guarantee deadlines for the low priority traffic, a low priority message stream Sl_i^k is merely characterised as:

$$Sl_i^k = \left(Cl_i^k\right) \quad (2)$$

where Cl_i^k is the maximum amount of time required to perform a Sl_i^k message cycle.

In the PROFIBUS MAC protocol, and as far as there are pending high priority messages, a master station is guaranteed to transmit, at least, one high priority message per received token (no matter if there is enough token holding time left). Assuming the worst-case scenario (token always arriving late), if there are m pending high priority messages, it will take m token visits to execute all those high priority message cycles. It is obvious that the queuing delay depends on the high priority outgoing queue policy, which in PROFIBUS is a First-Come-First-Served (FCFS) policy.

For the queuing delay analysis, it is important to note that the maximum number of pending messages will be nh^k, corresponding to one message per each Sh_i^k stream. Indeed, if at any time there are two pending message requests of the same stream, then a deadline for that message stream was missed.

It is now clear that, assuming that message deadlines are not missed (thus the maximum number of high priority pending messages is nh^k), the upper bound for the message queuing delay in a master k is:

$$Q^k = nh^k \times T_{cycle}^k \quad (3)$$

where T^k_{cycle} is the upper bound for the token inter-arrival time at a station k, hence it is the worst-case real token rotation time (T^k_{RR}). Theoretically, T_{TR} can be set to a value smaller than t. In the limit it can be set to 0. If this is the case, the token will always arrive late to a master, as T_{TR} will be at least t. Note also that under our assumptions the queuing delay for a message request in one station is independent of the message stream ($Q_i^k = Q^k$, $\forall_{i=1,..,\, nh}k$). For obvious reasons this does not applies for the definition of the worst-case message response time, which is:

$$R_i^k = Q^k + Ch_i^k = nh^k \times T_{cycle}^k + Ch_i^k \quad (4)$$

Defining the end-to-end communication delay as the aggregation of the g (generation delay), Q (queuing delay), C (transmission delay) and d (delivery delay) components, then, the deadline of a high priority message cycle is guaranteed if and only if the following condition is satisfied:

$$D_i^k \geq g_{Sh_i^k} + nh^k \times T_{cycle}^k + Ch_i^k + d_{Sh_i^k} \quad (5)$$

The main focus of this paper is on the evaluation of the T^k_{cycle} parameter, which in the context of this paper is defined as the worst-case time span between any two consecutive visits of the token to a PROFIBUS master.

3. Previous Relevant Work

The basic idea of the timed token protocol was presented by Grow [10]. In this protocol, messages are distinguished in two types. One concerns synchronous messages, which are

periodic messages that come to the system at regular intervals and have delivery time constraints. The other concerns asynchronous messages, which typically are non-periodic messages, and have no time constraints. Equivalence to PROFIBUS message types can be easily drawn: high priority and low priority are equivalent to, respectively, synchronous and asynchronous messages.

When a network is initialised, all the stations negotiate a common value for the T_{TR} parameter, which gives the expected token rotation time. The T_{TR} parameter should be chosen small enough to meet responsiveness requirements of all stations, i.e., the token must circulate fast enough to satisfy the most stringent timing requirements. Each station is assigned a fraction of T_{TR}, known as its synchronous capacity (H_i), which is the maximum time each station is allowed to transmit its synchronous messages, if any, every time it receives the token. The asynchronous messages can be transmitted, but only if the token has rotated fast enough, that is, the token is "ahead of schedule" with respect to its target rotation time.

In the timed token protocol, the time interval between two consecutive token arrivals at a specific station is upper bounded by $2\times T_{TR}$ and the average token rotation time is no more than T_{TR}. An intuitive explanation of these two timing properties can be found in [10] and a formal proof in [13].

In order to guarantee synchronous message deadlines, an upper bound to the token rotation time is a necessary but not sufficient condition. A node with inadequate synchronous capacity may be unable to guarantee message deadlines, and, on the other hand, allocating excess amount of synchronous capacities to the nodes increases T_{TR}, which may also cause message deadlines to be missed. Therefore, synchronous capacities must be properly allocated to individual nodes. As a consequence, synchronous capacities allocated to the nodes must satisfy two constraints [15,16]: a protocol constraint and a deadline constraint.

The protocol constraint states that the total sum of the allocated synchronous capacities should not be greater that the available portion of T_{TR}, i.e.,

$$\sum_{i=1}^{n} H_i \leq T_{TR} - t \qquad \textbf{(6)}$$

Theoretically, the total available time to transmit synchronous messages, during a complete token rotation, can be as much as T_{TR}. However, factors such as ring latency and other protocol or network overheads reduce the total available time and are denoted as t.

The deadline constraint states that the allocation of synchronous capacities to the nodes should be such that synchronous messages are always transmitted before their deadlines.

As a result, a message set can be guaranteed by an allocation scheme once the protocol and the deadline constraints are satisfied. Several allocation schemes have been proposed in the literature [17-19].

Both FDDI and IEEE802.4 are examples of network protocols based on the timed token protocol. Upper bounds for the time elapsed between two consecutive token arrivals can be found in [13] and [14]. These results cannot however be applied to PROFIBUS, as significant differences to the timed token protocol exist. We consider the following two as the most relevant ones:

1. In PROFIBUS there is no synchronous bandwidth allocation (H_i). If a station receives a late token (T_{RR} greater than T_{TR}), only one high priority message may be transmitted. Contrarily, in the timed token protocol the station can transmit synchronous (high priority) messages during H_i, even if the token is late.
2. In PROFIBUS, both high priority and low priority message cycles may overrun the T_{TH} timer. As previously stressed, in PROFIBUS a message cycle can be initiated

with a residual T_{TH} value and the message cycle will be performed until the end. In the timed token protocol this only happens with asynchronous (low priority) messages, as synchronous messages transmission can only be started if they fit within the time allocated for synchronous transmission.

Concerning the PROFIBUS protocol, in [20] the authors propose the use of the cyclic services to support real-time communication. In their approach these cyclic services support the polling of slaves and message deadlines are guaranteed since the token cycle time is bounded.

The major drawback of their approach is that, in order to evaluate the token cycle time, nor high priority traffic neither low priority traffic (other than cyclic traffic) are allowed. This prevents the transfer of event-driven messages of high priority, such as alarms. Furthermore, remote management services (which in PROFIBUS are mapped into low priority non-cyclic services) are also not covered by their approach.

In our approach, we propose the use of high priority services to support the real-time communication, instead of the cyclic low priority services. The major advantage is that the high priority traffic can be guaranteed, whichever the load of low priority messages.

In the following section, an accurate upper bound for the token cycle time in a PROFIBUS network supporting all types of traffic is given.

4. PROFIBUS Token Cycle Time Analysis

In PROFIBUS, the real token rotation time (T^k_{RR}) will always be smaller than T_{TR}, except when one or more masters in the logical ring induce the token to be late. Two reasons justify a late token at a master *k*:

1. As once a message cycle is started, it is always completed, even if T^k_{TH} has expired during its execution, a late token may be transmitted to the following stations. We define this occurrence as a T^k_{TH} overrun.
2. If a master receives a late token, it will still be able to send one high priority message, which may further increase the token lateness. This case is not considered to be a T^k_{TH} overrun since the T^k_{TH} timer is only released if the token arrives to the master in advance.

4.1. Analysis of Token Lateness

In this sub-section, we analyse causes and consequences of the token lateness. We will introduce and prove three theorems. Theorem 1 states that the token is never late unless a T^k_{TH} overrun occurs in one of the masters that form the logical token passing ring. Theorem 2 states that even if more than one master overruns its T^k_{TH} in a token cycle, only the last one (as seen from the master for which T_{RR} is being measured) will contribute to the token delay. Finally, theorem 3 states that, in a specific situation, all masters may contribute to the token lateness. These three theorems are the basis for the evaluation of T^k_{cycle} (which represents the T^k_{RR} upper bound), later on addressed in sub-section 4.2.

Theorem 1 *In PROFIBUS networks, if the master holding the token releases it before the T^k_{TH} expiration, then, the following master in the logical ring will receive an early token.*

Proof:

We denote $A^k(l)$ as the instant when the token arrives to the master k for its l^{th} visit, and $R^k(l)$ as the instant when master k releases the token that had just made its l^{th} visit to that master. If master k releases the token before the expiration of T^k_{TH} then, $R^k(l)–A^k(l–1)\leq T_{TR}$. Note that the real token rotation time is measured between token arrivals. If the successor of master k is denoted as k^{+1} (with $k^{+1}=(k+1)$ mod n), we want to prove that $A^{k+1}(l)–A^{k+1}(l–1)\leq T_{TR}$, that is, the successor of k will receive an early token. As $A^{k+1}(l–1)=R^k(l–1)+t/n$ (as t is the total logical ring latency, k^{+1} will receive the token t/n time after the release of the token in k) and $A^{k+1}(l)=R^k(l)+t/n$ then, combining these two expressions, it gives $A^{k+1}(l)–A^{k+1}(l–1)=R^k(l–1)–R^k(l)$. As $R^k(l–1)\geq A^k(l–1)$, and as $R^k(l)–A^k(l–1)\leq T_{TR}$ (starting assumption that there is no overrun of the T^k_{TH}), then $A^{k+1}(l)–A^{k+1}(l–1)\leq T_{TR}$ stands, i.e., the successor of k receives an early token.

❑

Figure 2 illustrates theorem 1, where master 2 releases the token before the expiration of $T^2_{TH}(l)$, and so master 3 receives an early token.

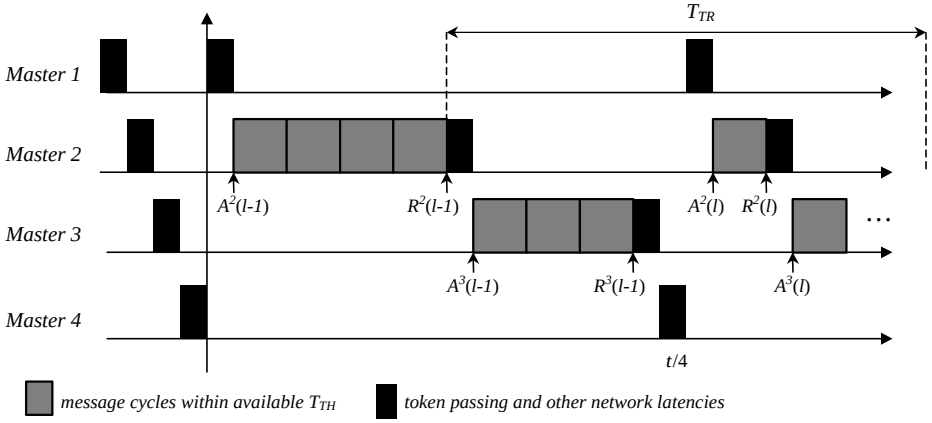

Fig. 2. Illustrative Example for Theorem 1

From theorem 1, two lemmas result.

Lemma 1.1: in PROFIBUS, master k receives an early token, if master k^{-1} releases it before the T^{k-1}_{TH} expiration, no matter if there were any T_{TH} overrun in masters k^{-2}, k^{-3}, …

Lemma 1.2: in PROFIBUS, if none of the k master stations overrun T^k_{TH}, the token will never be late.

Theorem 2 *In a PROFIBUS network, in a specific token cycle, only one overrun contributes to the token lateness.*

Proof:

Assume that a token delay is induced in the l^{th} token cycle. Hence the token arrives late in the next token cycle. Consider the analysis focused on master k, and the measurement of the time elapsed between $A^k(l)$ and $A^k(l+1)$, that is, between two consecutive token arrivals to master k (T^k_{RR}). The masters that may induce a delay in the token are, in sequence of token holding, the masters k itself until n, in the l^{th} rotation, and masters 1 until k^{-1} in the $(l+1)^{th}$ rotation, if $k<>1$, or simply masters k until n in the l^{th} rotation, if $k=1$. For simplification of this proof we assume that $k=1$. In this case, the last master, before the $(l+1)^{th}$ visit of the token to master 1, which may produce an overrun of the T_{TH}, is master n, hence an overrun in

T^n_{TH}. If in the l^{th} visit to master n a T^n_{TH} overrun occurs, then $A^n(l)–A^n(l–1)\leq T_{TR}$, that is, the token arrived early to master n. If we denote $b^n(l)$ as the time instant when T^n_{TH} expires during the l^{th} visit to master n, then, as $A^n(l)>A^n(l–1)$, $b^n(l)–A^k(l)\leq T_{TR}$, no matter if other overruns have occurred in the l^{th} rotation of the token in any of the predecessors of master n. Thus, only one overrun may contribute to the token lateness.
❑

Figure 3 illustrates theorem 2, where n is set to 4 and k is set to 1. In this illustrative example, two overruns occurred in the l^{th} token rotation (both in masters 1 and 4). Only the last one before $A^1(l+1)$ contributes to the token lateness in master 1, that is, the one occurred in master 4.

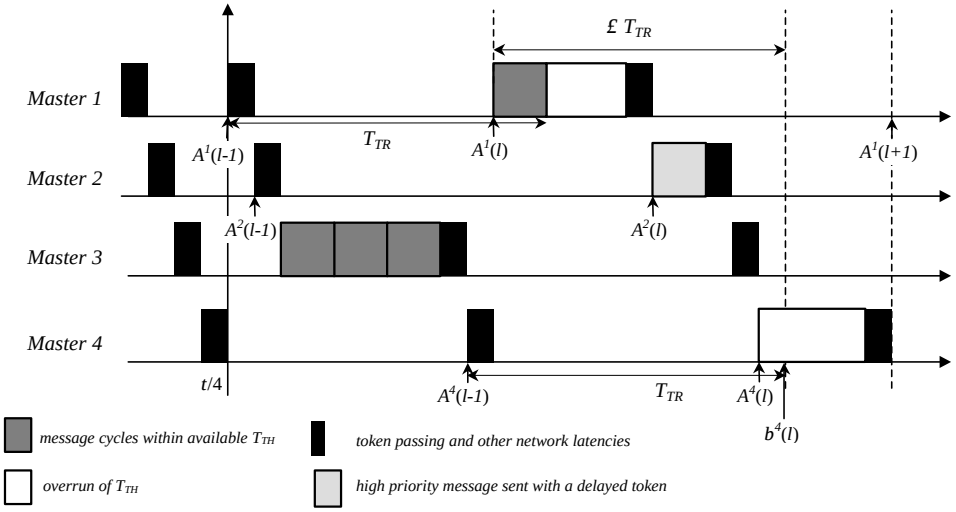

Fig. 3. Illustrative Example for Theorem 2

Theorem 3 *If a PROFIBUS master k holds the token for an interval greater than $T_{TR} – t$, all the following masters up to master k^{-1} will receive a late token.*

Proof:

Due to the token passing time and other network latencies, it results that $A^k(l)–A^{k+1}(l–1)\geq((n–1)/n)\times t$. That is the difference between the token arrival to a master k and the token arrival to its successor in the previous token cycle is at least $(n–1)/n\times t$ (corresponding to n-1 token passing times). $A^k(l)–A^{k+1}(l–1)\geq((n–1)/n)\times t$ can be re-written as $A^{k+1}(l–1)\leq A^k(l)–((n–1)/n)\times t$. As master k holds the token for an interval greater than $T_{TR}–t$, then $R^k(l)>A^k(l)+T_{TR}–t$. It is also evident that the arrival of the token to station k^{+1} occur at $A^{k+1}(l)=R^k(l)+t/n$, that is at the time the token is released in k added with the time to pass the token to k^{+1}. Thus, if we replace $R^k(l)$ in the equation $(A^{k+1}(l)=R^k(l)+t/n)$ with the inequality $R^k(l)>A^k(l)+T_{TR}–t$, it results that $A^{k+1}(l)>A^k(l)+T_{TR}–((n–1)/n)\times t$. Hence, using this last inequality and knowing that $A^k(l)–A^{k+1}(l–1)\geq((n–1)/n)\times t \Leftrightarrow A^{k+1}(l–1)\leq A^k(l)–((n–1)/n)\times t$, it results that $A^{k+1}(l)–A^{k+1}(l–1)>A^k(l)+T_{TR}–((n–1)/n)\times t–A^k(l)+((n–1)/n)\times t=T_{TR}$.
❑

Figure 4 illustrates theorem 3, where k is set to 1.

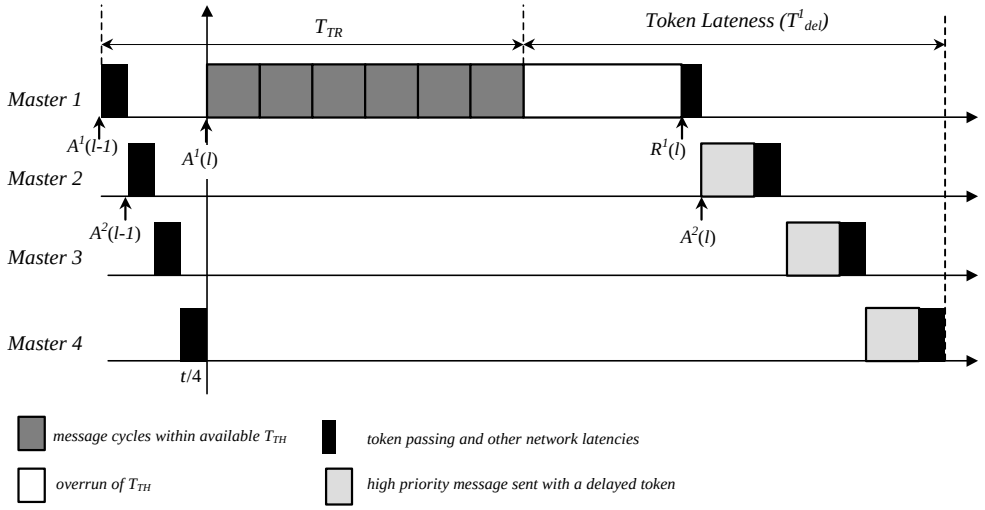

Fig. 4. Illustrative Example for Theorem 3

4.2. T^k_{cycle} as an Upper Bound for T^k_{RR}

Using theorem 3, we can define the token lateness in a master k (T^k_{del}) as the maximum excess to T_{TR} of a token arrival to the master k. The maximum time elapsed between two consecutive token arrivals to a master k (T^k_{cycle}) is then given by:

$$T^k_{cycle} = T_{TR} + T^k_{del} \quad (7)$$

Assuming for simplification that all the message cycle durations are equal to C_s then, the worst-case token lateness in a master k would result from the simultaneous occurrence of the three following conditions:

1. the actual token holding time is greater than $T_{TR} - t$.
2. master k itself causes an overrun, starting with a residual value of T^k_{TH}.
3. all following masters (until the master k^{-1}) transmit, each one, one high priority message cycle, having received a late token.

Observed these three conditions, in the next token cycle, T^k_{RR} reaches its upper bound, which is T^k_{cycle}. In the case of equal length for all the message stream cycles, $T^k_{del}=n\times C_s$, and thus:

$$T^k_{cycle} = T_{TR} + n \times C_s \,,\ \forall_{master\ k} \quad (8)$$

In the general case (message cycles with different length), the worst-case token lateness may result not from an overrun in the master k but from one occurring in one of the following masters (k^{+1} until k^{-1}).

Using figure 5 as an illustrative example, assume that master 1 do not overruns its T^1_{TH}. Then, master 2 may use its available token holding time and produce a T^2_{TH} overrun. If this T^2_{TH} overrun is longer than the sum of the maximum T^1_{TH} overrun in station 1 with the longest Ch^2_i, then this would led to a higher value for T^1_{del} (figure 5). Similarly, if the maximum T^3_{TH} overrun is longer than the sum of the maximum T^2_{TH} overrun and the longest Ch^3_i, then this would led to a higher value for T^1_{del}. Note that by theorems 2 and 3, for the

token lateness evaluation in master k, one must only consider one overrun in station j (from k to k^{-1}) and one high priority message cycle per each station whose address is between j and station k^{-1}.

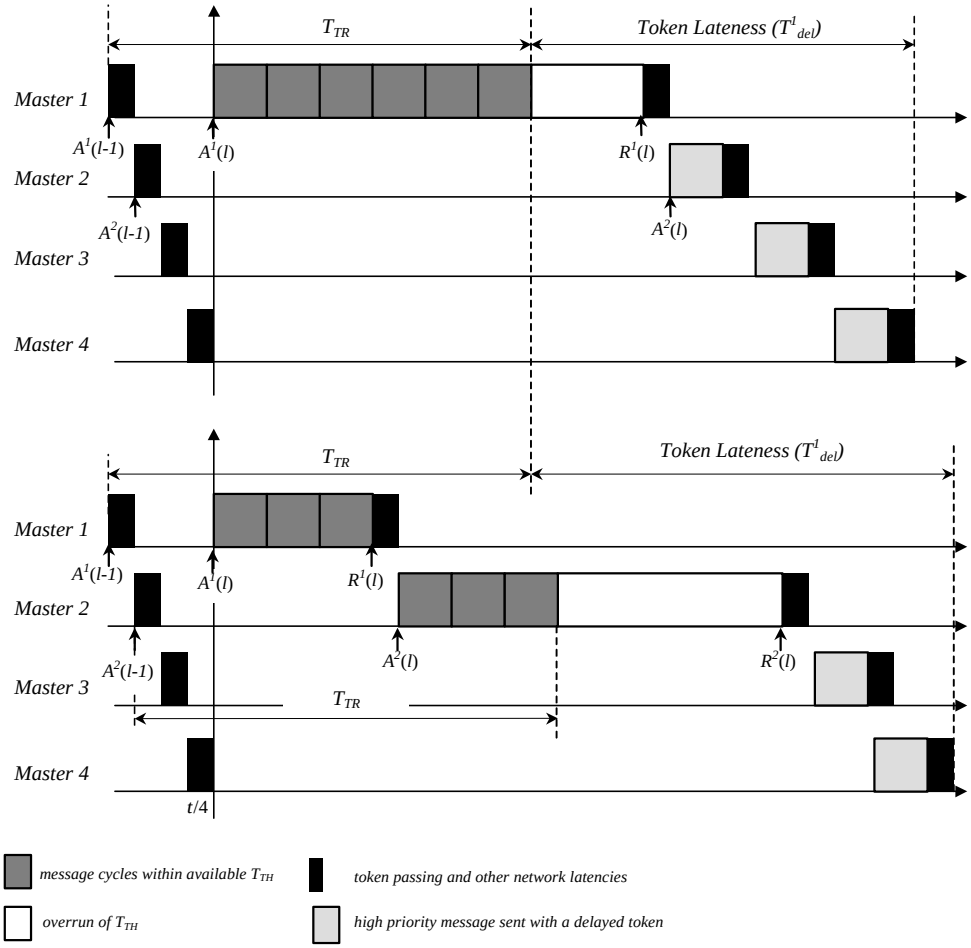

Fig. 5. Comparison between Two Overrun Situations

Basically, we can conclude that T^k_{cycle} depends on which master produces the worst--case overrun and on its relative position in the logical ring.

For the T^k_{del} evaluation, we introduce the following parameters: H^k, L^k and A^k. H^k is the longest high priority message cycle transfer requested by a station k:

$$H^k = \max_{i=1,..,nh^k} \left\{ Ch_i^k \right\} \quad (9)$$

L^k is the longest low priority message cycle transfer requested by a station k:

$$L^k = \max_{i=1,..,nl^k} \left\{ Cl_i^k \right\} \quad (10)$$

Finally, A^k is the longest message cycle transfer requested by a station k (including both types of message cycles):

$$A^k = \max\left\{H^k, L^k\right\} \quad (11)$$

Using the analysis outlined in this section, we can thus define the maximum token lateness in a PROFIBUS master station k (T^k_{del}) as being:

$$T^k_{del} = \max_{j=f_1}\left\{A^j + \sum_{i=f_2} H^i\right\} \quad (12)$$

where f_1 is defined as

$$f_1 = \begin{cases} k..n, & \text{if } k = 1 \\ k..n, 1..k-1, & \text{if } k > 1 \end{cases} \quad (13)$$

and f_2 as

$$f_2 = \begin{cases} j+1..n, & \text{if } k = 1 \\ j+1..n, 1..k-1, & \text{if } k > 1 \end{cases} \quad (14)$$

Finnaly, the worst-case token cycle time in a PROFIBUS fieldbus network is:

$$T^k_{cycle} = T_{TR} + \max_{j=f_1}\left\{A^j + \sum_{i=f_2} H^i\right\} \quad (15)$$

We can now re-write the deadline condition (5) as follows:

$$Dh^k_i \geq g_{Sh^k_i} + nh^k \times \left(T_{TR} + T^k_{del}\right) + Ch^k_i + d_{Sh^k_i}, \forall_{\text{master } k,\ \text{stream } Sh^k_i} \quad (16)$$

Therefore, the following condition for setting the T_{TR} parameter can be used:

$$0 \leq T_{TR} \leq \frac{Dh^k_i - Ch^k_i - d_{Sh^k_i}}{nh^k} - T^k_{del}, \forall_{\text{master } k,\ \text{stream } Sh^k_i} \quad (17)$$

where d aggregates both the generation and delivery delays of message stream.

5. Numerical Examples

Consider a PROFIBUS network scenario with 3 masters, each one with the following message streams:

Table 1. Numerical Example

Master Station 1	Master Station 2	Master Station 3
Ch_1^1=8 ms	Ch_1^2=8 ms	Ch_1^3=8 ms
Ch_2^1=6 ms	Ch_2^2=15 ms	Ch_2^3=18 ms
Ch_3^1=7 ms	-	-
Cl_1^1=10 ms	Cl_1^2=30 ms	-
-	Cl_2^2=18 ms	-

For this numerical example, the results for each T^k_{del} are (using 12):

Table 2. T^k_{del} Evaluation for the Numerical Example

Master Station 1	Master Station 2	Master Station 3
H^1=8 ms	H^2 = 15 ms	H^3 = 18 ms
A^1=10 ms	A^2 = 30 ms	A^3 = 18 ms
$T^1_{del}=A^2+H^3$=48 ms	$T^2_{del}=A^2+H^3+H^1$=58 ms	$T^3_{del}=A^3+H^1+H^2$=41 ms

For simplification, assume that the generation and delivery delays correspond to 10% of the message cycle length. Consider also that t = 1 ms. If we assume that the minimum value for T_{TR} should be marginally greater than t (otherwise low priority traffic would not be transferred at all and, as it will be clarified later, hence the T^k_{cycle} evaluation would be different), then (17) can be re-written as:

$$t < T_{TR} \leq \frac{Dh_i^k - 1.1 \times Ch_i^k}{nh^k} - T_{del}^k, \forall_{\text{master } k,\ \text{stream } Sh_i^k} \quad \textbf{(18)}$$

Then, to evaluate the shortest value for each message's deadline we have:

$$\frac{Dh_i^k - 1.1 \times Ch_i^k}{nh^k} - T_{del}^k > t \quad \textbf{(19)}$$

which can be re-written as:

$$Dh_i^k > \left(t + T_{del}^k\right) \times nh^k + 1.1 \times Ch_i^k \quad \textbf{(20)}$$

Using (20), the minimum deadline supported for each high priority stream of table 1 scenario is as shown in table 3.

Table 3. Minimum Admissible Deadlines for $T_{TR} = t$

Master Station 1	Master Station 2	Master Station 3
Dh_1^1>155.8 ms	Dh_1^2>126.8 ms	Dh_1^3>103.8 ms
Dh_2^1>153.6 ms	Dh_2^2>134.5 ms	Dh_2^3>103.8 ms
Dh_3^1>154.7 ms	-	-

From (16), it is obvious that T_{TR} can be set as small as *0*. Note however, as previously mentioned, that if this is the case, the low priority traffic would not be transferred at all. Notably, in this non-realistic situation, low priority traffic would not be considered for the evaluation of T^k_{del}. If T_{TR} is smaller than t, then (21) must be used to evaluate each T^k_{cycle}, instead of (12):

$$T_{del}^k = \sum_{i=1..n} H^i \quad \textbf{(21)}$$

which means that all T^k_{cycle} would have the same value in all masters. Using the same table 1 scenario, each T^k_{cycle} would then be:

Table 4. T_{del} Computations for the Numerical Example, with T_{TR} = 0

Master Station 1	Master Station 2	Master Station 3
H^1=8 ms	H^2=15 ms	H^3=18 ms
T^1_{del}=41 ms	T^2_{del}=41 ms	T^3_{del}=41 ms

and the minimum deadline supported for each high priority stream, would be as shown in table 5.

Table 5. Minimum Admissible Deadlines for $T_{TR} = 0$

Master Station 1	Master Station 2	Master Station 3
Dh_1^1>134.8 ms	Dh_1^2>92.8 ms	Dh_1^3>103.8 ms
Dh_2^1>132.6 ms	Dh_2^2>100.5 ms	Dh_2^3>103.8 ms
Dh_3^1>133.7 ms	-	-

6. Conclusions

In this paper, we have drawn a comprehensive study on how to use PROFIBUS to support real-time communications in distributed computer-controlled systems. The major contribution is to provide an accurate evaluation of the maximum PROFIBUS token cycle time, considering that all types of traffic are allowed, whereas in previous related works relevant traffic types were not considered. The relevance of this result is paramount as it is the basis for the setting of the T_{TR} parameter in PROFIBUS networks in order to guarantee messages' timing requirements.

We have shown that the maximum token cycle time is a consequence of the overrun of the token holding timer in a mater station and how such overrunning impacts the cycle time properties of the PROFIBUS timed token protocol.

References

1. Lenhart, G.: "A Field Bus Approach to Local Control Networks", Advances in Instrumentation and Control, 1993, 48 (1), pp. 357-365.
2. Cardoso, A. and Tovar, E.: "Industrial Communication Networks: Issues on Heterogeneity and Internetworking", Proceedings of the 6th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM'96), Atlanta, USA, pp. 139-148, Begell House Publishers, New York, May 1996.
3. Profibus, "Profibus Standard DIN 19245 part I and II". Translated from German, Profibus Nutzerorganisation e.V., 1992.
4. EN 50170, "General Purpose Field Communication System", European Standard, CENELEC, 1996.
5. Tindell, K., Burns, A. and Wellings, A.: "Analysis of Hard Real-Time Communications", Journal of Real-Time Systems, 1995, 9, pp. 147-171.
6. Fidge, C.: "Real-Time Schedulability Tests for Preemptive Multitasking", Journal of Real-Time Systems, Vol. 14, No 1, pp. 61-93, January 1998.
7. Stankovic, J., Spuri, M., Ramamritham, K. and Buttazzo, G.: "Deadline Scheduling for Real-Time Systems – EDF and Related Algorithms", Kluwer Academic Publishers, Boston, 1998.

8. Tovar, E. and Vasques, F.: “Guaranteeing Real-Time Message Deadlines in Profibus Networks”, Proceedings of the 10th Euromicro Workshop on Real-Time Systems, Berlin, Germany, IEEE Press, June 1998.
9. Tovar, E. and Vasques, F.: “Real-Time Fieldbus Communications using Profibus Networks”, ISEP Technical Report, July 1998 (accepted for publication in the IEEE Transactions on Industrial Electronics, December 1999).
10. Grow, R.: “A Timed Token Protocol for Local Area Networks”, Proceedings of Electro’82, Token Access Protocols, Paper 17/3, May 1982.
11. IEEE, “IEEE Standard 802.4: Token Passing Bus Access Method and Physical Layer Specification”, 1985.
12. ISO, “Information Processing Systems - Fibre Distributed Data Interface (FDDI) - Part 2: Token Ring Media Access Control (MAC)”, ISO International Standard 9314-2, 1989.
13. Sevcik, K. and Johnson, M.: “Cycle Time Properties of the FDDI Token Ring Protocol”, IEEE Transactions on Software Engineering, 13 (3), pp. 376-385, March 1987.
14. Montuschi, P., Ciminiera, L. and Valenzano, A.: “Time Characteristics of IEE802.4 Token Bus Protocol”, IEE Proceedings, 139 (1), pp. 81-87, January 1992.
15. Malcolm, N. and Zhao, W.: “The Timed-Token Protocol for Real-Time Communications”, IEEE Computer, pp. 35-41, January 1994.
16. Agrawal, G., Chen, B., Zhao, W., Davari, S.: “Guaranteeing Synchronous Message Deadlines with Timed Token Protocol Medium Access Control Protocol”, IEEE Transactions on Computers, Vol. 43, No. 3, pp. 327-339, March 1994.
17. Chen, B., Agrawal, G. and Zhao, W.: “Optimal Synchronous Capacity Allocation for Hard Real-Time Communications with the Timed-Token Protocol”, Proceedings of the 13th IEEE Real-Time Systems Symposium, pp. 198-207, December 1992.
18. Zhang, Z. and Burns, A.: “An Optimal Synchronous Bandwidth Allocation Scheme for Guaranteeing Synchronous Message Deadlines with the Timed-Token MAC Protocol”, IEEE/ACM Transactions on Networking, Vol. 3, pp. 729-741, December 1995.
19. Han, C.-C. and Shin, K.: “A Polynomial-time Optimal Synchronous Bandwidth Allocation Scheme for the Timed Token Protocol”, Proceedings of the IEEE Infocom’95, pp. 198-207, April 1995.
20. Li, M and Stoeckli, L.: “The Time Characteristics of Cyclic Service in PROFIBUS”, Proceedings of the EURISCON’94, Vol. 3, pp. 1781-1786, 1994.