



## Halmstad University Post-Print

# Towards reliable wireless industrial communication with real-time guarantees

Magnus Jonsson and Kristina Kunert

*N.B.: When citing this work, cite the original article.*

©2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Jonsson M, Kunert K. Towards reliable wireless industrial communication with real-time guarantees. Piscataway, NJ: Institute of Electrical and Electronics Engineers; IEEE transactions on industrial informatics. 2009;5(4):429-442.

DOI: [10.1109/TII.2009.2031921](https://doi.org/10.1109/TII.2009.2031921)

Copyright: IEEE

Post-Print available at: Halmstad University DiVA  
<http://urn.kb.se/resolve?urn=urn:nbn:se:hh:diva-74>

# Towards Reliable Wireless Industrial Communication With Real-Time Guarantees

Magnus Jonsson, *Senior Member, IEEE*, and Kristina Kunert, *Student Member, IEEE*

**Abstract**—Increased mobility coupled with a possible reduction of cabling costs and deployment time makes wireless communication an attractive alternative for the automation industry and related application areas. Methods compensating for the high probability of bit errors accompanying wireless transmissions are, however, needed. This is predominantly important in industrial applications with strict reliability and timing requirements, which cannot be met by standard communication protocols as, e.g., TCP. In this paper, a way of combining retransmissions with real-time worst-case scheduling analysis is presented that can offer both a high grade of reliability and hard real-time support. The presented solution handles one or several retransmission attempts of erroneous data without jeopardizing already guaranteed delay bounds of other packets. A real-time analysis for a full-duplex, asymmetric link, utilizing the novel retransmission scheme and supporting both piggybacked and nonpiggybacked acknowledgments, is provided. A simulation study is presented that evaluates the performance of the retransmission scheme for bit-error rates typically experienced in wireless communication. The results clearly indicate a possible reduction of the message error rate by several orders of magnitude.

**Index Terms**—Communication system reliability, data communication, industrial control, protocols, real-time systems.

## I. INTRODUCTION

INDUSTRIAL communication systems have traditionally utilized wired fieldbus systems to meet the demands of the applications they service. The reason for this is that industrial systems are characterized by their strict demands on both reliability and timing; requirements which can be met by the well-developed technology of fieldbuses [1]. However, the advantages of wireless communication solutions have lately become interesting also for industrial networks. The need for reduced cabling, which leads to both the possibility of faster setup times for equipment and the possibility of communication in areas too harsh for using cables, and the added mobility have triggered research on the use of wireless communication in industrial systems like production systems including robots [2], control loops [3]–[5], and other automation applications [6]. Even while still using existing fieldbus standards, wireless alternatives have been studied as a substitute for cables [7]–[9].

Even if wireless technology adds a lot of benefits to the context of industrial communication, it also suffers from a number of disadvantages. Wireless communication is characterized by its high error probability, leading to the risk of causing severe

problems for applications with strict reliability and timing requirements. A typical need of those applications is a bounded worst-case end-to-end delay or a low packet loss rate [10]. In order to fulfil those needs, conclude the authors in [11], industrial systems need both feed-forward error correction and automatic repeat request (ARQ). The presented research focuses on the question of how to use retransmissions in combination with real-time worst-case scheduling analysis. Although this paper does not explicitly touch upon the topic of feed-forward error correction, it is assumed to be used at a lower level in case it is needed. Its mechanism does not influence the design or functionality of the presented solution. Obviously, real-time adaptation can be, and has to be, implemented at each level. However, this work, as for this moment, has excluded the adaptation towards, e.g., specific medium access control (MAC) protocols or underlying wireless technologies. This solution is targeting an end-to-end delay-bound and is, therefore, located at the transport level.

This paper proposes a framework, containing both retransmission scheme and timing analysis, in order to give hard real-time communication support to an application, meaning that deadline misses are acceptable neither for the first instance of a packet nor for any retransmitted instances. The framework implements this by supporting the retransmission of erroneous packets only as long as their deadline is not reached (i.e., as long as the retransmitted packet will arrive in time), and at the same time the packet is only retransmitted if the extra network capacity it will consume does not jeopardize already granted real-time guarantees of any existing traffic flow in the network. The retransmission of any packet is triggered when, at a certain timeout value, no acknowledgment (ACK) has arrived at the sender, which might have different reasons. The packet for which the ACK is expected could either have arrived faulty at the receiver (this might be determined by, e.g., a checksum calculation), or it could have been lost completely on the way.

Using wireless communication in industrial settings comes with a quite high error rate compared to the one in copper wires or fibers. The use of feed-forward error correction is only helpful in the case of bit errors. When loosing complete packets, e.g., due to synchronization errors, only retransmitting the packet will help. Common retransmission protocols can have the problem of not being aware of the deadline, leading to countless retransmission attempts until the packet has been acknowledged successfully. However, the packet might already have missed its deadline, and is jeopardizing other packets' deadlines by (possibly unnecessarily) using network capacity. The presented retransmission scheme is a truncated ARQ scheme, limiting the number of possible retransmissions. Furthermore, the addition of the real-time scheduling analysis will prohibit the retransmission of a packet in case its retransmission will lead to another packet's deadline miss.

Manuscript received March 03, 2009; revised June 18, 2009 and June 18, 2009. First published September 29, 2009; current version published November 06, 2009. Paper no. TII-09-03-0035.R2.

The authors are with the Centre for Research on Embedded Systems (CERES), Halmstad University, Halmstad 30118, Sweden (e-mail: Magnus.Jonsson@hh.se; Kristina.Kunert@hh.se).

Digital Object Identifier 10.1109/TII.2009.2031921

Even when utilizing the retransmission scheme presented in this paper, the message error rate (MER) will never be zero, not even when using high-quality electrical wires and optical fibers. However, the proposed solution can reduce the MER substantially, which is of particular importance in wireless communication with its inherent high bit-error rates (BERs), especially when targeting real-time traffic as, e.g., in industrial applications. The solution is not aiming towards safety-critical industrial applications, where the missing of a deadline or the loss of a packet can have disastrous consequences, but the targeted hard real-time applications are still sensitive towards packet loss (as compared to, e.g., multimedia applications with hard deadlines, but low sensitivity regarding packet loss) [10]. The reaction following a packet loss is application dependent and normally taken care of by layers above the one implementing the presented solution. As an example one could mention applications relying on periodic transmissions of sensor data, where the application has to be prepared to, e.g., at rare occasions await the next transmission. The tradeoff between the advantages of wireless solutions and the increased probability of packet loss or a deadline miss is still debated in the literature [12].

ARQ for quality-of-service (QoS) is a well-studied area, so is guaranteed real-time communication. However, the approach of taking a holistic view on these areas together and targeting wireless communication, with the aim of being able to calculate guaranteed performance bounds, has been studied less thoroughly. Aiming for hard real-time communication in embedded and distributed wireless communication systems, the proposed approach uses traffic models and develops analysis methods originating from the area of real-time systems. This framework does not settle for a long-term statistical delay bound but wants to give a delay bound guarantee for every single packet by using worst-case delay analysis. This is necessary to support short delay bounds for periodic traffic in industrial real-time systems. Therefore, retransmissions of a packet are not allowed if they jeopardize the stated real-time guarantees of other packets, which would be possible in standard ARQ schemes.

Former research on ARQ for real-time communication includes work where only average performance, like average delay, is considered [13]. A promising technique to incorporate error correcting codes and ARQ in real-time communication is deadline dependent coding [14], [15]. However, no related real-time scheduling analysis framework has been developed.

In [16], Butt decreases the MER of hard real-time traffic by retransmitting erroneous packets as long as they still meet their hard deadline, a concept similar to ours. However, the solution is build upon an idle RQ approach and is only described on the packet level. Also, no queueing or scheduling analysis is provided. In contrast to idle RQ, which usually leads to a higher penalty in link utilization, the proposed retransmission scheme is more similar to a continuous RQ protocol. Additionally, the presented delay bound analysis, working on a real-time channel level and including queueing delay analysis, enables the provision of end-to-end delay guarantees.

Giancola *et al.* aim at providing real-time guarantees including retransmissions [17]. However, their methods use flow analysis, which has been shown to not fully exploit the available network capacity compared to real-time scheduling analysis [18]. Additionally, their analysis does not include the

timing details needed to support hard real-time communication in embedded and industrial real-time systems, while the here presented solution is targeting an as exact analysis as possible. Moreover, the solution proposed in [17] requires traffic regulators to be implemented in each node on the path to control the traffic. In this proposal, a more general framework is drawn up to combine ARQ with real-time scheduling analysis.

In this paper, partly based on earlier conference papers [19], [20], a complete framework for a full-duplex, asymmetric link is presented. The framework supports a certain number of retransmissions when needed, including several retransmission attempts, to support wireless industrial real-time communication. Based on earlier results, both timing analysis and real-time scheduling analysis have been developed for the new framework, still based on the earliest deadline first (EDF) scheduling policy [21]. Compared to [19] and [20], this paper no longer assumes a dedicated physical channel for ACK, and the asymmetry of the wireless medium has been taken into consideration, no longer assuming equal bit rates in each direction, which has led to incremental, but highly significant changes in the framework. Additionally, an analysis of both piggybacked and non-piggybacked ACKs is provided. A new simulation study was added, treating a new set of cases.

The rest of this paper is organized as follows. An overview of the framework is given in Section II. The protocol together with the timing analysis is presented in Section III. In Section IV, the real-time scheduling analysis assuming piggyback ACK is described, while in Section V this analysis is adapted for the case without the possibility of piggybacking ACKs. Section VI presents simulation results and this paper is concluded in Section VII.

## II. FRAMEWORK OVERVIEW

In order to increase the reliability for wireless industrial networks, a retransmission scheme is introduced that respects existing delay bounds of ordinary transmissions. This has been realized by defining a basic transport protocol which contains a real-time scheduling analysis comprising both ordinary transmissions and retransmissions. For the development of the framework, a network for which a real-time analysis method exists for the case of error-free communication (see, e.g., [22]) is assumed. All parameters used in the paper are summarized in Table I.

The traffic in the network is specified in the form of traffic flows (also called logical real-time channels, RT channels) denoted  $\tau_i$  with  $1 \leq i \leq Q$ , where  $Q$  defines the number of RT channels. All traffic flows are defined by the following parameters: a sending node  $m_{s,i}$ , a receiving node  $m_{d,i}$ , a period  $P_i$ , the message length  $L_i$  (excluding header bits in this section), and the end-to-end delay bound  $D_i$ . This means that each RT channel  $\tau_i$  can be defined completely by the following expression:  $\tau_i = \{m_{s,i}, m_{d,i}, P_i, L_i, D_i\}$ . Comparable to other end-to-end ARQ schemes, even the one described in this paper is assumed to be placed in the transport layer, and thereby thought to work on top of any MAC method. RT channels requested from the network layer are denoted as  $\tau_{N,i} = \{m_{s,i}, m_{d,i}, P_{N,i}, L_{N,i}, D_{N,i}\}$ , where  $N$  indicates the

TABLE I  
FRAMEWORK AND ANALYSIS PARAMETERS

Parameter	Description
<b>Parameters defining the logical real-time channels</b>	
$\tau_i$	Logical real-time channel $i$
$Q$	Number of RT channels
$m_{s,i}$	Source of logical real-time channel $\tau_i$
$m_{d,i}$	Destination of logical real-time channel $\tau_i$
$P_{N,i}$	Period of network level logical real-time channel $\tau_{N,i}$
$L_{N,i}$	Message length of network level logical real-time channel $\tau_{N,i}$ (excluding header)
$D_{N,i}$	End-to-end delay bound of network level logical real-time channel $\tau_{N,i}$
$\tau_{N,i}$	Network level logical real-time channel $i$
$\tau_{T,i}$	Transport level logical real-time channel $i$
$P_{T,i}$	Period of transport level logical real-time channel $\tau_{T,i}$
$L_{T,i}$	Message length of transport level logical real-time channel $\tau_{T,i}$ (excluding header)
$D_{T,i}$	End-to-end delay bound of transport level logical real-time channel $\tau_{T,i}$
$\tau_{re,i}$	Dedicated logical retransmission RT channel $i$
$m_{re\ S,i}$	Source of logical retransmission real-time channel $\tau_{re,i}$
$m_{re\ D,i}$	Destination of logical retransmission real-time channel $\tau_{re,i}$
$P_{re,i}$	Period of logical retransmission real-time channel $\tau_{re,i}$
$L_{re,i}$	Message length of logical retransmission real-time channel $\tau_{re,i}$
$D_{re,i}$	End-to-end delay bound of logical retransmission real-time channel $\tau_{re,i}$
$P_{ACK}$	Period of the acknowledgement RT channel
$D_{ACK}$	Deadline of the acknowledgement RT channel
<b>Parameters used for timing analysis in the framework</b>	
$D_{ord,i}$	End-to-end delay bound for ordinary transmission of logical real-time channel $\tau_i$
$D_{retr,i}$	End-to-end delay bound for the possible retransmission of one of the packets of logical real-time channel $\tau_i$
$L_{pack}$	Maximum length (in bits) of a packet including header
$L_{data}$	Maximum amount of pure data (in bits) per packet
$L_{header}$	Header length (in bits)
$L_{T,i}$	Amount of pure data per message belonging to logical RT channel $\tau_i$
$N_{pack,i}$	Number of packets per message belonging to logical RT channel $\tau_i$
$N_{pack\ max,i}$	Of $N_{pack,i}$ packets belonging to logical RT channel $\tau_i$ , the first $N_{pack\ max,i}$ packets are maximum sized packets
$L_{last,i}$	Length of the last packet of a message belonging to logical RT channel $\tau_i$
$R_{forward}$	Bit rate of the physical link from sender to receiver
$R_{reverse}$	Bit rate of the physical link from receiver to sender
$T_x\ forward$	Transmission time of a full-length data packet; also: maximum blocking time of one data packet
$T_x\ last\ forward,i$	Transmission time of the last packet belonging to logical RT channel $\tau_i$
$T_x\ tot\ forward,i$	Total transmission time of a message belonging to logical RT channel $\tau_i$
$T_{prop}$	Propagation delay over the physical link
$T_{ACK\ piggy,i}$	Transmission time of a piggybacked ACK packet acknowledging a packet belonging to logical RT channel $\tau_i$
$T_{proc\ 1}$	Processing time between the reception of a packet until its corresponding ACK packet is sent
$T_{proc\ 2}$	Processing time between the timeout without reception of an ACK packet until the retransmission can be initiated
$T_{timeout}$	Timeout instance
$T_{margin}$	Safety margin between the expected ACK reception and the timeout instance
$d_{ord,i}$	Maximum queuing delay of a packet belonging to logical RT channel $\tau_i$
$L_{max,i}$	Maximum length of a packet belonging to logical RT channel $\tau_i$
$T_x\ max\ forward,i$	Maximum transmission time of a maximum length packet belonging to logical RT channel $\tau_i$
$L_{retr,j}$	Maximum packet length among all packets belonging to any RT channel
$T_x\ retr\ forward,i$	Maximum transmission time among all packets belonging to any RT channel
$d_{retr,j}$	Maximum queuing delay, i.e., the deadline of the retransmission
$D_{retr\ last,i}$	Delay bound for last retransmission attempt of a packet belonging to logical RT channel $\tau_i$
$D_{retr\ other,i}$	Delay bound for any retransmission attempt except the last one of a packet belonging to logical RT channel $\tau_i$
$T_{ACK\ nonpiggy}$	Transmission time of nonpiggybacked ACK packet
$L_{ACK}$	Length of nonpiggybacked ACK packet including header
<b>System parameters provided by the user</b>	
$M$	Number of retransmission RT channels
$N_{attempt}$	Number of allowed retransmission attempts in a row per packet
<b>Parameters needed in the schedulability testing</b>	
$U$	Utilization by periodic real-time traffic
$HP$	Hyper period
$BP$	Busy period
$BP_1$	First $BP$ in the first $HP$ of the schedule where all periods start at time zero

network layer, while the corresponding RT channels requested by the application layer or, alternatively, by the system designer, from the transport layer are defined by  $\tau_{T,i} = \{m_{s,i}, m_{d,i}, P_{T,i}, L_{T,i}, D_{T,i}\}$ ,  $T$  indicating the trans-

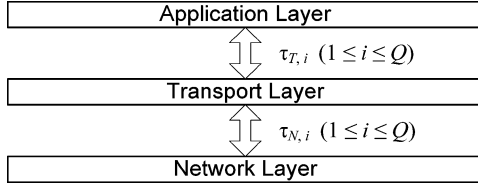


Fig. 1. Layering.

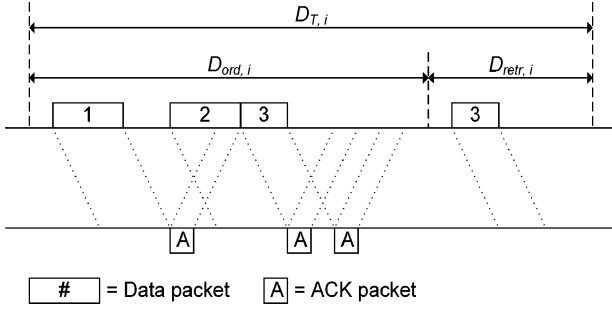


Fig. 2. Example of ARQ time-sequence diagram.

port layer. Throughout the remainder of this paper, a RT channel will be referred to as  $\tau_i$  in case the reasoning is applicable to both  $\tau_{T,i}$  and  $\tau_{N,i}$ . For an illustration of the layering and the real-time channel abstraction between the different layers see Fig. 1. Under the assumption of error free communication, a direct mapping of transport-layer channels onto network-layer channels means any existing analysis for the underlying network can be used to check whether the performance requested by the application can be guaranteed for all traffic flows on the network level. However, this direct mapping also infers that any necessary retransmissions are not taken care of by the transport protocol. In the following, the novel retransmission request (ARQ) scheme and a method of combining a timing analysis of this scheme with a real-time scheduling analysis will be described.

For every traffic flow, the delay bound of the transport layer,  $D_{T,i}$ , is divided into two separate deadlines (see Fig. 2), one shortened delay bound ( $D_{Ord,i}$ ) for the ordinary transmission of all packets belonging to one message, e.g., three packets as in the example shown in Fig. 2, and one delay bound ( $D_{retr,i}$ ) that renders possible the retransmission of a certain number of these three packets, and therefore it can be formally defined as

$$D_{T,i} = D_{Ord,i} + D_{retr,i}. \quad (1)$$

When mapping transport-layer channels onto network-layer channels, all parameters of  $\tau_{N,i}$  can be set to the corresponding values of  $\tau_{T,i}$  with the exception of the delay bound, where instead we have

$$D_{N,i} = D_{Ord,i}. \quad (2)$$

Targeting an improvement of the experienced MER in the network, retransmission channels are added which, when needed, can be used by any real-time channel. Those retransmission channels are also defined as traffic flows:  $\tau_{re,i} = \{m_{re-S,i}, m_{re-D,i}, P_{re,i}, L_{re,i}, D_{re,i}\}$ . The delay bound of the retransmission channel,  $D_{re,i}$ , decides upon the length of time allocated for a possible retransmission. The

period of the retransmission channels,  $P_{re,i}$ , is a predefined system parameter, decided upon by the system designer, by the help of which one can control the minimum interval of time between two retransmission requests (on that retransmission channel). In other words,  $P_{re,i}$  indicates the number of times one wants to be able to retransmit any electable packet of any arbitrary message. When using a, by the system designer predefined, number of  $M$  retransmission channels, with each of those supporting the retransmission of one packet every interval of  $P_{re,i}$ , up to  $M$  consecutive retransmission packets can be sent over the physical link, improving the retransmission support considerably. However, to reach this maximum of  $M$  packets, none of the  $M$  retransmission RT channels is allowed to have sent any retransmission packet during the last period of  $P_{re,i}$ . Each packet has an upper limit,  $N_{attempt}$ , for the number of allowed consecutive retransmission attempts before the end of its deadline. The value of  $N_{attempt}$  is chosen as a system parameter. A higher value of  $N_{attempt}$  will normally increase the probability for the correct transmission of a message, but will also put higher demands on the network or link capacity. The reason for that is that with a higher value of  $N_{attempt}$ , the deadline for each retransmission attempt must be shorter in order to manage to implement all attempts before the deadline of the message has expired. In order to be able to support  $N_{attempt}$ , a minimum of  $M = N_{attempt}$  retransmission RT channels have to be present over the corresponding physical link. However, this holds only true if the criteria regarding  $P_{re,i}$  is fulfilled for all retransmission RT channels, i.e., the demand that no retransmission was sent on any of the  $M$  retransmission RT channels during the previous period of  $P_{re,i}$ . By the usage of dedicated retransmission channels, resource allocation with real-time scheduling analysis can be done to guarantee both a certain amount of timely retransmissions per time unit and the delay bounds of ordinary transmissions.

The framework described above has up to now been used in a point-to-point link scenario with EDF scheduling. The network is exemplified by a full-duplex communication channel where each communication direction works at an individual bit rate due to the asymmetric properties characterizing wireless communication networks. The general concept, however, constitutes a generic framework that can be adapted to wireless networks with a deterministic MAC protocol (such as a time-division multiple access (TDMA)-based protocols [23]–[25]). Such adaptations are out of the scope of this paper and are planned future work. The details of this scheme are elaborated upon in more detail in the following.

### III. PROTOCOL DEFINITION AND TIMING ANALYSIS

The ARQ protocol defined is rather straightforward as it is assumed that both flow control and congestion control can be excluded from the specification. When agreeing to receive traffic at a certain packet rate [22], [26], as is the case in this design with RT channels, flow control is redundant, and as long as the traffic is complying with the real-time scheduling analysis (explained later in the paper), it is guaranteed that no congestion will occur. Although assuming a bidirectional channel, the analysis is implemented in one direction at a time, taking into consideration the asymmetric characteristic of the wireless channel. During the analysis, the bidirectional channel is assumed to carry data

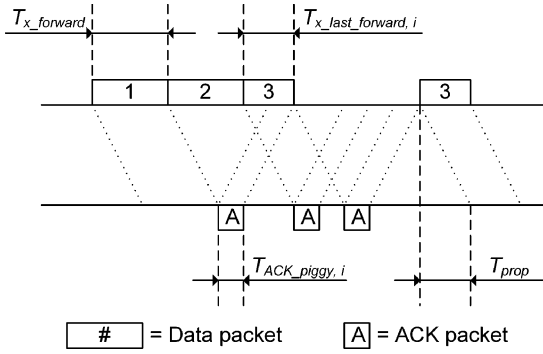


Fig. 3. Timing of the transmission of one message.

packets in one direction and ACK packets in the opposite direction. In order to decrease overhead, the ACK is piggybacked in a data packet in case there is a data packet queued in the node that wants to send the ACK. For circumstances when ACKs cannot be piggybacked, an alternative analysis is presented in Section V.

Messages sent over the link are divided into packets, where the maximum length  $L_{pack}$  of each packet, calculated in bits and including the header, is given by

$$L_{pack} = L_{data} + L_{header} \quad (3)$$

where the maximum amount of pure data per packet is denoted by  $L_{data}$ , and  $L_{header}$  specifies the length of the header. Given this, and the parameter  $L_{T,i}$  as the amount of pure data per message belonging to RT channel  $\tau_i$ , the number of packets per message can be calculated as

$$N_{pack,i} = \left\lceil \frac{L_{T,i}}{L_{data}} \right\rceil. \quad (4)$$

All packets except the last one in a message are assumed to be maximum sized, and the number of these maximum sized packets, denoted  $N_{pack\_max,i}$ , is given by

$$N_{pack\_max,i} = \left\lfloor \frac{L_{T,i}}{L_{data}} \right\rfloor. \quad (5)$$

The length of the last packet, including pure data and header bits, of a message is defined as  $L_{last,i}$  for the case of  $L_{last,i} < L_{pack}$ . Otherwise,  $L_{last,i}$  will be zero.  $L_{last,i}$  is calculated as

$$L_{last,i} = (N_{pack,i} - N_{pack\_max,i}) \cdot (L_{T,i} - N_{pack\_max,i} \cdot L_{data} + L_{header}). \quad (6)$$

For calculations on the timing of message transmissions (for a graphical illustration of this see Fig. 3), a parameter for the bit rate of the physical link has to be introduced. As the analysis assumes an asymmetric link between the nodes, the bit rate  $R_{forward}$ , experienced by the data packet from the sending node to the receiving node, and the bit rate  $R_{reverse}$  in the opposite direction, experienced by the ACK, are presupposed to be different. The transmission time of a maximum sized packet will therefore be

$$T_{x\_forward} = \frac{L_{pack}}{R_{forward}}. \quad (7)$$

In case of the last packet of a message not being of full length, the transmission time is instead given by

$$T_{x\_last\_forward,i} = \frac{L_{last,i}}{R_{forward}}. \quad (8)$$

Consequently, the total transmission time of one message is

$$T_{x\_tot\_forward,i} = \frac{N_{pack\_max,i} \cdot L_{pack} + L_{last,i}}{R_{forward}}. \quad (9)$$

Besides the propagation time,  $T_{prop}$ , over the physical link, Fig. 3 also shows the transmission time of the ACK packet belonging to RT channel  $\tau_i$  and denoted  $T_{ACK\_piggy,i}$ , where  $1 \leq i \leq Q$ . However, in this paper, the transmission time for all ACKs is assumed to be independent of the packet they are acknowledging and equal to  $T_{ACK\_piggy}$ . The transmission time for any piggybacked ACK is calculated as:

$$T_{ACK\_piggy} = \frac{L_{pack}}{R_{reverse}} \quad (10)$$

where  $L_{pack}$  denotes the length of the maximum-sized packet onto which the ACK is piggybacked.

Fig. 4 shows further details regarding the timing of the original transmission of one message including ACK. The time interval or processing time between the reception of a data packet and the sending of the corresponding ACK is called  $T_{proc,1}$ . Additional processing time, denoted  $T_{proc,2}$ , is needed between the timeout instance, triggered when not receiving an ACK packet, and the time when the retransmission of the corresponding data packet can be initiated. The usage of negative ACKs would be possible, but is not treated in this paper as this has no impact on the worst-case timing analysis.

The timeout instance is assumed to be identical for all packets belonging to the same message. This makes it possible, at the timeout instance, to check whether a sufficient amount of resources is available for the retransmission of all erroneous packets of one message. In case resources are found to be insufficient, none of the retransmissions will be sent in order to save bandwidth. It would be pointless to retransmit only some of the erroneous packets of a message, as the message still would be incomplete even after the retransmissions. Due to the timeout instance being identical for all packets of a message, it can be defined relative to the deadline of the ordinary message transmission

$$T_{timeout} = D_{ord,i} - T_{proc,2}. \quad (11)$$

As the delay bound  $D_{ord,i}$  is given relative to the generation instance of message  $i$ ,  $T_{timeout}$  is relative to this instance as well.

Finally,  $T_{margin}$ , a safety margin between the expected time for ACK reception and the timeout instance is introduced to cope with, e.g., uncertainties in the clock synchronization. In the following section, Section IV, any queueing delay introduced by the interference with the transmission of other messages will be analyzed.

In a runtime implementation of the retransmission protocol (packet handling, etc.) described above it is further assumed that the sending node has information about parameters as, e.g., the

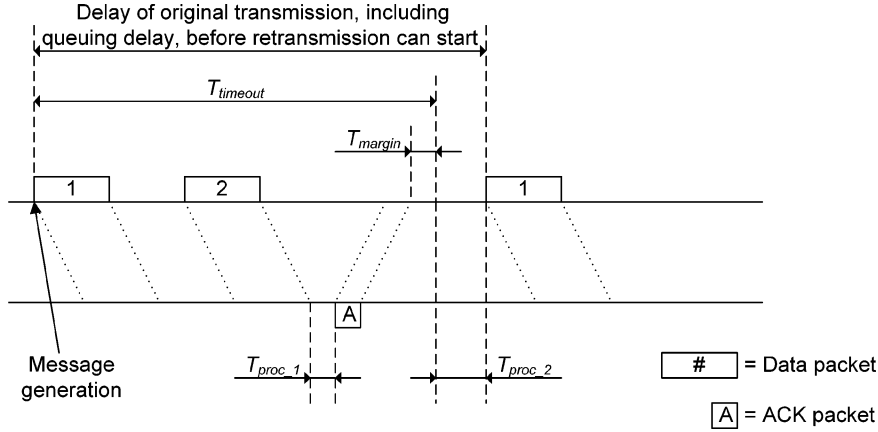


Fig. 4. Timing of the transmission of one message, including retransmission.

RT channel identification, the deadline of the individual packets, and also their transport-layer sequence numbers. Those are examples of parameters which are important for the accurate implementation of the retransmission scheme. If taking as an example the usage of sequence numbers in combination with the RT channel identification, then each RT channel can have its own set of sequence numbers, making it possible for the receiving side to reorder packets after arrival and for the sending side to identify which packet(s) need(s) to be retransmitted. The deadline for each packet, used when sorting packets according to EDF, is the absolute deadline obtained by adding the relative deadline to the arrival time. The relative deadline is the maximum queueing delay as derived in the next section.

#### IV. REAL-TIME SCHEDULING ANALYSIS

For the real-time scheduling analysis to work, the maximum delay introduced by queueing has to be isolated by subtracting all other delays before the analysis can be used to check whether all derived maximum queueing delays can be guaranteed. Additionally, all possible retransmissions have to be incorporated into the analysis, as previous versions of this analysis do not take retransmissions into consideration. The queueing discipline assumed for the packets is EDF, i.e., the packets are queued according to their relative deadline, where shorter deadlines are assigned higher priorities. Deadlines are not restricted to be equal to the periods, but arbitrary deadlines, i.e., shorter than, equal to or longer than the periods, are supported. The choice of EDF was guided by its well-documented capabilities to schedule successfully when having real-time requirements in terms of deadlines (or delay bounds). Additionally, EDF is one of the algorithms for which a well-proven analysis framework has been developed. Being a dynamic scheduling discipline, the priorities, determined by the deadlines, are updated dynamically as time proceeds. One example of the queueing delay under the assumption of EDF queueing is pictured in Fig. 5. As the real-time scheduling analysis only considers delay caused by queueing, including the total transmission time  $T_{x\_tot\_forward,i}$  of all packets of a message, all other delays need to be excluded before further analysis. The maximum queueing delay  $d_{ord,i}$  can, therefore, be isolated by subtracting all other delays from the relative

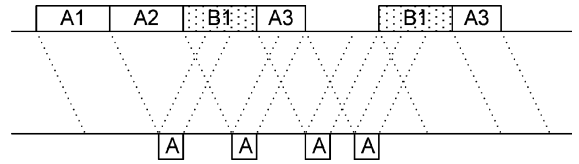


Fig. 5. Example of queueing delay. Message B arrives to the queue after Message A, but with an earlier deadline, thereby delaying packets of Message A.

delay bound of the ordinary transmission, resulting in the following maximum queueing delay (and new deadline of the ordinary transmission)

$$d_{ord,i} = D_{ord,i} - 2 \cdot T_{prop} - T_{proc1} - T_{proc2} - T_{margin} - T_{x\_forward} - 2 \cdot T_{ACK\_piggy} \quad (12)$$

where  $T_{x\_forward}$  is included for the maximum blocking time of one packet caused by the nonpreemptive transmission of a packet with a longer deadline. The two instances of  $T_{ACK\_piggy}$  are deducted from the ordinary delay bound as they correspond, first, to the worst-case waiting time of the ACK for a packet onto which it can be piggybacked, and, second, the actual transmission time of this packet containing the piggybacked ACK. In case the piggyback alternative is not possible due to the absence of another data packet, the usage of explicit ACK packets will not result in longer delays, as they will be able to be sent immediately.

As the retransmission guarantee is designed to extend to packets belonging to any of the RT channels, the maximum sized packet amongst all packets on a RT channel  $\tau_i$  is defined as

$$L_{max,i} = \begin{cases} L_{pack}, & N_{pack\_max,i} \geq 1 \\ L_{last,i}, & N_{pack\_max,i} = 0 \end{cases} \quad (13)$$

with the corresponding maximum transmission time

$$T_{x\_max\_forward,i} = \frac{L_{max,i}}{R_{forward}}. \quad (14)$$

The longest packet when comparing all RT channels can, therefore, be defined as

$$L_{retr,j} \geq \max_{i=1}^Q (L_{max,i}) \quad (15)$$

with its corresponding maximum transmission time taking all RT channels into consideration

$$T_{x\_retr\_forward,i} = \frac{L_{retr,i}}{R_{forward}}. \quad (16)$$

This means that  $T_{x\_retr\_forward,i}$  always implies the message length  $L_{pack}$ , if at least one RT channel has a message with at least one maximum-sized packet.

For all retransmission RT channels, their delay bound  $D_{re,i}$  is set to the common value  $D_{re}$ , a system parameter controlling the time span allocated for possible retransmissions. Therefore, we set

$$D_{retr,i} = D_{re} \quad 1 \leq i \leq Q. \quad (17)$$

In order to derive the delay bound for ordinary transmissions, (1) is rephrased as

$$D_{ord,i} = D_{T,i} - D_{retr,i}. \quad (18)$$

As for the ordinary transmission, the new deadline of each retransmission, i.e., the isolated maximum queueing delay  $d_{retr,i}$  can be calculated by subtracting all other delays from the delay bound (further elaborated on at a later point in this paper)

$$\begin{aligned} d_{retr,i} &= \frac{D_{re} - T_{prop} - T_{x\_forward} - (N_{attempt} - 1) \cdot T_{retr\_const}}{N_{attempt}} \end{aligned} \quad (19)$$

where

$$\begin{aligned} T_{retr\_const} &= 2 \cdot T_{prop} + T_{proc1} + T_{proc2} + T_{margin} \\ &\quad + T_{x\_forward} + 2 \cdot T_{ACK\_piggy}. \end{aligned} \quad (20)$$

The last possible retransmission has no ACK answer which results in fewer delay components. In (20),  $T_{x\_forward}$  is included as the maximum blocking time introduced by a packet with a longer deadline (thereby actually having a lower priority), as its nonpreemptive transmission already might have been initiated. Additionally,  $T_{ACK\_piggy}$  is deducted twice for each, but the last, retransmission attempt. One instance corresponds to the maximum amount of time an ACK might have to wait for a packet onto which it can be piggybacked, while the second instance corresponds to the actual transmission time of that packet (including the piggybacked ACK). The actual delay bound for the last retransmission attempt is

$$D_{retr\_last,i} = d_{retr,i} + T_{prop} + T_{x\_forward} \quad (21)$$

while the delay bound for each other retransmission attempt is calculated by

$$\begin{aligned} D_{retr\_other,i} &= d_{retr,i} + 2 \cdot T_{prop} + T_{proc1} + T_{proc2} \\ &\quad + T_{margin} + T_{x\_forward} + 2 \cdot T_{ACK\_piggy}. \end{aligned} \quad (22)$$

The reason for the calculation of these delay bounds is their necessity in the later runtime implementation part of the proposed method, where they are needed for the calculation of the timeout values.

Real-time scheduling analysis is used to check if any traffic allocation over the network, or in this example the point-to-point link, is feasible, i.e., it checks if all messages belonging to the scheduled traffic flows, including their retransmissions, will keep their deadlines. The first condition to be checked is link utilization. It is a necessary, but not sufficient (as the deadlines of any real-time channel are not restricted to be equal to the periods of that channel), condition that the allocated traffic over the network (link) does not exceed 100%. This traffic includes both the ordinary transmissions and the retransmissions. The utilization  $U$  of a link by periodic real-time traffic can be calculated as follows

$$U = \sum_{i=1}^Q \left( \frac{T_{x\_tot\_forward,i}}{P_{T,i}} \right) + \sum_{i=1}^M \left( \frac{T_{x\_retr\_forward,i}}{P_{re,i}} \right). \quad (23)$$

A second constraint has to be fulfilled in order to ensure the feasibility of the traffic allocation over the link. For the description of this second part of the feasibility check, a number of concepts, originating from the area of real-time scheduling, have to be introduced. Firstly, a hyper period,  $HP$ , is the least common multiple of all periods of the RT channels, i.e., the interval defined by the starting point when the periods of all traffic flows start at the same time to the finishing point when they again start at the same time. The next key word necessary for the analysis is the principle of a busy period,  $BP$ , which is any interval within  $HP$  when the link is busy. Lastly, the so called workload function,  $h(t)$ , has to be introduced. The workload function measures the traffic demand over the point-to-point link. Originally, this function was designed for the control of the processor demand in a real-time system, but due to the assumption of EDF scheduling, this analysis can be adapted for analyzing traffic allocations [27], and used for a network in a similar manner. Generally speaking,  $h(t)$ , is the sum of the transmission times for all message instances of all RT channels which have an absolute deadline that is less than or equal to a point in time  $t$ . This point in time  $t$  signifies the number of time units elapsed since the beginning of  $HP$ , the reason being that the synchronous traffic pattern where all RT channels' periods start at the same time is the worst-case in terms of workload (in this case leading to the worst-case delay) [28]–[30]. This also includes retransmission RT channels starting their periods at the same time even though not true in practice, since retransmissions always take place later than the ordinary transmissions. This assumption introduces thereby a certain amount of pessimism into the analysis. The workload function is calculated as

$$\begin{aligned} h(t) &= \sum_{\substack{i \in [1,Q], \\ d_{ord,i} \leq t}} \left( 1 + \left\lfloor \frac{t - d_{ord,i}}{P_{T,i}} \right\rfloor \right) \cdot T_{x\_tot\_forward,i} \\ &\quad + \sum_{\substack{i \in [1,M], \\ d_{retr,i} \leq t}} \left( 1 + \left\lfloor \frac{t - d_{retr,i}}{P_{re,i}} \right\rfloor \right) \cdot T_{x\_retr\_forward,i}. \end{aligned} \quad (24)$$

It is worth noticing that the summations only include terms for which  $t$  is equal or greater than the corresponding relative deadline. The workload function is, in its original form, intended for uniprocessor task scheduling and proven to be correct for

preemptive EDF scheduling in [30]. Since all delay constants (including blocking time of a nonpreemptable lower priority packet) already have been subtracted from the deadlines, (24) is used in a corresponding way as in [30], where the studied point-to-point link (one direction) corresponds to a processor. The second constraint, introduced by [28], [29], and generalized by [30], was added in order to be able to insure the continued feasibility of the traffic allocation even in case a new traffic flow is added. The constraint demands that

$$h(t) \leq t \quad \forall t. \quad (25)$$

Unfortunately, this constraint introduces a high amount of computational complexity into the feasibility analysis and therefore does not lend itself to calculation particularly well. However, in [31] a way of reducing the time and memory complexity is presented. It is possible to reduce the number of instances of evaluation to the instances of absolute message deadlines during an interval upper bounded by  $BP_1$ , the first busy period in the first hyper period of the schedule where all periods start at time zero, i.e.,

$$t \in \bigcup_{i=1}^Q \{m \cdot P_{T,i} + d_{ord,i} : m = 0, 1, 2 \dots\} \cup \{n \cdot P_{re,1} + d_{retr,1} : n = 0, 1, 2 \dots\} \quad (26)$$

where

$$t \in [1; BP_1]. \quad (27)$$

Only in the case when both the utilization constraint and the feasibility constraint are fulfilled, a feasible traffic allocation can be guaranteed, i.e., only then will it be possible to guarantee that no deadlines will be missed.

In the runtime implementation of this schedulability test, the utilization and feasibility are checked when a new ordinary RT channel is added, in order to guarantee the requested delay bounds for both the new and existing RT channels (including retransmission RT channels). As new RT channels are not expected to be requested very frequently, the computational demands of this admission control will not be very high. In case all RT channels are known at design stage or system startup, the analysis can be made offline instead.

## V. ANALYSIS WITHOUT THE ASSUMPTION OF PIGGYBACK ACKNOWLEDGMENT

In case ACKs are not piggybacked onto data packets, parts of the analysis have to be adapted. In order to be able to integrate this kind of ACK communication into the analysis, a dedicated RT channel for ACK packets is introduced, acknowledging data packets belonging to any of the RT channels in the opposite direction. The ACK RT channel is defined by its period,  $P_{ACK}$ , i.e., the minimum time allowed between two consecutive ACK packets eligible for transmission, its deadline  $D_{ACK}$ , i.e., the maximum queueing delay experienced by any individual ACK packet, and  $T_{ACK\_nonpiggy}$ , the transmission time of each ACK packet, calculated by

$$T_{ACK\_nonpiggy} = \frac{L_{ACK}}{R_{reverse}} \quad (28)$$

where  $L_{ACK}$  is the length of the ACK packet including the header. Consequently, (12) can now be rewritten in the following way:

$$d_{ord,i} = D_{ord,i} - 2 \cdot T_{prop} - T_{proc1} - T_{proc2} - T_{margin} - T_{x\_forward} - P_{ACK} - D_{ACK}. \quad (29)$$

The two instances of  $T_{ACK\_piggy}$  have been exchanged with one instance of  $P_{ACK}$  as the worst-case waiting time for an ACK packet until it is eligible for transmission, and one instance of  $D_{ACK}$  as the maximum queueing delay experienced by an ACK packet, including the actual transmission time of this ACK packet.  $P_{ACK}$  can be set sufficiently long to acknowledge several data packets at a time without resulting in a too high degree of utilization of the link capacity, but at the same time short enough to avoid the addition of too much delay in (29).  $D_{ACK}$  can, with benefit, be set to a low value, corresponding to a short deadline, to normally have priority over data packets.

Regarding the retransmission channels, (20) has to be adapted the same way as (12), affecting the result of (19). Equation (20) is therefore rewritten as

$$T_{retr\_const} = 2 \cdot T_{prop} + T_{proc1} + T_{proc2} + T_{margin} + T_{x\_forward} + P_{ACK} + D_{ACK}. \quad (30)$$

The delay bound for each retransmission attempt except the last one, originally calculated in (22), to be used in the runtime implementation for timeout calculations, is given by

$$D_{retr\_other,i} = d_{retr,i} + 2 \cdot T_{prop} + T_{proc1} + T_{proc2} + T_{margin} + T_{x\_forward} + P_{ACK} + D_{ACK}. \quad (31)$$

Further adaptation is needed in the utilization check [(23)] by adding the needed additional network capacity for ACK packets

$$U = \sum_{i=1}^Q \left( \frac{T_{x\_tot\_forward,i}}{P_{T,i}} \right) + \sum_{i=1}^M \left( \frac{T_{x\_retr\_forward,i}}{P_{re,i}} \right) + \frac{T_{ACK\_nonpiggy}}{P_{ACK}}. \quad (32)$$

Also, the workload function [(24)] is extended with a term for the ACK packets

$$h(t) = \sum_{\substack{i \in [1, Q], \\ d_{ord,i} \leq t}} \left( 1 + \left\lfloor \frac{t - d_{ord,i}}{P_{T,i}} \right\rfloor \right) \cdot T_{x\_tot\_forward,i} + \sum_{\substack{i \in [1, M], \\ d_{retr,i} \leq t}} \left( 1 + \left\lfloor \frac{t - d_{retr,i}}{P_{re,i}} \right\rfloor \right) \cdot T_{x\_retr\_forward,i} + I(D_{ACK} \leq t) \cdot \left( 1 + \left\lfloor \frac{t - D_{ACK}}{P_{ACK}} \right\rfloor \right) \cdot T_{ACK\_nonpiggy} \quad (33)$$

where  $I(A) = 1$  if  $A$  is true, otherwise, zero. The argument made in connection with (24) on why the workload function can be used in this context still holds.

## VI. SIMULATION ANALYSIS

For the validation of the suggested ARQ scheme and evaluation of its performance, two types of simulations have been conducted. In the first simulation, a schedulability analysis is implemented that calculates to which degree the communication link is utilized by the simulated hard real-time traffic, not including any possible retransmissions. The calculations are made by first letting the simulator generate requests for RT channels and by then checking the number of accepted channels according to the real-time scheduling analysis. The bandwidth utilization when using the presented scheme is compared to the case of solely having ordinary RT channels without any retransmission scheme. More strictly, the utilization ( $y$  axis) is defined as the sum of every accepted ordinary RT channel's utilization [the term inside the first pair of parentheses in (23)] for a specific number of requested ordinary RT channels ( $x$  axis). The second type of simulation is on the packet level, observing the communication channel in order to analyze the traffic which has been generated in the first simulation. Packet transmissions and retransmissions are measured making possible calculations of the average MER depending upon if the proposed retransmission scheme is utilized or not.

The simulator is implemented in MATLAB. In both simulations, the assumed bit rate over the full-duplex physical link is 50 Mbit/s in each direction for reasons of simplicity. This choice of bit rate is influenced by bit rates in current wireless local area network (WLAN) technology. The maximum packet length is set to  $L_{pack} = 1000$  bits, the propagation delay in one direction is assumed to be  $T_{prop} = 1 \mu s$  ( $\approx 200$  m, i.e., many control systems and embedded systems will have an even shorter propagation delay), and the length of each ACK packet is defined to be  $L_{ACK} = 100$  bits. The assumed packet length is slightly shorter than possible in today's WLAN technology. The reason for this is that industrial communication systems often use shorter packets than typical LANs do. This situation, however, might change in the future, as, e.g., Internet connection might be necessary in certain contexts. The parameters  $T_{proc1}$ ,  $T_{proc2}$  and  $T_{margin}$  are assumed to be negligible in the simulations, as they are constants that normally have values of negligible size, and are therefore set to zero. Moreover, ACKs are assumed to comprise a sufficient amount of redundancy for error correction resulting in a negligible error rate. Additionally, support for piggyback ACKs is assumed to exist. An implementation without the possibility of piggyback would lead to less efficient bandwidth utilization. While the formalities for an exact analysis are given in Section V, the study of other parameters, trying to examine the potential of the presented method, was prioritized in this simulation study.

The parameters for each generated ordinary RT channel are chosen randomly (with even distribution) from one of the four traffic classes listed in Table II (except for one simulation discussed later). The BER, is varied between the values  $10^{-6}$ ,  $10^{-5}$ , and  $10^{-4}$ . For substantially higher BERs, error correcting codes can be used to initially lower the BER (or packet-error rate) experienced by the ARQ protocol [11]. Another way to initially reduce the BER is using robust transceiver design, e.g., by using multiple antennas [32], [33]. While the error probability

TABLE II  
PARAMETERS OF THE FOUR DIFFERENT TRAFFIC CLASSES  
USED IN THE SIMULATIONS

Traffic class	$P_{T,i}$	$D_{T,i}$	$L_{T,i}$
1	2 ms	2 ms	4 000 bits
2	4 ms	4 ms	4 000 bits
3	8 ms	8 ms	4 000 bits
4	16 ms	16 ms	4 000 bits

TABLE III  
CASE DEFINITIONS FOR DIFFERENT NUMBER OF RETRANSMISSION CHANNELS  
USED IN THE SIMULATIONS

Case	Nbr of retransmission channels ( $M$ )	Nbr of retransmission attempts ( $N_{attempt}$ )	$P_{re,i}$	$D_{re,i}$	$L_{re,i}$
#1	4	1	2 ms	300 $\mu s$	1 000 bits
#2	4	2	2 ms	600 $\mu s$	1 000 bits
#3	8	4	2 ms	900 $\mu s$	1 000 bits

is constant for the first four cases presented in this simulation study, the section concludes with a study of the performance of the retransmission scheme under bursts of errors.

Three different cases are defined in Table III to specify traffic scenarios used in the first simulations. The number of retransmission channels is set to 4 or 8, while the number of retransmission attempts is varied from 1 to 2 and 4. While the period and packet length of the retransmission channels stay the same during all three cases, the values of the retransmission channel deadlines are set to values giving roughly the same utilization penalty when using the retransmission scheme. Cases with the number of retransmission channels equal to the number of retransmission attempts have also been simulated. However, no graphs are shown for those cases since it is advantageous, and therefore recommended, to always have more retransmission channels than attempts. The reason for this recommendation is that it is otherwise rather unlikely that enough retransmission channels are free to be used when the maximum number of attempts is needed. As an example, however, it can be mentioned that for a case with four retransmission channels and otherwise an identical setup as Case #3 in Table III, the MER showed to be more than two orders of magnitude higher than for Case #3.

In Figs. 6–11, the utilization (a) and the MER (b) are shown for both a link with the retransmission scheme (unbroken lines) and a link without it (dashed lines). For each  $x$  axis value, the  $y$  axis value is the average of at least 1000 simulations, with the duration of 10–100 hyper periods each for the packet level simulations. The MER for the cases without retransmissions can be verified by basic calculations due to the known BER and message length. [Simulated values are shown in all figures depicting the MER, i.e., labeled (b), and printed with a dashed line.] Assuming a BER and a message length  $L_{mess}$  in bits, the MER can be calculated as follows:

$$MER = 1 - (1 - BER)^{L_{mess}}. \quad (34)$$

Figs. 6 and 7 show the simulation results when using one retransmission attempt over four retransmission channels with the parameters  $P_{re,i} = 2\,000 \mu s$ ,  $D_{re,i} = 300 \mu s$ , and  $L_{re,i} = 1\,000$  bits. For this case, Case #1, BERs of  $10^{-4}$  (not shown in any figure),  $10^{-5}$  and  $10^{-6}$  were simulated. As shown in the two sets of figures for the lower BERs, when introducing the

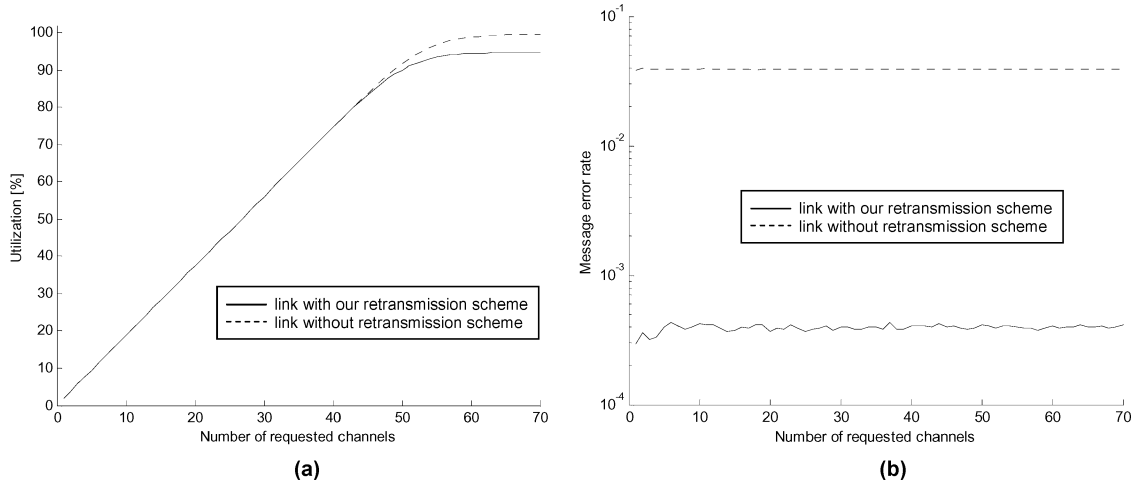


Fig. 6. Simulation results when supporting one retransmission attempt and having four retransmission channels, each with the parameters  $P_{re,i} = 2$  ms,  $D_{re,i} = 300$   $\mu$ s, and  $L_{re,i} = 1$  000 bits. The bit error rate was set to  $BER = 10^{-5}$ .

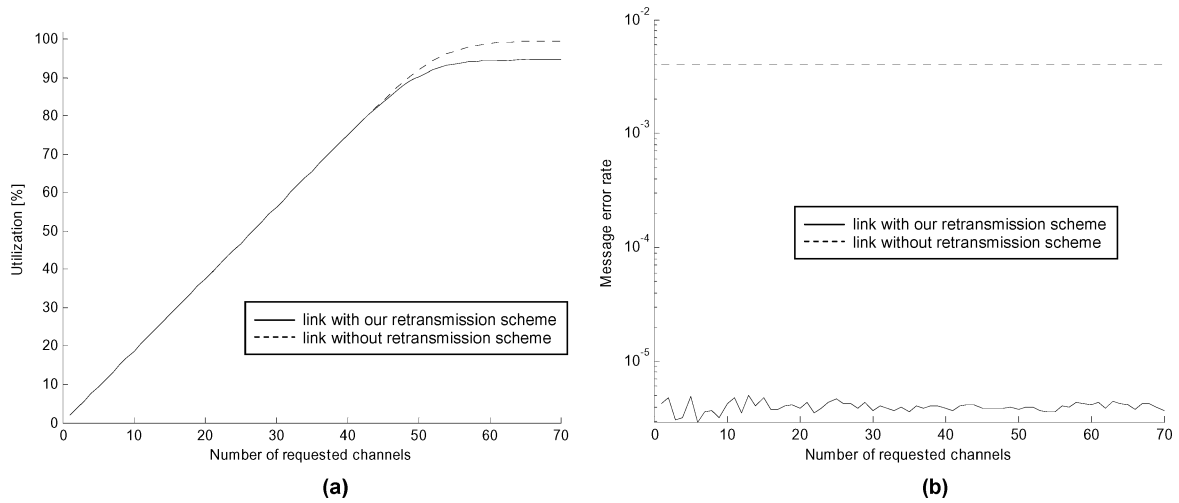


Fig. 7. Simulation results when supporting one retransmission attempt and having four retransmission channels, each with the parameters  $P_{re,i} = 2$  ms,  $D_{re,i} = 300$   $\mu$ s, and  $L_{re,i} = 1$  000 bits. The bit error rate was set to  $BER = 10^{-6}$ .

retransmission scheme for this case, the utilization for ordinary RT channels is just reduced by a few percentage points at saturation. Nevertheless, a reduction of the MER of almost two orders of magnitude is achieved for the BER of  $10^{-5}$ , and almost three orders of magnitude for the BER of  $10^{-6}$ . At the BER of  $10^{-4}$ , a 5% utilization penalty still improved the average MER by one order of magnitude. The lower the BER, the higher the grade of improvement that can be reached through the proposed retransmission scheme, while keeping the utilization penalty at a nearly constant level. At low numbers of requested RT channels, all RT channels are easily accepted regardless whether retransmission channels also shall be supported or not. The two curves in each utilization graph [Figs. 6(a)–11(a)] therefore follow each other well up to a certain number of requested RT channels.

The simulation results for Case #2 are shown in Figs. 8 and 9. Compared to Case #1, there now are two retransmission attempts over the four retransmission channels, while the deadline of the retransmission channels has been doubled to  $D_{re,i} = 600$   $\mu$ s. This change of deadline has been triggered by the fact that doubling the amount of (possible) retransmission

leads to more packets competing, and therefore the possibility of sending the retransmission before the deadline increases when the deadline is extended. The BERs for which the results are shown in the figures are  $10^{-4}$  and  $10^{-5}$ , while almost no message error could be detected for a BER of  $10^{-6}$  when using the retransmission scheme. Due to this lack of sufficient error measurements, no statistic conclusions could be drawn and therefore this figure is excluded from this evaluation. Fig. 8 shows the results for  $BER = 10^{-4}$ . An improvement of the MER of about 1.5 orders of magnitude can be reached, while the utilization for ordinary RT channels only is reduced by a few percentage points at saturation. When decreasing the BER to  $10^{-5}$ , a similar utilization penalty results in an improvement of four orders of magnitude of the MER (see Fig. 9). Again, one can see that the grade of improvement, regarding the MER, is highest at lower values of BER.

The last case specified in Table III assumes four retransmission attempts over eight retransmission channels. The parameters defining the retransmission channels are  $P_{re,i} = 2$  ms,  $D_{re,i} = 900$   $\mu$ s, and  $L_{re,i} = 1$  000 bits. Simulations with two

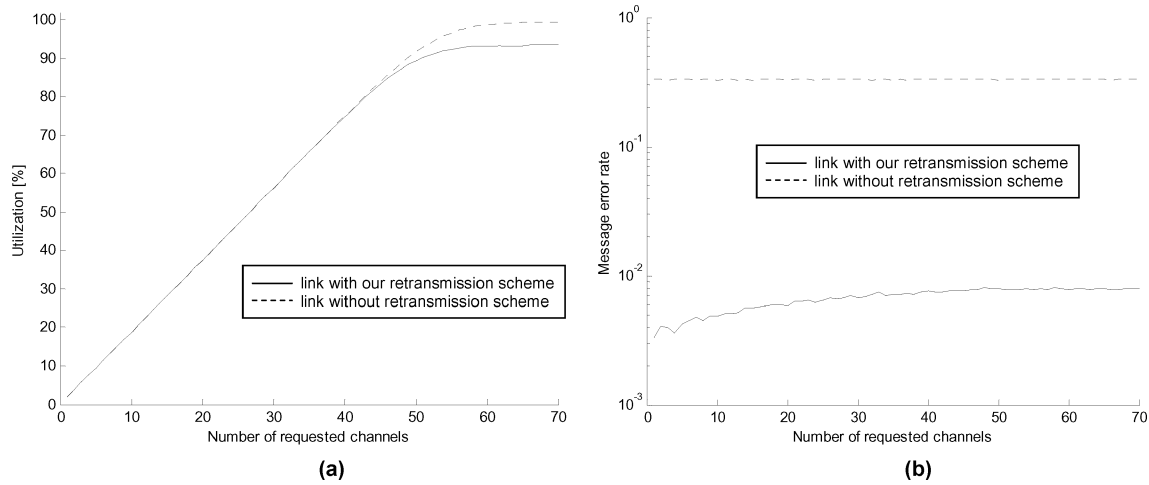


Fig. 8. Simulation results when supporting two retransmission attempts and having four retransmission channels, each with the parameters  $P_{re,i} = 2$  ms,  $D_{re,i} = 600$   $\mu$ s, and  $L_{re,i} = 1\,000$  bits. The bit error rate was set to  $BER = 10^{-4}$ .

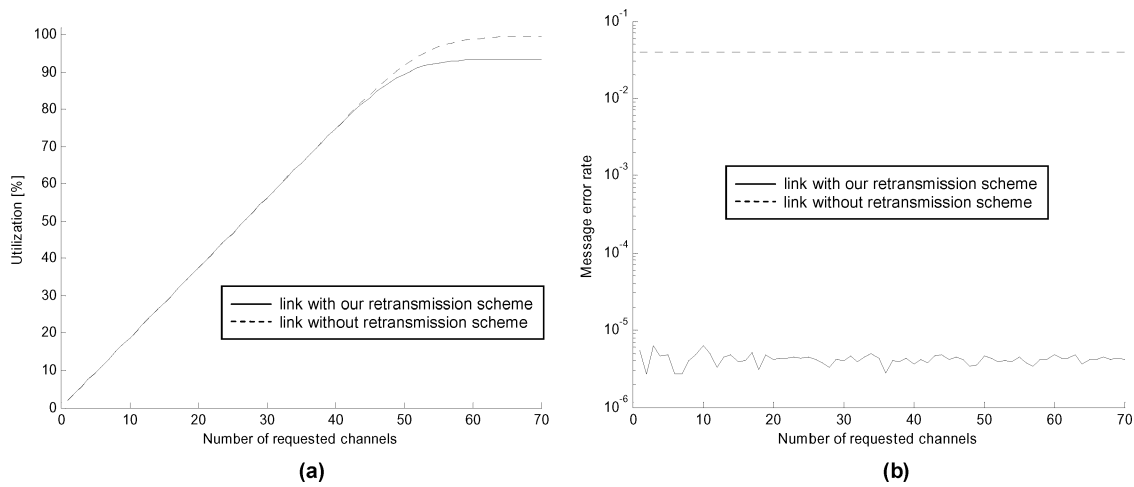


Fig. 9. Simulation results when supporting two retransmission attempts and having four retransmission channels, each with the parameters  $P_{re,i} = 2$  ms,  $D_{re,i} = 600$   $\mu$ s, and  $L_{re,i} = 1\,000$  bits. The bit error rate was set to  $BER = 10^{-5}$ .

different BERs were done,  $10^{-4}$  and  $10^{-5}$ . As  $BER = 10^{-5}$  already resulted in zero message errors registered by the simulator when using the presented retransmission scheme, no lower BER was tested. The figures showing the results for  $BER = 10^{-5}$  are omitted due to this lack of sufficient error measurements. Fig. 10 shows the results for the highest BER tested,  $10^{-4}$ . The choice of retransmission parameters (number of retransmission channels, number of retransmission attempts and deadline for the retransmission channel) leads to a higher utilization penalty at the saturation point compared to Case #1 and Case #2. However, the improvement of the MER of four orders of magnitude, despite the high BER, is considerable.

As mentioned in the beginning of this section, one simulation that was conducted is not included in the earlier traffic specifications. The reason for that are the different characteristics of this simulation. The choice of parameters this time has been guided by the question if a high grade of improvement could be reached for some traffic situations while still trying to keep the utilization penalty at a very low level. The chosen parameters for the traffic classes and the retransmission channels can be found in Tables IV and V, respectively.

Fig. 11 shows, assuming a BER of  $10^{-4}$  and the given parameters, that a substantial decrease of the MER can be achieved. The improvement was measured to almost five orders of magnitude, while the utilization penalty stayed at merely a few percentage points at saturation. These results are contributed to a favorable combination of different parameter characteristics, as, e.g., the high number of retransmission channels which simplify the schedulability of the retransmissions, and the high deadline-period ratio of the data traffic, where the relative deadlines are twice the length of the periods. Even this increases the possibility of a successful, i.e., feasible, retransmission schedule. Furthermore, the choice of retransmission channel parameters, in particular, the long period, results in a low utilization penalty. When lowering the BER to  $10^{-5}$ , no message errors could be detected by the simulator, and therefore the simulation for  $BER = 10^{-6}$  was omitted.

In order to include a study of the performance of the retransmission scheme under bursts of errors, not uncommon in wireless networks, Case #2 in Table III has been simulated assuming the Gilbert–Elliot error model. According to this model, two error states (“good state” and “bad state”) are defined. In this

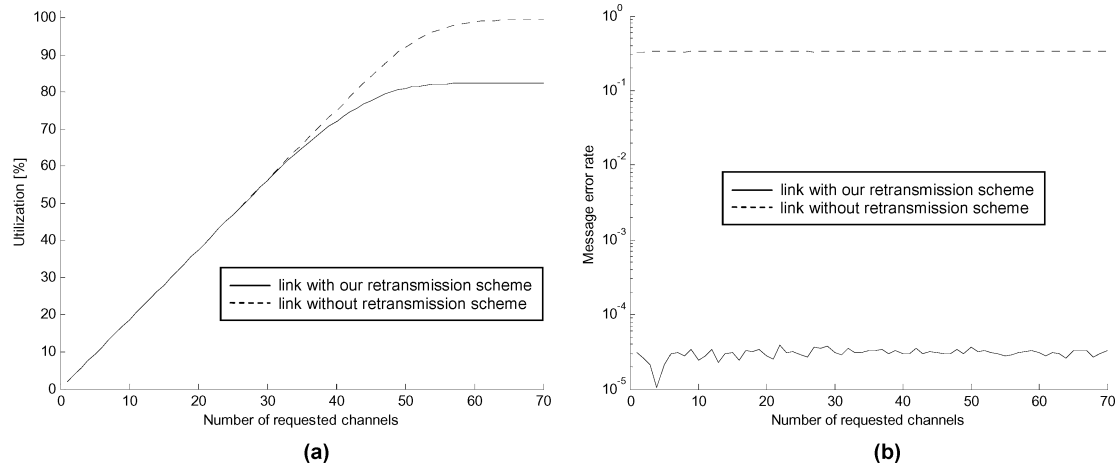


Fig. 10. Simulation results when supporting four retransmission attempts and having eight retransmission channels, each with the parameters  $P_{re,i} = 2$  ms,  $D_{re,i} = 900$   $\mu$ s, and  $L_{re,i} = 1\,000$  bits. The bit error rate was set to  $BER = 10^{-4}$ .

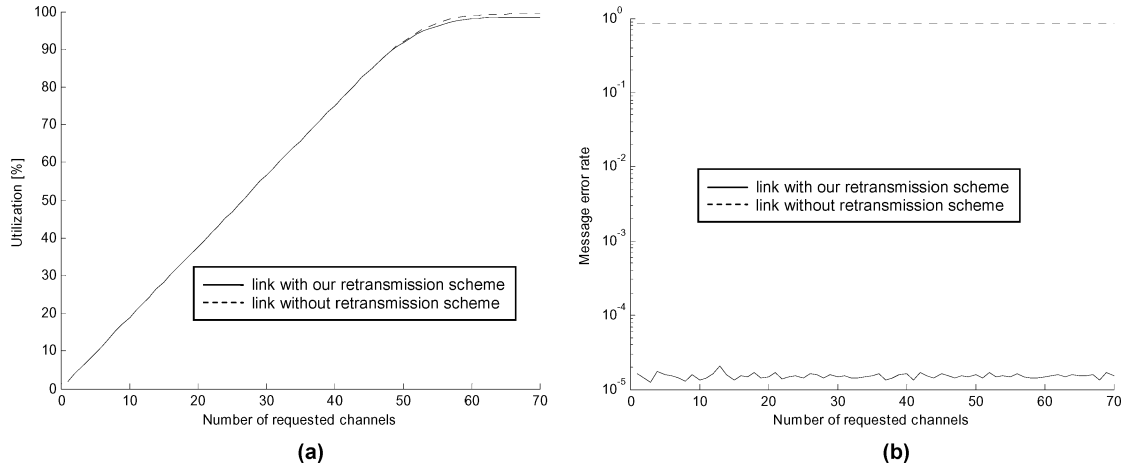


Fig. 11. Simulation results when supporting five retransmission attempts and having 25 retransmission channels, each with the parameters  $P_{re,i} = 50$  ms,  $D_{re,i} = 3$  ms, and  $L_{re,i} = 1\,000$  bits. The bit error rate was set to  $BER = 10^{-4}$ .

TABLE IV  
PARAMETERS OF THE FOUR DIFFERENT TRAFFIC CLASSES  
USED IN THE LAST SIMULATION

Traffic class	$P_{Ti}$	$D_{Ti}$	$L_{Ti}$
1	10 ms	20 ms	20 000 bits
2	20 ms	40 ms	20 000 bits
3	40 ms	80 ms	20 000 bits
4	80 ms	160 ms	20 000 bits

TABLE V  
CASE DEFINITIONS FOR DIFFERENT NUMBER OF RETRANSMISSION CHANNELS  
USED IN THE LAST SIMULATION

Case	Nbr of retransmission channels ( $M$ )	Nbr of retransmission attempts ( $N_{attempt}$ )	$P_{re,i}$	$D_{re,i}$	$L_{re,i}$
#4	25	5	50 ms	3 ms	1 000 bits

case, BERs of  $10^{-5}$  for the good state and  $10^{-4}$  the bad state are assumed. A state transition is possible every 2 ms with probabilities of 0.995 and 0.96 to remain in the good and the bad state, respectively. Those probabilities correspond to average times in

the good state of 400 ms and in the bad state of 50 ms. The results are shown in Fig. 12. The bad state can be seen to increase the MER substantially compared to a constant BER of  $10^{-5}$  [see Fig. 9(b)], but is still better than a constant BER of  $10^{-4}$  [see Fig. 8(b)].

In order to investigate the approximate execution time of the schedulability test, an isolated run of this test was conducted assuming the parameters from Fig. 12, i.e., Case #2 with the Gilbert–Elliot error model. Running each of the 70  $x$  axis values (i.e., sets of RT channels of increasing size) 1000 times in order to get smooth (averaged) curves resulted in 70 000 individual schedulability tests performed during the simulation. These runs took six seconds on a PC with a 3.0 GHz Pentium 4 processor with hyper-threading, which means an average execution time per test of 86  $\mu$ s. To verify this measurement, this simulation was run 10 000 times, which took 60 s. The average number of RT channels per test was 32.3, excluding the four retransmission channels. As the addition of new RT channels can be assumed to be relatively infrequent, 86  $\mu$ s is regarded as an acceptable value. Moreover, this execution time can be shortened significantly by optimizing the code for speed and by not running it by a MATLAB interpreter.

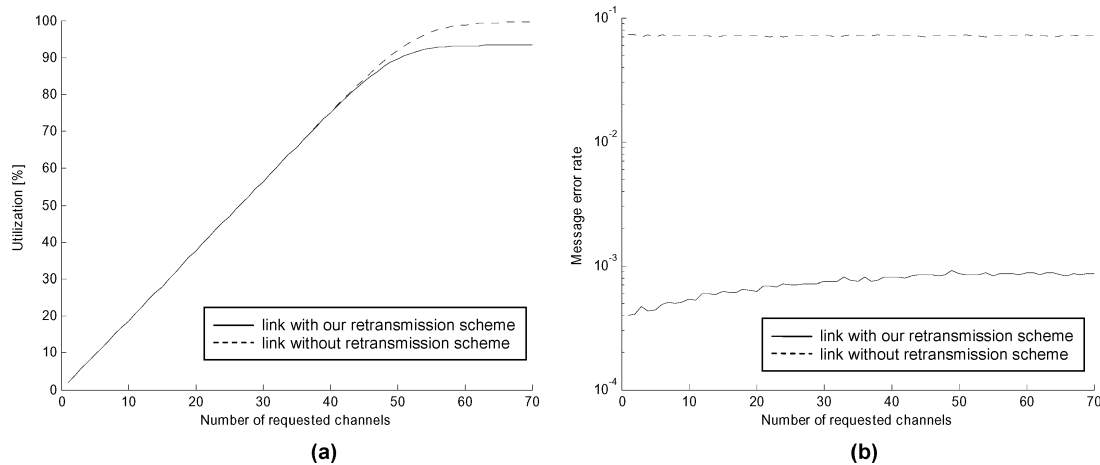


Fig. 12. Simulation results when supporting two retransmission attempts and having four retransmission channels, each with the parameters  $P_{re,i} = 2$  ms,  $D_{re,i} = 600$   $\mu$ s, and  $L_{re,i} = 1$  000 bits. The Gilbert–Elliot error model was used for bursts of errors, alternating between a bit error rate of  $BER = 10^{-4}$  and  $BER = 10^{-5}$ .

## VII. CONCLUSION

This paper summarizes research done so far for creating this framework for reliable communication over a wireless link, using retransmissions at the same time as delay bounds for ordinary transmissions can still be guaranteed. The framework is studied for the example of a point-to-point link, assuming EDF scheduling. An extensive simulation study was conducted, evaluating the performance of the proposed retransmission scheme in terms of both utilization overhead and improvement in MER. The study showed clearly that, under the assumption of parameters typical for a wireless network, a reduction of the MER by several orders of magnitude was possible while still keeping the utilization overhead at a reasonable level. A real-time analysis for the case of both piggybacked ACKs and nonpiggybacked ACKs has been developed.

Working with this framework has opened up several possibilities of future research. Work currently in progress is studying the relations and dependencies between different design parameters in the retransmission scheme, as, e.g., the number of retransmissions or the period of the retransmission channel. In the future, an extension to the framework is planned to address both wireless multihop networks, other scheduling policies, as, e.g., first-come first-serve scheduling, and make adaptations for specific underlying wireless communication standards. As retransmission schemes also exist on the link layer, a comparative study of link layer and transport layer retransmission could give valuable results on when per-hop retransmission might be preferable, compared to end-to-end retransmissions.

## REFERENCES

- [1] A. Willig, K. Matheus, and A. Wolisz, "Wireless technology in industrial networks," *Proc. IEEE*, vol. 93, no. 6, pp. 1130–1151, Jun. 2005.
- [2] D. Dzung, C. Apneseth, J. Endresen, and J.-E. Frey, "Design and implementation of a real-time wireless sensor/actuator communication system," in *Proc. 10th IEEE Conf. Emerging Technol. Factory Autom. (ETFA '05)*, Catania, Italy, Sep. 2005, vol. 2, pp. 433–442.
- [3] N. J. Ploplys, P. A. Kawka, and A. G. Alleyne, "Closed-loop control over wireless networks," *IEEE Control Syst. Mag.*, vol. 24, no. 3, pp. 58–71, Jun. 2004.
- [4] H. Ye, G. Walsh, and L. Bushnell, "Wireless local area networks in the manufacturing industry," in *Proc. 2000 Amer. Control Conf.*, Chicago, IL, Jun. 2000, vol. 4, pp. 2363–2367.
- [5] H. Ye and G. Walsh, "Real-time mixed-traffic wireless networks," *IEEE Trans. Ind. Electron.*, vol. 48, no. 5, pp. 883–890, Oct. 2001.
- [6] H.-J. Körber, H. Wattar, and G. Scholl, "Modular wireless real-time sensor/actuator network for factory automation applications," *IEEE Trans. Ind. Inform.*, vol. 3, no. 2, pp. 111–119, May 2007.
- [7] S. Cavalieri and D. Panno, "A novel solution to interconnect fieldbus systems using IEEE wireless LAN technology," *Comput. Standards and Interfaces*, vol. 20, no. 1, pp. 9–23, Nov. 1998.
- [8] A. Willig, "An architecture for wireless extension of PROFIBUS," in *Proc. 29th Conf. IEEE Ind. Electron. Soc. (IECON '03)*, Nov. 2003, vol. 3, pp. 2369–2375.
- [9] D. Miorandi and S. Vitturi, "A wireless extension of Profibus DP based on the Bluetooth radio system," *Ad Hoc Networks*, vol. 3, no. 4, pp. 479–494, Jul. 2005.
- [10] A. Willig, "A MAC protocol and a scheduling approach as elements of a lower layers architecture in wireless industrial LANs," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, Oct. 1997, pp. 139–148.
- [11] P.-A. Wiberg and U. Bilstrup, "Wireless technology in industry-applications and user scenarios," in *Proc. 8th IEEE Conf. Emerging Technol. Factory Autom. (ETFA '01)*, Antibes, France, Oct. 2001, vol. 1, pp. 123–131.
- [12] A. Willig, "Polling-based MAC protocols for improving real-time performance in a wireless PROFIBUS," *IEEE Trans. Ind. Electron.*, vol. 50, pp. 806–817, Aug. 2003.
- [13] S. Pejhan, M. Schwartz, and D. Anastassiou, "Error control using retransmission schemes in multicast transport protocols for real-time media," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 413–427, Jun. 1996.
- [14] E. Uhlemann, P.-A. Wiberg, T. M. Aulin, and L. R. Rasmussen, "Deadline dependent coding – A framework for wireless real-time communication," in *Proc. 7th Int. Conf. Real-Time Comput. Syst. Appl. (RTCSA '00)*, Cheju Isl., S. Korea, Dec. 2000, pp. 135–142.
- [15] E. Uhlemann and L. K. Rasmussen, "Incremental redundancy deadline dependent coding for efficient wireless real-time communications," in *Proc. 10th IEEE Conf. Emerging Technol. Factory Autom. (ETFA '05)*, Catania, Italy, Sep. 2005, vol. 2, pp. 417–424.
- [16] M. M. Butt, "Provision of guaranteed QoS with hybrid automatic repeat request in interleave division multiple access systems," in *Proc. 10th IEEE Singapore Int. Conf. Commun. Syst. (ICCS '06)*, Singapore, Oct. 2006, pp. 1–5.
- [17] G. Giancola, S. Falco, and M. G. Di Benedetto, "A novel approach to error protection in medium access control design," in *Proc. 4th Int. Workshop on Mobile Wireless Commun. Networks (MWCN '02)*, Stockholm, Sweden, Sep. 2002, pp. 337–341.
- [18] X. Fan, M. Jonsson, and J. Jonsson, "Guaranteed real-time communication in packet-switched networks with FCFS queuing," *Comput. Networks*, vol. 53, no. 3, pp. 400–417, Feb. 2009.

- [19] M. Jonsson and K. Kunert, "Reliable hard real-time communication in industrial and embedded systems," in *Proc. IEEE 3rd Symp. Ind. Embedded Syst. (SIES '08)*, Montpellier, France, Jun. 2008, pp. 184–191.
- [20] M. Jonsson and K. Kunert, "Meeting reliability and real-time demands in wireless industrial communication," in *Proc. 13th IEEE Int. Conf. Emerging Technol. Factory Autom. (ETFA '08)*, Hamburg, Germany, Sep. 2008, pp. 877–884.
- [21] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973.
- [22] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE J. Sel. Areas Commun.*, vol. 8, no. 3, pp. 368–379, Apr. 1990.
- [23] J.-F. Frigon, V. C. M. Leung, and H. C. B. Chan, "Dynamic reservation TDMA protocol for wireless ATM networks," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 2, pp. 370–383, Feb. 2001.
- [24] G. Scheible, D. Dzung, J. Endresen, and J.-E. Frey, "Unplugged but connected – Design and implementation of a truly wireless real-time sensor/actuator interface," *IEEE Ind. Electron. Mag.*, vol. 1, no. 2, pp. 25–34, 2007.
- [25] A. Mishra, C. Na, and D. Rosenburgh, "On scheduling guaranteed time slots for time sensitive transactions in IEEE 802.15.4 networks," in *Proc. IEEE Military Commun. Conf. 2007 (MILCOM '07)*, Oct. 2007, pp. 1–7.
- [26] Q. Zheng and K. G. Shin, "On the ability of establishing real-time channels in point-to-point packet-switched networks," *IEEE Trans. Commun.*, vol. 42, no. 2-4, pp. 1096–1105, Feb.-Apr. 1994.
- [27] H. Hoang and M. Jonsson, "Switched real-time ethernet in industrial applications – Deadline partitioning," in *Proc. 9th Asia-Pacific Conf. Commun. (APCC '03)*, Penang, Malaysia, Sep. 2003, vol. 1, pp. 76–81.
- [28] S. K. Baruah, A. K. Mok, and L. E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proc. 11th Real-Time Syst. Symp. (RTSS '90)*, Dec. 1990, pp. 182–190.
- [29] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Real-Time Syst.*, vol. 2, no. 4, pp. 301–324, Nov. 1990.
- [30] M. Spuri, "Analysis of deadline scheduled real-time systems," INRIA, France, Tech. Rep. 2772, 1996.
- [31] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, *Deadline Scheduling for Real-Time Systems – EDF and Related Algorithms*. Boston, MA: Kluwer, 1998.
- [32] P. S. Neelakanta and W. Preedalumpabut, "Robust operations of IEEE 802.11/WLAN and 802.15/WPAN wireless links in industrial informatic networks," in *Proc. IEEE Int. Conf. Ind. Inform. (INDIN'06)*, Singapore, Aug. 2006, pp. 276–281.
- [33] A. Willig, "Antenna redundancy for increasing transmission reliability in wireless industrial LANs," in *Proc. 8th IEEE Int. Conf. Emerging Technol. Factory Autom. (ETFA '03)*, Lisbon, Portugal, Sep. 2003, vol. 1, pp. 7–14.



**Magnus Jonsson** (S'95–M'00–SM'07) received the B.S. and M.S. degrees in computer engineering from Halmstad University, Halmstad, Sweden, in 1993 and 1994, respectively, and the Licentiate of Technology and Ph.D. degrees in computer engineering from Chalmers University of Technology, Gothenburg, Sweden, in 1997 and 1999, respectively.

He is a Full Professor of Real-Time Computer Systems at Halmstad University since 2003, where he also is the Head of the Computing and Communications Laboratory and Vice-Director of CERES – The Centre for Research on Embedded Systems. From 1998 to March 2003, he was an Associate Professor of Data Communication at Halmstad University (acting between 1998 and 2000). He has published more than 75 scientific papers and book chapters, most of them in the area of real-time communication, real-time and embedded computer systems, computer networking, optical networking, and optical interconnection architectures.

Prof. Jonsson is a senior member of the Association for Computing Machinery (ACM) and a member of the Optical Society of America (OSA). He has served on the program committees of many conferences and workshops and served as an International Liaison Co-Chair for the 28th International Conference on Distributed Computing Systems (ICDCS 2008).



**Kristina Kunert** (S'09) received the B.Sc. degree in information and communication technology and the M.Sc. degree in computer systems engineering with a specialization in embedded systems from Halmstad University, Halmstad, Sweden in 2003 and 2004, respectively, and the Licentiate of Technology degree from the Department of Computer Engineering, Chalmers University of Technology, Gothenburg, Sweden, in 2008. She is currently working towards the Ph.D. degree in the area of real-time communication at the Centre for Research

on Embedded Systems (CERES), Halmstad University, and Chalmers University of Technology, where her current research focus is on methods for increasing the reliability of real-time traffic in wired and wireless networks.

Ms. Kunert is a student member of the Association for Computing Machinery (ACM).