IPP HURRAY!

www.hurray.isep.ipp.pt

Technical Report

Bridge Architecture to Interconnect Hybrid Wired/Wireless PROFIBUS Networks

Paulo Baltarejo Sousa

Luis Lino Ferreira

HURRAY-TR-080102

Version: 0

Date: 05-01-2008

Bridge Architecture to Interconnect Hybrid Wired/Wireless PROFIBUS Networks

Paulo Baltarejo Sousa, Luis Lino Ferreira

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: {llf, pbs}@isep.ipp.pt

http://www.hurray.isep.ipp.pt

Abstract

The integration of wired and wireless technologies in modern manufacturing plants is now of paramount importance for the competitiveness of any industry. Being PROFIBUS the most widely used technology in use for industrial communications, several solutions have been proposed to provide PROFIBUS networks with wireless communications. One of them, the bridge-based hybrid wired/wireless PROFIBUS network approach, proposes an architecture in which the Intermediate Systems operate at Data Link Layer level, as bridges. In this paper, we propose an architecture for the implementation of such a bridge and the required protocols to handle communication between stations in different domains and the mobility of wireless stations.

Bridge Architecture to Interconnect Hybrid Wired/Wireless PROFIBUS Networks

Abstract: The integration of wired and wireless technologies in modern manufacturing plants is now of paramount importance for the competitiveness of any industry. Being PROFIBUS the most widely used technology in use for industrial communications, several solutions have been proposed to provide PROFIBUS networks with wireless communications. One of them, the bridge-based hybrid wired/wireless PROFIBUS network approach, proposes an architecture in which the Intermediate Systems operate at Data Link Layer level, as bridges. In this paper, we propose an architecture for the implementation of such a bridge and the required protocols to handle communication between stations in different domains and the mobility of wireless stations.

Keywords: Fieldbus, Wireless, Real-time, Industrial Automation.

1.INTRODUCTION

Any wired network can benefit from the integration of the wireless communications. Wireless communications present a set of advantages that can not be wasted, like easier equipment installation, configuration flexibility, ability to evolve and cuts in cabling and maintenance costs, just to mention some. Additionally, such technologies also enable the operation of mobile stations.

The PROFIBUS (PROcess FIeldBUS) [1] is the fieldbus most widely used. According to [2], there are more than 20 million PROFIBUS nodes on the market around the world and it continues to grow. The 20 million nodes goal has been achieved 8 months prior to its prediction, reinforcing the feeling that this 12 years old technology still has a very high market potential. Its main advantages are the possibility of being used in a wide range of applications, from discrete-part automation to process control and motion control.

Several solutions [3-7] have been proposed to provide PROFIBUS networks with wireless communications. The solutions proposed in [5-7] are quite limited either in terms of number of segments or wireless cells and in the support of mobility. In the solution proposed in [3] the interconnection between wired segments and wireless cells (hereafter, wired segments and wireless cells are referred as domains) is done by Intermediate Systems (ISs) operating as repeaters. No error containment between different domains and low responsiveness to failures are two identified drawbacks of this solution. The solution proposed in [4] is also compatible with standard PROFIBUS as also repeater-based solution [3] and solves its identified. In this solution the ISs operate at Data Link Layer (DLL) level, as bridges, and requires two new protocols, one for supporting the communication between stations in different domains – the Inter Domain Protocol (IDP), and another to support the mobility of wireless stations between different wireless domains – the Inter-Domain Mobility Procedure (IDMP).

In this paper, we propose an IS architecture for the bridge-based hybrid wired/wireless PROFIBUS network approach, focusing on the IDP implementation. Due to space limitation only a brief description of the IDMP will be presented in this paper, for details the reader is referred to [8].

This paper is structured as follows. Section 2 describes the main PROFIBUS characteristics which are relevant for the reasoning in the remaining of the paper. Then, in Section 3, the bridge-based hybrid wired/wireless PROFIBUS network approach is outlined. The proposed architecture for the Intermediate Systems is described in Section 4. Section 5 describes the IDP implementation proposal. Finally, in Section 6, we draw some conclusions.

2.BASICS OF PROFIBUS

PROFIBUS uses a master/slave protocol for communications, where the PROFIBUS DLL uses a token passing procedure to grant bus access to masters. The token is passed between masters in ascending Medium Access Control (MAC) address order, organizing the medium access in a logical ring.

After receiving the token, a PROFIBUS master is capable of dispatching transactions during its `Token Holding Time` (T_{TH}). A transaction (or message cycle) consists in the request frame from the initiator (a master) and of the associated acknowledgement or response frame of the responder station. The acknowledgement (or response) must arrive before the expiration of the `Slot Time` (T_{SL}), otherwise the initiator repeats the request a number of times defined by an internal DLL variable called `max_retry_limit`. The `Station Delay of Responder Time` (T_{SDR}), is the time required by a responder before transmitting a reply frame. `Idle Time` (T_{ID}) is a period of inactivity inserted by master stations between two consecutive message cycles. Fig. 1 illustrates the previous concepts.

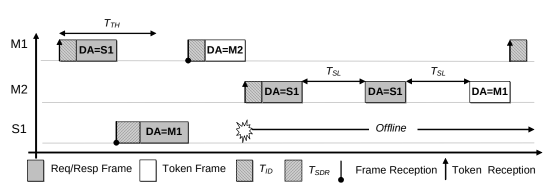

Fig. 1 – PROFIBUS message cycle timings

Traditionally, the PROFIBUS networks encompass only one domain (wired domain) and a Single Logical Ring (SLR), but the solution being proposed requires some add-ons to the protocol, enabling inter-domain communication and inter-domain station mobility.

3.BRIDGE-BASED HYBRID WIRED/WIRELESS PROFIBUS NTEWORK APPROACH

3.1. Basis of the Bridge-Based Approach

In the bridge-based approach the interconnection between wired and wireless domains is done by a IS operating at DLL level, as a bridge. Assuming, a two-port bridge interconnecting two different network domains, the incoming frames are only relayed to the other port if the destination address embedded in the frame corresponds to a MAC address of a station physically reachable through that other port.

With a MAC protocol as the one used in PROFIBUS (timed token passing), a bridge needs to have two network interfaces, both supporting the same DLL and specifically the same MAC protocols. This means that such a dual-port PROFIBUS bridge would contain two master stations. Each master station that belongs to a bridge is referred as `Bridge Master` (BM), which is a modified PROBIBUS master.

Fig. 2 presents a bridge-based hybrid network example. The network comprises two wired masters (M1 and M2), two wireless mobile masters (M3 and M4), five wired slaves (S1, S2, S3, S4 and S5) and one mobile wireless slave (S6).

The network comprises four domains: two wired domains (D^2 and D^4) and two wireless domains (D^1 and D^3) which are interconnected by three bridge devices (B1 (M8:M5), B2 (M6:M9), B3 (M10:M7)). Each bridge is composed by two BMs.

In the wireless domains all messages are relayed through `Base Stations` (BSs) which operate in cut-through mode as a wireless repeater, using two radio channels, one to receive frames from the wireless stations (the uplink channel), and another to transmit frames to wireless stations (the downlink channel).

Is assumed that the wireless communication interface of the BS and the wireless mobile stations is equal to the wireless communication interface defined in the RFieldbus [9].

Network operation is based on the Multiple Logical Ring (MLR) approach, proposed in [4]. Therefore, each wired/wireless domain has its own logical ring. In this example, four different logical rings exist: (D^1 (M3 → M8), D^2 (M1 → M5 → M6), D^3 (M4 → M9 → M10) and D^4 (M2 → M7)).

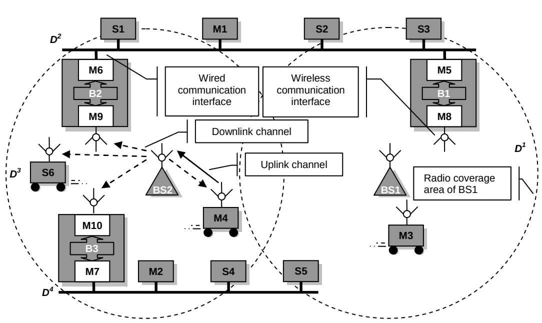

Fig. 2 – Bridge-based network example

As a consequence of the MLR, this approach requires two new protocols, one for supporting the communication between stations in different domains (the IDP) and another to support the mobility of wireless stations between different wireless domains (the IDMP).

3.2. Inter-Domain Protocol (IDP)

The IDP explores some PROFIBUS protocol features at the DLL and Application Layer (AL) level, which enables a master to repeat the same request until receiving a response from the responder station without generating error to the upper layers. It is also important to stress that this solution is compatible with standard PROFIBUS devices.

When a master starts a transaction with a station belonging to another domain, an Inter-Domain Transaction (IDT), it starts by transmitting a request frame addressed to the responder station (an Inter-Domain Request (IDreq) frame). This frame is then relayed by only one of the BMs (denoted as BM_{ini} – *ini* stands for initiator) belonging to the initiator domain, according to its `Routing Table` (RT) information. BM_{ini} receives the IDreq frame, codes it according to the IDP, an Inter-Domain Frame (IDF), and stores internally information about the transaction, in a structure called `List of Open Transactions` (LOT INI). Additionally, a timer, the `BM_IDT_Abort_Timer` ($T_{BM\text{-}IDTAbort}$), is started.

The initiator periodically sends a request frame until receiving a response frame. Note that the AL of PROFIBUS-DP can operate like this without generating errors.

The IDreq frame is relayed by the bridges until reaching the last BM, which belongs to the responder domain (denoted as BM_{res} – *res* stands for responder). BM_{res} decodes the original request frame, stores information about this request in another LOT (LOT RES) and transmits it to the responder, which can be a standard PROFIBUS station (for example a wired PROFIBUS slave). When decoding the frame, the BM_{res} reconstructs the original frame as transmitted by the initiator (it even puts the initiator address (SA) on the request frame). Thus, from the responder's perspective the initiator seems to belong to the same domain. When the BM_{res} receives the response to that request, it codes an Inter-Domain Response (IDres) frame, using the IDP, and forwards it through the reverse path until reaching the BM_{ini}, where it will be decoded and properly stored.

After that, the BM_{ini} is ready to respond to a new (repeated) request from the initiator. The response frame is exactly equal to the frame transmitted by the IDT responder.

If meanwhile, the $T_{BM\text{-}IDTAbort}$ expires the related entry at the LOT INI is deleted and a new IDT can be reinitialised by the next initiator's request.

Note that, the IDP uses the PROFIBUS `Send Data with Acknowledge` (SDA) service to forward the IDFs.

Fig. 3 presents a simplified timeline of an IDT between master M3 and Slave S6 assuming the network scenario presented in Fig. 2.

In this example a transmission error occurs when the IDF embedding the response is transmitted between BM M6 and BM M5. Since the frame has not been acknowledge by BM M5, BM M6 retransmits the frame after the expiration of T_{SL}.

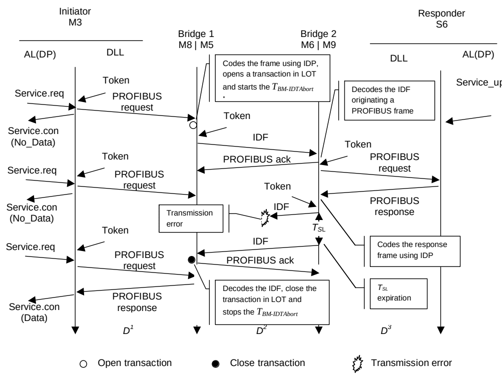

Fig. 3 – IDT between M3 and S6

3.3. Inter-Domain Mobility Procedure (IDMP)

The IDMP is a hierarchically managed procedure, where one master in the overall network – the `Global Mobility Manager` (GMM) – is responsible for periodically starting the IDMP and controlling some of its phases. Additionally, in each domain, one master controls the mobility of stations belonging to that domain – the `Domain Mobility Manager` (DMM). Finally, the BMs must implement specific mobility services. Note that, the GMM, DMM and BM can be the same master station (in the Section 4 a detailed description of the bridge architecture is given).

During the IDMP, the wireless mobile stations assess the quality of the radio channel using the functionalities provided by their Physical Layer (PhL) and change to the best radio channel detected.

The IDMP evolves through 4 phases, as shown in Fig. 4. The objective of these phases is to insure that the procedure will not generate errors, that the inaccessibility periods are minimal and that the wireless mobile stations are able to evaluate all wireless radio channels and switch to the best one. Recall that PROFIBUS has only been prepared to operate on a broadcast network with a single logical ring, therefore the IDMP must handle some problems associated with node mobility, particularly in order to avoid frame order inversion and synchronization problems due to the multiple logical ring architecture and node mobility.

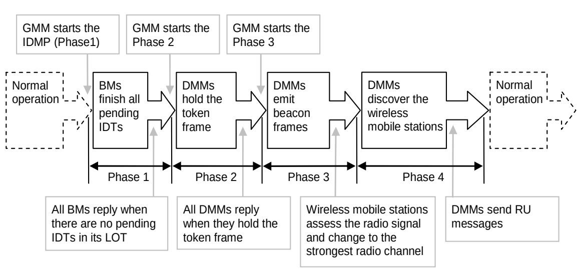

Fig. 4 –Inter-Domain Mobility Procedure Phases

This mechanism is synchronous in Phase 1 and Phase 2 as well as at the beginning of Phase 3 and controlled by the GMM. But in the case of Phase 3, the ending of it in the domains is not synchronized as well as Phase 4 runs asynchronously for each domain. These are controlled by each DMM in its domain.

The main goal of the Phase 1 is to allow that all pending IDTs end, therefore guaranteeing that order inversion problems will not occur. When there are no pending IDTs, the system evolves to Phase 2. The goal of Phase 2 is to assure that all DMMs in the network hold the token frame. Note that, according to the PROFIBUS protocol only stations that hold the token are able to transmit. Therefore, in order to provide a way for the mobility-related messages to be transmitted, the DMM inquiries all BMs in its domain, asking them to transmit mobility-related messages. After ending Phase 2, i.e. when all DMMs hold the token frame, the GMM starts the Phase 3 and its role ends. The DMMs of the wireless domains start emitting `Beacon` frames during a pre-defined time. The wireless mobile stations use the `Beacon` frames to evaluate the quality of the different radio channels and to decide if they want to handoff (or not). So, before the end of the `Beacon` transmission, every wireless mobile station that wants to handoff must switch to the new radio channel. After ending Phase 3, the DMMs start the Phase 4. The goal of this phase is to discover which wireless mobile stations are on the DMM domain. At the end of this phase the DMMs send `Route_Update` (RU) messages specifying the location of the wireless mobile stations. These messages are used by the BMs to update their RT and `List of Active Stations in Domain` (LASD). This protocol also supports some features which improve its error handling capability, as described in [8].

4.BRIDGE ARCHITECTURE

Fig. 5 illustrates the main building blocks of a two-port bridge. In order to support the required functions, there must be a set of mechanisms related to the IDP and to the IDMP. These mechanisms operate at DLL level and consequently the existing PROFIBUS DLL must be adapted. The operation of the IDP and IDMP are managed by three components: BM, DMM and GMM.

The BM component, which gives to the device its name and is mandatory, contains the routing and the IDF handling functions which are crucial to the IDP and to the IDMP. These two functions are associated with five data structures: the `Routing Table` (RT), the `List of Open Transaction on` $\mathtt{BM}_{\mathtt{ini}}$ (LOT INI), the `List of Open Transaction on` $\mathtt{BM}_{\mathtt{res}}$ (LOT RES), the `List of Active Stations in Domain` (LASD) and the `List of Wireless Mobile Stations in the Network` (LWMSN).

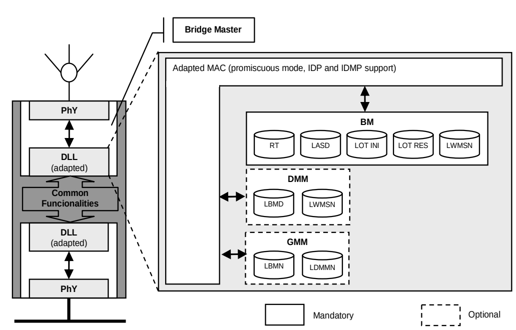

Fig. 5 – Bridge architecture

Frames are relayed by a BM according to the information contained in its RT. The LOT INI is used to store information about ongoing IDTs in which the BM assumes the role of BM_{ini} and the LOT RES is used when a BM assumes the role of BM_{res}. The LASD is a list of all masters and slaves that belong to the BM domain and is used by the BM to know if it is the transaction's BM_{ini}, BM_{res} or simply a bridge in the path between the initiator and the responder. The LWMSN is a list of addresses of all wireless mobile stations present in the network, and is used in the IDMP.

The other two components, DMM and GMM, are optional and their functions are related to the IDMP. The DMM functionalities require two data structures: the `List of Bridge Masters in the Domain` (LBMD) and the LWMSN. The LBMD is a list that contains the domain's BMs addresses and is used in the IDMP inquiry sub-phase. The LWMSN is a list of addresses of all wireless mobile stations present in the network and is used in the IDMP discovery sub-phase.

The GMM must also be provided with two data structures: the `List of Bridge Masters in the Network` (LBMN) and the `List of Domain Mobility Managers in the Network` (LDMMN). The LBMN is a list of address of all BMs present in the network and is used to control the received `Ready_to_Start_Mobility_Procedure` (RSMP) messages during the IDMP Phase 1. The LDMMN contains all network DMM addresses and is used to control the received `Ready_for_Beacon_Transmission` (RBT) messages during the IDMP Phase 2.

Fig. 5 also shows the `Common Functionalities` (ComFunc) box, which is supported by a shared memory area and is responsible for the communication between the two BMs of a bridge, however, if necessary this functionality can support more than two ports.

5.IDP IMPLEMENTATION

In this section we present the IDP implementation proposal details. Therefore, only the data structures and the procedures related to the IDP will be detailed. As referred previously, the IDMP implementation will not described due to space limitations. However a detailed description can be found in [8].

5.1. Inter-Domain Frame(IDF) Formats

IDFs are used by the IDP for proper transmission of frames between bridges. These frames must contain information that enables decoding the embedded original request/response and matching the information stored in the LOT and the respective response.

Basically, the IDFs are standard PROFIBUS frames that carry in `Transaction Identifier` (TI), `Embedded Function Code` (EFC) and the `Embedded Frame Type` (EFT) fields within the data field. The TI is a sequence number, assigned by the BM_{ini}, which must also be included in the response frame (similar to a TCP/IP sequence number). This field is used by the BM_{ini} to distinguish between response frames related to different pending transactions. The EFC is used to reconstruct the original frame and the EFT is an identifier which enables BM_{ini} and BM_{res} to identify the type of the embedded frame.

A detailed mapping between standard PROFIBUS frames and IDFs is found in [4].

5.2. List of Open Transaction (LOT)

The information about ongoing IDTs is stored in the two different LOTs, one maintained by the BM_{ini} (LOT INI) and another maintained by the BM_{res} (LOT RES).

When a BM acts as BM_{ini} (Fig. 6) and it receives a request frame, then it creates an entry in the LOT INI and starts the related $T_{BM\text{-}IDTAbort}$ timer. Each LOT INI entry is controlled by a state machine which initiates its operation in the WAITING state. It stays in this state until receiving an IDres frame or at expiration of the $T_{BM\text{-}IDTAbort}$ timer. In the first case, it state machine evolves to the FINALISED state and the information contained in the response frame is stored. In the second case, the IDT is aborted and its entry is removed from the LOT INI.

In the FINALISED state it waits for a repeated request from the initiator to finish the transaction. The transaction also ends if the $T_{BM\text{-}IDTAbort}$ timer expires. In both cases the entry is removed from the LOT INI.

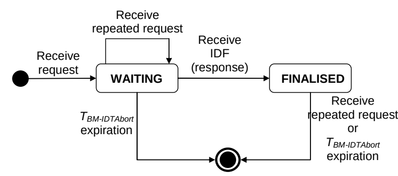

Fig. 6 - BM_{ini} LOT state machine

When a BM acts as BM_{res} (Fig. 7) and it receives an IDreq frame then it creates an entry in the LOT RES and starts the related $T_{BM\text{-}IDTAbort}$ timer. The entry state machine evolves to the WAITING state. It stays in this state until receiving a response frame or at expiration of the $T_{BM\text{-}IDTAbort}$ timer. In both cases the LOT entry is removed from the LOT RES. However, in the former case the IDF is forwarded and the in the latter case the IDT is aborted.

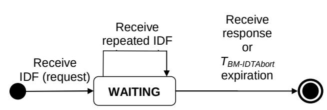

Fig. 7 – BM_{res} LOT state machine

Both LOTs can be implemented as a list of tuples, each of them composed by the following fields: `Destination Address` (DA), `Source Address` (SA), `Destination Address Extension` (DAE), `Source Address Extension` (SAE), TI, $T_{BM\text{-}IDTAbort}$ timer and DATA.

The DA and SA identify the frame sender and the frame responder, respectively. The DAE and SAE are extension fields that can be used to identify the AL service which originated the frame, as well as the destination service. These fields allow identifying different message streams (a message stream is a periodic sequence of message cycles, related for instance, to the reading of a sensor, in each message stream associates an initiator (a master) with a responder (usually a slave)) between the same stations. As mentioned, the TI is a sequence number to identify the transactions. The $T_{BM\text{-}IDTAbort}$ timer is used to avoid unfinished transactions. If a $T_{BM\text{-}IDTAbort}$ timer expires, the related entry is removed and this way allowing that new IDTs can be reinitialised. The DATA field is only used on the LOT INI to store the information contained in the data field of the received IDres frame. Therefore, it contains the data required by a BM to reply to an initiator's request when it has received a response to that particular transaction, Fig. 3.

5.3. Routing Table(RT) and List of Active Stations in Domain(LASD)

The RT and the LASD data structures are fundamental for the behaviour of the BM in the context of the IDP. The decision of relaying a frame is based on the RT, which determines whether an incoming frame is to be relayed to the other port or not.

A BM can receive a frame from its Physical Layer (PhL) (i.e., from its domain) or from ComFunc (i.e. from the other BM of a bridge) (Fig. 8). In the first situation, the LASD is used to check if the initiator belongs to its domain. If it succeeds then it assumes the role of BM_{ini} for this IDT. In the second case, the LASD is used to check if the frame responder belongs to its domain. If it succeeds then it assumes the role of BM_{res} for this IDT.

The RT can be implemented as a list of tuples that associate the station MAC address with information about routing. The LASD can be implemented as a list of station MAC address.

Note that, the RT and LASD structures must be configured prior to run-time. In run-time operation, these data structures are dynamically updated.

5.4. Interconnection Procedures

The interconnection schema between two BMs of a bridge is depicted in Fig. 8. Two situations are possible. The first situation occurs when a BM receives a frame from the domain to which it belongs, and the second occurs when it receives a frame from other BM. To handle such kind of events our architecrure bases on its operation in two procedures, the `handleFramePhL` procedure and the `handleFrameComFunc`, which are described next.

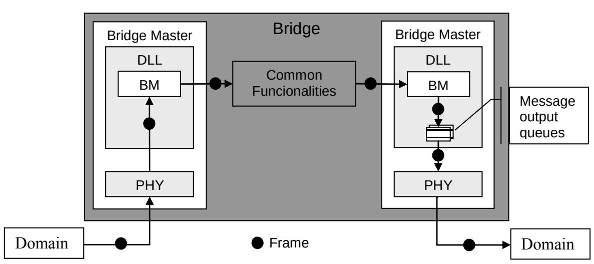

Fig. 8 – Interconnection schema in a bridge

Receive Frame (from domain) Procedure

Fig. 9 presents the procedure performed by a BM when it receives a frame from its domain. The BM starts by checking if the frame is addressed to a station in another domain using the `RT.isRoute(_frame)` function, which consults its RT. If no match is found, then the frame is discarded (`discard(_frame)`, line 43). Otherwise the frame is processed and it verifies if the transaction initiator (using SA of the frame) belongs to its domain (`LASD.isSAinDomain(_frame)`, line 4). If it does not belong, it forwards the frame to the ComFunc (`forward2ComFunc(_frame)`, line 39).

In the case of a frame which has to be forwarded and when the initiator belongs to the BM's domain, the next step is to check if it is a request or a response frame (`isReqFrame(_frame)`, line 5). In the first case, if it is a request, the BM is the BM_{ini} of this transaction. In the second case, the BM is the BM_{res} of this transaction. In this case, the BM has to check in the LOT RES for an entry related to this transaction. If it founds then it removes the entry from the LOT RES, stops the $T_{BM-IDTAbort}$ timer, codes an IDF (note that, this IDF has to use the same TI of the associated request) and forwards to ComFunc (line 29 to line 32). If it does not found then it discards the frame (`discard(_frame)`, line 34).

If it assumes the role of BM_{ini},, it has to check if the transaction requires a response (`requireRespFrame (_frame)`, line 6). If it does not require a response the BM forward it (`forward2ComFunc(_frame)`, line 23). If it requires a response there is the need to match the frame with LOT INI entries (`LOT_INI.match (_frame)`, line 7). Three possibilities are considered. First, there is an entry in the LOT INI related to this transaction in the state WAITING. In this case, the frame is discarded (line 9). Second, there is an entry in the LOT INI related to this transaction in the state FINALISED. In this case the entry is removed from the LOT INI, the $T_{BM-IDTAbort}$ timer is stopped and a PROFIBUS response is coded with the stored information and then the coded response frame is sent (line 11 to 14). Third, there is no entry related to this transaction, and then it is created an entry in the LOT INI, coded an IDF, started the $T_{BM-IDTAbort}$ timer and forwarded it to the ComFunc (line 16 to line 19).

The value of the $T_{BM-IDTAbort}$ timer must be set with time enough to which allows the execution of a transaction. The formulation for Worst Case Response Time (WCRT) of an IDT can be found in [4].

```
1. handleFramePhL(_frame) //from domain
2. {
3.   if RT.isRoute(_frame) { // if it has to be forward
4.      if LASD.isSAinDomain(_frame){ // if the SA belongs to its domain
5.         if isReqFrame(_frame){  //if it is a request frame
               //BM ini
6.            if requireRespFrame(_frame){ //if require response frame
7.               switch LOT_INI.match(_frame){
8.                  case WAITING:
9.                     discard(_frame);
10.                    case FINALISED:
11.                       LOT_INI.remove(_frame);
12.                       LOT_INI.stopIDTAbortTimer(_frame);
13.                       codePROFIBUS(_frame);
14.                       reply(_frame);
15.                    default:
16.                       LOT_INI.create(_frame);
17.                       LOT_INI.startIDTAbortTimer(_frame);
18.                       codeIDF(_frame);
19.                       forward2ComFunc(_frame)
20.              }
21.           }
22.           else{
23.                 forward2ComFunc(_frame); //it is a SDN
24.           }
25.        }
26.        else{
           //BM res - receive a response
27.           switch LOT_RES.match(_frame){
28.              case WAITING:
29.                 LOT_RES.remove(_frame);
30.                 LOT_RES.stopIDTAbortTimer(_frame);
31.                 codeIDF(_frame);
32.                 forward2ComFunc (_frame);
33.              default:
34.                 discard(_frame);
35.           }
36.        }
37.     }
38.     else{
39.        forward2ComFunc(_frame);//frm has to forward to the other BM
40.     }
41.  }
42.  else{
43.     discard(_frame);
44.  }
45.}
```

Fig. 9 - `handleFramePhL(_frame)` function, pseudo-code algorithm

Receive Frame (from ComFunc) Procedure

Fig. 10 shows the procedure when a BM receives a frame (an IDF) from the ComFunc (i.e., from the other BM of a bridge). The BM starts by consulting its RT, using the `RT.isRoute(_frame)` function, to determine if the frame should be relayed or not (line 3). In the case of not, the frame is discarded (`discard(_frame)`, line 40).

If the frame must be relayed, the BM verifies, using the DA frame field, if the addressed station belongs to its domain (`LASD.isDAinDomain(_frame)`, line 4). If it does not belong then the BM passes the frame to DLL to queue the frame (`pass2DLL(_frame)`,line 37). Otherwise, if it succeeds then this BM will act as either a BM_{ini} or a BM_{res}. For that, it has to check if it is a request or a response frame (`isReqFrame(_frame)`, line 6). If it is a request it can act as BM_{res} , otherwise it acts as BM_{ini}.

It acts as BM_{res} if the frame requires a response (`requireRespFrame(_frame)`, line 6). If it does not require a response then it passes the frame to the DLL (`pass2DLL(_frame)`, line 20). Otherwise, it checks in the LOT RES for an entry related to this transaction, if it founds then it discards the frame (`discard(_frame)`, line 9 and 11). If it does not found any entry, which is the expected, it creates an entry related to this transaction, starts the $T_{BM\text{-}IDTAbort}$ timer, codes a PROFIBUS frame containing the request and passes it to the DLL to queue in its message output queue (line 13 to line 16).

If it acts as BM_{ini}, it matches the frame with LOT INI entries. If it found an entry that is in the WAITING state it changes its state for FINALISED, stops the $T_{BM\text{-}IDTAbort}$ timer and store the response frame in the LOT INI (line 26 to line 28). In other cases, it discards the frame (line 30 and line 32).

```
1. handleFrameComFunc(_frame) //from BM
2. {
3.    if RT.isRoute(_frame) { // if it has to be forward
4.       if LASD.isDAinDomain(_frame){ // if the DA belongs to its domain
5.          if isReqFrame(_frame){  //if it is a request frame
6.             if requireRespFrame(_frame){ //if require response frame
                //BM res
7.                switch LOT_RES.match(_frame){
8.                   case WAITING:
9.                      discard(_frame);
10.                  case FINALISED:
11.                     discard(_frame);
12.                  default:
13.                     LOT_RES.create(_frame);
14.                     LOT_RES.startIDTAbortTimer(_frame);
15.                     codePROFIBUS(_frame);
16.                     pass2DLL(_frame);
17.               }
18.            }
19.            else{
20.               pass2DLL(_frame);
21.            }
22.         }
23.         else{
            //BM ini - receive a response
24.            switch LOT_INI.match(_frame){
25.               case WAITING:
26.                  LOT_INI.finalised(_frame);
27.                  LOT_INI.stopIDTAbortTimer(_frame);
28.                  LOT_INI.storeResponse(_frame);
29.               case FINALISED:
30.                  discard(_frame);
31.               default:
32.                  discard(_frame);
33.            }
34.         }
35.      }
36.      else{
37.         pass2DLL(_frame);//the frame is queued in the DLL
38.      }
39.   }
40.   else{
41.      discard(_frame);
42.   }
43.}
```

Fig. 10 - `handleFrameComFunc(_frame)` function, pseudo-code algorithm

6.CONCLUSIONS

In this paper an Intermediate Systems architecture for a bridge-based PROFIBUS hybrid wired/wireless network is presented. We describe the main components for the architecture including details for the main protocols, the Inter-Domain Protocol (responsible for communications between station in different domains) and the Inter-Domain Mobility Procedure (which permits station mobility between different wireless cells/domains). It is important to stress that these protocols are compatible with standard PROFIBUS, therefore standard stations are capable of operating and of performing transactions with stations belonging to different domains.

In this paper we also detail the implementation of the Inter-Domain Protocol which has been validated by a simulator developed using the OMNet++ framework [10] and the C++ language programming – the Bridge-Based Hybrid Wired/Wireless PROFIBUS Network Simulator [8].

REFERENCES

[1] IEC, "IEC61158 - Fieldbus Standard for use in Industrial Systems": European Norm., 2000.

[2] B. Weber, "PROFIBUS Cracks the 20 Million Barrier". *Press release, PROFIBUS International Support Center*, 2007. Available online at http:// www.profibus.com.

[3] M. Alves, "Real-Time Communications over Hybrid Wired/Wireless PROFIBUS-Based Networks", PhD. Porto, Portugal: University of Porto, 2003.

[4] L. Ferreira, "A Multiple Logical Ring Approach to Real-time Wireless-enabled PROFIBUS Networks", PhD. Porto, Portugal: University of Porto, 2005.

[5] K. Lee and S. Lee, "Integrated Network of PROFIBUS-DP and IEEE 802.11 Wireless LAN with Hard Real-Time Requirement". In proceedings of IEEE International Symposium on Industrial Electronics (ISIE'01), Pusan, Korea, pp. 1484-1489, 2001.

[6] D. Miorandi and S. Vitturi, "A Wireless Extension of PROFIBUS DP based on the Bluetooth System", *Journal of Computer Communications*, vol. 27, pp. 946-960, 2004.

[7] A. Willig, "Investigations on MAC and Link Layer for a Wireless PROFIBUS over 802.11", PhD: University of Berlin, 2002.

[8] P. Sousa, "Performance Analysis of Wireless-enabled PROFIBUS Networks", MsC. Lisboa: Instituto Superior Técnico, Universidade Técnica de Lisboa, 2007.

[9] L. Rauchhaupt, "System and Device Architecture of Radio Based Fieldbus - The Rfieldbus System". In proceedings of IEEE International Workshop on Factory Communication Systems., pp. 193-202, 2002.

[10] A. Varga, "OMNeT++ Discrete Event Simulation System," v2.3 ed, 2004. Available online at http:// www.omnetpp.org.