Hybrid wired/wireless implementations of Profibus DP: a feasibility study based on Ethernet and Bluetooth

Daniele Miorandi\textsuperscript{a}, Stefano Vitturi\textsuperscript{b,}\textsuperscript{*}

\textsuperscript{a}\textit{Department of Information Engineering, University of Padova, via Gradenigo, 6/B, IEIIT-CNR, Padua 35131, Italy}

\textsuperscript{b}\textit{Department of Information Engineering, Italian National Council of Research, University of Padova, via Gradenigo 6/B, IEIIT-CNR, Padua 35131, Italy}

Abstract

The communication networks used by factory automation systems are beginning to employ solutions that, until recently, were used in different fields. In particular, at the lowest level of automation, where, typically, real time communication between controllers and sensors/actuators is required, both Ethernet and wireless networks are becoming solutions of interest. However, the introduction of such technologies poses severe problems in terms of compatibility with already implanted fieldbuses. It is thus highly likely that, in the near future, modifications of the existing fieldbus standards will be considered, in order to accommodate hybrid network configurations, where communications take place on both Ethernet/wireless segments. This paper considers a very popular fieldbus protocol, Profibus DP, and analyzes the possibility of implementing it on hybrid wired/wireless networks, based on Ethernet and Bluetooth, respectively. Two basic configurations are analyzed and, after a description of how the Profibus DP protocol functions could be mapped on both Ethernet and Bluetooth, a theoretical study of the network behavior is carried out. Finally, simulations results are shown, which validate the theoretical analysis and show the soundness of the proposed protocol stack for industrial applications.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Fieldbus data link; Wireless networks; Bluetooth

1. Introduction

The industrial communication field has recently experienced the adoption of solutions which, until some years ago, were used exclusively in different technological areas. The most important of these innovations are those concerned with the ‘device level’, i.e. the lowest level of factory automation systems, where real time data exchange is required. In particular, they are represented by:

1. the introduction of Ethernet;
2. the use of wireless networks.

Ethernet, the well-known LAN standardized by IEEE [1], has been largely employed for several years in the factory environment as a backbone connecting different plant areas while its application to the device level has been prevented from the random medium access scheme. Recently, however, the Ethernet networks have gained the capability of communicating in real time, thanks to its enhancements, i.e. the increase of the transmission speed and the introduction of switches.

Ethernet interfaces working at transmission speeds of 100 Mbit/s are currently available also for very simple devices such as sensors/actuators typically employed at the device level. Moreover, an easy upgrade towards faster speeds (e.g. 1 Gbit/s) may be envisaged.

Switches are elements capable of linking either single Ethernet stations or network segments in an intelligent way. This means that a switch is able to recognize the addresses of the stations connected to it and to redirect the messages only to the destinations. Hence, the introduction of a switch eliminates or substantially reduces the number of collisions.

Switches are elements capable of linking either single Ethernet stations or network segments in an intelligent way. This means that a switch is able to recognize the addresses of the stations connected to it and to redirect the messages only to the destinations. Hence, the introduction of a switch eliminates or substantially reduces the number of collisions.

Wireless networks are helpful, at the device level, when there is the necessity of exchanging data with either a sensor or an actuator mounted on a moving device. Until some years ago there was no possibility of integrating the sensor/actuator on a network, since only wired implementations were available. Nowadays, instead, we can make use of wireless connections which are becoming available also in the industrial environment.
It has to be observed, however, that field networks (fieldbuses) have been employed since long time at the device level of factory automation systems. Thus, both Ethernet and wireless networks are going to be integrated with fieldbuses using different techniques, originating in such a way several types of hybrid solutions.

For example, the IEC 61158 standard [2] includes Ethernet versions of some fieldbus protocols, such as Interbus [3] and ControlNet [4], while a project set up in the Fifth Framework Program of the European Community, named R-FIELDBUS [5], is aimed at studying the implementation of a wireless fieldbus.

In this paper we take into consideration a very popular fieldbus protocol, Profibus DP [6], and analyze the possibility of implementing it on wired/wireless network configurations based on both Ethernet and Bluetooth. More in detail, we consider hybrid realizations of Profibus DP using Ethernet at both the physical and data link layers of the wired segments and, in the same way, the Bluetooth radio system for the wireless segments.

As it will be shown, our proposal maintains the Profibus DP interface towards the user applications, and, hence, ensures the compatibility with previous installations.

Bluetooth has been preferred to similar products, such as the IEEE 802.11 wireless LAN standard [7], because of both the expected low cost of the Bluetooth interfaces and the easy integration on sensors/actuators. It has also been considered that, both the short geographic range covered by Bluetooth transceivers and the relatively low number of devices imposed by the typical Bluetooth configuration, called piconet, do not represent a real limitation for this kind of network. In fact, most practical applications are characterized by the presence of a very short number of moving devices concentrated on a restricted area.

The paper is organized as follows: Section 2 illustrates shortly the main features of Profibus DP, Bluetooth and Ethernet, respectively. Section 3 introduces the hybrid configurations of Profibus DP we considered along with the description of how they could be implemented. Section 4 carries out a theoretical analysis to evaluate two important performance indexes of the proposed configurations; finally, the results are compared with those obtained from extensive numerical simulations.

2. Some basic features of Profibus DP, Bluetooth and Ethernet

2.1. Profibus DP

Profibus DP is a fieldbus designed to operate at the device level of factory automation systems where it allows real-time data exchange between controllers and field devices such as sensors and actuators.

Fig. 1 shows the Profibus DP communication profile. As can be seen, the high level protocol is implemented by two sublayers: the User Interface and the Direct Data Link Mapper (DDLM), whereas the access to the transmission medium is realized by the Fieldbus Data Link (FDL) protocol whose features are very similar to those defined by the IEEE 802.4 [8] token bus.

In detail, FDL specifies that either passive or active stations may be connected to the network. A token is let circulate among active stations, that form in this way a logical ring, granting the right to access the channel. The station which owns the token may hold it for an interval not longer than the token holding time, $T_{TH}$, calculated as the difference between the target token rotation time, $T_{TT}$ and the real token rotation time $T_{TR}$:

$$T_{TT} = T_{TR}$$

$T_{TT}$ represents the maximum token rotation time, and is set on all the stations during the network setup, while $T_{TR}$ is measured by the station at each token reception.

The Profibus DP protocol is based on the master-slave principle: every master station, after initialization, enters the cyclic data exchange phase, where it polls regularly its slaves. It is responsibility of the User Interface to handle the function requests which allow for the correct operation of the protocol. Examples of such functions are: data exchange, parameterization, configuration and diagnostic reading. In particular, the data exchange function is used to poll the slaves, while both parameterization and configuration are executed during the initialization phase. The diagnostic reading function is used to obtain information about the status of the slaves. Although such a function could be used in all phases of the Profibus DP operation, its main employment is during the data exchange, where it allows for the transmission of alarms. More in detail a slave, when polled, may signal the presence of a diagnostic message (alarm) to be read. As a consequence, at the end of the current polling cycle, the master is compelled to query that slave again to get the diagnostic data. The User Interface functions are mapped onto FDL services by the DDLM. In particular, the ‘Send and Request Data with reply’ (SRD) service is used. SRD is a confirmed service
which permits the transmission of up to 246 bytes of user data from a source station to a destination station. The latter, in the confirm frame, may include as many bytes as in the request frame. It is worth mentioning that the User Interface functions are identified by means of the Service Access Points, SAPs, which are specified in the FDL Protocol Data Units.

The most important performance index of Profibus DP is the cycle time, defined as the interval elapsing between two consecutive polling of the same slave. Such a time is computed as the sum of both the time necessary to poll all the slaves and the time required by the execution of acyclic activities on the slaves (typically the diagnostic reading).

2.2. Bluetooth

Bluetooth [9] is the leading radio technology for providing low-cost short-range radio connectivity. Its development, supported by the Bluetooth Special Interest Group, has lead to a protocol specifically designed for low-power low-cost communications, with the foreseen aim of providing ubiquitous connectivity. Even if major attention has been drawn by applications in the field of wireless personal communications, it is likely that such a technology may successfully make its way in the industrial environment. Indeed, many features of the Bluetooth radio system, namely the low cost of the chipset, its low power consumption and the ease of deployment due to the ad-hoc operation mode, are well suited to the demands and necessities of the industrial communication world.

Bluetooth operates in the worldwide available unlicensed Industrial, Scientific and Medical band, centered around 2.4 GHz. To comply with existing regulations, in order to avoid interference with other devices working on the same band, Bluetooth encompasses at the physical layer a spread-spectrum frequency-hopping scheme, in which the carrier frequency changes in a pseudo-random fashion among 79 possible choices every 0.625 ms. The carriers are 1 MHz spaced and the raw rate of 1 Mb/s is achieved by means of a binary modulation scheme (a GFSK with time-bandwidth product of 0.5). The basic network architecture consists of a cluster of no more than eight devices, called piconet in the Bluetooth lexicon, which share the common frequency-hopping channel in a master-driven way. Namely, one unit is elected master of the piconet, and regulates the access to the channel by polling the other units, which act as passive devices and are therefore referred to as slaves. The standard encompasses also the possibility of building up multi-hop networks, called scatternets, where time-shared units act as gateways among the piconets they belong to [10]. This issue is out of the scope of the paper, since we focus on the basic piconet configuration.

The system is slotted, one slot corresponding to $T = 0.625$ ms, and full-duplex is achieved by means of time division duplexing. According to the standard, master transmissions may start in even-numbered slots only, whereas odd-numbered slots are devoted to the reverse link. The standard defines two types of link, namely, Asynchronous Connectionless (ACL) and Synchronous Connection-Oriented (SCO). For ACL links, six different packet formats are supported, which differ for both forward error-correction features (either a (15, 10) shortened Hamming code or none), and packet length (I, 3 and 5 time slots), as shown in Fig. 2. A resume of the packet characteristics is reported in Table 1.

The protocol specifies a layered communication profile (reported in Fig. 3), in which the lowest level, called radio

<table>
<thead>
<tr>
<th>Type</th>
<th>Slot occupancy</th>
<th>Max. payload length (bytes)</th>
<th>FEC rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMI</td>
<td>1</td>
<td>17</td>
<td>213</td>
</tr>
<tr>
<td>DM3</td>
<td>3</td>
<td>121</td>
<td>213</td>
</tr>
<tr>
<td>DM5</td>
<td>5</td>
<td>224</td>
<td>213</td>
</tr>
<tr>
<td>DHI</td>
<td>1</td>
<td>27</td>
<td>No</td>
</tr>
<tr>
<td>DH3</td>
<td>3</td>
<td>183</td>
<td>No</td>
</tr>
<tr>
<td>DH5</td>
<td>5</td>
<td>339</td>
<td>No</td>
</tr>
</tbody>
</table>
The interface, specifies transmission power and pulse shaping masks. The baseband layer is in charge of operations such as forward error correction, automatic repeat request, encryption and CRC calculation. Link manager handles connection setup and release, whereas segmentation-and-reassembly, group management and quality-of-service provisioning are demanded to the Logical Link Control and Adaptation Protocol (L2CAP), which acts as interface to higher layers.

We remark that the standard does not specify the polling algorithm to be employed, whose choice is left to the manufacturer. In current implementations, however, a common solution is the basic Pure Round Robin (PRR) scheme. PRR results in very low-cost implementations and, since it does not require complex logic to be embedded in the chipset, helps maintaining the power consumption low. Hence, in the following, we will assume that PRR is the polling policy adopted to control the access to the channel.

2.3. Ethernet

The data link layer of Ethernet, as specified by the IEEE 802 committee, is composed of two sublayers: the Logical Link Control (LLC) [11] and the Medium Access Control (MAC). LLC is common to all the LANs standardized by the IEEE 802 committee and, depending on its class, it can provide various types of services to the users.

In particular, the acknowledged connectionless services which are available for both the LLC classes III and IV, are characterized by an interface to the higher layers which is very similar to that of FDL. For this reason, as we will show, the Ethernet implementation of the Profibus DP protocol that we propose makes use of LLC.

The medium access protocol of Ethernet is known as Carrier Sense, Multiple Access with Collision Detection (CSMA/CD), and the relevant standard has been issued by the IEEE 802.3 subcommittee. The Protocol Data Unit of the IEEE 802.3 is shown in Fig. 4.

The minimum length of the PDU is fixed at 512 bits. Consequently the field 'Pad', if necessary, is filled with a suitable number of bytes. The minimum length is computed starting from the 'Destination address' field, hence the size of the shortest frame is 576 bits. Moreover, the standard foresees a minimum inter-packet gap of 96 bits, that is, every transmitting station has to ensure the presence of this minimum interval between any two subsequent frames sent on the network. It is worth noting that data fields containing up to 48 bytes are transmitted with a PDU of the minimum size. Thus, considering that the data exchanged between controller and sensors/actuators typically consist of a few bytes, it is very likely that, when Ethernet is employed for the communication at the device level, only PDUs of the minimum size are used.

3. Hybrid wired/wireless configurations of Profibus DP

In this section we will consider two different hybrid Profibus DP networks based on both Ethernet and Bluetooth: a monomaster configuration in which the slaves are located on both the wired and wireless segments, and a multimaster configuration in which, with respect to the previous one, a master with slaves located only on the wired segment is added.

The analysis of only the above two configurations is not a limitation since, as it will be clear, the results obtained may be easily extended to more complex network configurations comprising a higher number of devices.

3.1. Monomaster configuration

This configuration, as shown in Fig. 5 is characterized by the presence of only one master device with two different hardware interfaces toward both the wired (Ethernet) and wireless (Bluetooth) segments, necessary for the connections with the slaves located on both segments.

From the protocol point of view, the master has to be able to access both the segments and, at the same time, it must maintain the interface specified by Profibus DP toward the user applications. A suitable communication profile of the master is shown in Fig. 8. As can be seen, for the wired segment, a modified version of the Profibus DP Direct Data...
Link Mapper, named MD-DLM, is used. MD-DLM has the task of mapping the functions specified by the User Interface onto the LLC acknowledged connectionless services. In particular, the DL-REPLAY service has been used, since it is very similar to SRD. Moreover, LLC allows for the adoption of Service Access Points; hence they can be used to specify the Profibus DP functions in the same way as done in the original protocol. As an example, Fig. 6 shows how the Profibus DP data exchange function might be implemented. The User Interface at the master side issues the MDDLM_Data_Exchange request primitive. As a consequence, the MDDLM generates the DL-REPLAY request primitive to the LLC. The output data are then encapsulated into a LLC PDU and subsequently transmitted on Ethernet. At the slave side, the input data for the master have to be prepared in advance by means of the MDDLM_Data_Exchange_Upd function. The request primitive of such a function triggers the execution of the DL-REPLAY-UPDATE request to the LLC, which has the effect of storing the input data. At the arrival of the master request PDU, the LLC of the slave sends back a response PDU carrying the input data. Contemporaneously, the DL-REPLAY indication primitive is issued to the MDDLM of the slave which, in turn, makes available the output data to the User Interface by means of the MDDLM-Data-Exchange indication primitive. In the same way, the arrival of the PDU carrying the input data at the master side triggers the generation of the DL-REPLAY-STATUS indication primitive to the LLC which in turn issues the MDDLM-Data-Exchange confirm primitive carrying the input data to the User Interface.

A complete description of how the Profibus DP protocol may be implemented on Ethernet is given in Ref. [12].

A similar solution has been adopted to realize the communication on the wireless segment, as described in Ref. [13]: in this case the modified version of the DDLM, named BTDDLM uses the services of the Bluetooth Logical Link Control and Adaptation Protocol, L2CAP, to implement the User Interface functions.

However, differently from LLC, L2CAP does not allow for the use of SAPS. For this reason, it is necessary to reserve the first two bytes of the payload to specify the User Interface functions. Moreover, both this solution and the native polling of Bluetooth prevent of implementing the User Interface functions in the same way of Profibus DP, since it is not possible to store data in advance at a specified SAP and wait for the arrival of a service primitive to transmit them. Consequently, a different

---

Fig. 5. Monomaster hybrid configuration with wired/wireless slaves.

Fig. 6. Implementation of the Profibus DP data exchange function on Ethernet.
approach has been chosen. As an example, Fig. 7 shows how the data exchange function could be realized. As can be seen we have assumed that, independently, both master and slave prepare the data to be transmitted (with the primitives BTDDLM-Data-Exchange.req and BTDDLM-Data-Exchange_Upd.req, respectively). Both the output and input data are transferred to the Bluetooth radio system by means of the L2CAP Write service. When the packet carrying the output data arrives at the Bluetooth slave, the latter may include the input data in the return packet, sent in the immediately following slot. The transmitted data are then acquired by both master and slave via the L2CAP Read service.

It has also to be observed that the diagnostic reading function of the proposed wireless implementation of Profibus DP is slightly different from that of the original protocol: in fact, when a slave signals the presence of diagnostic, the master, at the end of the current cycle, has to stop the polling of the slaves and to issue the diagnostic reading request. This function may require some Bluetooth polling cycles, during which the data exchange function is necessarily not executed. For these reasons we use a different technique: during the data exchange, a slave may replace the input data with the diagnostic data. The master, when analyzing the data received from the polled slave, recognizes the diagnostic message and undertakes the appropriate actions. It is clear that, with this procedure, when an alarm occurs on a slave, the input data of that slave are not updated for one polling cycle. Nevertheless, we believe that this drawback is much more acceptable than the delays that could arise following the traditional implementation.

Fig. 8 also shows that the User Interface uses an adaptation sublayer to access both the modified versions of the DDLM. This is necessary because the User Interface is not aware of the type of link (either wired or wireless) the slave is connected to. In practice, the adaptation sublayer has to maintain a table with the location of the slaves, in order to route the requests coming from the User Interface over the right segment.

![Fig. 7. Implementation of the Profibus DP data exchange function on Bluetooth.](image)

![Fig. 8. Communication profile of a profibus DP hybrid master.](image)
3.2. Multimaster configuration

The considered configuration is shown in Fig. 9. The presence of a second master on the wired segment requires a careful analysis of the network access procedure because, unless a switch is used, the multiple access specified by Ethernet could generate collisions with consequent unpredictable delays on the message delivery. A possible solution to this problem is represented by the introduction of a token passing scheme. However, contrarily to the original version of Profibus, where such a feature is supplied by the data link layer, in this case it is convenient to realize it directly in the modified version of the DDLM. An interesting possibility, for example, could be the implementation of an implicit token passing procedure, similar to that of ControlNet. In detail, every master station connected to the network maintains an implicit token register which contains the address of the token-holding master. This value is incremented by one at the end of the operations performed on the slaves belonging to that master (i.e. when the master would pass the token). Immediately after, each master station compares its address with that contained in the implicit token register and, if they match, that master has the right to transmit. If a station is not present, in the fraction of time reserved to that station there will not be activity on the network. In this case, after a time-out, the address contained in the token register is newly incremented by all the stations and the implicit token is passed on.

Such a procedure fails if the time required to poll the wireless slaves is greater than the above time-out. In this case, after a time-out, the address contained in the token register is newly incremented by all the stations and the implicit token is passed on.

4. Performance analysis

In this section we provide a mathematical framework which, under some simplifying but reasonable assumptions, allow us to evaluate two metrics of interest, namely the average cycle time and the mean alarm delay (defined as the time encompassed between an alarm generation at a slave and its reception by the master). We will limit ourselves to the two basic configurations described in the previous section. However, as it will become apparent, the framework straightforwardly extends to more complex scenarios. We begin by introducing some notation; for any given random variable $X$, we will indicate its mean with $\bar{x} = E[X]$, where $E[\cdot]$ is the expectation operator, its second-order moment as $\bar{x}^2 = E[X^2]$ and its variance with $\sigma_x^2 = E[(X - \bar{x})^2]$. In the analysis, $T_C$ will denote the cycle time, $B_i$ the service time (poll + response) for the $i$-th slave, and $A_i$ the service time for the acyclic services (diagnostic reading) which may be required by slave $i$. To distinguish between wired and wireless slaves, we will add to the random variables of interest a superscript of the form $w_d$ or $w_l$, respectively. Finally, $D_i$ is a binary random variable, taking values in the alphabet $\mathcal{B} = \{0, 1\}$, which indicates whether slave $i$ presents a diagnostic message to be read. We model alarm generations by means of independent Poisson arrival processes of intensity $\lambda_i$. Hence, in steady-state conditions, the number of arrivals at queue $i$ during a cycle, denoted by $N_i$, turns out to be a Poisson random variable of mean $\lambda_i T_C$. Furthermore, since a slave may generate no
more than one diagnostic reading request in a cycle, we have:

\[ E[D_i] = P[D_i = l] = P[N_i ≥ 1] = 1 - e^{-\lambda t_C} \approx \lambda t_C. \]  

Note that the last approximation holds as far as \( \lambda t_C \ll 1 \), which is usually the case. For the sake of simplicity, we take \( \lambda = \lambda V_i \). In this work, we do not consider neither link failures nor access collisions which may occur on the wired segment.

4.1. Monomaster configuration

Let us start by considering a monomaster configuration, presenting \( N^{wd} \) slaves connected to the wired segment and \( N^{wl} \) slaves operating on the Bluetooth radio system, as depicted in Fig. 5.

We denote by \( T^{wd} \) the time spent for serving the wired slaves (both data exchange and acyclic services). Thus:

\[ T^{wd} = \sum_{i=1}^{N^{wd}} (B_i^{wd} + D_i^{wd} + A_i^{wd}). \]  

In a symmetrical scenario, \( B_i^{wd} = B_i^{wd} \), \( A_i^{wd} = A_i^{wd} \) and \( D_i^{wd} = D_i^{wd} \). The mean service time for a wired slave may be easily computed by recalling that Profibus packets are highly likely to be encapsulated in minimum-size Ethernet PDUs. Taking into account both the MAC preamble, start frame delimiter and the interframe gap (set to the minimum, i.e. 0.96 \( \mu \)s), we get:

\[ B^{wd} = 2 \left( \frac{512 + 64 + 96}{96} \right)b = 13.44 \text{ \( \mu \)s}, \]  

where \( R^{wd} = 100 \text{ Mbps} \) is the Ethernet transmission rate. By considering that also diagnostic messages nicely fit into minimum-size packets, we get \( A^{wd} = B^{wd} \). Since both \( A^{wd} \) and \( B^{wd} \) turns out to be pseudo-random variables, we will use in the following, with an abuse of notation, \( a^{wd} \) and \( b^{wd} \), respectively, to indicate the (unique) values assumed by these random variables.

Let us denote with \( T^{wl} \) the time elapsed between the polling of the first wireless slave and the reception by the master of the data of the last wireless slave. Neglecting propagation time and processing time at the receiver, we have:

\[ T^{wl} = \sum_{i=1}^{N^{wl}} B_i^{wl} - T + W_{tx}^{wl}, \]  

where the last term represents the transmission time for a Bluetooth packets. For a DH1 packet, taking into account both access code and header, we have:

\[ W_{tx}^{wl} = \frac{72 + 58 + 216}{R^{wl}} = 346 \text{ \( \mu \)s}, \]  

where \( R^{wl} = 1 \text{ Mbps} \) is the transmission rate of a Bluetooth transceiver. After the completion of operations on the wireless slaves, the master starts serving the wired ones. During this period, the Bluetooth piconet’s clock keeps on going. If the operations on the wired segment are completed in no more than:

\[ T^{max} = T - W_{wx}^{wl} = 279 \text{ \( \mu \)s}, \]  

the cycle time stabilizes on the Bluetooth cycle time, and we thus have:

\[ T_C = T^{wd}. \]  

In case \( T^{wd} > T^{max} \), the network loose synchronism. Assuming that the Blue-tooth master may schedule the beginning of a polling cycle at any even slot boundary, regardless of the client actually in service, we waste an even number of slots. Indeed, after some easy algebra, we get:

\[ T_C = \begin{cases} T^{wd} + T^{wd} - \hat{T}^{wd} + 2T, \quad \hat{T}^{wd} > T^{max}; \\ T^{wd} + T^{wd} - \hat{T}^{wd}, \quad \hat{T}^{wd} < T^{max}; \end{cases} \]  

where \( \hat{T}^{wd} = T^{wd} \mod(2T) \).

Let us limit ourselves to consider \( 0 < T^{wd} < 2T \) (which is an interval of time sufficient to poll about 94 wired slaves). In order to compute the mean cycle time we need to find \( P[T^{wd} > T^{max}] \). The latter clearly depends on the mean cycle time \( t_C \). However, under the hypothesis that \( b^{wd} = a^{wd} = 13.44 \text{ \( \mu \)s} \) (i.e. that only minimum size Ethernet PDUs are used, either for polling or reading diagnostic from the slaves), then

\[ N^{wl}_T \leq \left\lfloor \frac{T^{max} b^{wd}}{2T} \right\rfloor = 20 \]  

is the maximum number of queries of the wired slaves per cycle which allows to maintain the synchronism on the network. Hence, for \( N^{wd} \leq 10 \), we will always have \( t_C = t^{wl} \), since even if all wired slaves require the reading of the diagnostic, \( T^{wd} < T^{max} \), and we say that the network is operating in synchronous regime. On the other hand, for \( 10 < N^{wd} \leq 20 \), the network will show a quasi-synchronous behavior, in the sense that most of the cycles will be regular and last \( i^{wl} \), but some sporadic ‘anomalous’ cycles of length \( i^{wl} + 2T \) will occur.

In the same way, the condition

\[ 20 < N^{wd}_T \leq \left\lfloor \frac{2T + T^{max} b^{wd}}{2T} \right\rfloor = 114 \]  

will ensure that \( t_C = i^{wl} + 2T \) if \( 20 < N^{wd} \leq 57 \), while, for \( 57 < N^{wd} \leq 114 \) there could be cycles of length \( i^{wl} + 4T \).

In the calculus of wireless slaves service times, we cannot neglect the impact of a lossy channel on network performance. We consider an additive white gaussian noise (AWGN) channel, with a signal-to-noise ratio (SNR) at the receiver equal to \( \Gamma \), and we denote with \( P_e(\Gamma) \) the packet error rate at a SNR of \( \Gamma \). Referring to [13] for more details, and recalling just that we use DH1 packets only, we may write the service time of a wireless slave as:

\[ B^{wd} = 2T[1 + \beta(1 - P_e(\Gamma))^2)], \]
where $\beta(x)$ is a geometric random variable, having probability mass function

$$p_{\beta(x)}(k) = P[\beta(x) = k] = x(1-x)^k, \ k = 0, 1, ...$$

Hence,

$$b^{wl} = 2T\{1 + E[\beta[1 - P(I)^2]]\} = \frac{2T}{(1 - P(I))^2}, \ (10)$$

The relationship between the SNR and $b^{wl}$ is reported in Fig. 10.

The analysis of the cycle time is thus completed; now we proceed to consider the other performance metric, the alarm latency. The alarm latency for a wire-less slave may be computed by noting that it consists of two terms: the access delay, that is the time elapsed between the generation of the alarm message and the reception by the slave of the master query, and the transmission delay (sum of the transmission time, propagation delay and processing time at the receiver). The access delay may be thought as the residual life in a renewal process having renewal period equal to the cycle time \[14\], while the transmission delay may be upper-bounded by $b^{wl} - T$. Thus, we obtain:

$$w^{wl} = \frac{t^2_c}{2t_c} + T \frac{2 - [1 - P(I)]^2}{[1 - P(I)]^2}.$$  \hspace{1cm} (11)

As far as the wired slaves are concerned, we face a slightly different formulation: alarms generated during a cycle time trigger a diagnostic read request at next cycle, which is served at the end of that cycle. Let us consider two possible situations: in the first, the wired slaves are polled first:

$$w^{wd} = \frac{t^2_c}{2t_c} + N^{wd}b^{wd} + N^{wl}b^{wl} + \frac{b^{wd}}{2}.$$  \hspace{1cm} (12)

In the second case, the wired slaves come after the wireless ones in the polling cycle, and hence we have:

$$w^{wd} = \frac{t^2_c}{2t_c} + N^{wd}b^{wd} + \frac{b^{wd}}{2}.$$  \hspace{1cm} (13)

The second option, which is clearly preferable from the point of view of network performance, turns out to increase the fairness of the network. In Fig. 11 we reported a comparison (in terms of outcomes of a numerical simulation) of the mean alarm delay for wired slaves in a simple scenario ($N^{wd} = 3, N^{wl} = 4$). In Fig. 12 we show that, choosing the second option, the alarm latency of wired and
wireless slaves turns out to have similar values. On the contrary, if the first polling order was chosen, the alarm latency of wired slaves would clearly overcome that of the wireless ones, thus resulting in a less fair scheme.

The above equations show that, in order to evaluate the alarm latency, it is necessary to evaluate the second-order moment of the cycle time. Assuming that only minimum size PDUs are employed, and neglecting the randomness due to the occurrence of 'anomalous' cycles, we can write:

\[ \sigma^2_{T_c} = N^{wl} \sigma^2_{B^{wl}}, \]

where the variance of \( B^{wl} \) may be expressed as:

\[ \sigma^2_{B^{wl}} = 4T^2 \frac{1 - [1 - P(I)]^2}{[1 - P(I)]^4}. \]

Eventually,

\[ \tau_C^2 = (\tau_C)^2 + \sigma^2_{T_c}. \]

### 4.2. Multimaster configuration

Now we proceed to consider a basic multimaster configuration, made up of two masters, the first presenting \( N^{wd}_1 \) and \( N^{wl}_1 \) wired and wireless slaves, respectively, and the second having \( N^{wd}_2 \) slaves. Furthermore, we denote with \( r \) the time it takes to pass the token from one master to the other. According to the implicit token passing procedure described above, each station continuously monitors the channel, and, when no transmission are detected for a time interval of length \( r \), the token register is updated. We set \( r = b^{wd}/2 = 6.72 \text{ ms} \). For the calculus of the mean network polling cycle time, we tread the footprints of the monomaster case. Here we have to consider the values assumed by:

\[ \Phi^{wd} = \sum_{i=1}^{N^{wd}_1}(B_i^{wd} + D_i^{wd} + A_i^{wd}) + \sum_{i=1}^{N^{wd}_2}(B_i^{wd} + D_i^{wd} + A_i^{wd}) + 2r. \]

For a symmetrical scenario,

\[ \Phi^{wd} = (N^1 + N^2)(b^{wd} + Da^{wd}) + 2r. \]

As above, if \( \Phi^{wd} < T^{max} \), \( \tau_C = \tau^{wd} \), whereas, otherwise, we have to add to \( \tau^{wd} \) a multiple of \( 2T \). Neglecting the time spent to read diagnostic data on the wired slaves, we may take \( \Phi^{wd} = (N^1 + N^2)b^{wd} + 2r \), and the analysis follows straightforwardly. For the configuration under study, after some easy algebra we get:

\[ w^{wd}_1 = \frac{\tau_C^2}{2\tau_C} + N^1 b^{wd} + b^{wd} \]

\[ w^{wd}_2 = \frac{\tau_C^2}{2\tau_C} (b^{wd} - T). \]
Finally, the average delay for wired slaves may be computed as:

\[ w_{wd} = \frac{C^2}{2T_C} + N_{wd}^2 b_{wd} + \frac{b_{wd}^2}{2}. \]  

(21)

4.3. Simulation results and model validation

In order to validate our analysis, the proposed protocol stack has been simulated by using a commercially available software tool [15]. In Fig. 13 we reported some simulation results for very high SNR (>30 dB), where three parameters of interest (cycle time, alarm latency for the wired slaves and alarm latency for the wireless slaves) are plotted versus time for a scenario with \( N_{wd} = 3, N_{wl} = 7 \). In all
simulations, we assume $1/\lambda = 200$ ms, i.e. each slave generates, on average, five diagnostic messages per seconds. In such a scenario, the network operates in synchronous regime, and, furthermore, $\lambda t_c \ll 1$ holds. As it may be seen from Fig. 14, the network behavior stabilizes after a few seconds.

We simulated also a quasi-synchronous configuration, with $N^{wd} = 15$ and $N^{wl} = 1$ and $1/\lambda = 50$ ms. The behavior
of the cycle time is plotted in Fig. 15, where it is apparent that only two anomalous cycles of length $4T$ occur for a simulation time of 40 s.

Starting from the monomaster configuration, one of the basic thing to understand is the impact of the number of wireless devices on the network performance. Analytical and simulation results for the parameters of interest are plotted in Fig. 16, where the simulations have been run at high SNR. As expected, in the synchronous regime, the number of wired slaves has a negligible impact on the delay experienced by alarm messages. Another parameter of interest is the sensitivity of the system to a noisy environment: cycle time and alarm latency for various SNR are plotted in Fig. 17.

![Fig. 17. Network performance (mean values) for a monomaster configuration vs. SNR ($N_{wd} = 6, N_{wl} = 4$).](image)

![Fig. 18. Network behavior for a basic multimaster configuration ($N_{wd1} = 6, N_{wl1} = 3, N_{wd2} = 2$).](image)
As far as multimaster configurations are concerned, as above we plotted the simulation outcomes versus time, at very high SNR, for a simple configuration operating in the synchronous mode ($N_{wd}^1 = 6, N_{wl}^1 = 3, N_{wd}^2 = 2$) in Fig. 18 and then plotted the time-averaged results in Fig. 19. Table 2 reports a comparison of analytical and simulation results for this configuration.

On the whole, the good match between analytical and simulation results allow us to validate the model. It is clear that, in the framework of the proposed implementation of Profibus DP for hybrid networks, the wireless section behavior dominates that of the wired segment. However, the use, for connecting wired devices, of a high-speed Ethernet segment, helps keeping the alarm delay low. In particular, for SNR greater than 20 dB (that is, in normal operating condition for a Bluetooth network), the proposed implementation of Profibus keeps the performance well beneath the typical real-time constraints of industrial communications.

5. Conclusion

In this paper, we studied the feasibility of hybrid wired/wireless implementations of Profibus DP based on Ethernet and Bluetooth.

We considered two basic Profibus DP configurations, a monomaster and a multimaster one, and showed how they could be implemented using the services provided by the underlying networks.

Furthermore, we presented a theoretical framework, which enables the evaluation of two performance metrics, namely the cycle time and the alarm latency. The analysis, which may be straightforwardly extended to more complex configurations, has then been validated by means of extensive numerical simulations.

The results show that, in normal operating conditions, such an implementation is able to cope with the strict real time requirements of industrial communication systems.

References

[3] IEC 61158 parts 3 to 6: Digital data communications for measurement and control-Fieldbus for use in industrial control


