Asynchronous Traffic Signaling over Master-Slave Switched Ethernet protocols

R. Marau, P. Pedreiras, L. Almeida
DETI-IEETA Universidade de Aveiro
3810-193 Aveiro, Portugal
{marau,pedreiras,lda}@det.ua.pt

Abstract

Network protocol designers have always been divided between the adoption of centralized or distributed communication architectures. Despite exhibiting negative aspects like the existence of a single point-of-failure in the master as well as computational overhead and an inefficient handling of the asynchronous communications, Master-Slave protocols have always found their space mainly given their simplicity of operation and deployment as well as good control over the communication medium. Along the years, many protocols based in this paradigm have been proposed, with many of them being still used today. Some of the negative aspects traditionally exhibited by these protocols have also been attenuated, e.g. with master replication and master/multi-slave control, but the handling of asynchronous requests is still a limitation concerning the response time and overhead. In this paper, we address the specific case of micro-segmented switched Ethernet networks, where Master-Slave protocols are used to control the load submitted to the switch and prevent high queuing jitter and memory overflows. Particularly, we propose a novel signaling mechanism that reduces the asynchronous traffic response time and network overhead, by exploiting the full duplex channels, and analyze the integration of this mechanism in the FTT-SE and Ethernet Powerlink protocols.

1 Introduction

Ethernet became generalized as a communication protocol in many application domains, well beyond office automation, for which it was originally conceived. Particularly, Ethernet is now a de facto standard in the industrial automation domain, as well as in large embedded systems. Reasons for this dissemination include its low cost, the scalable and high bandwidth, mainly comparing to other fieldbuses, the mature technology, the existence of device drivers for many operating systems and the easy integration with Internet protocol stacks [5].

At the lower levels of automation systems on the plant floor, the applications are commonly time constrained. To cope with this requirement several Ethernet-based protocols were proposed to fill the intrinsic gap associated to the original medium access indeterminism of Ethernet, e.g., EPL (ETHERNET Powerlink) [2], EthernetIP [3], EtherCAT [1], PROFINET [4] and FTT-Ethernet (Flexible Timed Triggered) [9].

The recent evolution to Switched Ethernet brought multiple forwarding paths in a collision free domain, which allows a less stringent control in the transmission instants and guarding windows that were typically required with shared Ethernet. However, switches essentially shifted the collision problem at the communication medium to a congestion problem in the queues, which still introduces some level of timing indeterminism associated to the queuing policies and queue lengths, possibly leading to blocking effects among traffic streams. Consequently, achieving low communication jitter and latency values still depends on the use of specific RT protocols to enforce strict temporal transmission control, as it was typical with shared Ethernet, assuring queuing free transactions (e.g. EPL [2] and PROFINET [4]). For instance in the case of EPL that temporal control is such that all transactions are serialized, thus not even exploiting the availability of multiple forwarding paths made available by the Ethernet switches. Other protocols act upon the input data flows [7] to bound the maximum queuing depths and, consequently, the transmission jitter and latency, and to prevent queue overflows.

Some of the real-time (RT) communication protocols recently developed for switched Ethernet networks are based on the Master-Slave paradigm (e.g. EPL, EtherCAT and FTT-Ethernet). The use of this paradigm reduces the traffic management requirements on the slaves, reduces inconsistent communication scenarios and favors predictability by preventing nodes from transmitting out of time. However, it is also well known that this paradigm is not very efficient in handling asynchronous communications, requiring special mechanisms for that purpose. A common technique consists in having queues within the nodes for the asynchronous traffic, possibly several queues with different priorities, and having the master polling those queues with rates adequate to the degree of responsiveness required by the asynchronous traffic. That poll mechanism encompasses two different phases. The first one, called *signalling phase*, in which the master inquires the nodes about the existence of asynchronous traffic ready to be transmitted, and a *transmission phase*, in which the master polls the transmission of that traffic in adequate instants in time. These two phases can either be separate where the *transmission phase* comes as consequence of the *signalling phase*, or merged in a periodic poll mechanism that leads to unused bandwidth when no messages are to reply the poll.

Notice that whether the two phases are jointly or separately implemented, in both there is an intrinsic compromise between responsiveness and overhead; a higher responsiveness requires polling at higher rates, but polling at higher rates potentially implies an higher overhead bandwidth. In this paper it is proposed exploring the full duplex features of common Ethernet switches to implement a new signaling mechanism in Master-Slave Switched Ethernet protocols, which does not suffer from the referred responsiveness versus overhead compromise and may dramatically improve the QoS given to the asynchronous traffic.

The mechanism herein described can be potentially applied to any master-slave protocol based on full-duplex micro-segmented switched Ethernet infrastructures. For illustration and performance assessment purposes the paper describes how the mechanism can be adapted to the EPL and FTT-SE [8] protocols. The remainder of the paper is structured as follows: the next section describes possible signaling mechanisms. Then the novel signaling mechanism is presented. Afterwards the paper describes how the signaling mechanism can be adapted to the FTT-SE and EPL protocols. The following section presents a response time analysis for the RT asynchronous traffic within FTT-SE and an assessment of the throughput and jitter enhancements brought by the new mechanism. Finally the conclusions close the paper.

2 Background

One of the key aspects that must be considered when developing a hard-RT protocol over shared Ethernet networks is that collisions must be completely avoided, in face of the non-deterministic collision resolution mechanism employed by this standard. Therefore, all the message transmission instants are rigorously controlled to avoid collisions. Protocols such as EPL and FTT-Ethernet enforce this behavior with a Master-Slave architecture, centralizing all the communication control in the Master node; slave nodes can only issue message transmission after being explicitly polled by the Master. Despite the intrinsic absence of collisions, the Master-Slave paradigm is still useful in micro-segmented switched Ethernet topologies to control the switch queuing and thus enforce timeliness guarantees, particularly when very constrained values for jitter and end-to-end latency are required.

One negative aspect of the Master-Slave paradigm is the overhead. This overhead comes in two forms, one due to the need to transmit the *master poll* and another caused by the time spent by the slaves decoding and preparing the reply to the poll, which is normally called the *turnaround time*. This interval of time, which mediates between the poll reception and the beginning of the reply transmition, depends on the slave implementation technology. For standard PC-based solutions and network interface cards (NICs) can reach 200 μs, even using RT kernels and stacks. Reducing this value to a few tens of microseconds is possible with special hardware support.

The absence of transmission autonomy makes the slaves fully master dependant, which turns out to be a bottleneck when developing asynchronous messaging schemes in Master-Slave protocols. Several techniques have been developed to override this constraint:

- Using a periodic polling mechanism (e.g. FTT-Ethernet), oriented to the individual asynchronous streams in a blind way. The Master treats the asynchronous messages (AM) like the synchronous (periodic) ones, periodically scheduling them, with a period equal to their minimum inter-transmission times ($Tmit$), without knowing whether there are pending requests for its transmission or not. The only difference concerning the periodic traffic is that for the AMs the absence of an actual message transmission after a Master poll does not mean an error but instead signals the absence of pending asynchronous requests. Deterministic time guarantees can also be given for these messages but with a, normally, very large degree of pessimism associated to the inefficient use of the bandwidth. The signaling latency, i.e., the interval of time from the moment an asynchronous message becomes ready in a node and that information reaching the master, can grow up to two times $Tmit$ in the worst case.

- A bandwidth reclaiming mechanism can be applied on top of the periodic polling mechanism above described, as a feedback scheme to detect unanswered polls, re-using that bandwidth and improving the average network throughput. Unanswered polls can be either autonomously sensed by the master or explicitly reported by the respective slave with a short message. The unused time can then be re-allocated with a follow up polling message. Notice that this mechanism has no impact on the AM worst-case response time, which happens when there are pending requests in all Master pools, but may have a positive impact on the average throughput since unused bandwidth may be reallocated. This technique is easily deployed in processor systems [6] to re-use the budget left free by a task. However, this model is not always efficiently transportable to the network level. An effective adaptation is proposed by Nolte in the Server-CAN protocol [10] (Master-Slave with cyclic scheduling windows in CAN) that reclaims the bandwidth by anticipating the next cycle. In this case the Master takes advantage of the broadcast nature of the CAN network to readily infer the state of the nodes asynchronous queues. However, switched Ethernet networks support unicast and multicast traffic, and thus this mechanism in not directly applicable, and a practical implementation would require an explicit acknowledge mechanism, which would be much more complex to manage and would consume bandwidth.

- An in-band backward signaling scheme is used, e.g. EPL, where the backward information is piggybacked onto the synchronous traffic. This approach has two main problems. On one hand, additional bandwidth is used if the application does not require the node to produce any synchronous message. In that case, a synchronous message must be added just to implement the backward channel. On the other hand, the responsiveness to asynchronous requests dependends on the polling rate of the backward channel. This may also result in overhead since it may be necessary to poll synchronous message more often than necessary just to allow the polling of the asynchronous queues at rate compatible with the desired asynchronous traffic responsiveness.

- A backward signaling mechanism allowing the slaves to periodically report to the master the current status of their asynchronous queues. The master, then, only polls the asynchronous traffic when it is ready in the node queues. Provided that such backward channel is available and does not consume extra bandwidth, e.g. using out-of-band communication, this solution is optimal with respect to bandwidth utilization since all the polls correspond to an asynchronous request and no bandwidth is consumed by the signaling procedure .

In this paper we propose a signaling mechanism similar to the backward signaling mechanism above listed, i.e., using a backward channel that is, essentially, out-of-band with respect to the regular messages, exploiting the full duplex features of current Ethernet switches.

3 Out-of-band signaling scheme

In Master-Slave protocols, the traffic is scheduled and controlled by means of a Master poll message. This token may be associated with a single slave message transmission or may delimit a transmission window for multiple slave messages (Master/Multi-slave). In any case the slave nodes are synchronized by the poll message reception and, therefore, it is of utmost importance transmitting this message with no interference. This behavior is typically achieved by reserving transmission windows wide enough to ensure that all the traffic related with a transaction is completely dispatched by the switch before

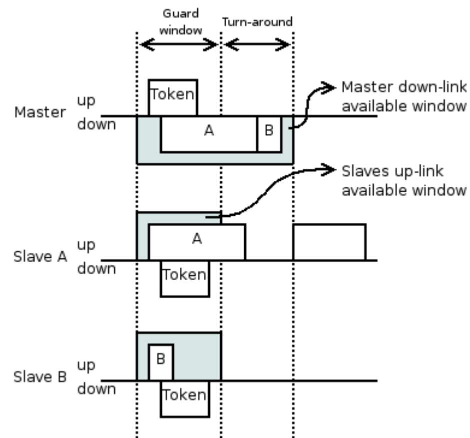

Figure 1. Token

the beginning of the subsequent transaction. In practice, this window includes the polling message and the reply transmission time, the turn-around time necessary for the slaves to decode and prepare the reply and guarding windows to compensate the jitter inherent to each one of these operations.

In a full-duplex switch, the Master download link is not included in the token switching path, i.e messages transmitted in unicast to the Master do not interfere with the actual token message transmission (see Fig. 1). Therefore, the slave nodes may safely report to the Master node the internal state of its asynchronous queues during the guarding window and the turn-around window. Provided that these messages end their transmission before the end of the turn-around window there is no impact on the regular protocol messages and the slaves turn-around time is also not affected, since the report messages are submitted to the NIC during the transmission of the poll message by the Master. Figure 1 shows one situation in which two nodes send their status messages (A and B) to the Master. Both messages are completely received within the time gap defined by the poll message transmission plus the turn-around windows, thus not interfering with the normal protocol operation.

Thus, our proposal relies on this transmission scheme to handle the backwards signaling information containing the asynchronous queuing status in the Slaves. It can be applied to any Master-Slave full-duplex Switched Ethernet protocol, as long as the Slaves are able to synchronize with the transmission instant of the following Master polls.

The following Sections describe the inclusion of this signaling mechanism in two distinct protocols, FTT-SE and EPL, which have different ways to handle the asynchronous traffic.

4 Implementation on FTT-SE

FTT-SE [9] is a RT protocol that exploits the advantages brought by the Ethernet micro-segmentation. Such advantages include a throughput enhancement by taking

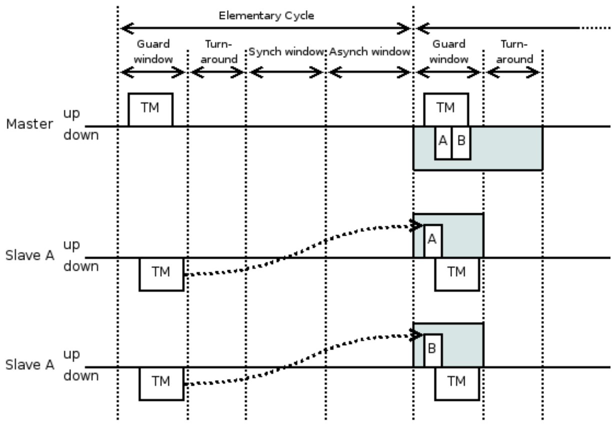

Figure 2. FTT-SE signaling proposal

advantage of the multiple forwarding paths and the full-duplex links, as well as the timing relaxation in the slaves by taking advantage of the collision free domain. In this protocol the Master poll message, known as Trigger Message (TM) is periodically transmitted, polling the Slaves to transmit messages in two consecutive and disjoint time windows, designated synchronous and asynchronous windows. The synchronous window is associated with the transmission of the synchronous (periodic) traffic, which is centrally scheduled by the Master node. The asynchronous traffic (sporadic) is transmitted in the asynchronous window. Associated with each sporadic message there is a minimum inter-transmission time ($Tmit$) that, when associated with the message size, defines the maximum bandwidth consumed by the respective message stream. The FTT-SE protocol, uses the $Tmit$ to periodically poll the asynchronous messages as if the messages had a periodic activation. However, contrarily to what happens with the periodic traffic, the poll of a sporadic message may or may not result in an actual transmission, depending on the existence or not of pending requests. Therefore, this polling mechanism consumes all the bandwidth allocated to a given sporadic message independently of the actual transmission request rate.

In the FTT-SE protocol the system granularity is stated by the *Elementary Cycle* (EC) which is headed by the TM transmission and followed by the turn-around window, Synchronous window and Asynchronous window. Figure 2 sketches the EC structure with an example of the signaling scheme herein proposed.

As described in Section 3, the signaling channel uses the reverse path taken by the periodic Master message, avoiding interfering with the normal protocol operation. Therefore, the slaves are required to synchronize themselves with the Master before being able to report its status.

Each signaling message include information regarding the asynchronous queues status of the associated node and shall not use more than the minimum payload required for an Ethernet frame. The idea is to transmit one signaling message per node in each EC. This option limits the number of nodes in the system to the ability of fitting all the messages below the TM size and the turn-around time.

The turn-around time plays a significant role in this scheme. In FTT-SE this parameter is configurable and must be properly tuned according to the node's performance. Typically must be set to a value no lower than the worst-case turn-around time among all the nodes present in the system.

The following equation allows computing the maximum number of nodes ($MaxN$) with respect to a system s with a given turn-around time, Tr:

$$MaxN(s) = \frac{Tr(s) + TM\_size(s)}{SIG\_size(s)}$$

where $TM\_size(s)$ stands for the trigger message transmission time and $SIG\_size(s)$ for the signaling message transmission time. Assuming a Fast Ethernet network (100 Mbps), $TM\_size(s)$ equals $24\mu s$ and $SIG\_size(s)$ takes the minimum Ethernet frame transmission time, which is $6.72\mu s$. Assuming also that the system nodes are based on standard PCs architectures and use a RT kernel and stack, yielding a typical turn-around time (Tr) of $200\mu s$, the maximum number of nodes ($MaxN$) is 33. Notice that for applications requiring more nodes or exhibiting lower turn-around times (specialized hardware) two approaches can be used to avoid this scalability constraint. One is to reduce the rate at which nodes issue the signaling messages. This approach implies a negative impact on the signaling latency but permits supporting an arbitrary number of nodes. Other possible approach is to extend the Master downloading window beyond the turn-around time using the synchronous window for the purpose. This would require the proper adjustment of the synchronous messages scheduler to include this extra traffic in the Master downlink but has the advantage of potentially not affecting the Slaves application since typically regular application messages are not directed to the Master.

5 Implementation on EPL

The Ethernet Powerlink protocol [2] is a Master-Slave protocol that supports both the shared and the switched Ethernet media. This protocol aims at real-time jitter-sensitive applications and enforces strict time restrictions.

Similarly to the FTT-SE protocol, the EPL protocol employs a bi-phase cyclic communication structure, with the time divided in *Elementary Cycles* (EC) and the ECs comprising an isochronous and an asynchronous phase. Heading the EC, a Start of Cycle poll message (SoC) indicates the beginning of the isochronous phase. Thereafter, for each message (Pres) polled in the isochronous phase, a Master poll is transmitted (Preq). Each individual polling is issued in a specific time slot within the isochronous phase to reduce the message jitter. The number and size of the slots is parametrized offline. The asynchronous phase

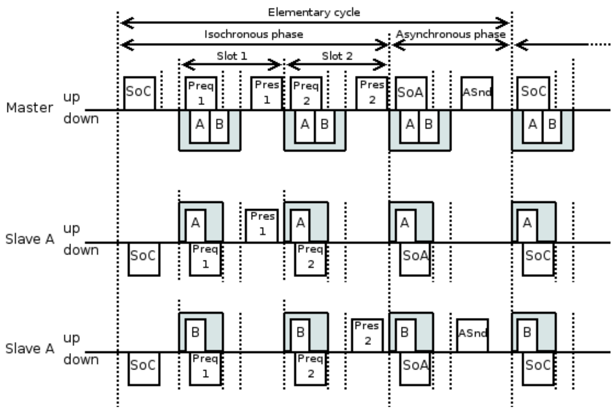

Figure 3. EPL signaling proposal

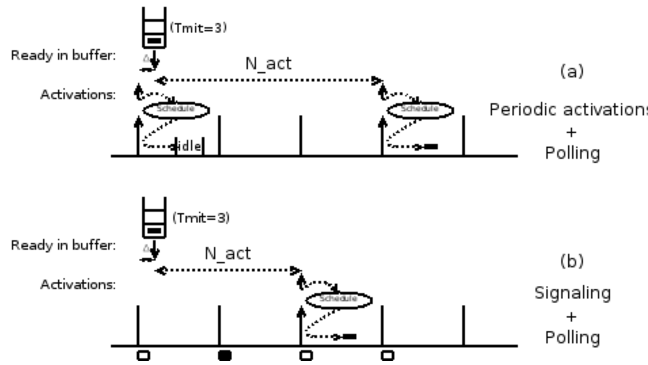

Figure 4. FTT-SE activations response time

starts with a Master message (SoA) polling a single asynchronous message in each EC.

The asynchronous traffic is totally managed in the *Managing Node* (MN) after explicit transmission requests by the *Controlled Nodes* (CN). The request for transmission can be either sent piggybacked in the synchronous messages or when explicitly requested by the MN, using in this latter case a specific asynchronous message for the purpose. The asynchronous requests include also the requested message priority. The MN keeps track of all pending requests, including their owns, and in the asynchronous phase of each EC schedules the highest priority one for transmission, as explained above.

Besides the explicit AM status requests, the EPL protocol also considers the polling of the nodes identification to verify the network status and take the necessary procedures on connectivity events.

The out-of-band signaling mechanism described in Section 3 may also be used in EPL both to reduce the signaling latency and minimize the protocol overhead due to the explicit polling for the asynchronous status and nodes connectivity or due to the over-scheduling of isochronous messages for faster signaling purposes. Figure 3 sketches an EPL elementary cycle including the signaling scheme. Like in the FTT-SE approach, the Slaves are synchronized with all the Master polls so that the signaling messages can be unicasted to the Master.

6 Comparative analysis

The $Tmit$ parameter sets the maximum transmission rate that a given AM may have and, in conjunction with the message length, bounds the message maximum bandwidth utilization ($C/Tmit$). However, frequently the sporadic messages have average activation rates significantly lower than the maximum one. With the backward signaling scheme the network bandwidth resulting from the missed activations can be reclaimed. The RT traffic is typically admitted against the worst-case scenario, and thus reclaimed extra bandwidth does not result in better schedulability levels for the RT assynchronous traffic. However, the reclaimed bandwidth allows for reducing the average response time of those AMs, since the activations absent anticipates the pools for the lower priority traffic. Furthermore, the reclaimed bandwidth may also be used by the background non-RT traffic, which can see its average throughput significantly improved.

Besides the average gains in bandwidth and response time, the backward signaling scheme may also have a positive impact on the runtime schedulability of the asynchronous RT messages due to a potentially prompter acquisition of the nodes state by the Master node. Let us define the *activation delay* (N_act) as the time that goes between the AM deployment in a Slave queue and its inclusion in the Master scheduler. The activation delay is part of the AM response time (AM_Rt), which can be computed as follows:

$$AM\_Rt = N\_act + Jsch + C$$

where C stands for the AM size and $Jsch$ for the scheduling jitter which varies with the scheduling algorithm and the applied load. Therefore, taking two similar scenarios differing only in the applied signaling mechanism and maintaining the same scheduler and load, it is possible to evaluate the impact on the AM_Rt due the N_act variation.

Taking the FTT-SE implementation case study, Fig. 4 sketches a timeline with the worst case situation for the *activation delay* (N_act) in two scenarios. In (a) it is used a pure polling mechanism where the master periodically schedules the AM in a blind way, while in (b) it is represented the backward signaling mechanism operation. In both scenarios the asynchronous message is registered with a minimum inter-transmission time of 3 ECs.

When activations are periodically triggered (a), the Slave may ultimately have an AM transmission request right after the last poll, leading to an $N\_act = Tmit - \Delta = 3 * LEC - \Delta$, where Δ is an infinitesimal and LEC in the EC length. In this scenario the activation delay N_act varies linearly with message's $Tmit$. However, when the backward signaling scheme is applied (b), once an AM transmission in requested it produces a signal that is transported in the following EC and is then included in the Master scheduler. The worst-case delay is, in this case, independent on the message's $Tmit$

and fixed to $N\_act = 2 * LEC - \Delta$. The message activation signal takes at worst case 1 EC to reach the Master. After reaching the Master the request is eligible by the scheduler and the schedule result is dispatched on the following EC.

The best case activation delay happens in (a) when the message is queued right before its poll ($N\_act = \Delta$), and in (b), when it is queued before the signaling message ($N\_act = 1 * LEC + \Delta$). Assuming that the messages are asynchronous and randomly triggered by the application we may preview a uniform distribution in the queuing events, leading to an average delay between the best and the worst cases.

Table 1. Generalized activation delays

scenario	Worst Case	Best Case	Average
(a)	$Tmit$	0	$Tmit/2$
(b)	$2 * LEC$	$1 * LEC$	$1.5 * LEC$

Table 1 outlines the activation delays for the two depicted scenarios and for the average case. In the assumed average condition, we may see that the proposed signaling scheme (b) induces a better asynchronous responsiveness for discrete values of $Tmit$ greater than 2 ECs. It is thus obvious the benefits for messages registered with long inter-transmission times.

7 Conclusions

The advent of switched Ethernet has opened new perspectives for real-time communications over Ethernet. However, a few problems subsist related with queue management policies, queue overflows and limited priority support. To overcome such difficulties several real-time protocols were proposed. Some of these are based in the Master-Slave paradigm, which, despite exhibiting many interesting properties like simplicity of operation and deployment and good control over the communication medium, is also known by its inefficient handling of the asynchronous traffic. This paper presents a novel signaling mechanism, suitable for master-slave protocols based on full-duplex micro-segmented switched Ethernet networks. It allows the slave nodes to periodically inform the Master about their status in particular time instants during which the communication path between the Slaves and the Master is idle, thus without interfering with the normal protocol operation. The inclusion of this signaling mechanism brings important advantages both in terms of the asynchronous messages responsiveness and protocol overhead. The paper illustrates how the mechanism can be applied to the FTT-SE and EPL RT protocols and shows the potential gains in terms of responsiveness and overhead reduction. The signaling of the AMs status is the most direct use for the permanent backward channel, however, as future work it is planned extending the use of this channel for other protocol operations like configuration during setup phase, supporting e.g. plug&play protocol.

References

[1] EtherCAT - Ethernet for Control Automation Technology specifications. http://www.ethercat.org/.

[2] Ethernet Powerlink protocol home page. http://www.ethernet-powerlink.org/.

[3] Ethernet/IP (Industrial Protocol) specifications. http://www.odva.org.

[4] Real-Time PROFINET IRT. http://www.profibus.com/.

[5] J. Decotignie. A perspective on Ethernet as a fieldbus. In *Proceedings of FeT'2001, 4^{th} International Conference on Fieldbus Systems and Their Applications Nancy, France*, pages 138–143, 2001.

[6] Deepu C. Thomas, Sathish Gopalakrishnan, Marco Caccamo, and Chang-Gun Lee. Spare CASH: Reclaiming Holes to Minimize Aperiodic Response Times in a Firm Real-Time Environment. In *Proc. on the 17th Euromicro Conference on Real-Time Systems (ECRTS'05).*, pages 147–156. IEEE, July 2005.

[7] J. Loeser and H. Haertig. Using Switched Ethernet for Hard Real-Time Communication. In *Proc Parallel Computing in Electrical Engineering, International Conference on (PARELEC'04)*, pages 349–353, Dresden, Germany, Sept. 2004.

[8] R. Marau, L. Almeida, and P. Pedreiras. Enhancing real-time communication over COTS Ethernet switches. In *WFCS'06: IEEE International Workshop on Factory Communication Systems*, pages 295–302, 27 June 2006.

[9] P. Pedreiras, P. Gai, L. Almeida, and G. C. Buttazzo. FTT-Ethernet: a flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems. *IEEE Transactions on Industrial Informatics*, 1(3):162–172, Aug. 2005. ISSN: 1551-3203.

[10] Thomas Nolte, Mikael Nolin, and Hans Hansson. Server-based scheduling of the CAN bus. In *Proc of the 9th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA'03)*, pages 169–176 (volume1), Lisbon, Portugal, Sept. 2003. IEEE Industrial Electronics Society.