Integration of TCP/IP and PROFIBUS Protocols

N. Pereira [1], F. Pacheco [1], L. M. Pinho [1], A. Prayati [2], E. Nikoloutsos [2], A. Kalogeras [2], E. Hintze [3], H. Adamczyk [3], L. Rauchhaupt [3]

[1] School of Engineering
Polytechnic Institute of Porto
Portugal

[2] Industrial Systems Institute,
University of Patras
Greece

[3] Institut f. Automation u.
Kommunikation e. V. Magdeburg
Germany

Abstract

Recent technological developments are pulling fieldbus networks to support a new wide class of applications, such as industrial multimedia applications. These applications are usually supported by the widely used TCP/IP stack. It is thus essential to provide support to TCP/IP based applications, in fieldbus networks.

This paper presents an effort that is being carried out to integrate the TCP/IP and PROFIBUS stacks, in order to support industrial multimedia applications, whilst guarantying the timing requirements of control-related traffic.

1. Introduction

Recent developments in the factory floor technologies together with the widespread use of TCP/IP and the Internet are increasing the eagerness to support a new wide class of applications, such as industrial multimedia applications, in fieldbuses such as PROFIBUS [1]. Examples of such applications for the industrial environment include monitoring applications, interfacing to microphones and cameras, remote access to maintenance data including graphics and videos, etc. These applications are usually supported by the widely used TCP/IP stack. Thus, the most effective way to integrate such applications within the PROFIBUS communication stack is to tunnel the TCP/IP telegrams into PROFIBUS telegrams.

The TCP/IP - PROFIBUS integration must be correctly specified, in order to provide not only the adequate Quality of Service to the supported TCP/IP applications but also to guarantee the timing requirements of the PROFIBUS control-related traffic. Furthermore, other details that must be assessed are:

– Performance issues due to the small maximum transmission unit size of PROFIBUS (when compared to typical TCP/IP environment)
– A solution to adapt the master/slave paradigm of PROFIBUS to the symmetric nature of a IP network
– The integration must be transparent from the application point of view

In this paper we present the on-going work to integrate a standard TCP/IP stack (Windows NT NDIS) with the PROFIBUS stack. Section 2 provides a brief presentation of the architecture of the proposed solution. Section 3 presents a discussion of the use of Windows network drivers, while Section 4 presents the current implementation of the architecture.

2. Architecture

The proposed solution is to use a dual stack architecture (Fig. 1) with a Dispatcher sub-layer connecting the TCP/IP stack and the PROFIBUS Stack over the Fieldbus DLL. This architecture adds extra sub-layers (IP Mapper, IP ACS and Dispatcher) to the standard TCP/IP and PROFIBUS stacks.

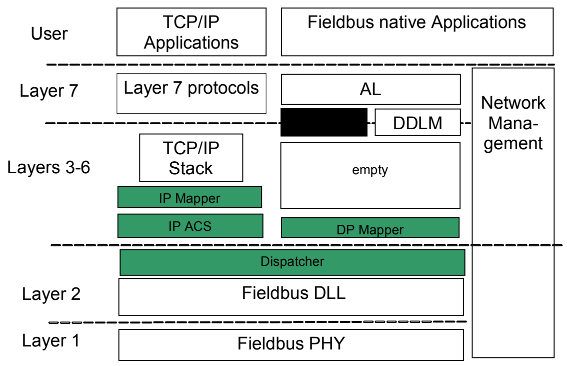

Fig. 1. PROFIBUS and TCP/IP integration

Since this paper focuses in the implementation currently being performed, the interested reader may refer to [2-4] where the architecture of the RFieldbus solution is presented with more detail.

2.1. IP Mapper

The IP-Mapper sub-layer resides directly below the TCP/IP Protocol Stack mapping the TCP/IP services into the PROFIBUS DLL services. It performs the identification, fragmentation and re-assembly of the IP packets to/from PROFIBUS DLL frames. In addition, an appropriate mapping of the Data Link Layer services has to be performed, in order for these datagrams to be properly transmitted. This layer is also responsible for the integration of the client-server model of the IP protocol into the fieldbus communication model. Since the communication model of PROFIBUS is master-slave, it is required to provide extra functionalities to compensate the lack of initiative in slave stations.

2.2. IP ACS

The Admission Control and Scheduling (ACS) sub-layer resides between the IP Mapper and the Dispatcher. This sub-layer is responsible for the control/limitation of the network resources usage by the TCP/IP applications. Each IP packet is classified examining IP Header fields like destination address and port. Given this classification the corresponding fragments are placed in a specific queue.

Moreover, this sub-layer implements the appropriate scheduling policies, in order to provide the desired Quality of Service for the multimedia applications [5].

2.3. DP Mapper

The DP Mapper module is responsible for identifying the type of DP traffic, feeding the appropriate dispatcher queue, accordingly to the DP priority.

2.4. Dispatcher

The Dispatcher sub-layer interfaces the PROFIBUS DP Mapper and the IP ACS to the PROFIBUS Data Link Layer (DLL) (Fig. 1). For transmission, it provides 5 queues concerning the priority of service requests:

- DP high priority (DPH),
- DP low priority (DPL),
- IP high priority (IPH) → IP QoS traffic,
- DP best effort (DPBE)
- IP best effort (IPBE).

The Dispatcher transfers requests from these queues to the DLL, limited by the master allocation time. The requests are transferred at least within the Dispatcher Cycle Time and according to the queue priority. In the receiving side, the Dispatcher identifies the type of message (DP or IP), sending them to the DP Mapper or to the IP ACS, respectively.

3. Windows NDIS interface

In order to obtain a transparent integration of the IP and PROFIBUS stacks, the implementation of the dual stack is being performed at the operating system driver level. Therefore, standard DP and TCP/IP applications do not require changes. It also allows the use of the already available interfaces for the development of network drivers, also facilitating future unforeseen developments.

The implementation is being performed in the Microsoft Windows NT 4 operating system (OS). Within this OS, networking protocols use the Network Device Interface Specification (NDIS) [6] to communicate with network card drivers. Much of the OSI model functionality is implemented in this interface, which allows an easier development of network card drivers.

NDIS defines a fully abstracted environment for network driver development. For every external function that a Network Interface Card (NIC) driver needs to perform, it can rely on NDIS routines to perform the operation. These drivers can be recompiled with a system-compatible compiler to run in any NDIS environment.

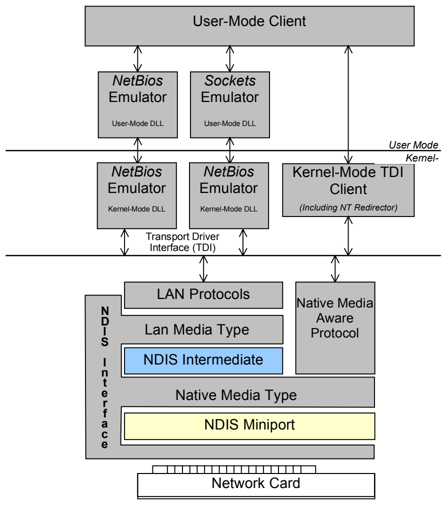

Fig. 2. Windows NT network driver components

The overall structure of the Microsoft Windows NT network driver support is presented in Fig. 2. Within it, three types of NDIS drivers may exist: NIC drivers, Intermediate drivers and Upper-level drivers.

NIC drivers directly manage Network Interface Cards. NIC drivers can be either Miniport drivers or legacy Full NIC drivers. The older Full NIC drivers were built with all the required functionalities to support different network cards. With a Miniport driver, much of these functionalities have been moved to the NDIS library, thus drivers are smaller and easier to write.

Intermediate protocol drivers interface between an upper-level driver such as a legacy transport driver and a Miniport. A typical reason for developing an intermediate protocol driver is to perform media translation between an existing legacy transport driver and a Miniport that manages a NIC for a new media type unknown to the transport driver.

An upper level protocol driver implements a Transport driver Interface, or possibly another application-specific interface at its upper-edge to provide services to users of the network.

4. RFieldbus Prototype implementation

The implementation of the RFieldbus prototype is based on existing portable software, which supports PROFIBUS master and slave functionality [7]. As depicted in Fig. 3, it consists of three main parts: the PROFIBUS firmware, the NDIS Miniport driver and the NDIS Intermediate driver. In addition a card driver DLL is necessary for the PROFIBUS control application.

The NDIS Intermediate driver interfaces with the TCP/IP upper layers and implements the IP Mapper and IP ACS functionalities. The NDIS Miniport is responsible for interfacing with both the Intermediate driver and the DP native Applications.

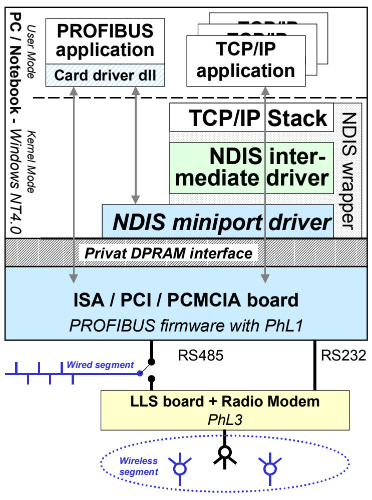

Fig. 3. Architecture of the RFieldbus prototype

4.1. Intermediate Driver

The Intermediate driver (Fig. 4) is responsible for interfacing with the upper protocols (i.e. TCP/IP) on its upper edge, and with the Miniport driver on its lower edge. Both the IP Mapper and the IP ACS modules rely on a common driver support, and interacting using a defined interface. The Intermediate driver's entry/exit functions are NDIS standard calls.

In order to send datagrams, the Transport layer uses the send function registered by the Intermediate driver, by using a standard NDIS call. In the opposite case, where a reassembled IP Datagram is ready to be forwarded to TCP/IP, another standard NDIS call is used. To deliver and receive fragments IP ACS uses standard NDIS calls to communicate with the lower Miniport in a similar manner.

The interface between the IP Mapper and IP ACS modules is performed by using a well-defined interface. Upon reception of fragments from the lower layers, the IPM_Unconfirmed_Fragment_Delivery_Ind primitive is used by the IP ACS to forward them to the IP Mapper. By calling this primitive, the IP ACS passes to the IP Mapper the pointer to the fragment as well as its source address. This extra information is necessary for the identification of the fragment.

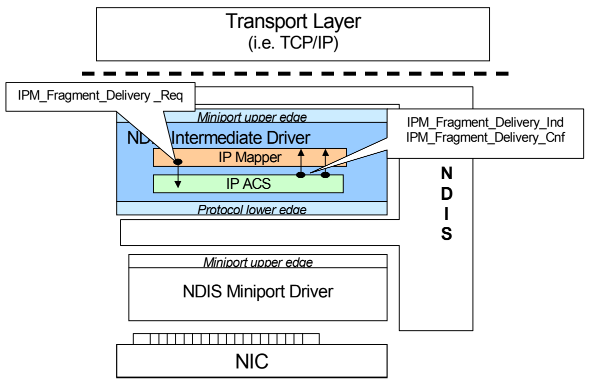

Fig. 4. Intermediate driver

On the other hand, for fragments destined to the lower layers, the IPM_Unconfirmed_Fragment_Delivery_Req service primitive is used to send them to the IP ACS. For every fragment sent to the IP ACS, the IP Mapper expects a confirmation (indicated by the primitive IPM_Unconfirmed_Fragment_Delivery_Cnf). This confirmation, sent by the IP ACS, depends on the delivery status of the fragment to the lower layers.

4.2. Miniport driver

The PROFIBUS Miniport driver has to perform two main tasks:

– to set up the hardware access according to the different board types and
– to manage the exchange of service primitives between the TCP/IP protocol and the PROFIBUS firmware.

The architecture of the Miniport driver is shown in Fig. 5. The access from the network protocols (via the Intermediate driver) is made using an NDIS Miniport interface managed by the NDIS wrapper. The access from the PROFIBUS application, running in user mode, is made using a WDM interface.

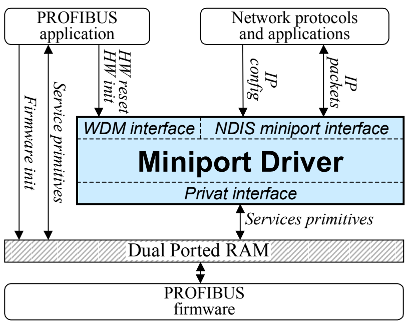

Fig. 5. Architecture of the Miniport driver

The PROFIBUS application is the responsible for initialising and controlling the network board. The driver is started with the Windows boot process. However, it rejects all send packet requests from the network protocols until the PROFIBUS firmware is initialised.

After a successful firmware initialization, the device driver delivers packet send requests from the network protocols to the PROFIBUS firmware, and delivers received packet indications to the upper protocols. Send packets are returned together with state information, when receiving the related send confirmations.

Receiving service primitives from the PROFIBUS firmware is implemented by polling the report area of the Dual Port RAM interface.

4.3. Support to DP and IP Traffic

Fig. 6 shows the architecture of the RFieldbus firmware prototype. The uncolored modules are unchanged and provide the standard functionality of the PROFIBUS-DP protocol. The colored modules are new implementations corresponding to the RFieldbus specification.

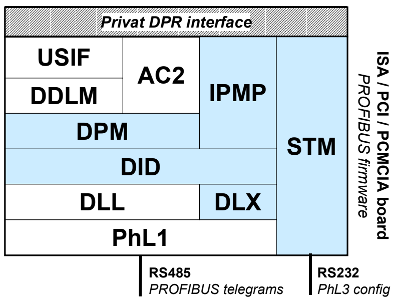

Fig. 6. Architecture of the RFieldbus firmware

The IPMP module is the PROFIBUS application layer extension that realizes the transparent transmission of IP packets via PROFIBUS [8]. It uses standard Data Link Layer services with a special service access point (SAP) and manages configurable application relationships to the remote stations.

The DP-Mapper (DPM) module adds a message cycle time parameter to each send request of the standard DP protocols. This allows the DP/IP-Dispatcher (DID) module to dispatch the DP and IP frames according to the required QoS.

The requests are transferred from the DPM or IPM into the dispatcher queues, if the dispatcher cycle timer is running and the master allocation time is not exceeded. The DID checks the queues when the dispatcher cycle time is expired and puts the requests into one of the DLL queues (high/low priority) as long as the target message cycle time of a request does not exceed the remaining target rotation time. In addition, the traffic of each dispatcher queue is monitored, because the traffic of each dispatcher queue can be limited by a separate limitation time.

In receiving direction the DID has to identify the confirmations and indications and has to pass them from the PROFIBUS DLL to the standard DP modules or to the IPM module respectively.

The DPM and DID modules can be disabled. In this case the requests are transferred from the standard DP and IPM modules directly to the DLL module.

A Station Management (STM) module has been implemented to reset and configure the newly introduced RFieldbus modules. In addition, a control interface has been implemented to pass the management functions to the radio physical layer (PhL3), which is situated on a separate hardware (Fig. 3).

5. Summary and Current Work

This paper presented the architecture of a proposed solution to incorporate support to TCP/IP based applications within PROFIBUS networks. The provided solution allows to guarantee the timing requirements of control-related traffic and to support Quality of Service in the multimedia applications.

Currently, such solution is being implemented within the Windows NT NDIS interface. Additionally, two field trials are being implemented in order to demonstrate the suitability of the proposed approach for the development of multimedia-enabled industrial communication systems.

Acknowledgements

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

References

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

[2] RFieldbus Deliverable D1.3, “General System Architecture for the RFieldbus System”, Technical Report, Sep. 2000.

[3] Pacheco, F., Tovar, E., Kalogeras, A and Pereira, N., “Supporting Internet Protocols in Master-Slave Fieldbus Networks”. Proceedings of 4th IFAC FET Conference, 2001.

[4] E. Nikoloutsos, A. Prayati, A. Kalogeras, V. Kapsalis, S. Koubias, G. Papadopoulos, ”Integrating IP Traffic into Fieldbus Networks”, IEEE – ISIE, Italy 2002

[5] E. Tovar, F. Pacheco, F. Vasques and L. Ferreira, “Industrial Multimedia over Factory-Floor Information Networks”, 10th IFAC INCON Conference, 2001.

[6] Microsoft Windows NT Driver Development Kit documentation. Microsoft, 2000.

[7] Deike, P. ; Hähniche, J. ; Hintze, E. ; Pöschmann, A. : Development of a portable PROFIBUS protocol software (in German). DFAM Research Report Nr. 9/96, Frankfurt (Germany), 1996.

[8] Krogel, P.; Pöschmann, A.; Rauchhaupt, L.: Open Internet Protocol Fieldbus System Tunneling Specification. Addendum to Open Internet API considering the real-time conditions of field devices" (in German). DFAM Research Report Nr. 18/2002, Frankfurt (Germany), 2002