Cycle time properties of the PROFIBUS timed-token protocol

Eduardo Tovar

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

Francisco Vasques

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 computing elements by means of multi-point broadcast networks. Since the network medium is shared between a number of network nodes, access contention exists and must be resolved by a 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 a MAC protocol that guarantees bounded access and response times to message requests. PROFIBUS is a communication network in which the MAC protocol is based on a simplified version of the timed-token protocol. In this paper we address the cycle time properties of the PROFIBUS MAC protocol, since the knowledge of these properties is of paramount importance for guaranteeing a real-time behaviour of a distributed computer-controlled system which is supported by this type of network.

Keywords: Real-time communications; 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 economic aspects, the use of LANs is also reinforced by the increasing decentralisation of control and measurement tasks, as well as by the increasing 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 scope of fieldbuses was dominated by vendor-specific solutions, which were mostly restricted to specific application areas. Moreover, the concepts behind each proposed network were highly dependent on the manufacturer of the automation system. Each had different technical implementations and also claimed to fulfil different application requirements, or to fulfil the same requirements but with different technical solutions [2]. More recently, vendor-independent standardised fieldbuses, supporting the open system concept, have started to be commonly used. PROcess FIeld BUS (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, whatever 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 called 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 next master. Typically, the process-relevant devices (sensors and actuators) are accessed through a slave network-interface, whereas the distributed control algorithms reside at master stations. Therefore, in PROFIBUS, the end-to-end communication delay [5] for 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 to 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 of the contention not only 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 the message requests, while the second factor depends on the behaviour of the token-passing procedure. Therefore, an evaluation of the worst-case queuing delay of the message requests is of paramount importance to guaranteeing the messages' timing requirements. The aggregate value that includes the queuing and transmission delays is referred to, in this paper, as the message response time.

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

The remainder of this paper is organised as follows. In Section 2 we provide a methodology for the evaluation of the worst-case PROFIBUS messages' response time. The section presents an improved version of the analysis previously proposed in [8,9], which assumes the worst-case scenario for the token cycle time. In Section 3 we survey previous relevant works addressing the evaluation of the token cycle time in other network protocols based on the timed-token protocol [10], such as the IEEE802.4 [11] and the FDDI [12]. PROFIBUS uses a simplified version of the timed-token protocol. We highlight the differences to the original timed-token protocol and justify why it is not possible to apply the results available for the case of other timed-token based networks. 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 messages' response time. Finally, in Section 5 we show how the methodology and the results provided in this paper can be used to set the target token rotation time (T_{TR}) parameter of a PROFIBUS network, in a way that the real-time requirements of a DCCS application are guaranteed.

2. Real-time analysis for PROFIBUS networks

2.1. A brief description of the PROFIBUS timed-token protocol

In PROFIBUS, messages belong to one of the two following categories: high-priority messages and low-priority (including cyclic, non-cyclic and management) messages. For the approach we propose in this paper, real-time traffic is supported by PROFIBUS high-priority messages.

The PROFIBUS MAC protocol is based in a simplified version of the timed-token protocol [10]. We describe now its basic timing characteristics.

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 master 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}) is greater than the

target rotation time (T_{TR}), the master may execute, at most, one high-priority message cycle. Otherwise, the master may execute high-priority message cycles while $T_{TH}>0$. T_{TH} is always tested before starting the execution of a message cycle. It follows that once a message cycle is started it is always completed, even if during message cycle execution the master's T_{TH} reaches the value 0. We call this situation an overrun of the T_{TH} overrun timer. The low-priority message cycles are executed if there are no high-priority message requests pending, and while $T_{TH}>0$ (also evaluated before starting the message cycle execution, thus also leading to a possible overrun of the master's T_{TH} timer).

Below we give a pseudo-code description of the PROFIBUS token-passing algorithm:

```
/* Initialisation procedure */
At each master k, DO:
   T_TH ← 0 ;
   T_RR ← 0 ;
   Start T_RR ;                /* count-up timer */
/* Run-time procedure */
At each master 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 master (k + 1) (modulo n);
```

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

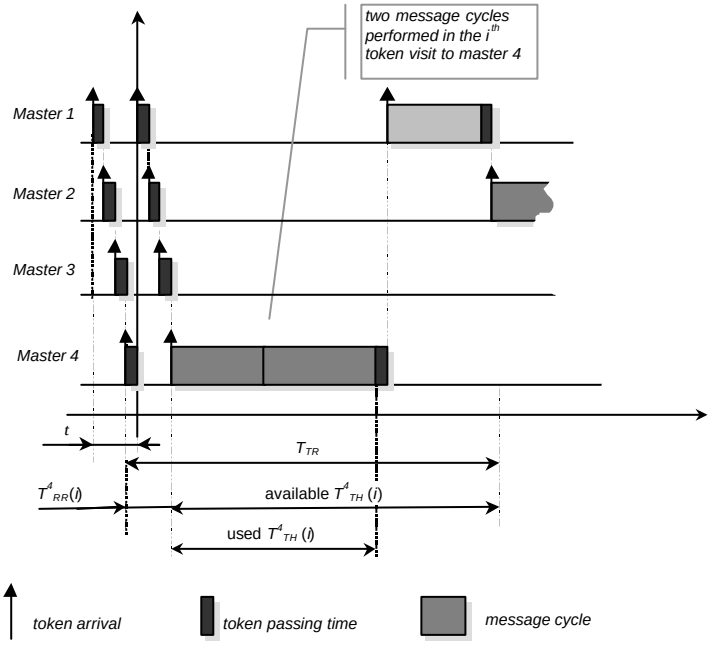

Fig. 1. An example of token usage in a master

In PROFIBUS, a message cycle consists on a master's action frame (request or send/request frame) and a responder's immediate acknowledgement/response frame. User data may be transmitted in the

action frame or in the response frame. Note that a PROFIBUS master is allowed to send up to a limited number of retries, in case the response does not come within a predefined time. Thus, for the analysis, the message cycle time length must also include the time needed to process the allowed number of retries.

2.2. Worst-case response time for the high-priority messages

We consider a bus topology with n PROFIBUS masters, with addresses ranging from 1 to n. A special frame, the token, circulates between the masters, resulting in a logical token ring. Each master accesses the network according to the token passing sequence; hence, first master 1, then masters 2, 3, ... until master n, and then again masters 1, 2, ... Slaves will have network addresses higher than n.

The logical ring latency (token walk time, including node latency delay, media propagation delay, etc.) is denoted with the symbol τ.

We 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 stream Sh_i^k is characterised as:

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

where Ch_i^k is the maximum amount of time required to perform a message cycle of Sh_i^k, and Dh_i^k is the stream's relative deadline. In other words, the relative deadline is the maximum admissible end-to-end communication delay a stream Sh_i^k may tolerate. Th_i^k is the minimum inter-arrival time interval between any two consecutive requests of a stream Sh_i^k.

As we do not mean to guarantee deadlines for the low-priority traffic, a low-priority stream Sl_i^k is characterised as:

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

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

As previously said, one of the main characteristics of the PROFIBUS timed-token protocol, is that it guarantees that a master may always transmit at least one high-priority message per token visit, independently of the real token rotation time. Assuming the worst-case scenario (the token is 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 how the master's high-priority queue is implemented. In PROFIBUS communication queues operate in a first-come-first-served (FCFS) basis.

Consequently, 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 belonging to the same stream, then it would mean that a deadline for that message stream was missed. It also follows that for all streams Sh_i^k, Th_i^k must be greater or equal to Dh_i^k

Therefore, the upper bound for the message queuing delay in a master k is:

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

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

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

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:

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

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 synchronous and asynchronous messages, respectively.

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 must 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 can be found in [13].

In order to guarantee the deadlines of synchronous messages, the existence of an upper bound to the token rotation time is a necessary but not a sufficient condition. A node with inadequate synchronous capacity may be unable to guarantee messages' deadlines, and, on the other hand, allocating excess amount of synchronous capacities to the nodes increases T_{TR}, which may also cause messages' 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 a protocol constraint and a deadline constraint [15,16].

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

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

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 differences as the most relevant ones.

1. In PROFIBUS there is no synchronous capacity allocation (H_i). If a station receives a late token (T_{RR} is greater than T_{TR}), then, at most, only one high-priority message may be transmitted. Contrarily, in the original timed-token protocol the station can transmit synchronous (high-priority) messages during H_i time, even if the token arrived late to that station.
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 messages' 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 neither low-priority traffic (other than the cyclic services) are allowed. This prevents the transfer of high-priority event-driven messages, 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 timing requirements of the high-priority traffic are guaranteed whatever the load of low-priority messages.

In the following section, we derive an accurate upper bound for the token cycle time, valid for PROFIBUS networks supporting all types of traffic.

4. Analysis of the token cycle time in PROFIBUS networks

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 arriving to a master k.

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

4.1. Analysis of the 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 an overrun of T_{TH} 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_{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} (an upper bound for T^k_{RR}), which is later on discussed in Section 4.2.

Theorem 1:

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

Proof:

We denote $A^k(l)$ as the time instant when the token arrives to the master k for its l^{th} visit, and $R^k(l)$ as the time instant when master k releases the token in that l^{th} visit.

If master k releases the token before the expiration of T^k_{TH} then, $R^k(l)$-$A^k(l$-1)<T_{TR}. Note that the real token rotation time is measured between token arrivals. Therefore, at the time instant $A^k(l)$, T^k_{TH} is given the value T_{TR}-T^k_{RR}, a positive value (the token do not arrives late). If at the time instant $R^k(l)$ master k releases the token and the T^k_{TH} timer has not yet reached 0, then $R^k(l)$-$A^k(l$-1)<T_{TR}. This equation is the starting assumption for the remainder of the proof.

We denote the successor of master k as master k^{+1} (with k^{+1}=(k+*1*) mod n). We want to prove that if $R^k(l)$-$A^k(l$-1)<T_{TR} is true, then $A^{k+1}(l)$-$A^{k+1}(l$-1)<T_{TR} is also true; that is, after releasing the token at time instant $R^k(l)$, the successor of k will receive an early token.

As $A^{k+1}(l$-1)=$R^k(l$-1)+τ/n (since τ denotes the total logical ring latency, k^{+1} will receive the token τ/n time after the release of the token in k) and as $A^{k+1}(l)$=$R^k(l)$+τ/n then, combining these two expressions, it follows that $A^{k+1}(l)$-$A^{k+1}(l$-1)=$R^k(l)$-$R^k(l$-1).

The starting assumption is that $R^k(l)$-$A^k(l$-1)<T_{TR}. As $R^k(l$-1)=$A^k(l$–1)+τ/n, it is true saying that $R^k(l$-1)>$A^k(l$-1). Therefore, if $R^k(l)$-$A^k(l$-1)<T_{TR} then $R^k(l)$-$R^k(l$-1)<T_{TR}.

As $A^{k+1}(l)$-$A^{k+1}(l$-1)=$R^k(l)$-$R^k(l$-1), if $R^k(l)$-$R^k(l$-1)<T_{TR} then $A^{k+1}(l)$-$A^{k+1}(l$-1)<T_{TR} is true; that is, the successor of k receives an early token.

❑

Fig. 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.

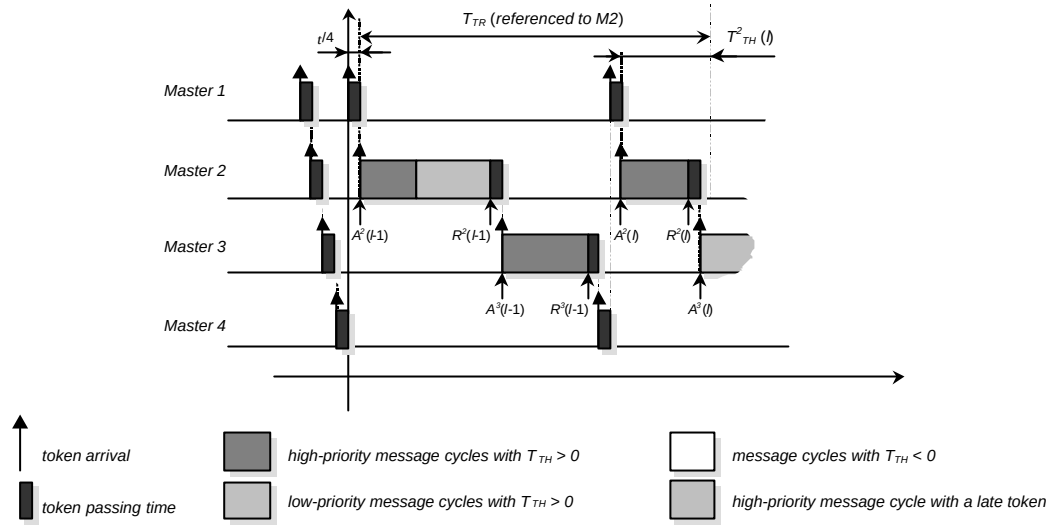

Fig. 2. An illustrative example for Theorem 1

From Theorem 1, two lemmas result.

Lemma 1.1:

In PROFIBUS, a 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 overrun of a T_{TH} in masters k^{-2}, k^{-3}, …

Lemma 1.2:

In PROFIBUS, if none of the masters overrun their T_{TH}, the token will never be late.

Theorem 2:

In a PROFIBUS network, in a specific token cycle, only one overrun of T_{TH} 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 ($T^k{}_{RR}$).

If $k \neq 1$, then the masters that may induce a delay in the token are, in sequence of token holding, the master k itself and all other masters up to master n, in the l^{th} token rotation, and master 1 and all masters up to master k^{-1} in the $(l+1)^{th}$ token rotation. If $k=1$, then the masters that may induce a delay in the token are, in sequence of token holding, the master k itself and all other masters up to master n, all in the l^{th} token rotation.

For simplification of this proof, and without loss of generality, 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 $T^n{}_{TH}(l)$.

If in the l^{th} visit to master n an overrun of $T^n{}_{TH}$ occurs, then $A^n(l)$-$A^n(l\text{-}1) \leq T_{TR}$; that is, the token arrived early to master n. If we denote $\beta^n(l)$ as the time instant when $T^n{}_{TH}$ expires during the l^{th} visit to master n, then, as $A^n(l) > A^n(l\text{-}1)$, it follows that $\beta^n(l)$-$A^k(l) \leq T_{TR}$, 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.

❑

Fig. 3 illustrates Theorem 2, where n is set to 4 and k is set to 1. In this illustrative example, two overruns occur in the l^{th} token rotation, in master 1 and in master 4. Only the last one before $A^1(l+1)$, the one that took place in master 4, contributes to the lateness of the token arriving to master 1 at $A^1(l+1)$.

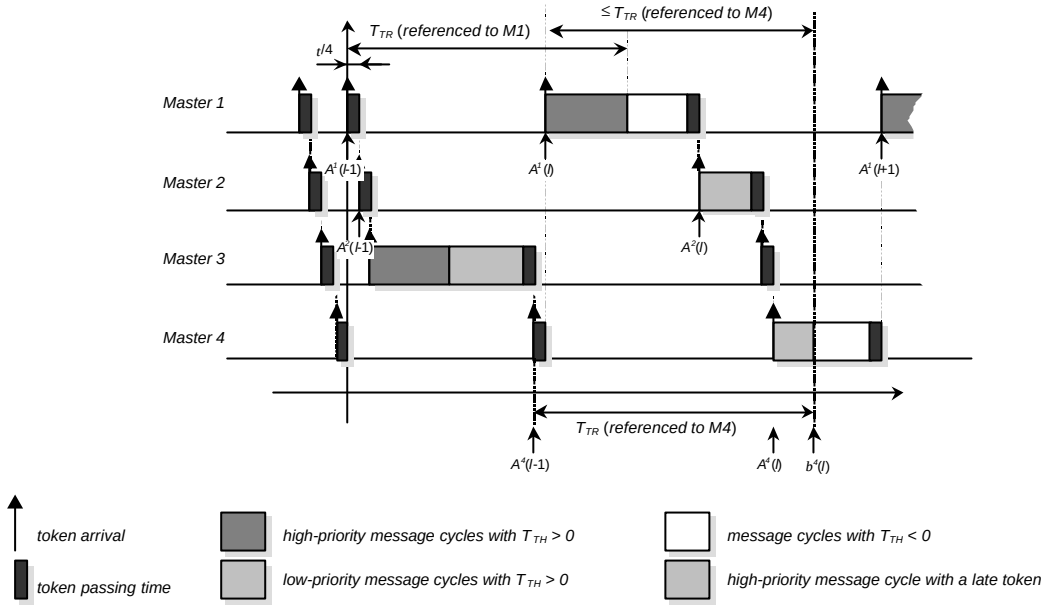

Fig. 3. An illustrative example for Theorem 2

Theorem 3:

If a PROFIBUS master k holds the token for an interval greater than T_{TR}-τ, 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 follows that $A^k(l)$-$A^{k+1}(l\text{-}1)\geq((n\text{-}1)/n)\times\tau$; 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 $((n\text{-}1)/n)\times\tau$ (corresponding to n-1 token-passing times).

$A^k(l)$-$A^{k+1}(l\text{-}1)\geq((n\text{-}1)/n)\times\tau$ can be re-written as $A^{k+1}(l\text{-}1)\leq A^k(l)$-$((n\text{-}1)/n)\times\tau$. As the master k holds the token for an interval greater than T_{TR}-τ, then $R^k(l)>A^k(l)+T_{TR}$-τ.

It is also evident that the arrival of the token to master k^{+1} occurs at $A^{k+1}(l)=R^k(l)+\tau/n$, that is at the time the token is released in k added to the time to pass the token to k^{+1}.

Thus, if we replace $R^k(l)$ in the equation ($A^{k+1}(l)=R^k(l)+\tau/n$) with the inequality ($R^k(l)>A^k(l)+T_{TR}$-τ), it follows that $A^{k+1}(l)>A^k(l)+T_{TR}$-$((n\text{-}1)/n)\times\tau$. Hence, using this last inequality and knowing that $A^k(l)$-$A^{k+1}(l\text{-}1)\geq((n\text{-}1)/n)\times\tau \Leftrightarrow A^{k+1}(l\text{-}1)\leq A^k(l)$-$((n\text{-}1)/n)\times\tau$, it follows that $A^{k+1}(l)$-$A^{k+1}(l\text{-}1)>A^k(l)+T_{TR}$-$((n\text{-}1)/n)\times\tau$-$A^k(l)+((n\text{-}1)/n)\times\tau=T_{TR}$.

Obviously this result extends to all the following masters which range from master k^{+2} up to master k^{-1}. The starting assumption is that the token holding time in master k is $R^k(l)$-$A^k(l)>T_{TR}$-τ. As the token arrived in the other masters before $A^k(l)$, and after its release from master k at time instant $R^k(l)$ it will arrive at the other masters after $R^k(l)$, the token rotation time as measured in the other masters will give, for all of them, a value greater than T_{TR}.

❑

Fig. 4 illustrates Theorem 3, where k is set to 1.

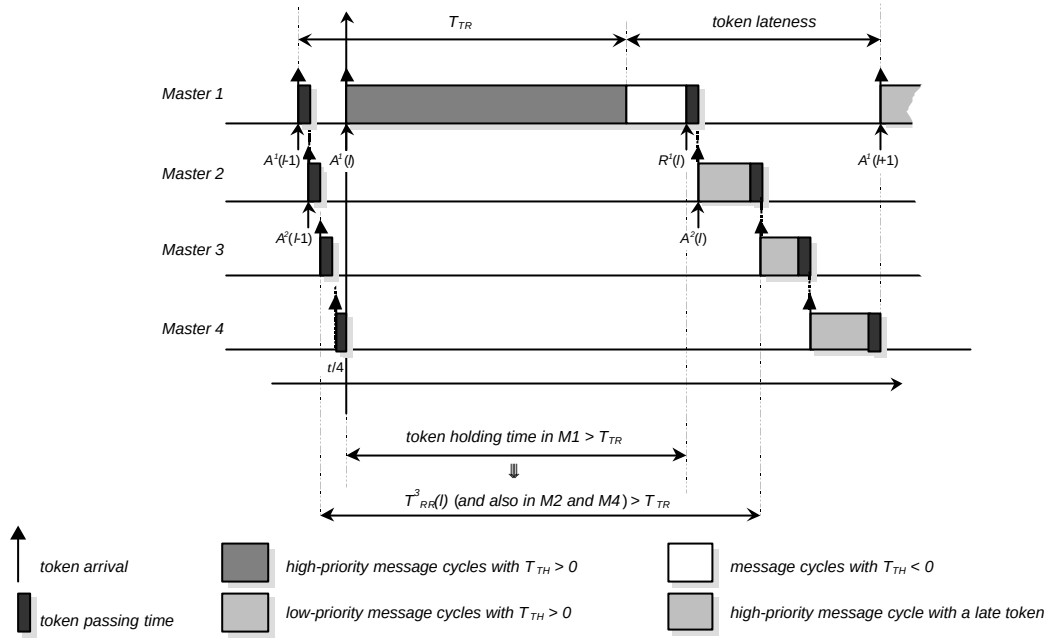

Fig. 4. An illustrative example for Theorem 3

4.2. T^k_{cycle} as an upper bound for T^k_{RR}

By 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:

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

Assuming, for simplification, that all the message cycle have the same duration (represented by 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 in master k is greater than T_{TR}-$\boldsymbol{t}$.
2. The master k itself causes an overrun of T^k_{TH}, and this overrun starts with a residual value of T^k_{TH}.
3. All the 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:

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

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

Using Fig. 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 an overrun of its T^2_{TH} overrun timer. If this overrun is longer than the sum of the longest overrun of T^1_{TH} added to the longest Ch^2_i, then this would led to a higher value for T^1_{del} (Fig. 5b). Similarly, if the longest overrun of T^3_{TH} is longer than the longest overrun of T^2_{TH} added to 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 evaluation of token lateness in master k, we can only consider one overrun in master j (with j ranging from master k to master k^{-1}), and one high-priority message cycle per each master whose address is between j and k^{-1}.

a)

b)

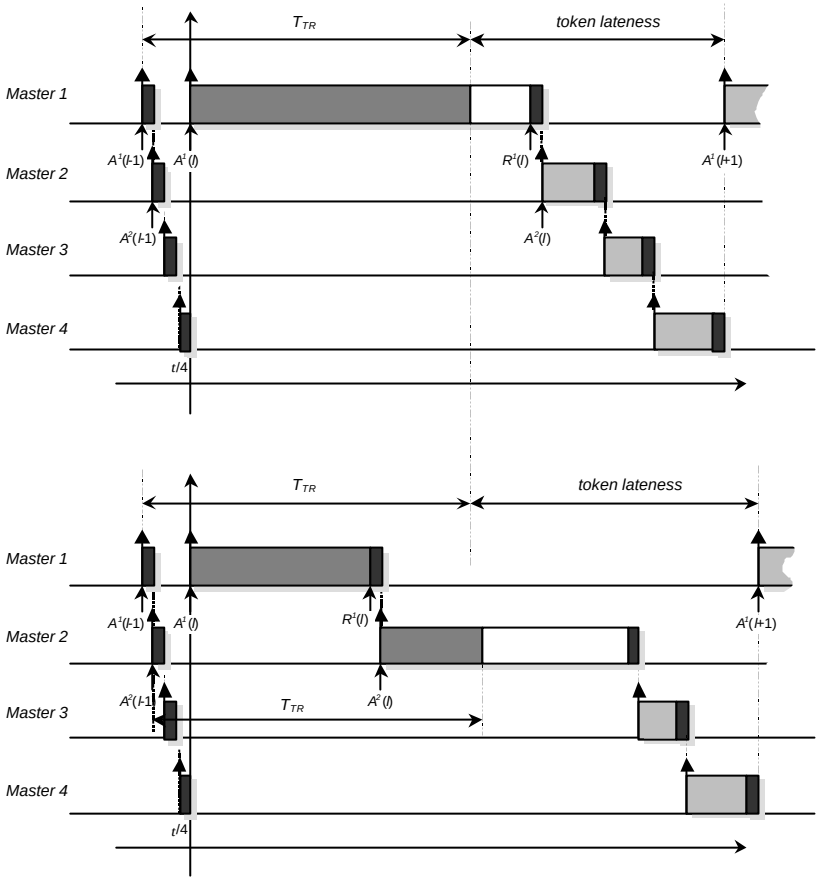

Fig. 5. A comparison between two scenarios with overrunning of T_{TH}

Consequently, we conclude that the evaluation of T^k_{cycle} depends on which is the master that produces the worst-case overrun of its T_{TH} and on its relative position within the logical ring sequence of token-passing.

For the evaluation of T^k_{del}, we introduce the following parameters: H^k, L^k and A^k. H^k is the longest high-priority message cycle that can be requested by a master k:

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

L^k is the longest low-priority message cycle that can be requested by a master k:

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

Finally, A^k is the longest message cycle that can be requested by a master k (including both low and high-priority message cycles):

$$A^k = \max \left\{ H^k, L^k \right\} \tag{11}$$

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

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

where f_1 is defined as being the following set of values:

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

and f_2 is defined as being the following set of values:

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

Consequently, the worst-case token cycle time in a PROFIBUS fieldbus network may be defined as follows:

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

Then, we can now re-write the deadline condition (5) as follows:

$$Dh_i^k \geq g_{Sh_i^k} + nh^k \times \left(T_{TR} + T_{del}^k\right) + Ch_i^k + d_{Sh_i^k}, \forall_{\text{master } k,\ \text{stream } Sh_i^k} \quad \textbf{(16)}$$

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

$$0 \leq T_{TR} \leq \frac{Dh_i^k - Ch_i^k - d_{Sh_i^k}}{nh^k} - T_{del}^k, \forall_{\text{master } k,\ \text{stream } Sh_i^k} \quad \textbf{(17)}$$

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

5. Numerical examples

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

Table 1: A numerical example

Master 1	Master 2	Master 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 (case of $T_{TR} > t$)

Master 1	Master 2	Master 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 have a value equal to 10% of the message cycle duration. Consider also that t=1ms. 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), 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 can use the following inequality:

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

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 shown in Table 1 would be as shown in Table 3.

Table 3: Minimum admissible deadlines for (case of $T_{TR} = t$)

Master 1	Master 2	Master 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 to a value as small as *0*. In this case however the low-priority traffic would not be transferred at all. It also follows that in this non-realistic situation, the low-priority traffic would not be considered for the evaluation of T^k_{del}. In fact, if T_{TR} is smaller than $\boldsymbol{t}$, 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 would mean that T^k_{cycle} would have the same value for all masters. Using the same scenario as shown in Table 1, each T^k_{del} would then be:

Table 4: T^k_{del} evaluation for the numerical example for (case of $T_{TR} = 0$)

Master 1	Master 2	Master 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 (case of $T_{TR} = 0$)

Master 1	Master 2	Master 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 for 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 important traffic types were not considered. This result is of paramount importance as it is the basis for the setting of the T_{TR} parameter in PROFIBUS networks in order to guarantee the timing requirements of the high-priority messages.

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

Acknowledgements

This work was partially supported by ISEP, FLAD, DEMEGI/FEUP and FCT.

References

[1] G. Lenhart, A field bus approach to local control networks, Advances in Instrumentation and Control 48 (1) (1993) 357-365.
[2] A. Cardoso, E. Tovar, Industrial communication networks: issues on heterogeneity and internetworking, in: Proceedings of FAIM 1996, pp. 139-148.
[3] Profibus Standard DIN 19245 part I and II, translated from German, Profibus Nutzerorganisation e.V., 1992.
[4] EN 50170 - General purpose field communication system, CENELEC, 1996.
[5] K. Tindell, A. Burns, A Wellings, Analysis of hard real-time communications, Real-Time Systems 9 (1995) 147-171.
[6] C. Fidge, Real-time schedulability tests for preemptive multitasking, Real-Time Systems 14 (1) (1998) 61-93.
[7] J. Stankovic, M. Spuri, K. Ramamritham, G. Buttazzo, Deadline scheduling for real-time systems – EDF and related algorithms, Kluwer Academic Publishers, Boston, MA, 1998.
[8] E. Tovar, F. Vasques, Guaranteeing real-time message deadlines in PROFIBUS networks, in: Proceedings of the 10th Euromicro Workshop on Real-Time Systems 1998, pp. 79-86.
[9] E. Tovar, F. Vasques, Real-time fieldbus communications using PROFIBUS networks, Polytechnic Institute of Porto, Technical Report HURRAY-TR-9823, 1998, accepted for publication in the IEEE Transactions on Industrial Electronics.
[10] R. Grow, A timed token protocol for local area networks, in: Proceedings of Electro 1982, Token Access Protocols, paper 17/3.
[11] IEEE Standard 802.4: Token passing bus access method and physical layer specification, 1985.
[12] Information processing systems - fibre distributed data interface (FDDI) - part 2: token ring media access control (MAC), ISO international standard 9314-2, 1989.
[13] K. Sevcik, M. Johnson, Cycle time properties of the FDDI token ring protocol, IEEE Transactions on Software Engineering 13 (3) (1987) 376-385.
[14] P. Montuschi, L. Ciminiera, A. Valenzano, Time characteristics of IEE802.4 token bus protocol, IEE Proceedings 139 (1) (1992) 81-87.
[15] N. Malcolm, W. Zhao, The timed-token protocol for real-time communications, IEEE Computer (January 1994) 35-41.
[16] G. Agrawal, B. Chen, W. Zhao, S. Davari, Guaranteeing synchronous message deadlines with timed token medium access control protocol, IEEE Transactions on Computers 43 (3) (1994) 327-339.
[17] B. Chen, G. Agrawal, W. Zhao, Optimal synchronous capacity allocation for hard real-time communications with the timed-token protocol, in: Proceedings of RTSS 1992, pp. 198-207.
[18] Z. Zhang, A. Burns, An optimal synchronous bandwidth allocation scheme for guaranteeing synchronous message deadlines with the timed-token MAC protocol, IEEE/ACM Transactions on Networking 3 (1995) 729-741.
[19] C.-C. Han, K. Shin, A polynomial-time optimal synchronous bandwidth allocation scheme for the timed token protocol, in: Proceedings of INFOCOM 1995, pp. 198-207.
[20] M. Li, L. Stoeckli, The time characteristics of cyclic service in PROFIBUS, in: Proceedings of EURISCON 1994, pp. 1781-1786.