Enabling Inter-Domain Transactions in Bridge-Based Hybrid Wired/Wireless PROFIBUS Networks

Luís Ferreira, Eduardo Tovar, Mário Alves
Polytechnic Institute of Porto (ISEP-IPP)
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto, Portugal
E-mail: {llf@dei, emt@dei, malves@dee}.isep.ipp.pt

Abstract - **The marriage of emerging information technologies with control technologies is a major driving force that, in the context of the factory-floor, is creating an enormous eagerness for extending the capabilities of currently available fieldbus networks to cover functionalities not considered up to a recent past. Providing wireless capabilities to such type of communication networks is a big share of that effort. The RFieldbus European project [6,7,10] is just one example, where PROFIBUS was provided with suitable extensions for implementing hybrid wired/wireless communication systems. In RFieldbus, interoperability between wired and wireless components is achieved by the use specific intermediate networking systems operating as repeaters, thus creating a single logical ring (SLR) network. The main advantage of the SLR approach is that the effort for protocol extensions is not significant. However, a multiple logical ring (MLR) approach provides traffic and error isolation between different network segments. This concept was introduced in [8], where an approach for a bridge-based architecture was briefly outlined. This paper will focus on the details of the Inter-Domain Protocol (IDP), which is responsible for handling transactions between different network domains (wired or wireless) running the PROFIBUS protocol.**

I. INTRODUCTION

PROFIBUS [1] is one of the most popular fieldbuses, with several hundreds of thousands of installations currently in operation worldwide. It was standardised in 1996, as EN 50170 [2] and more recently, in 2000, by IEC as IEC61158 [1].

The research works on the timing behaviour of PROFIBUS networks [3-5] have proved the capabilities of this protocol to support distributed computer-controlled systems with stringent real-time requirements. More recently, there has been an eagerness for extending the capabilities of PROFIBUS to cover new functionalities like: industrial wireless communications [6, 7] and the ability to support industrial multimedia traffic [9].

The RFieldbus European project [6,7,10] is just one example of that effort, where PROFIBUS was extended to implement hybrid wired/wireless communication systems. In RFieldbus, repeaters are used to interconnect wired and wireless domains, resulting in just one token rotating between masters. The main advantage of such a single logical ring (SLR) approach is that the effort for protocol extensions is not significant.

However, there are a number of advantages in using a multiple logical ring (MLR) approach to support such type of hybrid systems. This concept was introduced and discussed in [8], where a bridge-based approach (thus, layer 2 interoperability) was briefly outlined. The paper included references to how some complex functionalities (such as the handoff between adjacent wireless cells) could be supported with minimum protocol extensions and still maintaining the compatibility with legacy PROFIBUS technologies.

The main advantage of a bridge-based solution is that it provides traffic segmentation, thus improved responsiveness for transactions between stations belonging to the same logical ring, and error containment within each logical ring. In such a system, transactions between stations in different logical rings are handled by an Inter-Domain Protocol (IDP). This protocol defines the format of the frames that are exchanged between bridges and the functionalities that must be supported by the bridges. The main contribution of this paper is the definition of the IDP.

The reminder of this paper is organised as follows. In Section II, some aspects of the PROFIBUS protocol that are relevant for the understanding and reasoning of the solutions and mechanisms outlined in Sections IV and V are presented. In Section III, we introduce the context and describe the main concepts related to bridge-based hybrid wired/wireless PROFIBUS networks. Then, in Section IV, we describe the main characteristics of the Inter-Domain Protocol (IDP), and in Section V, we describe how this protocol can be implemented. In Section VI, we compare the approach proposed in this paper with the single-logical ring approach and discuss the compatibility of the protocol with the PROFIBUS-DP application layer. Finally, in Section VII, we draw some conclusions.

This work was partially supported by the European Commission under the project R-FIELDBUS (IST-1999-11316) and by FCT under the project CIDER (POSI/1999/CHS/33139).

II. RELEVANT ASPECTS OF PROFIBUS

This section addresses some features of PROFIBUS that are relevant within the context of this paper.

A. Message Cycle

The PROFIBUS Medium Access Control (MAC) protocol uses a token passing procedure to grant bus access between masters, and a master-slave procedure used by masters to communicate with slaves.

A master station that sends an *Action Frame* (the first frame transmitted in a transaction) is said to be the initiator of the transaction, whereas the addressed one is the responder (a master or a slave). A transaction (or message cycle) consists on the request or a send/request frame from the initiator and of the associated acknowledgement or response frame of the responder.

Generally, all the stations except the initiator monitor all the requests and acknowledge/respond only if they are addressed. Moreover, the acknowledgement (or the response) must arrive before the expiration of the *Slot Time* (T_{SL}), otherwise the initiator repeats the request the number of times defined by the *max_retry_limit*, DLL variable.

A PROFIBUS master is capable of dispatching transactions during its token holding time (T_{TH}), which is given the value corresponding to the difference, if positive, between the target token rotation time (T_{TR}) parameter and the real token rotation time (T_{RR}). For further details, the reader is referred to [1, 2, 4].

B. Ring Maintenance Mechanisms

In order to maintain the logical ring, PROFIBUS provides a decentralised ring maintenance mechanism. Each PROFIBUS master maintains two tables – the *Gap List* (GAPL) and the *List of Active Stations* (LAS), and may optionally maintain a *Live List* (LL).

The *Gap List* consists of the address range from *TS* ('This Station' address) until *NS* ('Next Station' address, i.e., the next master in the logical ring). Each master station in the logical ring starts checking the addresses in its GAPL every time its *Gap Update Timer* (T_{GUD}) expires. This mechanism allows masters to track changes in the logical ring: addition (joining) and removal (leaving) of stations. This is accomplished by examining (at most) one *Gap address* per token visit, using the *FDL_Request_Status* frame.

The LAS comprises all the masters in the logical ring and is generated in each master station when it is in the *Listen Token* state, after power on. It is also dynamically updated during operation, upon receipt of token frames.

The *Live List* mechanism requires an explicit request from the PROFIBUS DLL user (via a management FMA1/2 request). This service returns the list of all active stations (masters and slaves).

C. Frame Formats

PROFIBUS defines 3 types of request/response frames: fixed length with no data field, fixed length with data field and variable data field length (Fig. 1.a), 1.c) and 1.d), respectively).

Each of these three types includes the following fields: *Destination Address* (DA), *Source Address* (SA), *Frame Control* (FC) and *Start Delimiter* (SDx). These frames also include the *Frame Check Sequence* (FCS) and the *End Delimiter* (ED).

Variable data field length frames additionally contain the field *Data Length* (LE and LEr) and they can optionally include the *Destination Address Extension* (DAE) and *Source Address Extension* (SAE) [2], in the *Data* field.

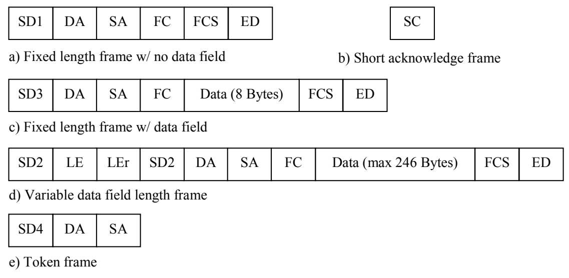

SD1 DA SA FC FCS ED
a) Fixed length frame w/ no data field
SC
b) Short acknowledge frame
SD3 DA SA FC Data (8 Bytes) FCS ED
c) Fixed length frame w/ data field
SD2 LE LEr SD2 DA SA FC Data (max 246 Bytes) FCS ED
d) Variable data field length frame
SD4 DA SA
e) Token frame

Fig. 1 PROFIBUS frame formats

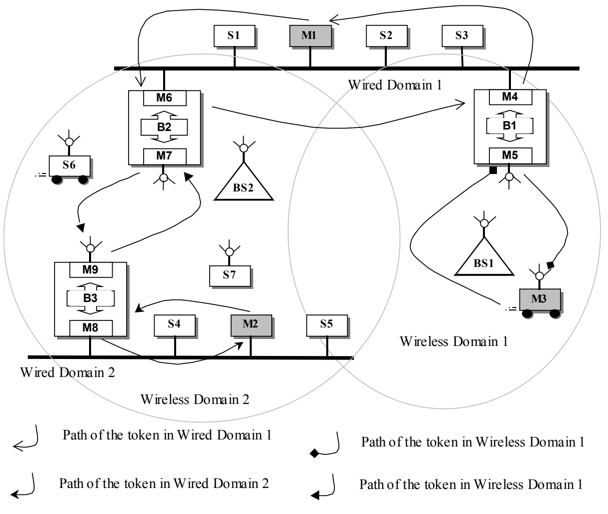
PROFIBUS also defines the *Short aCknowledge* frame (SC) and the Token Frame (Fig. 1.b) and 1.e)). The first consists of a single byte frame, and it is used as negative or positive acknowledge to a request. The second is passed between masters to grant medium access.

III. BASICS ON HYBRID WIRED/WIRELESS PROFIBUS NETWORKS

A hybrid wired/wireless fieldbus network is composed by stations with a wireless interface (usually radio) that are able to communicate with wired (legacy) stations.

The wireless part of the fieldbus network is supposed to include at least one *radio cell*. Basically, a radio cell can be described as a 3D-space where all associated wireless stations are able to communicate with each other. Our architecture considers two types of domains. A *Wired Domain* is a set of (wired) stations intercommunicating via a wired physical medium. A *Wireless Domain* is a set of (wireless) stations intercommunicating via a wireless physical medium. In the example of Fig. 2 the following set of wired PROFIBUS master (M) and slave (S) stations are considered: M1, M2, S1, S2, S3, S4 and S5. Additionally, the following set of wireless stations is considered: M3, S6 and S7. Within this set, only M3 and S6 are mobile. All wireless stations are assumed to be PROFIBUS stations with a wireless physical interface, capable of supporting radio communications and the mobility functionalities, like in RFieldbus [10]. Three bridge devices are considered: B1, B2 and B3.

In such a system, all communications are relayed through base stations: BS1 and BS2. Each base station uses two radio channels, one to transmit frames to wireless stations (the downlink channel), and another to receive frames from the wireless stations (the uplink channel). We will assume, in the remainder of the paper, that M5 and M7 include the base station functionalities in their wireless front-end, thus, structuring radio cells (wireless domains) 1 and 2, respectively.



S1
M1
S2
S3
Wired Domain 1
M6
B2
M7
M4
B1
M5
S6
BS2
BS1
M9
B3
M8
S7
M3
S4
M2
S5
Wireless Domain 1
Wired Domain 2
Wireless Domain 2
Path of the token in Wired Domain 1
Path of the token in Wireless Domain 1
Path of the token in Wired Domain 2
Path of the token in Wireless Domain 1

Fig. 2 Wireless PROFIBUS example network

Network operation is based on the Domain-Driven Multiple Logical Ring (MLR) approach, described in [8]. Therefore, each wired/wireless domain has its own logical ring. In Fig. 2, and in the remainder of this paper we are only representing bridges with two ports, but it should be noted that this approach could be generalised to bridges interconnecting *n* domains.

In this example, four different logical rings exist: {(M3 → M5), (M1 → M4 → M6), (M7 → M9), (M8 → M2)}.

We are also assuming that the network has a tree-like topology and that bridges perform routing based on MAC addresses.

IV. INTER-DOMAIN PROTOCOL

The communication between stations in different domains (Inter-Domain Transactions) is to be supported by the Inter-Domain Protocol (IDP). The IDP not only defines the format of frames exchanged between bridges, but also the functionalities that bridge devices must support.

A. Inter-Domain Transactions (IDT)

An Inter-Domain Transaction (IDT) is a transaction between an initiator and a responder belonging to different domains, i.e. with one or more bridges in the communication path.

When an initiator makes a request addressed to a station in another domain (an Inter-Domain Request), all stations belonging to the initiator's domain discard the frame, except the bridge masters belonging to that domain. The inter-domain request frame is relayed by only one of the bridge masters belonging to the domain (according to the routing mechanism). We denote this bridge master (the first bridge master in the path from the initiator to the responder) as BM_i, where *i* denotes the initiator. The relayed frame, denoted as an Inter-Domain Frame (IDF), is coded using the Inter-Domain Protocol (IDP). Bridges perform routing based on the MAC addresses contained in the DLL (frames). Frames are forwarded from one bridge master to the other if the destination address is included in the routing table of the incoming side. Obviously, every bridge must include two tables (one for each masters). This approach imposes the use of a single address space, where every station in the overall network has a unique MAC address.

The IDF embeds the original request or response and additional information that allows the decoding (of the embedded original frame) and the matching between the request and the respective response, as it will be detailed later on.

The IDF embedding the request is relayed by bridges until reaching the last bridge master in the path, bridge master BM_r (*r* denotes the responder). Then, this bridge decodes the original request frame and transmits it to the responder, which is a standard PROFIBUS station (for example a PROFIBUS-DP slave).

When BM_r receives the response to that request, it encodes the frame using the IDP and forwards it. The other bridges will relay this IDF until reaching bridge master BM_i, where it will be decoded and stored.

As the actual response to the original request takes more time than if the responder belongs to the same domain as the initiator, the initiator must periodically repeat the same request until receiving the related response. This means, in practice, that the request is not immediately responded. After BM_i having received (and stored) the correspondent response frame, then it is ready to respond to a new (repeated) request from the initiator. One of the objectives of this mechanism is to provide complete transparency from the point of view of both the initiator and the responder (to cope with PROFIBUS compatibility requirements). This is achieved by BM_i emulating the responder in a way that the initiator station considers the responder station as belonging to its domain, and by BM_r emulating the initiator in a way that the responder considers the initiator as belonging to its domain.

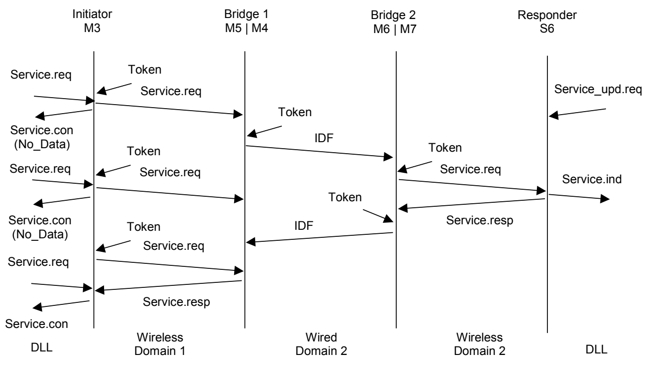


Fig. 3 Example for an Inter-Domain Transaction (IDT)

Considering the network scenario illustrated in Fig. 2, Fig. 3 represents a simplified timeline regarding a transaction between master M3 and slave S6.

Regarding Fig. 3 and the operation of the IDP, we assume that slaves read their inputs periodically, placing their image in the Data Link Layer (DLL), using the *Service_upd.req* primitive. The image of the input values is placed in a buffer, which is used by the DLL to build a response to a specific request. An indication is returned to the higher layers every time a slave receives a request. This behaviour is implemented by the PROFIBUS-DP protocol. On the initiator side, it is also necessary that the user of the DLL periodically repeats the same request. For every request, the DLL returns a confirmation, which can include “no data” if the response data is not available yet.

The mechanisms that must be implemented in the bridges are described in detail in the following section.

B. Bridge Inter-Domain Functionalities

A bridge must include one bridge master for each of its network accesses, which we assume to be one wired and one wireless (Fig. 4). In order to support the required functionalities, there must be a set of mechanisms related to the IDP, and two data structures: the *Routing Table* (RT) and the *List of Open Transactions* (LOT), associated to each bridge master.

When an initiator sends a request frame addressed to a station in another domain, the associated bridge masters in the path must be capable of relaying the IDF. This is possible only if the bridge masters are capable of receiving all request frames, notwithstanding the destination address (this is a functionality commonly found in bridges).

Each bridge must also support two routing tables (RT), one for each bridge master. The routing tables allow the receiving master of the bridge to know whether the received frame should be relayed to the other bridge master or not.

The bridge master belonging to the domain of the initiator (BM_i) must also be capable of matching a response to the related pending request. This is achieved using the information contained in the IDF embedding the response, and by using the *List of Open Transactions* (LOT).

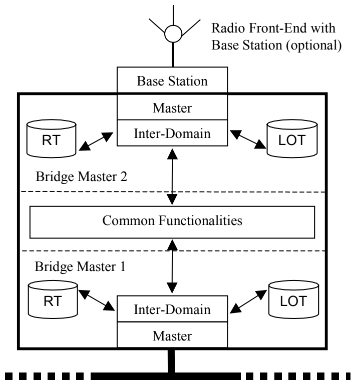


Fig. 4 Bridge components

The LOT contains information about the request frame, such as the Destination and Source Addresses. It also contains a transaction identification tag, the *Transaction Identifier* (TI), which must be included in both the IDF related to the request and also in the respective IDF response.The LOT is also used to manage the repetitions of the same request. Thus, for every arriving request, a BM_i consults its LOT and if that request is already listed, then it is discarded.

When an IDF embedding a response arrives at the BM_i, the respective request is searched in the LOT, and the response is associated and stored. This response is returned to the initiator when it repeats the original request. Refer to Section C for further details on the IDF format.

To comply with this, each bridge master must know which stations belong to its domain. This is possible by the use of the ring maintenance mechanisms defined in the PROFIBUS protocol - the *List of Active Stations* (LAS) and the *Live List* (LL).

Fig. 4 also depicts the “common functionalities” box. These functionalities are shared by both bridge masters, implement the interfacing between them and some features related to the support of inter-domain mobility [11]. This last feature will not be addressed in this paper.

C. Inter-Domain Frame Formats

Inter-Domain Frames (IDF) are used by the IDP for the proper transmission of frames between bridges. The operation of the protocol requires that these frames contain information that enables decoding the embedded original request/response and the matching between the information stored in the BM_i LOT and the respective response.

The PROFIBUS protocol defines that requests using variable data field length frames can be replied with a short acknowledge (SC) frame. Obviously, if no special IDF format was used, the bridges would be unable to route the SC frame back to the initiator station, since that type of frame does not have a Destination Address (DA) field. Also, the PROFIBUS protocol allows a request using a variable data field length frame with Destination Address Extension (DAE), to be answered by a response using fixed length frames without data field (thus not supporting DAE). So, BM_i would not be capable of matching two different requests from the same initiator, addressed to the same responder, but with different DAE. Therefore, to solve the first problem, it is required that every IDF must have a destination address field, while the second problem can be solved by using a *Transaction Identifier* (TI), which enables the matching of the request and the respective response.

It is also required that the IDF includes the *Embedded frame Function Code* (EFC) and the *Original Frame Type* (OFT), in order to allow decoding the embedded frame.

The TI is a sequence number, assigned by the BM_i, which should be included in the response frame (similar to a TCP/IP sequence number). This field is used by BM_i to distinguish between response frames related to different pending transactions.

Table 1 illustrates the proposed mapping between standard PROFIBUS frames and the IDFs. The EFC contains the Function Code (FC) of the embedded frame and finally the OFT field identifies the type of frame thus enabling its reconstruction. In the table, a grey rectangle means that the field is not used in the IDF because it is not present in the original frame. A dash indicates that the field is not available on that type of IDF. The equal symbol means that the field must the equal to the original embedded frame field.

Table 1 Mapping between standard PROFIBUS frames and IDFs

		Frame Header					**Frame Data**					
Type of Frame		**LE**	**SD**	**DA**	**SA**	**FC**	**DAE**	**SAE**	***TI***	***OFT***	***EFC***	**Data Unit**
Fixed length no data	Req	-	SD3	=	=	FC	▨	▨	TI	1	EFC	▨
	Ack	-	SD3	=	=	FC	▨	▨	TI	2	EFC	▨
	Short ack	-	SD3	Req. SA	Req. DA	FC	▨	▨	TI	3	EFC	▨
Fixed length w/ data	Req	Data len	SD2	=	=	FC	=	=	TI	4	EFC	=
	Res	Data len	SD2	=	=	FC	=	=	TI	5	EFC	=
Var. length	Req	Data len	SD2	=	=	FC	=	=	TI	6	EFC	=
	Res	Data len	SD2	=	=	FC	=	=	TI	7	EFC	=

In the conversion, the IDFs preserve the same DA and SA, except in the case of the short acknowledge frame, which does not have DA or SA. In this case, the IDF carries the DA and SA obtained from the request. To distinguish IDFs from other frame types, the Function code of the FC field must be equal to 0x0A, and its remaining sub-fields should be filled with the appropriate values (for a PROFIBUS frame). Finally, SDN frames do not need any conversion, so they can be relayed by the bridges as received (without being coded). Note that response frames are transformed into request frames by the IDP.
In this approach, the maximum size of the data unit is reduced by 3 bytes, i.e. to 241 bytes in frames using address extension, and to 243 bytes in frames without address extension. Nevertheless, this overhead of the protocol has a minor impact on network performance.

D. Illustration of the IDP

This section describes an example of one Inter-Domain Transaction (IDT), considering the network scenario of Fig. 2.

We assume that the only traffic in the network is related to the token passing and one IDT between master M3 and slave S6, respectively in wireless domain 1 and wireless domain 2. Fig. 6 illustrates all the frames exchanged, related to the transaction between M3 and S6.

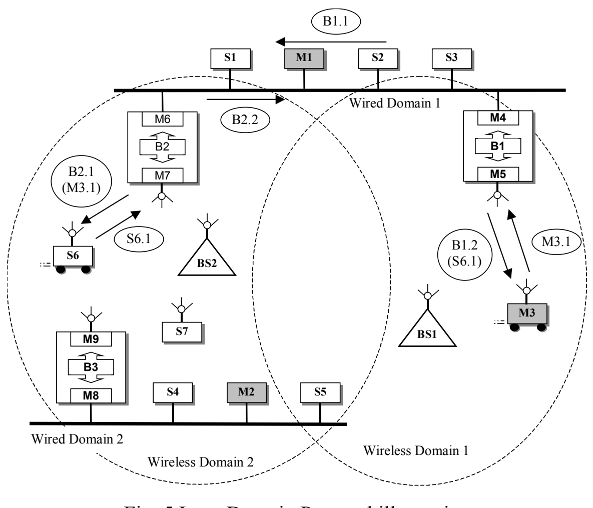


Fig. 5 Inter-Domain Protocol illustration

We denote a message related to a station as *Station_ID.n*, where *Station_ID* is the station or the bridge reference (e.g. M3) and *n* is a number identifying that message. In Fig. 5, the temporal order of the frames is the following: M3.1, B1.1, B2.1, S6.1, B2.2, M3.1 and B1.2. According to the IDP, request M3.1 must be repeated several times. Also note that frames B2.1 and B1.2 are equal to frames M3.1 and S6.1, respectively.

The request issued by M3 (M3.1) is addressed to S6, thus it is converted using the IDP and relayed to wired domain 1, without sending any reply to M3. Since M3 belongs to the same domain as bridge master M5, the latter adds a pending IDT to its LOT.

Frame B1.1, transmitted by M4, preserves the destination and source addresses of the original request. So, bridge master M6 receives the frame and forwards it to bridge master M7.

Since S6 belongs to the same domain as M7, M7 must decode the original request frame (M3.1) and send it. S6 receives the frame B2.1, decodes it and responds. Note that bridge master M7 will not create another entry on its LOT, since M3 does not belong to wired domain 1.

After receiving the response from S6 (S6.1), B2 codes it again using the IDP and relays to wired domain 1, with destination address M3 (B2.2). Meanwhile, M3 continues repeating request M3.1. When bridge master M5 receives that request, it consults its LOT and detects that a transaction with the same data is already going on, so it takes no action.

The response to request M3.1 is received by bridge B1 embedded in the IDF frame B2.2. Since the destination station belongs to the same domain as M5 and there is a related entry in the LOT, then M5 stores the response to M3.1 in a format equal to response S6.1.

When M3 repeats request M3.1, B1 replies using frame S6.1 and closes that particular pending transaction.

Fig. 6 shows a timeline that details the description above. Note the influence of the independent token rotations for the overall latencies of this particular transaction and the delay that exists in the bridges before a request is decoded and converted to an IDF (in this particular case the delays represented in the figure are exaggerated; in a real situation they should be much smaller).

As it can be seen from the figure, when bridge B1 receives the first request (M3.1), bridge master M5 initialises a pending transaction in its LOT. This transaction will only be deleted from the LOT when the respective response is received, and after another M3.1 request.

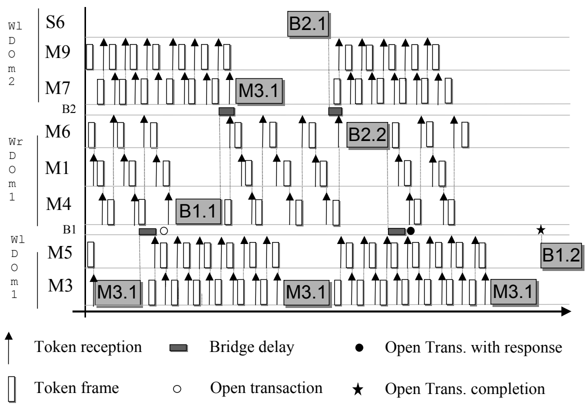


Fig. 6 Timeline example for an Inter-Domain Transaction

V. IMPLEMENTATION APPROACH

This section addresses how the bridge functionalities related to the IDP can be implemented. We assume that when a bridge master receives a request, it calls the *Indication_Handler* function. This function is responsible for replying to the initiator station or for forwarding the original request coded using the IDP to the other bridge master. We also assume that the reception of a response will be handled by a *Confirmation_Handler* function, which codes the response using the IDP and forwards it to the other bridge master.

A. Indication Handler

The *Indication_Handler* (Fig. 7) starts by checking the received *req_frame*, in order to determine the operation that will follow (line 5). For that propose, it uses the *req_frame* destination address, source address and FC code together with the information contained in the bridge master (RT, LL and LAS).

If the *req_frame* is addressed to the bridge, then it is processed according to its content. Requests can be addressed to the bridge during the mobility management procedure [11] or in case the bridge also integrates the functionalities of a common PROFIBUS master (e.g. a PLC).

If the bridge receives an IDF, then this frame is forwarded to the other bridge master, using the *Fwrd_ID_Request* function (described in Fig. 8). In case the initiator station belongs to the domain of the bridge master and the *req_frame* is addressed to a station in another domain, then the bridge master must try to initialise the LOT with another pending IDT, using function – *Init_ID_Request* (described in Fig. 9).

Broadcast frames must also be relayed to other domains and, at the same time, be processed by the bridge. In the other cases (e.g. when the frame is addressed to a station belonging to the same domain as the bridge master), the bridge master will not process the *req_frame*.

```
1.  Indication_Handler(req_frame)
2.  {
3.  // Checks the req_frame to determine
4.  // the operation that will follow
5.   res = check_addr(req_frame);
6.   Switch (res)
7.   {
8.  // When the req_frame is addressed
9.  // to the bridge
10.    case BRIDGE_ADDRESS:
11. // Process the message according
12. // to its contents
13.       process(req_frame);
14.    end;
15. // When the req_frame is an IDF that must
16. // be forwarded by the other bridge master
17.    case FWRD_ID_REQUEST:
18.       Fwrd_ID_Request(req_frame);
19.    end;
20. // When the initiator is on the bridge
21. // master domain
22.    case ID_REQUEST:
23.       Init_ID_Request(req_frame);
24.    end;
25. // req_frame sent in broadcast
26.    case BROADCAST:
27.      process(req_frame);
28.      Fwrd_ID_Request(req_frame);
29.    end;
30.    default:
31. // Do nothing
32.    end;
33.    …
34.  }
35. }
```

Fig. 7 Indication Handler, pseudo-code algorithm

The function *Fwrd_ID_Request* (Fig. 8) is called by the *Indication_Handler* function and it operates with the resources of the other bridge master (of the bridge). It starts by determining if the destination station is on its domain (line 3). If not, then the frame is queued on the output queue of the bridge master (line 33). Otherwise, the function determines the type of Inter-Domain Request (line 7).

If the *ID_req_frame* embeds a response frame that matches one entry in the LOT, then the *ID_req_frame* is decoded and a response (using the standard PROFIBUS format) is stored (lines 10 to 18). If the *ID_req_frame* embeds a request frame, then this frame is decoded. The information concerning this request is stored in order to enable the identification of the related response. Additionally, the frame is put in the output queue (using the standard PROFIBUS format) (lines 23 to 30).

```
1.  Fwrd_ID_Request(ID_req_frame)
2.  {
3.   res = is_station_on_domain(ID_req_frame);
4.   if res == 1 then // on the domain
5.   {
6.  // Det. type of frame
7.     type = type_of_ID_req(ID_req_frame);
8.     if type == RESP then
9.     {
10. // Accesses the LOT to find match
11.       res = LOT_match_resp(ID_req_frame);
12. // if yes, stores the corresponding reply
13.       if res == 1 then
14.      {
15. // Decodes the ID_req_frame
```

```
          std_resp_frame =
          prepare_std_resp(ID_req_frame);
          store_reply(std_resp_frame);
       }
    }
    else // type = REQ
    {
// Decodes the request embedded in
// the ID_req_frame
       std_req_frame =
       prepare_std_req(ID_req_frame);
// Stores the information necessary to
// identify the respective conf.
       Store_info(std_req_frame);
       queue(std_req_frame);
    }
  }
  else // station in another domain
  {
    queue(ID_req_frame);
  }
}
```

Fig. 8 Fwrd_ID_Request, pseudo-code algorithm

The function *Init_ID_Request* (Fig. 9) is also called by the *Indication_Handler* when it needs to initialise a pending IDT in the LOT.

```
1.  Init_ID_Request(req_frame)
2.  {
3.   // Test if there is a match with any
4.   // other pending transaction on the LOT
5.   res = check_LOT(req_frame);
6.   if res != 1 then // No entry on the LOT
7.   {
8.   // Updates the LOT
9.     handler = updt_LOT(req_frame);
10.    start_error_handling_timer(handler);
11.  // Codes an IDF
12.    ID_req_frame = prepare_IDF(req_frame)
13.    Fwrd_ID_Request(ID_req_frame);
14.  }
15.  else // There is a match on the LOT
16.  {
17.  // Do nothing
18.  }
19. }
```

Fig. 9 Init_ID_Request, pseudo-code algorithm

The *Init_ID_Request* function starts by checking (in the LOT) if there is another entry with the same data (line 4). In the affirmative case, the bridge will not do any additional processing on this frame. Otherwise, it stores data relative to this pending IDT in the LOT, and starts a count down timer that will clean the pending transaction from the LOT, upon expiration. This timer guarantees that if a transaction is not completed, the bridge will remove that transaction from the LOT. The value for the timer can be calculated based on the worst-case response time analysis of the network. Finally, this function codes the frame using the IDP and relays it to the other bridge master.

B. Confirmation Handler

The *Confirmation_Handler* function is called when a bridge receives a response to a request. In Fig. 10, we are only detailing the part relative to the response to an Inter-Domain Request.

```
Confirmation_Handler(resp_frame)
{
 res = type_of(resp_frame)
 Switch (res)
 {
   case INTER_DOMAIN_RESP:
    ID_req_frame =
  prepare_ID_req(res_frame, req_data1);
Fwrd_ID_Request(ID_req_frame);
   end;
…
 }
}
```

Fig. 10 Confirmation handler, pseudo-code algorithm

This function starts by determining the message type, using the Destination Address of the *resp_frame* and the information stored by the *Forward_ID_Request* function.

If the received frame is a response to an Inter-Domain Request, then the bridge prepares a new frame (an IDF) using the IDP and forwards it through the other bridge master. The other cases handle standard PROFIBUS functionalities, e.g. any request addressed to the bridge.

VI. DISCUSSION AND ONGOING WORK

The use of bridges to interconnect wired and wireless domains of a PROFIBUS-based network has several advantages over the use of repeaters.

In a repeater-based approach, it is necessary to increase the value of several network timing parameters in order to encompass the latencies of the repeaters, different data rates and different frame formats in order to guarantee a predictable behaviour for the network [12]. In a bridge-based network architecture, these issues do not impact on important PROFIBUS parameters (e.g. the Slot Time). Thus, in a bridge-based network we can set network parameters in the same way as in a common PROFIBUS network (at least on its wired parts). Consequently, the same degree of responsiveness to failures can be achieved, far superior to a repeater-based approach.

Wireless networks are usually more error-prone than their wired counterparts. Thus another advantage of the bridge-based approach is that when an error occurs its consequences will be confined to a single network domain (error containment). Another consequence of network segmentation is that transactions between stations in the same domain will have their response times reduced. However, transactions involving stations in different domains may, in some cases, have increased response times.

These characteristics allow bridge-based networks to be more scalable than repeater-based networks. Nevertheless, the proposed bridge-based network requires more complex Intermediate Systems (the bridges), due to the need for supporting the extra functionalities of the Inter-Domain Protocol.

One of the main objectives of the proposed protocol is to maintain the compatibility with existing solutions. The most used upper layer for PROFIBUS is the Decentralised Peripherals (DP) application layer. This protocol is specially suited for the exchange of data between PLCs, PCs or process control systems with field devices like I/O, drives or valves. The PROFIBUS-DP application layer provides the functionalities to configure and diagnose devices as also for the cyclic exchange of data.

A PROFIBUS-DP slave is controlled and configured by a single master. Before being operational (able to exchange data), a slave has to pass through several configuration and parameterisation phases. During normal operation, a master periodically sends requests to the slave, which replies using the data previously stored. During this phase, if a PROFIBUS-DP master does not receive a response to its request, the DLL will return a confirmation without data to the DP layer. This behaviour does not generate errors in the master; it simply requires to continue the periodical inquiry of the slave, until receiving a response. This characteristic allows a PROFIBUS-DP master to transparently use the IDP.

However, the configuration and parameterisation of a slave station involve the exchange of messages between the master and the slave. During these phases, a master station expects an immediate answer from the slave, and if no answer is received the process is restarted. This leads to problems when a master tries to initialise a slave located in another domain. A solution to this problem can be based on bridges acting as proxies (for the slaves). In this way, a bridge could emulate the behaviour of a slave during the initialisation phase. These issues are currently being addressed.

A key factor to characterise the proposed protocol is to carry out a timing analysis. An analytical worst-case model can be adapted from the analysis proposed in [8]. The authors are also developing a tool for simulating the protocol, which will further help on its temporal characterisation. Simulations will also enable to test the different functionalities that are required for the operation during the configuration and parameterisation phases. This timing analysis will also take into consideration the support of inter-domain mobility [11], which is under assessment.

VII. CONCLUSIONS

In this paper, we have proposed an architecture and the mechanisms which extend the capabilities of the PROFIBUS protocol to support a hybrid wired/wireless network interconnected by intermediate systems acting as bridges.

In such an architecture, the communication between different domains is supported by an Inter-Domain Protocol (IDP), which allows the use of standard PROFIBUS stations, since the required functionalities are implemented by the bridges. The bridges emulate the behaviour of the initiator and the responder stations and are able to relay frames coded using the Inter-Domain Protocol.

The proposed architecture has several advantages in relation to the repeater-based Single Logical Ring architecture proposed in [6]. Namely, it provides traffic and error containment between different domains and better responsiveness to errors. Also, the response times for transactions between nodes in the same domain will be reduced. However, transactions involving stations in different domains may, in same cases, have increased response times. Finally, the proposed architecture is more scalable than the Single Logical Ring architecture.

VIII. REFERENCES

[1] IEC, *IEC61158 – Fieldbus Standard for use in Industrial Systems, PROFIBUS (Type 3)*, 2000.

[2] CENELEC, *EN 50170 – General Purpose Field Communication System, Volume 2 – PROFIBUS*, 1996.

[3] Tovar, E. and Vasques, F., "Real-Time Fieldbus Communications Using PROFIBUS Networks", *IEEE Transactions on Industrial Electronics*, vol. 46, no. 6, December 1999, pp. 1241-1251.

[4] Tovar, E. and Vasques, F., "Cycle Time Properties of the PROFIBUS Timed-Token Protocol", *Computer Communications* 22 (1999), pp. 1206-1216, Elsevier.

[5] Tovar, E. and Vasques, F., "Non Pre-Emptive Scheduling of Messages on SMTV Token-Passing Networks", *in Proceedings of the 12th IEEE Euromicro Conference on Real-Time Systems, Stockholm*, Sweden, pp. 209-218, June 2000.

[6] Haehniche, J. and 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.

[7] Alves, M., Tovar, E., Vasques, F., Roether, K. and Hammer, G., "Real-Time Communications over Hybrid Wired/Wireless PROFIBUS-based Networks", *in Proceedings of the 14th Euromicro Conference on Real-Time Systems*, pp. 142-151, June 2002.

[8] Ferreira, L., Alves, M., Tovar, E., "Hybrid Wired/Wireless PROFIBUS Networks Supported by Bridges/Routers", *in Proceedings of the 2002 IEEE International Workshop on Factory Communication Systems*, pp. 193-202, August 2002.

[9] Pereira, N., et al, "Integration of TCP/IP and PROFIBUS Protocols", *in WIP Proceedings of the 2002 IEEE International Workshop on Factory Communication Systems*, August 2002.

[10] Rauchhaupt, L., "System and Device Architecture of a Radio Based Fieldbus – The RFieldbus System", *in Proceedings of the 2002 IEEE International Workshop on Factory Communication Systems*, pp. 193-202, August 2002.

[11] Ferreira, L., Tovar, E., Alves, M., "PROFIBUS Protocol Extensions for Enabling Inter-Cell Mobility in Bridge-based Hybrid Wired/Wireless Networks", *in Proceedings of the 5th IFAC International Conference on Fieldbus Systems and their Applications*, Aveiro, Portugal, pp 283-290, July 2003.

[12] Alves, M., *Real-Time Communications over Hybrid Wired/Wireless PROFIBUS-based Networks*, PhD dissertation, University of Porto, Portugal, February 2003.