# A Time-Triggered Ethernet (TTE) Switch

Klaus Steinhammer Petr Grillinger Astrit Ademaj Hermann Kopetz
Vienna University of Technology
Real-Time Systems Group
Treitlstr. 3/182-1, A-1040 Vienna, Austria
E-mail:{klaus,grilling,ademaj,hk}@vmars.tuwien.ac.at

## Abstract

*This paper presents the design of a Time-Triggered Ethernet (TTE) Switch, which is one of the core units of the Time-Triggered Ethernet system. Time-triggered Ethernet is a communication architecture intended to support event-triggered and time-triggered traffic in a single communication system. The TTE Switch distinguishes between two classes of traffic. The Event-Triggered (ET) traffic is handled in conformance with the existing Ethernet standard, while the Time-Triggered (TT) traffic is transmitted with temporal guarantees. A TTE Switch is used in the Time-Triggered Ethernet system for exchanging time-triggered messages in a time-predictable way while continuing the support of standard Ethernet traffic in order to use existing networking protocols such as IP, UDP or IPX without any modifications.*

*In this paper we present the mechanisms the TTE Switch uses to guarantee a constant transmission delay for time-triggered traffic. Also an experimental validation of these mechanisms is given.*

## 1. Introduction

The temporal accuracy interval of information in a distributed real-time system is affected by the precision of the global view of time. This precision depends on the jitter of the message transmission delay over the network and the communication protocol. A predictable real-time communication system must therefore guarantee the message transmission within a constant transmission delay and bounded jitter [7]. Due to the open nature of Ethernet it is difficult to guarantee bounded transmission delays in systems that use standard Ethernet networks.

The duration of the message transmission over the network depends on the characteristics of the network traffic. We distinguish between two different scenarios: *cooperative senders* and *competing senders*. If senders are competing (as in standard Ethernet [5]) there is always the possibility that two ore more messages are sent to a single receiver simultaneously. There are two possibilities to resolve this conflict: back-pressure flow control to the sender (this is the choice of the bus-based “Vintage Ethernet” [6]) or the storage of the messages within the network (this is the choice of switched Ethernet). Both alternatives involve increased message transmission jitter which is unsatisfactory from the point of view of real-time performance. We therefore conclude that in real-time systems the senders must cooperate to avoid conflicts. This cooperation can be achieved through coordination by reference to a global view of time.

In many industrial domains, e.g. the automotive and aerospace domain, the functional range of distributed applications increases, and therefore the bandwidth requirements are reaching the limits of the currently available communication mediums. This creates a demand for a new communication medium for implementing these distributed applications. To drive towards certification, many applications are designed using the time-triggered architecture. So a new communication medium is needed, which supports this architecture. As Ethernet is well established in the event-triggered world, there are many approaches which try to adapt Ethernet in a way, that it can be deployed in applications where temporal guarantees are necessary [3, 4, 9, 11, 16]. A summary of these implementations can be found on the Real-Time Ethernet web-page [1],a comparison in [3] and [4]. All these solutions use the concept of *cooperative senders* in order to guarantee constant transmission delays.

Time-Triggered Ethernet [8] allows competing senders to *coexist* with cooperative senders on the same network while yet preserving the temporal predictability of the traffic among the cooperative senders. To integrate competing traffic seamlessly in the same network without interference of the cooperative traffic, Time-Triggered Ethernet introduces two message classes using the standardized Ethernet frame format [5]:

- The standard Ethernet messages used for Event-Triggered (ET) traffic by competing transmitters.

- The Time-Triggered (TT) messages, transferred among cooperative senders.

Standard Ethernet messages are transmitted in those time intervals where the communication medium is not needed for transmission of time-triggered messages. In order to avoid that standard Ethernet traffic affects the temporal properties of the time-triggered traffic, time-triggered messages preempt all standard Ethernet messages which are in their transmission path.

Introducing these two classes guarantees the compatibility to standard Ethernet "commercial of-the-shelf" (COTS) components and existing networking protocols without any modification of the competing transmitters on the one hand and the possibility to realize a scheduled message transfer between all cooperative senders by establishing and maintaining a global time base on the other hand.

This paper presents the design and implementation of the Time-Triggered Ethernet (TTE) Switch that integrates standard Ethernet traffic and time-triggered traffic in one communication architecture by guaranteeing predictable transmission delays for time-triggered traffic. The TTE Switch is part of a TT Ethernet system that uses the TT Ethernet protocol to integrate the required services:

(i) Predictable transmission of time-triggered messages,
(ii) Flexible communication schedule for standard Ethernet messages, and
(iii) Clock synchronization.

This paper is organized as follows: Section 2 gives an overview of the TT Ethernet architecture. The hardware architecture of the TTE Switch is presented in Section 3, whereas the functionality is given in Section 4. Section 5 presents the experimental validation results. It compares the TTE Switch to a standard COTS switch. The results of these experiments are discussed in Section 6 and the paper is concluded in Section 7.

## 2. Time-Triggered Ethernet Architecture

A TT Ethernet system consists of a set of computer nodes that communicate via the TT Ethernet protocol. A node consists of a host computer and an Ethernet controller that is connected to a switch via a bidirectional point-to-point link (Figure 1). User applications are executed on the host computer, whereas the communication controller is in charge of executing the TT Ethernet communication protocol.

The TT Ethernet protocol distinguishes between two different classes of traffic:

- The event-triggered traffic that is not required to meet strict temporal requirements regarding the transmission delay.
- The time-triggered traffic whose transmission is scheduled based on a conflict-free scheme and which have to meet strict temporal requirements regarding the transmission delay.

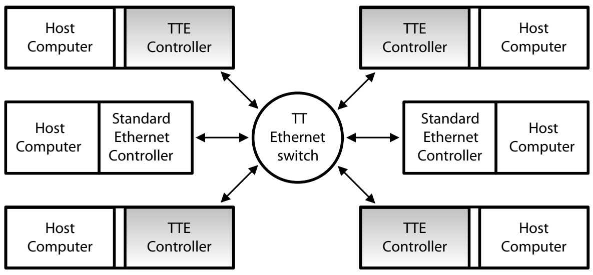

**Figure 1. Time-Triggered Ethernet System**

In the case of standard Ethernet messages, the transmission of a message is not anticipated (not scheduled) and therefore, standard Ethernet messages, originating from networking protocols like IP, UDP or IPX, are ET messages.

The TT Ethernet message format is based on the format of the standardized Ethernet message (Figure 2) according to the IEEE 802.3 standard [5]. The standardized Ethernet frame header contains a two-byte *Type Field* that specifies the type of an Ethernet message, which provides a context for interpretation of the data field of the frame (for protocol identification). The contents of the Type Field is administered by the Ethernet standard authority of the IEEE in order to assure a consistent development of Ethernet. This standard authority has assigned the value $0x88d7$ as the content of the Type Field to uniquely identify a TT Ethernet message and the associated message protocol.

Since TT messages are scheduled and ET messages are not, a run-time conflict situation between a TT message and ET traffic might occur. A COTS switch will handle all messages with identically and therefore, the jitter caused by a COTS switch is not bounded: In case of a buffer overflow at a port, some messages will be lost. An "intelligent" COTS switch that implements a strict priority based Quality of Service (QoS) mechanism will provide better performance. The jitter will be bounded by the duration of longest packet transmission (including the inter-frame gap) which is $123\,\mu s$ at 100 Mbit/sec network speed. Additionally, it is possible to implement an early packet dropping mechanism for the ET message queues that prevents buffer exhaustion for the TT packets. Some COTS switches support these features (e.g. the "Catalyst" switches manufactured by Cisco). However, even with an "intelligent" COTS switch, the jitter would be too high for a communication system like the experimental setup described in Section 5. Moreover, no available solution provides a deterministic resolution of collisions between equal message classes (i.e. a consistent order of delivery to all receivers).

To avoid the listed problems of standard Ethernet sys-

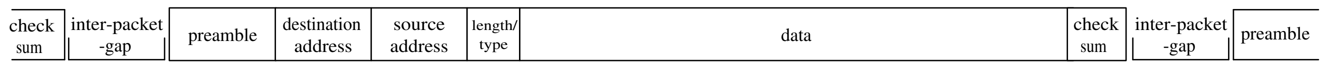

**Figure 2. IEEE 802.3 layer 2 frame format**

tems, TT Ethernet introduces a specific switch, which is denoted as TTE Switch. The TTE Switch guarantees predictable transmission delays for TT traffic, even in the case when ET and TT messages are transmitted using the same Ethernet network. Whenever there is a run-time conflict between ET traffic and a TT message the TTE Switch preempt the ET traffic and transmits the TT message with a constant transmission delay. The TTE Switch autonomously retransmits any preempted ET message immediately after the transmission of the TT message has finished (using a best-effort method). The TTE Switch also guarantees that all receivers receive TT messages in the same order.

By implementing these mechanisms it is possible to use the TT Ethernet communication network for both real-time and non-real-time network traffic and achieve a global time with a precision in the range of microseconds.

For details on the TT Ethernet protocol see [8].

## 3. TTE Switch Architecture

Unlike a COTS switch, the TTE Switch provides the following features:

- Classification of traffic into ET and TT packets,
- A constant transmission delay for TT traffic which is a priori known,
- Memory-less transmission path for TT traffic,
- Preemption of ET packets,
- Retransmission of preempted ET packets, and
- ET packet handling according to the Ethernet standard.

To achieve predictable temporal behavior we have developed a TTE Switch using a custom made board based on an Altera EP1C20 FPGA and an eight-port Ethernet physical layer transceiver (Ethernet PHY).

The transceiver is used for signal shaping and voltage level conversion between the I/O standard used by the FPGA and the Ethernet signalling standard. The exchanged data are *Ethernet layer 2* packets with the frame format shown in Figure 2.

The Ethernet frame header contains all data that are needed by the TTE Switch. The *source* and *destination addresses* are used for routing and the *Type Field* is used to distinguish between ET and TT frames.

After the identification of the frame type, the ET frames are stored in buffers. This is necessary since the whole message has to be sent again if the transmission has been preempted. If the message is fetched from the buffer, it is switched to the output port(s) according to its destination address.

TT frames on the other hand are directly switched to their destination ports. They are not written into a buffer and whenever there is a run-time conflict between a TT message and a ET message, the TTE Switch preempts the ET message and transmits the TT message. The transmission of two or more TT frames at the same time is considered as an error and must be prevented by the communication protocol. Anyway, if this case occurs, only one packet is forwarded — the one that has arrived first. If two TT packets arrive simultaneously, the TTE Switch has priorities assigned to its ports and only the packet coming from the port with the highest priority is transmitted.

As the classification of a packet is stored in the packet itself, the TTE switch does not need additional configuration data to operate, this means that it does not have to be configured and can be used “out of the box”.

## 4. TTE Switch Functionality

To achieve determinism of the transmission delay, it is important that each TT packet takes the same route through the FPGA. For this reason a symmetric layout of the data paths was chosen. The data-flow of a single channel is shown in Figure 3.

To keep the clock rate in the core design low, the serial data from the transceiver is parallelized to a 9 bit wide data stream which consists of eight data bits and one status bit to mark the validity of data. While the Ethernet PHY is capable to support 10 and 100 Mbit/sec network speeds, the TTE Switch supports *only 100 Mbit/sec* due to the fact that lower transmission speeds are covered by fieldbus communication systems (TTPC: up to 25 Mbit/sec [15], FlexRay: up to 10 Mbit/sec [2], CAN: up to 1 Mbit/sec [13]).

At the *input shift register*, the source address and the packet type are extracted from the data stream. By using a shift register, header data is extracted out of the data stream of a frame after a dedicated amount of clock cycles following the start of a frame. Applying a shift register with 26 stages all the data can be obtained after 25 clock cycles. The last stage is used to buffer the data while the *control module* empties the data path through the TTE Switch if a TT packet is identified.

As the whole packet travels through the shift register, all data are delayed by a fixed amount of time depending on the length of the shift register. To meet the requirements of the memory-less transmission path, the length of one shift register has to be less than the minimum frame length, so that there is not enough space to store a complete packet. If

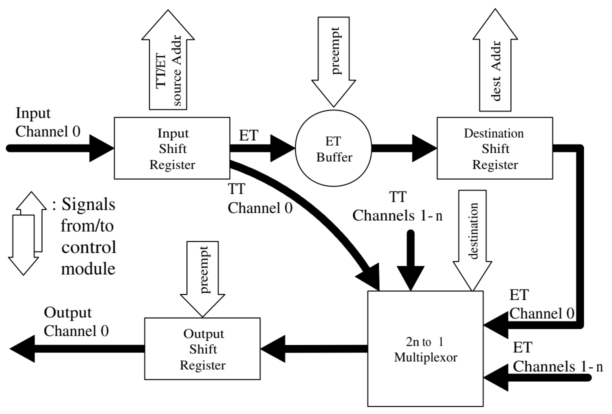

**Figure 3. TTE Switch single channel data-flow**

now a partially stored TT packet is transmitted twice due to an error, it will be discarded by the receiver because either the minimum packet length is violated or a frame checksum mismatch is detected.

In the case when a ET packet leaves the shift register, the data is written into the *ET-Buffer*. This is an enhanced ring buffer with three modes of operation:

- Normal read/write operation.
- ET packet preemption or ET packet collision — These cases are handled in the same way by the buffer: the data extraction out of the buffer is stopped and the read position is reset to the beginning of the actual frame. After the preemption the whole frame is read from the beginning.
- Buffer overflow — In this case the incomplete frame is discarded.

After the data leaves the buffer, another shift register is used to extract the destination address. With this information the control module is able to set the *multiplexors* so that the packet is directed to the respective *output shift registers*. Every multiplexor is connected to all *destination shift registers* where the ET packets arrive from and additionally to every input shift register so that a TT packet can be immediately connected to any output shift register bypassing the ET-Buffers. These shift registers are capable of generating the mandatory inter-packet-gap (Figure 2) if a ET packet is preempted because of a TT packet.

With preemption of a ET packet, the ET-Buffers are set to preemption mode and the output shift registers are cleared. The transmission of data is stopped and the Ethernet PHY transmits an end-of-packet symbol. After the TT data is shifted through the output shift registers, and therefore the inter-packet-gap has elapsed, the TT packet is sent. By preempting a packet in this way no error condition is generated on the network — the receiver discards the preempted packet because it is either to short or there is a mismatch at the frame checksum comparison.

The address management and the switching computations are made in the control module. As these computations have to be done in a single clock cycle, comparators are used to find the correct output ports for every packet.

## 5. Experiments

In order to demonstrate the operation of the TTE Switch we have performed experiments with different configurations of a TT Ethernet cluster. We used a cluster of 4 nodes. Three nodes with TTE Controllers (denoted as *tte1*, *tte2* and *tte3*) transmitted TT messages and one node with a standard Ethernet controller (denoted as *et1*) transmitted ET messages in broadcast mode. All nodes are operating at 100 Mbit/sec network speed.

The TTE Controller was implemented on single-board computers “net4801” manufactured by Soekris Engineering [14]. The TT Ethernet protocol functionality is implemented in software and uses the Linux operating system (kernel version $2.4$) with the *real-time application interface* (RTAI) version $3.1$ [12].

The TT Ethernet nodes maintain the synchronization based on the synchronization frames sent by the primary rate-master node (*tte1*). TT Ethernet frames are sent periodically based on the configuration file stored at each node, which contains the message sending instants and message arrival instants of all TT Ethernet messages. The difference of the expected message arrival time and the actual arrival time presents the difference of the local view of the global time between the sender and the receiver node. A receiver node expects a TT message within a time interval $[t_{arrival} - \Pi + prop_j^i, t_{arrival} + \Pi + prop_j^i]$, where $\Pi$ is the precision of the global time, and $prop_j^i$ is the propagation delay of a frame from sender $i$ to receiver $j$. If a frame is received outside this receive-time window it is discarded and classified as a *lost frame*. Note that the frame propagation delay consists of:

- The delay caused by the switch,
- The time needed to transmit a signal over a communication wire, and
- The time delay caused by processing in the hardware and software of the TTE Controller.

The experiments were executed with 5 different configurations, using TT message periods from $7.8\,ms$ up to $2\,s$.

The aim of these experiments was to show that the transmission delay of TT messages caused by the TTE Switch is constant, regardless of the load of the ET traffic on the network. The transmission delay caused by the prototype implementation of a TTE switch with 8 ports is $4.6\,\mu s$.

The switch transmission delay is validated by measuring the time difference values between the expected frame arrival time and the actual frame arrival time of TT messages.

The jitter caused by the TTE Controller is in the range of $80\,\mu s$. This jitter is due to the clock drift of the TTE Controller timer unit and the unpredictable program execution times, caused by the hardware architecture. With the current software implementation of the TTE Controller we get a global time with the precision of $120\,\mu s$. We are working on a hardware implementation, which shall reach a precision up to a single microsecond.

| No. | Period | ET | $\Delta$ | Lost |
|---|---|---|---|---|
| 1 | $7.8\,ms$ | no | $64\,\mu s$ | $0\,\%$ |
| 2 | $7.8\,ms$ | yes | $77\,\mu s$ | $0\,\%$ |
| 3 | $31.2\,ms$ | no | $53\,\mu s$ | $0\,\%$ |
| 4 | $31.2\,ms$ | yes | $54\,\mu s$ | $0\,\%$ |
| 5 | $125\,ms$ | no | $69\,\mu s$ | $0\,\%$ |
| 6 | $125\,ms$ | yes | $79\,\mu s$ | $0\,\%$ |
| 7 | $500\,ms$ | no | $46\,\mu s$ | $0\,\%$ |
| 8 | $500\,ms$ | yes | $52\,\mu s$ | $0\,\%$ |
| 9 | $2000\,ms$ | no | $47\,\mu s$ | $0\,\%$ |
| 10 | $2000\,ms$ | yes | $80\,\mu s$ | $0\,\%$ |

**Table 1. TTE Switch validation results**

The experimental results are based on the observation of the measured time differences and the number of lost TT messages. For each experimental run (*No. 1 - 20* in Table 1 and 2), 33000 TT messages have been transmitted. The $\Delta$ column presents the maximum measured time difference between the expected and the actual frame arrival time, the percentage of lost messages (i.e, messages received outside the precision window) are shown in the *Lost* column. The *ET* column indicates if the node *et1* was transmitting or not.

Only TT messages were transmitted in the first experimental campaign (*No. 1,3,5,7,9*). The second campaign (*No. 2,4,6,8,10*) of experiments was performed while the node with the standard Ethernet controller ($et1$) continually transmitted UDP messages of 1500 bytes length in broadcast mode. The time interval between two UDP transmissions was $1\,ms$. The experimental results of the first and second campaign using the TTE Switch are presented in Table 1. The slightly increased $\Delta$ values in the second campaign can be explained by the increased CPU work load in the software TTE Controller caused by the additionally received UDP packets.

In the third campaign (*No. 11,13,15,17,19*), the experiments were performed with an 8 port COTS switch (*Netgear GS608*) and only TT messages were transmitted. In the fourth campaign (*No. 12,14,16,18,20*), experiments were performed with the same setup as in the third campaign but the node *et1* transmitted ET messages. The experimental results are presented in Table 2.

| No. | Period | ET | $\Delta$ | Lost |
|---|---|---|---|---|
| 11 | $7.8\,ms$ | no | $70\,\mu s$ | $0\,\%$ |
| 12 | $7.8\,ms$ | yes | $119\,\mu s$ | $1.92\,\%$ |
| 13 | $31.2\,ms$ | no | $65\,\mu s$ | $0\,\%$ |
| 14 | $31.2\,ms$ | yes | $119\,\mu s$ | $1.98\,\%$ |
| 15 | $125\,ms$ | no | $56\,\mu s$ | $0\,\%$ |
| 16 | $125\,ms$ | yes | $119\,\mu s$ | $1.63\,\%$ |
| 17 | $500\,ms$ | no | $51\,\mu s$ | $0\,\%$ |
| 18 | $500\,ms$ | yes | $119\,\mu s$ | $1.79\,\%$ |
| 19 | $2000\,ms$ | no | $50\,\mu s$ | $0\,\%$ |
| 20 | $2000\,ms$ | yes | $119\,\mu s$ | $1.72\,\%$ |

**Table 2. COTS switch validation results**

Table 2 shows that the measured time difference values have significantly increased and that the number of lost frames is not equal to zero when the standard COTS switch is used. The value of $119\,\mu s$ that we can see in the $\Delta$ is the size of the receive-time window of the software implementation of the TTE Controller. Transmitted ET messages delay the transmission of TT messages at the COTS switch, and therefore TT messages are received outside the receive-time window. This scenario does not occur when the TTE Switch is deployed, as the TTE Switch preempts the transmission of ET messages and transmits the TT messages with a constant delay.

## 6. Discussion

As long as the jitter caused by the switch is much smaller than the jitter caused by the controller ($\varepsilon_{switch} \ll \varepsilon_{controller}$), one can use any switch in a TT Ethernet system. The experimental results have shown that even when a low performance TTE Controller like our software implementation is used, a switch is required with less jitter than that introduced by a COTS switch. Also an "intelligent" COTS switch introduces jitter that does not meet this requirement.

The current prototype of the TTE Controller is implemented *in software* using PC boards with RTAI-Linux and it shows a jitter of $80\,\mu s$. In the case where a global time is required with the precision in the range of several microseconds, a software implementation of the TTE Controller is not a suitable solution, as the precision of the clock synchronization in a distributed real-time system depends on the jitter of the message delivery [10]. We are working on an FPGA implementation of the TTE Controller that shall have a jitter lower than $1\,\mu s$, meaning that the achieved precision is in the range of $1\,\mu s$. In this case, the deterministic transmission of TT messages and the jitter value of the used switch become even more important.

The mechanism of the TTE Switch that preempts the transmission of ET messages upon the reception of TT messages enables the transmission of TT messages with a constant delay resulting in a bounded jitter. The jitter caused by the current implementation of the TTE Switch is $70\,ns$ plus the jitter caused by 2 Ethernet PHYs, which is $100\,ns$ for each (one for reception, and one for transmission).

Most of the related works in enabling predictable message transport via Ethernet are based on time-triggered message transfers on a fixed schedule. Some protocols, like FFT-Ethernet [11], can integrate ET with TT traffic, but ET messages are exchanged only in statically allocated time slots. Using a standard Ethernet controller in such systems is not possible without changes in the software of the host computer. TT Ethernet can seamlessly integrate time-triggered traffic and standard Ethernet traffic into the same communication network. The TTE Switch mechanisms enable this integration without having to change the networking protocols in a system using a standard Ethernet controller, as standard Ethernet messages are handled according the IEEE 802.3 standard [5].

## 7. Conclusion

This paper presents the design of a Time-Triggered Ethernet Switch, which is one of the core units of the Time-Triggered Ethernet system.

In TT Ethernet time-triggered traffic can coexist with standard Ethernet traffic in the same communication network. The TTE Switch mechanisms enable this integration without having to change the networking protocols used for exchange of standard Ethernet messages, because these messages are handled according the IEEE 802.3 standard [5]. Scheduled time-triggered messages preempt any standard Ethernet traffic in their transmission path, so standard Ethernet traffic has no temporal effects on jitter and transmission delay of the time-triggered messages. Preempted messages are retransmitted when the medium is free of time-triggered traffic.

Performed experiments, which compare the usability of a standard COTS switch and the TTE Switch in a TT Ethernet system, have shown that transmission delays and jitter caused by a standard COTS switch are not satisfactory for use in a TT Ethernet system in combination with our software implementation of a TTE Controller. COTS switches that implement a strict priority-based QoS mechanism are theoretically able to provide a bounded jitter defined by the transmission time of a maximum length Ethernet message ($123.1\,\mu s$), which does also not meet the requirements of the used TTE Controller.

Experimental results have further shown that a standard COTS switch and the TTE Switch show similar properties when nodes exchange only TT messages. TT traffic is coordinated by a schedule that ensures conflict free frame transmissions. When the nodes exchange both ET and TT messages, a COTS switch cannot predictably guarantee transmission delays of the TT messages. In the setup where the TTE Switch was used, the transmission delays of TT messages remain the same as in case when no ET traffic is deployed.

The TTE Switch mechanism that preempts transmission of ET messages upon the reception of TT messages enables the transmission of TT messages with a constant delay resulting in a bounded jitter. This enables the establishment of the global time with the precision in the order of $1\,\mu s$.

## References

[1] Information about Real-Time Ethernet in Industry Automation, 2004. http://www.Real-Time-Ethernet.de.

[2] R. Belschner, J. Berwanger, C. Ebner, T. Führer, F. Hartwitsch, and et al. FlexRay Requirements Specification Version 2.0.2. Available at: http://www.flexray-group.com, 2002.

[3] J.-D. Decotignie. Ethernet-Based Real-Time and Industrial Communications. *Proceedings of the IEEE*, 93(6), 2005.

[4] M. Felser. Real-Time Ethernet-Industry Prospective. *Proceedings of the IEEE*, 93(6):1118–1129, June 2005.

[5] IEEE Standard 802.3, 2000 Edition. Carrier Sense Multiple Access with Collision Detect on (CSMA/CD) Access Method and Physical Layer Specifications.

[6] B. Jouga. How Ethernet becomes industrial. *Workshop: The use of Ethernet as a fieldbus*, Sept. 2001.

[7] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997. ISBN 0-7923-9894-7.

[8] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The Time-Triggered Ethernet (TTE) Design. In *8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC)*, Seattle, Washington, May 2005.

[9] S.-K. Kweon and K. G. Shin. Achieving Real-Time Communication over Ethernet with Adaptive Traffic Smoothing. In *Sixth IEEE Real-Time Technology and Applications Symposium*, pages 90–100, May 2000.

[10] J. Lundelius and N. Lynch. An Upper and Lower Bound for Clock Synchronization. *Information and Control*, 62, 1984.

[11] P. G. P. Pedreiras, L. Almeida. The FTT-Ethernet Protocol: Merging Flexibility, Timeliness and Efficiency. In *14th Euromicro Conference on Real-Time Systems*, June 2002.

[12] Real-Time Application Interface, 2004. http://www.rtai.org.

[13] SAEJ1850. *Controller Area Network CAN, an In Vehicle Serial Communication Protocol*, pages 20341–20355. 1992-SAE Handbook, 1990.

[14] Soekris. Soekris Engeneering, the *net4801 Embedded Computer Board*, 2005. http://www.soekris.com.

[15] TTTech. Time-Triggered Protocol, High Level Specification Document. Vienna, Austia, D-032-S-10-28 Available at http://www.tttech.com, 2002.

[16] C. Venkatramani and T. Chiueh. Supporting Real-Time Traffic on Ethernet. In *Real-Time Systems Symposium (RTSS)*, pages 282–286, Dec. 1994.