Supporting Internet Protocols in Master-Slave Fieldbus Networks

F. Pacheco[1], E.Tovar[1], A. Kalogeras[2], N. Pereira[1]

[1] Department of Computer Engineering, ISEP, Polytechnic Institute of Porto, Portugal
e-mail: {ffp, emt, i960820}@dei.isep.ipp.pt

[2] Industrial Systems Institute, University of Patras, Greece
e-mail : kalogeras@isi.gr

Abstract

In this paper we describe how to integrate Internet Protocols (IP) into a typical hierarchical master-slave fieldbus network, supporting a logical ring token passing mechanism between master stations. The integration of the TCP/IP protocols in the fieldbus protocol rises a number of issues that must be addressed properly. In this paper we particularly address the issues related to the conveyance of IP fragments in fieldbus frames (fragmentation/de-fragmentation) and on how to support the symmetry inherent to the TCP/IP protocols in fieldbus slaves, which lack communication initiative.

1. Introduction

Fieldbus is a generic name given to fully digital communication protocols for industrial measurement and control applications. Several fieldbus protocols exist, while the general-purpose Fieldbus Communication System European Standard - EN 50170, proposed PROFIBUS, WorldFIP and P-Net as fieldbus protocol standards. [1]

The fieldbus protocol addressed in this paper distinguishes between two types of devices - masters and slaves – and supports both mono-master and multi-master systems. A master can send a message on its own initiative (without an external request), once it gains the right to access the bus. On the other hand, slaves do not have bus access rights and they can only acknowledge or respond to requests from masters.

The medium access control (MAC) mechanism of the fieldbus protocol is based on a simplified logical ring timed token protocol, which is a well-proven solution for real-time communication systems [2,3]. Bus access is based on a hybrid, decentralised/centralised method: masters use a token-passing procedure to grant bus access rights and a master-slave procedure to communicate with slave stations. The token, which represents the right to access the bus, circulates in a logical ring composed by the masters.

An important concept is the *Message Cycle*, which comprises the *Action Frame* sent by the *initiator* (always a master) and the associated A*cknowledge* or R*esponse Frame* from the *responder* (slave or master). The protocol distinguishing between high and low priority messages. The latter can further be divided in three subtypes:

- *Cyclic low priority message cycles (Poll Cycle)*, that represent the execution of the requests contained in the poll-list;
- *Acyclic low-priority message cycles*, which comprise application and remote management services;
- *Gap maintenance cycles*, that are actions taken to determine the status of the others station in order to support dynamic changes in the network.

The need to integrate the TCP/IP stack in a fieldbus protocol, which is mandatory for such applications as industrial multimedia, raises several issues:

1. The transfer of IP packets through fieldbus networks.
2. The provision of full IP functionalities to fieldbus slaves, lacking communication initiative.
3. The impact of such an integration on the fieldbus usual real-time control traffic.
4. The Scheduling and Admission Control mechanisms needed to correctly integrate the TCP/IP traffic (supporting multi-media applications) with the fieldbus control-related traffic.

In this paper we particularly focus on issues 1 and 2.

2. General Design Aspects

The integration can be done transparently from the perspective of the TCP/IP applications (Figure 1) taking into consideration the use of the TCP/IP stack on top of the fieldbus Data Link Layer (DL). Therefore, the fieldbus DL controls the data link, while the TCP/IP stack implements the desired network and transport functionalities. From the perspective of the TCP/IP applications, the fieldbus DL and the adequate interface provide the data link communication services needed to support the exchange of IP packets between TCP/IP communication peers.

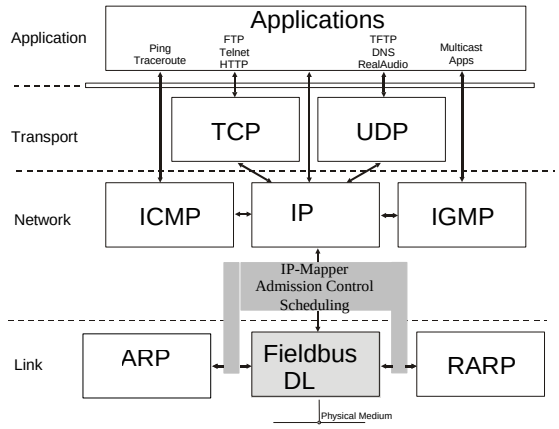

Figure 1 - Integration of a Fieldbus network in a TCP/IP stack

From the Fieldbus DL perspective (Figure 2), this integration can be done considering the TCP/IP stack as an extra DL User, which will be scheduled together with native DL Users. The adequate interface ensures that the TCP/IP related traffic will be converted to fieldbus DL traffic and that it will not jeopardise the timing requirements of the control-related traffic.

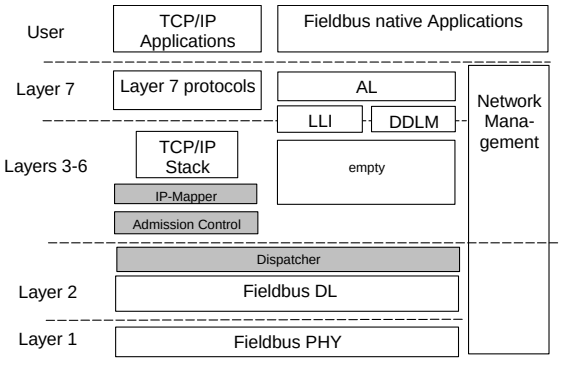

Figure 2 - Higher Layer General Model

Figure 2 shows the layers that need to be inserted on top of the fieldbus DL for the handling of the IP traffic: the IP Mapper, the IP ACS (Admission Control and Scheduler) and the Dispatcher. These layers are responsible for the transparent integration of the TCP/IP stack with the fieldbus protocol stack, taking into account the issues raised by this integration. IP traffic passes through the IP Mapper, the IP ACS and the Dispatcher. In the following paragraphs the functionalities of each one of these layers are presented.

The IP Mapper layer resides directly below the TCP/IP Protocol Stack. This layer is responsible for the conversion of IP packets into/from fieldbus DL frames. Therefore, it maps the TCP/IP services into the fieldbus DL services and performs the identification, fragmentation and reassembly of the IP packets to/from fieldbus DL frames. The IP-Mapper layer is also responsible for the transparent support of the peer-to-peer relationship inherent to the IP protocol, mapping it to the fieldbus DL master/slave structure.

The Admission Control and Scheduling (ACS) layer resides directly below the IP Mapper Layer. This layer is responsible for the control/limitation of the network resources usage by the TCP/IP applications. This limitation can be accomplished through specific traffic scheduling policies, distinguishing the traffic generated by different TCP/IP applications. Such scheduling policies must provide the desired Quality of Service for multimedia applications, while guaranteeing that the timing requirements of the control-related traffic are always satisfied.

The Dispatcher layer resides under the IP ACS Layer. Both fieldbus native traffic and IP traffic pass through this layer. This layer is responsible for the transfer of both kinds of traffic to the fieldbus DL. The dispatcher is responsible for maintaining proper timing constraints, so that the Dispatcher Cycle Time (T_{DCY}) is met. The value of T_{DCY} must be set during the system planning (pre-runtime), according to the timing requirements of the transactions. See [4] for details on these aspects and also on admission control and scheduling.

3. IP Mapper

The IP Mapper is responsible for the conversion of IP packets into/from fieldbus DL frames as well as the transparent support of the peer-to-peer relationship inherent to the IP protocol, when mapping it to the fieldbus DL master/slave structure.

To meet these needs it maps the TCP/IP services into the fieldbus DL services, performs identification, fragmentation and reassembly of the IP packets to/from fieldbus DL frames. The IP Mapper (Figure 3) must include the following entities:

- Fragmentation Entity;
- Reassembly Entity;
- Identification and Routing Entity;
- Switching Entity;
- ID Generation Entity.

3.1. IP-Mapper Fragmentation

A clarification has to be made here on the meaning of IP Fragments in the context of this document. The IP Fragments that the IP Mapper passes to its lower layers take into account the limitations that are imposed by the underlying fieldbus network. In this context, the IP Mapper may receive from the TCP/IP Stack an already fragmented IP packet and fragment it again according to these limitations.

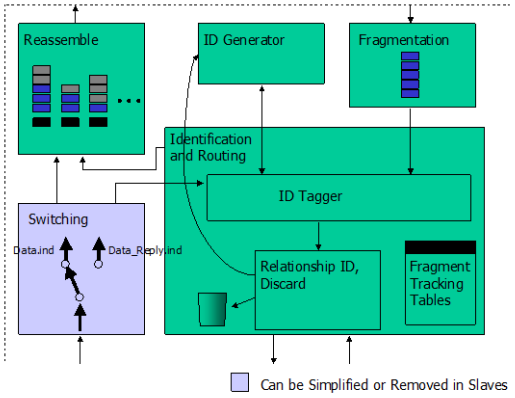

Figure 3 - IP Mapper Structure

There are several reasons for the IP to use a 60-byte header, and most of them do not apply in the fieldbus scenario. In effect, typical fieldbus networks have a very small error rate, a small number of hosts. Therefore the number of IP packets being transmitted between two machines can be limited, without limiting the TCP/IP intended functionality, and the underlying network supports host-to-host communication. In this way the IP encapsulation needed for applications is simplified. For each original IP packet a specific fragmentation protocol can be used, utilising a very small overhead in each fragment, and a larger one on the first one (to send the original IP header). The necessary data is: a Fragment Number and a Packet ID:

- The Fragment Number will be coded in the following way:

Value	Meaning
0	Not Fragmented
1	First Fragment
2	Second Fragment
…	…
127	Last Fragment
128 to 255	Reserved

Table 1 - Fragment Encoding

- The Packet ID will identify the packet to which the fragment belongs.

The implementation of this solution is quite straightforward and very few resources of the machines and the network are used.

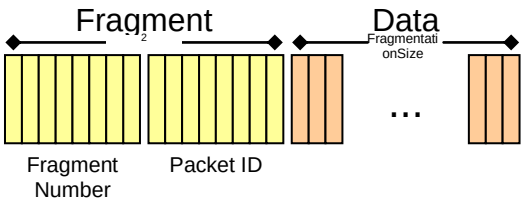

Figure 4 - 2 byte Fragment Header

3.2. Fragmentation Entity

The Fragmentation entity is responsible for the fragmentation of IP packets received from the upper layer into IP fragments.

The IP Datagrams will be fragmented according to the FIFO principle. That means the next fragmentation process will start after the previous one is done. Two control bytes will be added to each fragment. They correspond to the Fragment Header. Figure 5 shows a fragment structure. The packet ID of the newly generated fragments is of the default value, namely 0. The fragment size (FragmentationSize) will be determined by a parameter to be defined at system planing (pre-runtime).

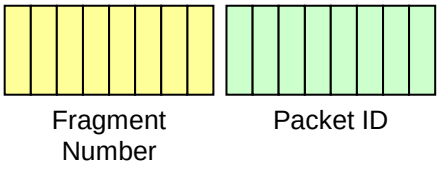

Figure 5 - Fragment Structure

3.3 Reassembly Entity

By the Reassembly entity the IP Datagrams will be reconstructed from the received fragments. The reconstruction will be done based on the algorithm shown in Figure 6: when receiving a first fragment a buffer identified by the packet ID, source address and with a size depending on the amount of the IP datagram will be set up. The other received fragments, with the 2 control bytes ripped off, will be concatenated and kept in the buffer according to their packet ID. These steps will be repeated again and again, until the last fragment of an IP datagram is received, as far as there is no disorder. The reconstructed IP datagram is to be moved to a reserved buffer (PickUpQueue) that is dependent on the interface between TCP/IP stack and the IP Mapper. Since the TCP/IP stack is generally part of an operating system, this interface may be operation system specific. After that the reserved buffer may be released.

Were any disorders among the received fragments that may be determined by the fragment number or there is no more buffer space available, the fragments are to be discarded.

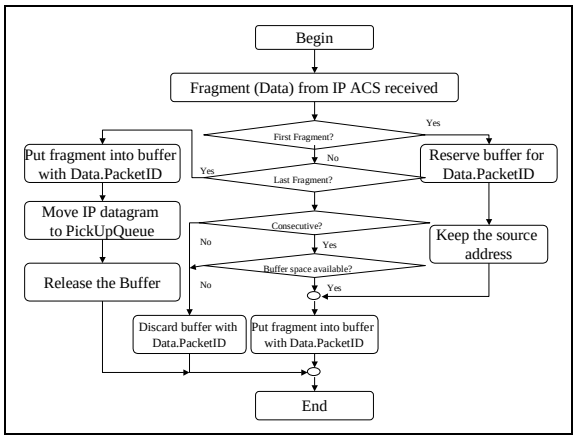

Figure 6 - Algorithm for the Reassembly

3.4. Identification and Routing Entity

As its name suggests, this entity is responsible for the identification of the IP data traffic and its appropriate routing whenever the received fragments are not destined to the local station.

To accomplish this, it relies on information like:

- Protocol;
- Source address;
- Destination address;
- Source port address;
- Destination port address.

This information is collected from the IP header that is available only in the first fragment of each IP datagram. Thus, the packet ID is used to keep track of the fragments that belong to the same IP datagram.

An important characteristic of TCP/IP application protocols is the fact that many servers wait for requests at a well-known port so that their clients know to which port they must direct their requests.

This entity relies on this information (port number, protocol, source and destination address) to correctly identify the different kinds of IP data fluxes. Therefore, it is a task of the system planning to associate these parameters with the Relationship ID of the Relationship entity, where the IP traffic with the defined characteristics will be delivered.

In this way, all the different kinds of traffic can be efficiently handled, indifferently of the application (server or client) or the node (master or slave). Based on this information the appropriate Relationship Entity ID will be determined and fragments will be accordingly transferred.

Routing decisions for fragments bearing the same packet ID will also be made. Possible destinations are the Reassembly entity or the underlying IP ACS. In the case that no match is possible for a fragment, it will be trashed.

In order for the source and destinations addresses to follow the IP addressing conventions, IP Class C network addresses need to be used for all fieldbus network stations. Therefore, they should follow the format Y1.Y2.Y3.X, where X is the fieldbus DLL address and Y1.Y2.Y3 a system wide Class C Network ID (like 192.0.1) as shown in Figure 7.

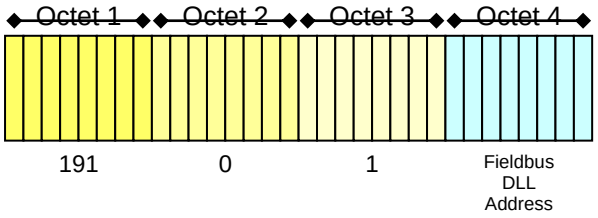

Figure 7 - IP Address Convention

IP fragments are conveyed to the Identification and Routing Entity either through the local user application (via the Fragmentation Entity) or in the case of master stations through the remote application (via the Switching Entity). The ID Tagger will assign a unique packet ID to all fragments of the same IP packet, issued by the ID Generator entity.

The Identification and Routing entity is also responsible for the discarding of packet fragments. Whenever a fragment is discarded, the entity also discards all the other fragments in the IP Mapper layer with the same Packet ID. The discarding of all relevant fragments, will result in the associated Packet ID release.

In the Slave stations, this entity is simplified, since IP fragments are only conveyed to the Identification and Routing Entity through the local user application.

3.4.1. ID Generation Entity

The ID Generation entity is responsible for the management of the packet ID that identifies an IP datagram. All the fragments, that an IP datagram is fragmented to, have the same packet ID value. Packet IDs are needed in order to keep tracking of the received and routed fragments of the same IP datagram.

Depending on the kind of request of the Identification and Routing Entity the ID Generation Entity will:

- search for an unused ID number, reserve it and pass it to the requesting entity;
- release a previously reserved ID number from the Fragmentation Entity.

An ID number will be released in the following cases:

- after the sending of the last fragment of an IP datagram;

- after the discarding of the last fragment of an IP datagram.

3.4.2. Fieldbus - TCP/IP Gateway

This approach also supports the concept of Fieldbus - TCP/IP Gateway, i.e. a station that will forward IP packets to other networks. Each station that supports traffic to the "outside" world must have a configured fieldbus address of the gateway. When sending packets the system checks the network address ID of the packet and the network address ID of the station. If they match, the Host ID will be used as fieldbus destination address. Else the gateway address will be used. In the latter case if there is no gateway address configured, the packet will be discarded. In the TCP/IP Gateway side, these packets will be delivered as any other IP packet to the TCP/IP stack that is responsible to do the routing to the correct network interface (Ethernet, Dial Up…). On the opposite direction (outside world to fieldbus) the TCP/IP Stack will route the packets from the outside to the fieldbus that will treat them as any other packets.

3.5. Switching Entity

The switching entity is the one that receives from the lower layers IP fragments being transferred from other stations. Relevant to the DLL service used, the switching entity passes the fragments appropriately to either the identification and routing entity or the reassembly entity.

On the slave, this Entity can be simplified or even removed, since the slave will not handle IP traffic that is not destined to it, and all the traffic received is directly passed to the reassembly entity.

4. IP Admission Control and Scheduling

The Admission Control and Scheduling (ACS) layer resides directly under the IP Mapper Layer. This layer is responsible for the control/limitation of the network resources usage by the TCP/IP applications. Using specific traffic scheduling policies, capable of distinguishing the traffic generated by different TCP/IP applications this limitation can be accomplished. Such scheduling policies must provide the desired Quality of Service for multimedia applications, while guaranteeing that the timing requirements of the control-related traffic are always satisfied.

4.1. Functionality

The ACS Layer is composed of several Relationship Entities, and a Scheduler implementing a determined scheduling algorithm.

The multimedia traffic can typically be of two types: traffic that does not require stringent timing characteristics (denoted as IP Best Effort traffic – IPBE), and multimedia traffic characterised by specific QoS characteristics, namely bandwidth and jitter (referred to as IPH).

The term scheduling (or scheduling algorithm), used here, refers to the production of a sequence (or order) for the transmission of IP traffic. This schedule is destined mainly for the IPH traffic. IPBE Traffic Queues will be served in a round-robin way.

The scheduling will be done based on differentiated services to be given to the different types of IP traffic. Thus, the scheduler has to be aware of the different types of traffic and the related QoS. The differentiation of the IP traffic will be made using Relationship Entities. Each Relationship Entity receives fragments delivered from the IP Mapper to the IP ACS layer related to a specific QoS (including "Best Effort") and peer to peer relationships. See [4] for more details on scheduling and timing characteristics of the proposed architecture.

The general functionality of the ACS Layer is shown in Figure 8.

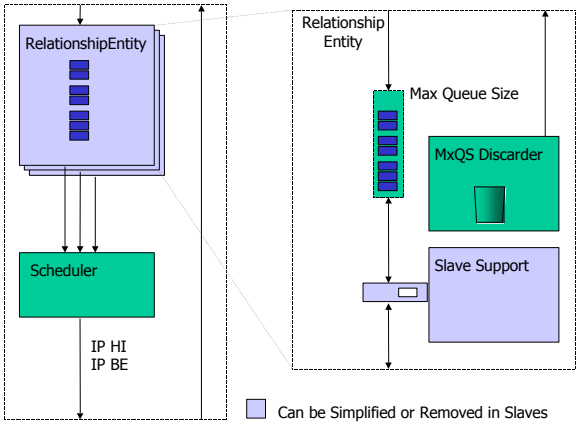

Figure 8 - General Functionality of the ACS Layer

4.2. Relationship Entities

Each Relationship Entity accepts requests from the Scheduler to deliver the traffic to the Dispatcher. When the Relationship Entity successfully delivers a fragment to the dispatcher it will generate a positive confirmation to the IP Mapper.

Each Relationship Entity has a FIFO Queue where the received fragments are stored. This FIFO Queue is characterised by a Maximum Queue Size, when the Relationship Entity receives a request from the IP Mapper that would overload the Queue, it will discard all fragments in the Queue that have the same Packet ID as in the request. It will also generate negative confirmations to the IP Mapper matching all the discarded fragments, including the one that overloaded the Queue.

Each entity should have (IP ACS internal) services to the Scheduler that provide information about the contents of the Entity's Queue and let the Scheduler change in run time the Maximum Size of the Queue.

After the Relationship Entity receives a request for a change on the Maximum Size of the Queue it checks if there are any fragments in the Queue that will overload the Queue. If this is the case, it will discard all fragments with the same Packet ID in the queue and generate negative confirmations to the IP Mapper.

The Relationship Entity will also provide the Scheduler with timing characteristics concerning the message cycles related to the transmission of the fragments in the queue.

Since a fieldbus Slave Station can transfer data only as a response to a Master's request, it is impossible to support multimedia IP traffic from a Slave Station, when no Master originated traffic exists. To overcome this limitation, when a request for a fragment is issued by the Scheduler Entity to an empty Relationship Entity Queue, then the Relationship Entity generates a special frame if the Slave Poll Option is set. This feature of the Relationship Entities is especially useful for the support of multimedia capabilities by DLL Slave Stations, maintaining the QoS defined in System Planning.

Due to limitations on the Slave side, it is a System Planning task to guarantee that one and only one Master polls a Multimedia traffic enabled Slave. However several Relationship Entities can be Slave-Enabled in that Master. If the Slave Pull Option of the Relationship Entity is disabled, it will simply take no action when it receives a request from the Scheduler and no data is available on the Queue.

4.3. Scheduler

A Scheduler Entity is responsible for the appropriate serving of the different Relationship Entity Queues so that all different QoS are respected. The Scheduler Entity uses a service interface, internal to the ACS, for the emptying of the different Relationship Entity Queues or the acquisition of information relevant to their contents.

The Scheduler uses an interface to the Dispatcher layer in order to determine whether it is allowed to fill the Dispatcher queues. The queues that may be fed by the ACS layer are the IPH queue and the IPBE queue.

The Scheduler is executed periodically. There are two different ways to perform the Scheduling of the Relationship Entity queues:

- It may be done off-line. In this case a table in the Station Management gives the actual schedule of the different Relationship Entity queues. This schedule is created off-line, taking into account all needed information so that the diverse QoS requirements of the different Relationship Entity queues are met. In this case the Scheduler actually works as a dispatcher for IP Traffic.
- It may be done on-line. In this case a table in the Station Management gives the parameters needed by the Scheduling Algorithm so that the actual schedule is determined each time the Scheduler is executed. In addition to these QoS specific parameters, the Scheduling algorithm has to take into account the time allocated for IP HP traffic and the remaining time for BE traffic.

The reasoning for these algorithms, timing parameters, and examples are fully approached in [4].

5. Conclusions

This paper gives a general insight to an architecture for the integration of Internet protocols for industrial multimedia over a master slave fieldbus network. The integration of the TCP/IP stack in the fieldbus protocol raises several issues. In this paper we addressed mainly the functionalities that must be added to the fieldbus in order to support transparently (both in the masters and slaves) TCP/IP transactions. Emphasis was particularly given to the issues of fragmentation/de-fragmentation of IP packets and to the support of symmetric internet protocols over the master-slave model of the fieldbus networks.

References

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

[2] Tovar, E. and Vasques, F., "Cycle Time Properties of the Profibus Timed Token Protocol", in Computer Communications, Elsevier Science, 22(13), pp. 1206-1216, August 1999.

[3] Tovar, E. and Vasques, F., "Real-Time Fieldbus Communications Using Profibus Networks", in IEEE Transactions on Industrial Electronics, Vol. 46, No. 6, pp. 1241-1251, December 1999.

[4] Tovar, E., Vasques, F. and Pacheco, F., "Scheduling IP Traffic in RFieldbus", Technical Report, Polytechnic Institute of Porto, HURRAY-TR-0122, March 2001.