# On the Adaptation of Broadcast Transactions in Token-Passing Fieldbus Networks with Heterogeneous Transmission Media

Mário Alves, Eduardo Tovar
ISEP, Polytechnic Institute of Porto
R. São Tomé, Porto, Portugal
{malves@dee,emt@dei}.isep.ipp.pt

Francisco Vasques
FEUP, University of Porto
R. Roberto Frias, Porto, Portugal
vasques@fe.up.pt

**Abstract**

Broadcast networks that are characterised by having different physical layers (PhL) demand some kind of traffic adaptation between segments, in order to avoid traffic congestion in linking devices. In many LANs, this problem is solved by the actual linking devices, which use some kind of flow control mechanism that either tell transmitting stations to pause (the transmission) or just discard frames. In this paper, we address the case of token-passing fieldbus networks operating in a broadcast fashion and involving message transactions over heterogeneous (wired or wireless) physical layers. For the addressed case, real-time and reliability requirements demand a different solution to the traffic adaptation problem. Our approach relies on the insertion of an appropriate idle time before a station issuing a request frame. In this way, we guarantee that the linking devices' queues do not increase in a way that the timeliness properties of the overall system turn out to be unsuitable for the targeted applications.

## 1. Introduction

A number of local computer networks (LANs) impose or benefit from the use of multiple segments interconnected by linking devices such as repeaters or bridges. This is true since many applications demand the maximum number of nodes and/or the maximum bus length to be extended. Moreover, the same network may include physical layers (PhL) with different bit rates, PhL protocol data unit (PDU) formats and even different transmission media (wired or wireless). The heterogeneity in bit rates and PhL PDU formats in a broadcast network imposes the consideration of some kind of traffic adaptation, in order to have a deterministic behaviour in the linking devices.

In many kinds of LANs this problem is solved by the linking devices, either by controlling traffic generation in transmitting stations (flow control) or by discarding frames. In a fieldbus network working in a broadcast fashion (all messages must be received by all stations) and where there are strict real-time and reliability requirements, another approach must be followed. In this paper, we propose a solution where the responsibility of traffic adaptation is given to the stations, based on the insertion of additional idle time between the transmission of consecutive frames.

We are considering a fieldbus network where the medium access control (MAC) protocol is based in a token passing procedure used by master stations to grant the bus access to each other, and a master-slave procedure used by master stations to communicate with slave stations. A master station is able to perform transactions during the token holding time. A transaction consists of the request from a master and the associated response frame. Transactions are atomic since requests will be followed by a synchronous response (positive or negative), i.e. the master will only process another transaction (or pass the token) upon completion of the ongoing transaction and waiting a pre-defined idle time. If an erroneous response frame is received or a timeout (before receiving any response) occurs, the master station may retry the request. A master station can also send unacknowledged requests. In this case, as there is no associated response frame, it will be able to start another transaction (or pass the token) just after a pre-defined idle time. The idle times between consecutive frames in the network should always be respected due to physical layer (PhL) requirements (namely for synchronisation).

In order to have a broadcast network, linking devices (which interconnect two different physical media) must act as repeaters. For simplicity we assume that the linking devices have a store-and-forward behaviour, i.e. a frame must be completely received by one port of the linking device before being re-transmitted to the other port. Obviously, the linking devices may have to support functionality such as encapsulation/decapsulation (due to potentially different PhL PDU formats).

The previously described MAC protocol, the linking devices' characteristics and the heterogeneous characteristics of the interconnected media impose the need to insert additional idle time between consecutive transactions, as a means of implementing traffic adaptation. In section 2 we describe how to compute the minimum idle time each master station must wait before issuing a request frame, in a way

that we guarantee that no linking device will experience increasing queue length. The inserted idle time for a certain master station mainly depends on the characteristic of the message streams of that master and on the frame formats and bit rates in the network. The results achieved in section 2 may be applied to any fieldbus network that copes with the previously described characteristics, seamless of the kind of transmission media that are involved.

Section 3 introduces a hybrid wired/wireless fieldbus system based on the EN50170 Profibus profile [1] that is under development - the RFieldbus system [2]. The wired and wireless PhL PDUs are also described. Then, in Section 4, the idle time parameters are computed, taking into account the particular characteristics of the RFieldbus system. The last section draws some conclusions about this paper, namely on the interest of the inserted idle time approach in the scope of the RFieldbus system.

## 2. Computation of the idle time

There are some particular conditions that must be satisfied in order to guarantee a correct behaviour of the overall token-passing fieldbus networks involving broadcast message transactions over heterogeneous transmission media. In this section, we show how traffic congestion can be avoided by inserting idle time before issuing request frames.

### 2.1 PhL PDU Length and Duration

In a broadcast fieldbus network, linking devices interconnect domains that use the same data link layer (DLL) protocol, but may have different physical layer (PhL). Therefore, we have to define some parameters for each domain:

| Parameter | Description | Units |
|---|---|---|
| L | Length of the DLL PDU | Chars |
| $k_a$ | Length of a char in the PhL of domain *a* | bits/char |
| $l_{a+}$ | PhL overhead of domain *a* (header, preamble, SFD) | Bits |
| $r_a$ | Bit rate in domain *a* | Mbit/s |

**Table 1: Parameters for frame length and duration**

A character (char) may be defined as the smallest unit of information in the DLL. A DLL PDU (Protocol Data Unit) is a set of chars delivered to the PhL for transmission. In order to proceed with this transmission, the PhL may have to introduce additional information (header) or synchronisation (preamble, start frame delimiter) bits. Moreover, a character of the DLL may have different lengths at the PhL, depending on the type of PhL.

Taking into account the parameters outlined in Table 1, we can define $C_a$ as the duration of a PhL PDU in domain *a*:

$$C_a = \frac{L \cdot k_a + l_{a+}}{r_a}$$

### 2.2 Traffic Congestion in Linking Devices

When interconnecting domains with different PhL frame formats and data rates, the queuing delay in the linking devices may increase, from one transaction to the next. The timing diagram depicted in Figure 1 illustrates a sequence of transactions between an initiator and a responder both in the same domain ($D_a$), and the resulting frames in the other domain ($D_b$). One linking device interconnects the two domains and it is assumed that the frame duration in $D_b$ is twice the frame duration in $D_a$.

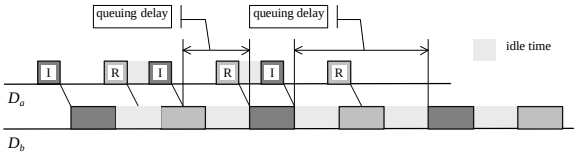


**Figure 1: Increasing queuing delay in a linking device**

Note that since the idle time is usually defined as the duration of a predefined number of (idle) bits separating consecutive frames in the network, its duration may be different for the two domains.

Clearly, if a request from an initiator in $D_a$ and a responder in $D_b$ appears after the last response shown in Figure 1, this transaction will be affected by the cumulative queuing delay in the linking device. The queuing delay in such a linking device depends on the number and duration of consecutive transactions where initiator and responder belong to $D_a$. Even a sequence of short frames may lead to very long message response times. For instance, a sequence of token passing between master stations that have nothing to transmit may also cause traffic congestion.

A way to avoid traffic congestion in linking devices (and long message response times) is through the insertion of an additional idle time before initiating a transaction. Obviously, the insertion of this additional idle time reduces the number of transactions per time unit when the responder is not in the same domain as the initiator. Nevertheless, the advantage of avoiding traffic congestion is enormous. It leads to a better responsiveness to failure (when an error occurs, retransmissions are undertaken sooner) and to smaller worst-case message response times.

### 2.3 Two Different Idle Time Parameters

In every master station, two different DLL idle time parameters should be defined - $T_{ID1}$ and $T_{ID2}$, related to acknowledged and unacknowledged requests, respectively. $T_{ID1}$ is the time that expires at the initiator after receipt of a response frame's last bit, until a new frame's bit is transmitted on the medium. $T_{ID2}$ is the time that expires between transmitting the last bit of an unacknowledged frame and transmitting

the first bit of the next frame. The need for having these two distinct idle time parameters is explained next.

For a single segment network, all stations may set their idle time parameters to a minimum value, usually big enough to cope with synchronisation requirements. In the following subsections, we are going to assume that all stations set these "minimum" idle time parameters to the same value[1], i.e. $T_{ID1}=T_{ID2}=T_{ID}$. Then, we compute the additional idle time each station must insert, in order to perform traffic adaptation. These inserted idle times are represented by $t_{ID1+}$ and $t_{ID2+}$. Finally, we merge the corresponding components into single parameters – $T'_{ID1}$ and $T'_{ID2}$.

As we will see, a master station could hold a unique idle time, i.e. wait the same idle time after receiving response frames or sending unacknowledged requests. Nevertheless, this would demand this unique idle time to be the maximum between $T'_{ID1}$ and $T'_{ID2}$. Obviously, this would lead to a non-optimal situation. Indeed, previous results say that $T'_{ID2}$ is usually smaller than $T'_{ID1}$. If we considered a unique idle time (that is the maximum between the two), we would penalise unacknowledged requests (inserting more idle time than needed). The following sections show how to set both idle times.

### 2.4 Computing $t_{ID1}$

In order to compute the inserted idle time after receiving a response frame ($t_{ID1a+}$), we will refer to Figure 2. A sequence of message cycles similar to the one shown in Figure 1 is presented, but now including the inserted idle time. For the sake of simplicity, the timing diagram depicted in Figure 2 assumes that the frame duration in $D_b$ is twice the frame duration in $D_a$.

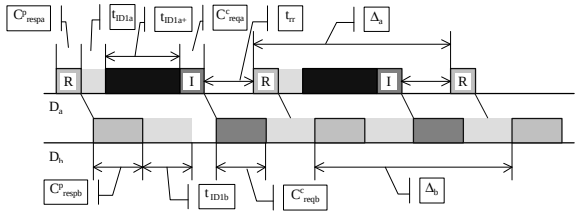


**Figure 2: Inserting additional idle time (acknowledged request sequence)**

The responder's reaction time is represented by $t_{rr}$ (assumed to be constant for every station) and the superscript indexes *p* and *c* correspond to *previous* and *current* (transaction), respectively.

Clearly, the increase in idle time ($t_{ID1a+}$) guarantees that there will be at most two messages in a linking device's queue, one being processed and the other one waiting to be served.

Reporting to Figure 2, we can state that $D_a$ should be greater or equal to $D_b$, in order to be able to avoid the increase in the queue. That is, considering that $\Delta_a = C^p_{respa} + t_{ID1a} + t_{ID1a+} + C^c_{reqa} + t_{rr}$ and $\Delta_b = C^p_{respb} + t_{ID1b} + C^c_{reqb} + t_{ID1b}$, then:

$$t_{ID1a+} \geq \left(C^p_{respb} - C^p_{respa}\right) + \left(C^c_{reqb} - C^c_{reqa}\right) + \left(2 \cdot t_{ID1b} - t_{ID1a}\right) - t_{rr} \quad (1)$$

In order to compute the value for $t_{ID1+}$ for a given master station, there is the need to know the characteristics of the message streams related to that master. Therefore, we must know the length of the different DLL request/response PDUs for every acknowledged request and of the different DLL unacknowledged request PDUs for that master.

### 2.5 Computing $t_{ID2}$

The condition expressed in (1) is just related to acknowledged request frames, though. The case of a sequence of non-acknowledged request (or token) frames must also be analysed. Figure 3 shows a sequence of unacknowledged requests and the variables that are necessary to evaluate this second idle time.

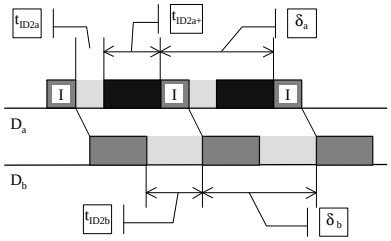


**Figure 3: Inserting additional idle time (unacknowledged request sequence)**

In this case, $d_a$ should be greater or equal to $d_b$. That is, considering $d_a = C_{reqa} + t_{ID2a} + t_{ID2a+}$ and $d_b = C_{reqb} + t_{ID2b}$, then:

$$t_{ID2a+} \geq \left(C_{reqb} - C_{reqa}\right) + \left(t_{ID2b} - t_{ID2a}\right) \quad (2)$$

Again, the different DLL unacknowledged request PDUs lengths for that master must be known, in order to compute $t_{ID2+}$.

Finally, assuming only one register for $T_{ID1}$ and one register for $T_{ID2}$, there is the need to merge both the "conventional" idle time with the inserted idle time in one variable, i.e.:

$$t'_{ID1a} = t_{ID1a} + t_{ID1a+} \wedge t'_{ID2a} = t_{ID2a} + t_{ID2a+}$$

Or, in bit times:

[1] Note that this is the idle time all hopping devices will use, when relaying traffic from one port to the other.

$$T'_{ID1a} = T_{ID1a} + T_{ID1a+} \wedge T'_{ID2a} = T_{ID2a} + T_{ID2a+}$$

We are considering also that a master station will insert $T'_{ID2}$ after receiving a token frame[2]. This is true since a sequence of token passing between master stations that have nothing to transmit may also "jam" the linking device.

Concerning the idle time used by the linking devices when relaying frames, they only insert the "conventional" idle times, i.e. $T_{ID1}$ and $T_{ID2}$. In order to avoid the need for the linking devices to decode the DLL PDU (to know if it is a acknowledged or unacknowledged request), both "conventional idle times should be set to the same value.

### 2.6 Algorithm for the computation of the idle times

The methodology presented in sections 2.4 and 2.5 permits to set both idle time parameters in a per-station basis, taking into account all possible transactions (message streams) for that master station. In this sense, each master station in the network would have a unique pair ($T'_{ID1}$, $T'_{ID2}$) of idle time parameter values.

For the sake of simplicity, we present a simplified algorithm that returns the same idle time parameter values for all masters stations in a given domain (therefore, in a per-domain basis). Therefore, instead of considering the particular set of message streams for each master station, we opt for a worst-case scenario where maximum and minimum frame lengths for the overall network are considered. This requires the definition of some additional network parameters:

| Description | Symbol |
|---|---|
| Maximum length of DLL request PDU | $L_{req}^{max}$ |
| Maximum length of DLL response PDU | $L_{resp}^{max}$ |
| Minimum length of DLL request PDU | $L_{req}^{min}$ |
| Minimum length of DLL response PDU | $L_{resp}^{min}$ |

**Table 2: Additional parameters**

Moreover, the maximum length of acknowledged and unacknowledged DLL request PDUs is considered to be the same. The following algorithm for the computation of $T'_{ID1}$ and $T'_{ID2}$ is proposed:

*Define values for network-specific parameters:*

$L_{req}^{max}$ , $L_{resp}^{max}$ , $L_{req}^{min}$ , $L_{resp}^{min}$

$t_{rr}$

*For every domain i, define values for domain-specific parameters*

$L_{i+}$ , $k_i$ , $r_i$

$T_{ID1i} = T_{ID2I} = T_{ID}$

*For every domain i*

*Computation of the idle time after receiving response frame ($T'_{ID1i}$)*

*For every domain j¹i*

*Choose* $L_{req}$ *and* $L_{resp}$ *, such as*

$$L_{req}^{min} \le L_{req} \le L_{req}^{max} \wedge L_{resp}^{min} \le L_{resp} \le L_{resp}^{max}$$

*and in order to maximise the ratio* $\frac{C_j}{C_i}$

*Compute* $t_{ID1i+}$ *using the formula*

$$t_{ID1i+} \ge \left(C_{respj} - C_{respi}\right) + \left(C_{reqj} - C_{reqi}\right) + \left(2 \cdot t_{ID1j} - t_{ID1i}\right) - t_{rr}$$

*Choose the highest idle time (considering all domains)*

*Set*

$$t'_{ID1i} = t_{ID1i} + t_{ID1i+} \wedge T'_{ID1i} = T_{ID1i} + T_{ID1i+}$$

*Computation of the idle time after sending unacknowledged request or receiving the token ($T'_{ID2i}$)*

*For every domain j¹i*

*Choose* $L_{req}$ *such as*[3]

$$L_{req}^{min} \le L_{req} \le L_{req}^{max}$$

*and in order to maximise the ratio* $\frac{C_j}{C_i}$

*Compute* $t_{ID2i+}$ *using the formulas*

$$t_{ID2i+} \ge \left(C_{reqj} - C_{reqi}\right) + \left(t_{ID2j} - t_{ID2i}\right)$$

*Choose the highest idle time*

*Set*

$$t'_{ID2i} = t_{ID2i} + t_{ID2i+} \wedge T'_{ID2i} = T_{ID2i} + T_{ID2i+}$$

*end*

## 3. The RFieldbus System

The RFieldbus System is being specified in the scope of the European Union Project IST-1999-11316 RFieldbus - High Performance Wireless Fieldbus in Industrial Multimedia-Related Environment [2]. Within this project, Profibus [1] was chosen as the fieldbus platform. Essentially, extensions to the current Profibus standard are being developed in order to provide Profibus with wireless, mobility and industrial-multimedia capabilities. In fact, providing these extensions means fulfilling strong requirements, namely to encompass the communication between wired (currently available) and wireless/mobile devices and to support real-time

[2] This demands the decoding of the DLL PDU, in order for the master station to know if it received a token frame.

[3] While the maximum length of acknowledged and unacknowledged DLL request PDUs was considered to be the same, we present a more general algorithm, since it allows having two different values.

control traffic and multimedia traffic in the same network.

### 3.1 RFieldbus Network Topology

The RFieldbus network topology [3] is exemplified in Figure 4:

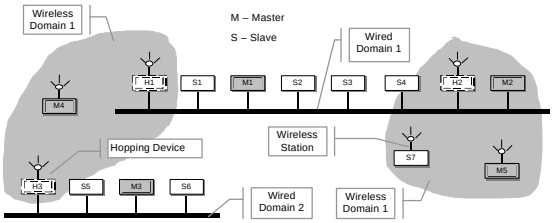


**Figure 4: RFieldbus network topology and components**

A domain consists of a set of stations communicating between them via a shared communication channel. Note that no registering mechanisms are needed since we assume that wireless domains operate in different radio channels. It is also important to note that inter-domain mobility is supported if in each wireless domain messages are relayed through a base station (with up-link and down-link channels instead of direct communication between the wireless nodes) and mobile stations are able to perform channel assessment and channel switching [4].

### 3.2 Wired and Wireless PhL Parameters

In the wired domains, the asynchronous version of Profibus (RS485) is used. Each PhL PDU consists of a number of characters – the UART characters, that are composed of 11 bits each (8+3). The wired PhL adds no overhead. When relaying a Profibus PhL PDU to a wireless domain, the linking device removes every extra 3 bits and encapsulates the entire data octets in the data part of the wireless PDU, adding a specific header. Also, there is the need to insert a preamble and a start frame delimiter (SFD) to the wireless PhL PDU, in order to allow its correct reception. Considering a 10 bytes header plus 53 μs preamble and SFD (resulting in a total overhead of 186 bit times) in the wireless PhL PDU, and referring to Table 1, we get:

| Wired domain | Wireless domain |
|---|---|
| $k_{wr}$=11 bits/char | $k_{wl}$=8 bits/char |
| $l_{wl+}$=0 bits | $l_{wl+}$=186 bits |
| $r_{wl}$=1,5 Mbit/s | $r_{wl}$=2 Mbit/s |

**Table 3: Parameter values in the RFieldbus system**

Thus, the duration of wired and wireless PhL PDUs can be computed as:

$$C_{wr} = \frac{L \cdot 11}{1{,}5} = \frac{22}{3} \cdot L \quad (ms) \quad \wedge \quad C_{wl} = \frac{L \cdot 8 + 186}{2} = 4 \cdot L + 93 \quad (ms)$$

Table 4 presents the PhL PDU duration of some DLL PDUs:

| PDU Type | L (chars) | $C_{wr}$ (μs) | $C_{wl}$ (μs) |
|---|---|---|---|
| Short acknowledge | 1 | 7.3 | 97 |
| Token | 3 | 22 | 105 |
| Fixed length no data | 6 | 44 | 117 |
| 50 data octets | 59 | 432.7 | 329 |
| 100 data octets | 109 | 799.3 | 529 |
| 150 data octets | 159 | 1166 | 729 |
| 246 data octets | 255 | 1870 | 1113 |

**Table 4: Frame length and duration**

From Table 4, it is clear that short frames have a longer duration in wireless domains while long frames take longer to transmit in wired domains.

## 4. Evaluating the Idle Time for the RFieldbus System

If the number of bits in the wireless and wired PhL PDUs was the same, there would be only need for inserting idle time in the stations belonging to the wireless domains (assuming 2 Mbit/s for wireless and 1,5 Mbit/s for wired). As this does not happen (different PhL PDU formats), wired stations also need to insert idle time (due to the sequence of short frames). The idle time parameters of wired and wireless stations are going to be computed taking into account the RFieldbus characteristics defined in Section 3. Additionally, the following values are assumed:

$$T_{ID1} = T_{ID2} = T_{ID} = 50 \text{ bit times} \quad | \quad t_{rr} = 100 \text{ ms}$$

### 4.1 Computation of $T'_{ID1}$

#### 4.1.1 For the wired stations

In order to see if there is need for additional idle time in the wired stations, we must consider the smallest frames possible. This is true since as the frames are shorter, the wireless frame duration gets increasingly higher than the wired frame duration. Thus, (1) turns into:

$$t_{ID1WR+} \geq \left(C_{respWL} - C_{respWR}\right) + \left(C_{reqWL} - C_{reqWR}\right) + \\ + \left(2 \cdot t_{ID1WL} - t_{ID1WR}\right) - t_{rr}$$

Now considering that in Profibus:

- The shortest response/acknowledgement frame is 1 character long (short ack frame).
- The shortest request frame is 6 characters long (fixed length with no data field)

This results in (refer to Table 4):

$$t_{ID1WR+} \geq (97 - 7.3) + (117 - 44) + \left(2 \cdot \frac{50}{2} - \frac{50}{1.5}\right) - 100 \approx 80 \text{ ms}$$

For a 1,5 Mbit/s data rate, this would imply an inserted idle of 120 bit times. Therefore, the idle time is:

$$T'_{ID1WR} = T_{ID1WR} + T_{ID1WR+} = 50 + 120 = 170 \text{ bit times}$$

4.1.2 For the wireless stations

In this case, the longest request/response frames should be considered (255 characters each) to compute the worst-case idle time. Equation (1) turns into:

$$t_{ID1WL+} \geq \left(C_{respWR} - C_{respWL}\right) + \left(C_{reqWR} - C_{reqWL}\right) + \\ + \left(2 \cdot t_{ID1WR} - t_{ID1WL}\right) - t_{rr} = \\ = (1870 - 1113) + (1870 - 1113) + \left(2 \cdot \frac{50}{1.5} - \frac{50}{2}\right) - 100 \approx 1556\, ms$$

For a 2 Mbit/s data rate, this would imply an inserted idle of 3112 bit times. Therefore, the idle time is:

$$T'_{ID1WL} = 50 + 3112 = 3162 \text{ bit times}$$

### 4.2 Computation of $T'_{ID2}$

4.2.1 For the wired stations

Equation (2) results in:

$$t_{ID2WR+} \geq \left(C_{reqWL} - C_{reqWR}\right) + \left(t_{ID2WL} - t_{ID2WR}\right)$$

Considering that the smallest unacknowledged frame is the token frame (3 characters):

$$t_{ID2WR+} \geq (105 - 22) + \left(\frac{50}{2} - \frac{50}{1.5}\right) \approx 75\, ms$$

For a 1,5 Mbit/s data rate, this would imply an inserted idle of 113 bit times. Therefore, the idle time is:

$$T'_{ID2WR} = 50 + 113 = 163 \text{ bit times}$$

4.2.2 For the wireless stations

Equation (2) results in:

$$t_{ID2WL+} \geq \left(C_{reqWR} - C_{reqWL}\right) + \left(t_{ID2WR} - t_{ID2WL}\right)$$

Considering the longest unacknowledged frame (255 characters):

$$t_{ID2WL+} \geq (1870 - 1113) + \left(\frac{50}{1.5} - \frac{50}{2}\right) \approx 766\, ms$$

For a 2 Mbit/s data rate, this would imply an inserted idle of 1532 bit times. Therefore, the idle time is:

$$T'_{ID2WL} = 50 + 1532 = 1582 \text{ bit times}$$

### 4.3 Summarising table

The following table summarises the values (in bit times) for the idle times:

| Station Type | $T'_{ID1}$ (bit times) | $T'_{ID2}$ (bit times) |
|---|---|---|
| Wired Master | 170 | 163 |
| Wireless Master | 3162 | 1582 |
| Linking Device | 50 | 50 |

**Table 5: Idle time values for the RFieldbus system**

## 5. Conclusion

This paper addresses the problem of traffic adaptation in broadcast fieldbus networks sharing the same DLL but containing different physical layers. In some LANs, linking devices solve the problem of traffic adaptation. They use flow control mechanisms to tell transmitting stations to pause or just discard frames, when an overflow situation occurs (or is predicted). However, we assumed a fieldbus network working in a broadcast fashion and with real-time and reliability requirements. Therefore, another solution to the traffic adaptation was proposed which principle is to force master stations to insert additional idle time before transmitting frames. Two different idle time parameters were defined – $T'_{ID1}$, $T'_{ID2}$. The former must be inserted when a master station has received a response frame. The latter must be respected after issuing an unacknowledged request. An algorithm for the computation of both idle times was presented. Then, we present a very basic description of the RFieldbus system, with its specific data rates and frame formats in the wired and wireless domains. Finally, section 4 is devoted to the computation of the idle time parameters considering the RFieldbus case.

The methodology to evaluate the idle time parameters is not only important for the traffic adaptation between wired and wireless domains (or generally between heterogeneous PhLs), but also to determine the duration of a message transaction [5], a relevant parameter in any real-time system's analysis.

## 6. References


[1] "General Purpose Field Communication System, Volume 2" – Profibus, European Norm EN 50170, 1996.

[2] Haehniche, J., Rauchhaupt, L., "Radio Communication in Automation Systems: the R-Fieldbus Approach", in Proceedings of the 2000 IEEE International Workshop on Factory Communication Systems, pp. 319-326, September 2000.

[3] RFieldbus Deliverable D1.3, "General System Architecture of the RFieldbus", Technical Report, June 2000

[4] RFieldbus White Paper, "Mobility Management in RFieldbus", RFieldbus Technical Report, January 2001.

[5] Alves, M., Tovar, E., Vasques, F., "Evaluating the Duration of Message Transactions in Broadcast Wired/Wireless Fieldbus", HURRAY-TR-0121, May 2001.