Enhancing reliability in IEEE 802.11 based real-time networks through transport layer retransmissions

Kristina Kunert, Elisabeth Uhlemann and Magnus Jonsson

Kunert K, Uhlemann E, Jonsson M. Enhancing reliability in IEEE 802.11 based real-time networks through transport layer retransmissions. In: 2010 International Symposium on Industrial Embedded Systems (SIES). IEEE; 2010. p. 146-155.

DOI: 10.1109/SIES.2010.5551388

Enhancing Reliability in IEEE 802.11 Based Real-Time Networks through Transport Layer Retransmissions

Kristina Kunert, Elisabeth Uhlemann, and Magnus Jonsson
CERES – Centre for Research on Embedded Systems
Halmstad University, Halmstad, Sweden
{Kristina.Kunert, Elisabeth.Uhlemann, Magnus.Jonsson}@hh.se

Abstract - **As the number of application areas for wireless technologies grows, the need for providing both predictable and reliable communication over wireless networks becomes apparent. Cooperative embedded systems for industrial automation are one example of systems with these needs. Previously, we developed a framework for reliable real-time communication in a single-hop wireless network with a logical star topology. The framework was placed on top of IEEE 802.15.4 and combines transport layer retransmissions with real-time analysis admission control. IEEE 802.15.4 was selected due to its advantageous energy saving techniques, making it an interesting choice for wireless sensor networks in industrial contexts. However, its achievable data rate is rather low, especially when voice or video for industrial surveillance and monitoring need to be transferred. Hence, we adapt our framework to fit the IEEE 802.11 standard and evaluate its performance using a data traffic model from industrial control and surveillance systems. The performance of the framework is evaluated in terms of network utilization, message error rate and delay distribution using theoretical analysis as well as computer simulations.**

I. Introduction

Many of today's industrial automation systems need wireless communication. This implies that applications requiring reliable real-time services often attempt to run over noisy wireless networks with inherently high packet error probabilities. We have previously presented a theoretical framework [1] for reducing the message error rate (MER) over a wireless link through retransmissions, while still being able to guarantee delay bounds for real-time traffic. We adopt a joint view of real-time constraints and truncated retransmission schemes, such that the number of retransmissions is adjusted to the delay bound, thereby avoiding unnecessary retransmissions of packets after their deadline, or retransmissions interfering with other delay bound guaranteed traffic. Placing the retransmission functionality in the transport layer rather than in the link layer of a single-hop network avoids unnecessary retransmissions. Such unnecessary retransmissions would occur e.g. if two packets of a message consisting of five packets need to be retransmitted, but due to deadline constraints there is only time to retransmit one of these packets in time.

Using existing standards and commercial off the self components is vital in industrial settings, to reduce costs and maintain interoperability. Hence, in [2] our framework was placed on top of IEEE 802.15.4 in a typical wireless industrial network with a logical star topology. We showed that our approach is able to reduce the MER introduced by transient noise on the medium without jeopardizing delay bound guarantees given to ordinary transmissions. Further, the retransmissions can be constrained to require only a reasonable increase of the bandwidth. However, even though IEEE 802.15.4 has advantageous energy saving techniques, the available data rate is rather low. This is cumbersome for two reasons. Firstly, since data rate can be traded for reliability, the IEEE 802.15.4 standard is generally less robust against noise and interference compared to e.g., IEEE 802.11 WLAN. Secondly, albeit many industrial applications consist of some tens of nodes transmitting only a few bytes of data, there are also industrial control and monitoring systems consisting of hundreds of nodes transmitting up to 80 bytes of data. In this paper, we therefore extend our framework to target industrial monitoring and surveillance applications and demonstrate how it can be used on top of commercially available IEEE 802.11 transceivers in a typical single-hop network with a logical star topology.

Previously, we assessed performance using numerical evaluation of the network utilization, U, together with Monte-Carlo computer simulations to determine the message error rate (MER). This paper also extends the general framework to include a theoretical analysis and numerical evaluation of the MER as well as computer simulations to determine the actual U experienced. In addition, the effects of retransmissions on the delay distribution are evaluated, as jitter is an important parameter for voice and video transfer.

The reminder of this article is organized as follows. Section 2 presents related works, and in Section 3 our system model is described. In Section 4 the transport layer scheme with retransmissions and real-time analysis admission control is introduced, followed by the analytical performance evaluation in Section 5. The computer simulation results based on schedulable traffic flows are presented in Section 6, while Section 7 contains our conclusions.

II. Related Works

Prior research in the area of real-time communication is plentiful, as are the solutions for increased reliability through Automatic Repeat reQuest (ARQ) schemes. Related research results include e.g. information redundancy to increase the probability of a successful transmission [3], but where even error-free packets are retransmitted creating unnecessary additional traffic. In [4], a fixed amount of extra

E. Uhlemann is funded in part by the Swedish Governmental Agency for Innovation Systems, Vinnova, through the VINNMER program, www.vinnova.se.

time for possible retransmissions is claimed by each node based on error probabilities. But, if not all erroneous packets in a message have time to be retransmitted, resources are wasted without adding benefits. Further related work targets improvement of the average delay, but failing to give any deadline guarantees [5][6], or considering each communication link separately, not using real-time scheduling analysis to ensure overall network performance [7][8]. In [9], a solution similar to ours is developed, reducing the MER by introducing retransmissions that are sent only until the message deadline has passed. However, this approach contains no queuing delay analysis and therefore provides no end-to-end delay bound guarantees. A promising solution was presented in [10], providing hard real-time delay guarantees for traffic flows including retransmissions. However, the solution uses flow analysis, which has been proven to give a lower network utilization capacity than the real-time analysis used in our framework [11]. Since many existing solutions lack timing details, or require traffic regulators to be present in each node, the need for our approach is well motivated.

Ideas corresponding to our approach can be found in the area of fault-tolerant task scheduling on uniprocessor systems, which could be mapped onto a communication scenario. The reexecution of a task can then be seen as the retransmission of a packet. In [12] a schedulability analysis for periodic tasks on a single processor was carried out, but assuming a maximum of only one error at a time. In [13] those results were improved, providing real-time guarantees for task sets with higher utilizations and exact schedulability analyses for certain failure hypotheses. However, both [12] and [13] assume static priority scheduling algorithms, while we use earliest deadline first (EDF) with dynamic priorities.

There are also several publications on the timing analysis of the controller area network (CAN) [14]-[18], but also here the priority assignment differs from our approach. For variants of the original CAN bus, retransmission schemes for time-constrained traffic have been proposed, as e.g. for Timely-CAN [19] and FTT-CAN [20][21], but both are based on fixed priority assignments. Further, in [20] and [21] retransmissions exceeding the fault hypothesis are made in a best-effort manner (but before the deadline). The results of [22] and [23] are based on EDF scheduling, but assume preemptive tasks, while in our context packets are assumed to be nonpreemptive. Additionally, the results in [22] are only valid for tasks with deadlines shorter than their periods, while [23] does not treat periodic tasks at all.

III. System Model

Since our framework assumes EDF scheduling, a deterministic medium access control (MAC) protocol enforcing EDF is needed. Consequently, a simple master-slave organization with polling is used, suitable for placement on top of a basic IEEE 802.11 chipset. The decision whether or not a retransmission should be initiated is based on the real-time schedulability analysis placed in the transport layer protocol. Since all administration of retransmissions is located in the transport layer, link layer retransmissions are redundant and thus assumed to be turned off. The placement of the retransmission functionality in the transport layer makes it possible to use the bandwidth more effectively, as unnecessary retransmissions can be avoided. An unnecessary retransmission would e.g. occur if two packets of a longer message consisting of five packets need to be retransmitted in order to achieve a correct message, but due to deadline constraints there is only time to retransmit one of these packets in time.

A. Network

In accordance with a characteristic industrial network application, we assume a wireless, logical star network with one central node. Since the geographical size of factory networks is typically small, each node is assumed to be in communication range of the central node experiencing approximately the same propagation delay. Further, we assume a common frequency channel and a fixed data rate. The central node is organizing the channel access in a master-slave fashion. Slaves are polled frequently by the master, and once polled a slave may use the channel exclusively to transmit one packet. We assume that all packets hold a perfect CRC checksum and the probability of lost acknowledgement messages (ACK) is negligible.

B. Data traffic specification

The data traffic in the framework is characterized by traffic flows τ_i, i=1, 2,…, a.k.a. real-time channels (RTCs), each defined by a sender node, a receiver node, a minimum interarrival time (period), a message length (in bits) and a relative deadline (indicating the maximum time from the start of the period until the message must have arrived at the receiver): $\tau_i = \{S_i, R_i, P_i, C_i, D_i\}$. For each traffic flow, the transport layer deadline, D_i, is divided into two parts, one earlier deadline, ($D_{ord,i}$), for transmission of all ordinary packets of the message and one later deadline, ($D_{retr,i}$), enabling retransmissions of a certain limited number of these packets such that $D_i=D_{ord,i}+D_{retr,i}$. Next, we add retransmission channels, ReRTCs, to reduce the overall MER. Even retransmission channels are defined as traffic flows: $\tau_{re,j} = \{S_{re,j}, R_{re,j}, P_{re,j}, C_{re,j}, D_{re,j}\}$. The period of a retransmission channel, $P_{re,j}$, is a system parameter defining the shortest time between two consecutive uses of the retransmission channel in order to maintain a certain maximum bandwidth. The deadline, $D_{re,j}$, refers to the time allocated for a potential retransmission. Through these dedicated retransmission channels, it is possible to separately allocate network resources for retransmissions. In many hard real-time application contexts (e.g. industrial communication or distributed embedded systems), data traffic characteristics such as period, deadline or data rate are known to the master node. This makes it possible to define logical RTCs for ordinary transmissions or retransmissions. Note that several logical RTCs may exist between the same two nodes, e.g., belong-

ing to different applications. For our intended scenario, the traffic flows may exist in either direction between the master and any of the slaves.

C. Medium access control

Access to the medium can be organized in different ways, but not all of them are advisable for use with real-time traffic since queuing delays must be upper-bounded. Carrier Sense Multiple Access (CSMA), the contention-based, random access MAC method used in IEEE 802.11 WLAN, is very successful in providing high throughput for best-effort traffic, but highly unsuitable when targeting deterministic delay bound guarantees for real-time traffic. This is due to the unbounded channel access delays which may occur as a result of random back-off times in case of packet collisions. To provide deterministic treatment of real-time traffic, we extend the IEEE 802.11 MAC layer with a deterministic MAC method, where a master polls the connected slaves in a specific, deadline-dependent order. The intelligence of this polling mechanism (when to poll which slave for what packet) resides in the transport layer. Packets are thereby already handed down from transport to MAC layer in a deadline-ordered fashion. Any link layer ARQ mechanisms are therefore assumed to be turned off and only its error detection functionality is used. This is a necessary condition as retransmissions only can be permitted when they are schedulable, i.e., the retransmitted packet can meet its deadline without jeopardizing delay bound guarantees already given to ordinary transmissions or other retransmissions.

IV. Real-Time ARQ Transport Layer

A transport layer with real-time ARQ functionality and EDF scheduling of traffic is present in both master and slave nodes. The choice of EDF enables efficient scheduling of retransmissions with short deadlines amongst ordinary transmissions with longer deadlines. Since it is our goal to provide end-to-end service on a message level, the retransmission policy is placed in the transport layer. Flow and congestion control can be omitted as the use of schedulability analysis in connection with traffic flows will ensure that only traffic that can be handled by the network will be accepted. ACKs are not retransmitted and used only for data traffic from the master to a slave. Data packets from individual slaves to the master need not be acknowledged since the master is responsible for polling for retransmissions.

Master Node

At start-up, the master runs a schedulability analysis based on traffic specifications given by the system manager. All ReRTCs plus all RTCs that pass the real-time feasibility check are accepted. Next, an EDF-sorted queue is created, containing two types of packets: polling messages for data to be sent from slaves to master and data packets for transmission from master to a slave. The master chooses the first packet in the queue and checks if it is a polling packet or a data packet. A time-out value specifies the period of time

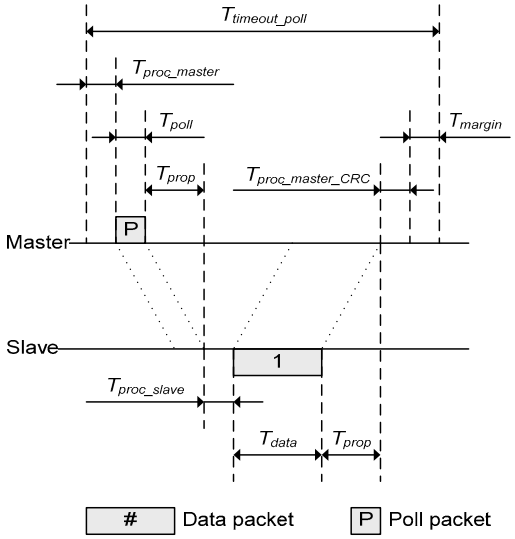

Figure 1. Time-sequence diagram for the calculation of the timeout in case of a traffic flow from a slave to the master.

the master node has to wait for either the requested packet, or the acknowledgement of the transmitted packet, before it is allowed to continue with the next packet in the queue. The value of the timeout parameter depends on several factors, e.g., the length of the data packet, the actual bit rate, and the propagation delay, Fig. 1. In the case of a polling packet, the time from extracting the poll from the queue until a poll for a (re)transmission will be possible is given by:

$$T_{timeout\_poll,i} = T_{proc\_master} + T_{poll} + T_{prop} + T_{proc\_slave} + \\ + T_{data,i} + T_{prop} + T_{proc\_master\_CRC} + T_{margin}, \quad (1)$$

where T_{poll} and $T_{data,i}$ are the transmission times of the polling packet and the data packet belonging to τ_i, respectively. The propagation delay for both directions, $2 \cdot T_{prop}$, is added, as well as T_{proc_master} and T_{proc_slave}, accounting for, e.g., MAC layer processing delays at the master and slave nodes, respectively. $T_{proc\_master\_CRC}$ also includes the time to check for errors in the MAC layer, while T_{margin} is a safety margin between the expected reception of the data packet and the actual end of the timeout to cater for propagation variations. In case the transmitted packet is data from the master to one of the slaves, an ACK is sent directly by the slave after reception of the data packet. The time before a (re)transmission is possible in this case is given by, Fig. 2:

$$T_{timeout\_data,i} = T_{proc\_master} + T_{data,i} + T_{prop} + T_{proc\_slave\_CRC} \\ + T_{ACK} + T_{prop} + T_{proc\_master} + T_{margin}. \quad (2)$$

T_{ACK} is the transmission time of the ACK. In addition to the processing time, $T_{proc\_slave\_CRC}$ also includes the time for error checking. Here, T_{margin} indicates the safety margin between the expected reception of the ACK and the actual timeout.

Once the timeout value has been calculated by the master node, it sends either a data or a poll packet down to the MAC layer. If it is expecting data from a slave, it sends a

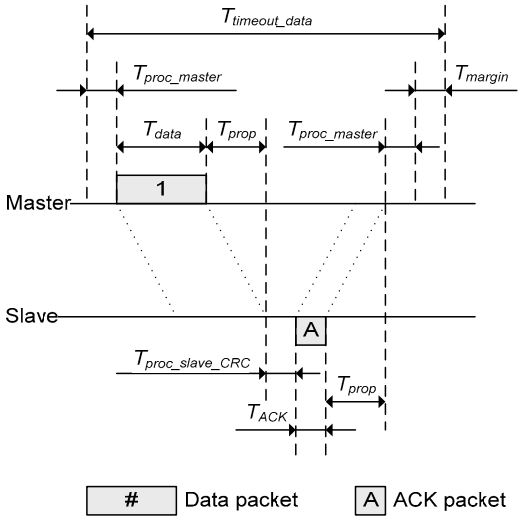

Figure 2. Time-sequence diagram for the calculation of the timeout in case of a traffic flow from the master to a slave.

poll packet requesting either an ordinary transmission or a retransmission. Then the master waits for the interval of time defined by the timeout value. In case an ACK arrives, the master removes the corresponding data packet from the buffer. In the absence of an ACK, or when an expected data packet is missing, the polling or data packet will be saved until its ordinary or retransmissions deadline has expired. Not until the ordinary deadline of a complete message has passed, will the master check the availability of the retransmission channels, as only then it will know how many packets in the message actually need retransmitting. This holds also true for retransmission deadlines unless it is the deadline of the last allowed retransmission attempt. To be considered available, the last use of a retransmission channel must be at least as long ago as its retransmission period. If a satisfying number of retransmission channels is found, and the maximum number of retransmissions for the packets in question has not been exhausted, new polling or data packet retransmissions will be scheduled and added to the queue. Note that the entire retransmission time, including timeout values and potential polling or ACK, needs to be taken into account when determining if a retransmission deadline can be met. If not enough retransmission channels are available, the request is denied, ensuring that deadlines of previously accepted transmissions are never violated.

Slave Node

In the slave, no logic for retransmissions administration or schedulability analysis is needed. Each slave only holds a software queue for ordinary transmissions, continuously populated with the periodic real-time traffic generated by the application(s). Further, a memory for previously transmitted packets is necessary, where all packets are kept until their deadline expires in order to be available for possible retransmissions. ACKs are not queued as they are sent directly after reception of a correct packet. Whenever a slave receives a polling message, the protocol identifies the right packet in the transmission queue or the retransmission memory and hands the packet down to the MAC layer for transmission. Next the retransmission memory is checked for packets with expired deadlines, which are immediately removed. Packets transmitted for the first time are relocated from the transmission queue to the retransmission memory.

V. Analytical Performance Evaluation

Since it is our goal to guarantee that the deadlines of all real-time traffic flows (including retransmissions) are met, real-time analysis plays a vital part in our framework. Extending the work in [1], our analysis is applied to a polling-based MAC method, where master-to-slave and slave-to-master communication is denoted by M→S and S→M, respectively. Additionally, an analytical expression is derived to calculate the probability of message error under the assumption of our retransmission scheme.

A. Utilization

The maximum number of RTCs and ReRTCs simultaneously present in the system is denoted Q and M, respectively. Each RTC is defined as $\tau_i=\{S_i, R_i, P_i, C_i, D_i\}$. In the transport layer, the deadline D_i, given by the application, is, as previously mentioned, divided into one deadline for ordinary transmissions $D_{ord,i}$ and one for retransmissions $D_{retr,i}$. Further, the message length given in the RTC specification indicates only the length of the actual data packets, but does not take into consideration the polling packet prior to a slave's channel access. Neither does it account for the ACK sent by the slave upon reception of a correct data packet. Both have to be included when analysing the real-time feasibility of the traffic set. Considering the total transmission time $T_{S\rightarrow M,i}$ for a traffic flow τ_i, we have:

$$T_{S\rightarrow M,i} = N_i \cdot \left(T_{data,i} + T_{poll,i} + 2 \cdot T_{prop} + T_{proc\_master} + T_{proc\_slave} + T_{proc\_master\_CRC} + T_{margin}\right), \quad (3)$$

where N_i is the number of packets transmitted by τ_i in each period. Note that (3) relates to the timeout value given by (1), but includes all packets required to transmit one instance of τ_i. Considering the transmission time $T_{M\rightarrow S}$, no polling packet is included. Instead an ACK is sent by the slave after successful reception of a data packet, and the corresponding transmission time for all packets of τ_i is:

$$T_{M\rightarrow S,i} = N_i \cdot \left(T_{data,i} + T_{ACK} + 2 \cdot T_{prop} + 2 \cdot T_{proc\_master} + T_{proc\_slave\_CRC} + T_{margin}\right). \quad (4)$$

The deadline for each ordinary transmission has already been reduced to $D_{ord,i}$ to allow for one or more retransmission attempt(s). However, further adaptation of this deadline is necessary to isolate the queuing delay. The worst case blocking time has to be considered, as experienced by a packet (with an earlier deadline) arriving at the EDF queue and being blocked by a packet with a later deadline that has just started to be transmitted and cannot be stopped due to

the assumption of nonpreemptiveness for packets in a communication context. The blocking time, $T_{blocking}$, is defined as the maximum timeout in any direction, i.e.,

$$T_{blocking} = \max\left\{T_{timeout\_poll,i}, T_{timeout\_data,i}\right\} \quad (5)$$

The new deadlines for regular M→S and S→M communication are, respectively:

$$d_{ordM\to S,i} = d_{ordS\to M,i} = D_{ord,i} - T_{blocking}. \quad (6)$$

Analogously, the total transmission times for $\tau_{re,j}$ are:

$$T_{S\to M\_retr,j} = T_{data,j} + T_{poll,j} + 2\cdot T_{prop} + T_{proc\_master} + T_{proc\_slave} + T_{proc\_master\_CRC} + T_{margin} \quad (7)$$

$$T_{M\to S\_retr,j} = T_{data,j} + T_{ACK,j} + 2\cdot T_{prop} + T_{proc\_master} + T_{proc\_slave} + T_{proc\_master\_CRC} + T_{margin}. \quad (8)$$

By dividing the deadline in two parts, we obtain the deadline $D_{retr,i}$ for all retransmission attempts for the packets belonging to τ_i. Being identical for all channels, it is henceforth denoted D_{retr}. As D_{retr} has to accommodate $N_{attempt}$ retransmission attempts, a deadline per retransmission attempt ($D_{retrM\to S,i}$, $D_{retrS\to M,i}$) for each RTC is defined as:

$$D_{retrM\to S,i} = D_{retrS\to M,i} = \frac{D_{retr}}{N_{attempt}} \quad (9)$$

The retransmission channel $\tau_{re,j}$ is partly defined by the parameters $P_{re,j}$, $D_{re,j}$, and $C_{re,j}$, which are system parameters used to control the length of the retransmission time span and thereby its allowed bandwidth. As the deadline guarantee has to hold also for retransmissions, all retransmission periods D_{re} are defined to be of the same length as the per retransmission deadline, i.e.:

$$D_{re} = D_{re,j} = D_{retrM\to S,i}. \quad (10)$$

Again, we need to account for the worst case blocking time $T_{blocking}$ when isolating the maximum queuing delays $d_{retrM\to S,i}$ and $d_{retrS\to M,i}$ which are calculated by:

$$d_{retrM\to S,i} = d_{retrS\to M,i} = D_{re} - T_{blocking}. \quad (11)$$

The real-time analysis itself is conducted in two mandatory parts, as each of the steps is necessary, but not sufficient in itself under given assumptions. The first condition to be fulfilled is that the utilization of any link is never to exceed 1. Adapting well-known EDF scheduling theory [24] for periodic real-time tasks to our case yields a utilization of:

$$U = \sum_{i=1}^{V}\left(\frac{T_{S\to M,i}}{P_{T,i}}\right) + \sum_{i=1}^{W}\left(\frac{T_{M\to S,i}}{P_{T,i}}\right) + \sum_{j=1}^{X}\left(\frac{T_{S\to M\_retr,j}}{P_{re,j}}\right) + \sum_{j=1}^{Y}\left(\frac{T_{M\to S\_retr,j}}{P_{re,j}}\right). \quad (12)$$

Here, V and W denote the number of RTCs (M→S and S→M, respectively), where $V + W = Q$. X and Y are the number of retransmission channels used for M→S communication and vice versa, where $X + Y = M$. The transmission time parameters are used according to their definition in (3), (4), (7) and (8), respectively. The second step of the real-time analysis includes calculating the workload imposed on the network by the RTCs. Originally intended for calculating the processor workload in uniprocessor task scheduling, and proven correct for EDF scheduling in [25], this method was mapped onto a networking context in [26]. A workload function, $h(t)$, is calculated as the sum of the transmission times of all messages sent by all RTCs in all periods (only message instances which have their absolute deadline before time t are included). The time t is measured relative to the start of the hyperperiod of all RTCs. The length of the hyperperiod is calculated as the least common multiple of all RTC periods and the start of the hyperperiod is defined as the time when all periods start at the same time, and it ends when they do so again. This concurrent start of all periods represents the proven worst case workload on the network, and therefore also leads to the worst case delay [25][27][28]. In our case, the workload is:

$$h(t) = \sum_{\substack{i\in[1,V],\\ d_{ordS\to M,i}\le t}}\left(1+\left\lfloor\frac{t-d_{ordS\to M,i}}{P_{T,i}}\right\rfloor\right)\cdot T_{S\to M,i} + \sum_{\substack{i\in[1,W],\\ d_{ordM\to S,i}\le t}}\left(1+\left\lfloor\frac{t-d_{ordM\to S,i}}{P_{T,i}}\right\rfloor\right)\cdot T_{M\to S,i} + \sum_{\substack{j\in[1,X],\\ d_{retrS\to M,j}\le t}}\left(1+\left\lfloor\frac{t-d_{retrS\to M,j}}{P_{re,j}}\right\rfloor\right)\cdot T_{S\to M\_retr,j} + \sum_{\substack{j\in[1,Y],\\ d_{retrM\to S,j}\le t}}\left(1+\left\lfloor\frac{t-d_{retrM\to S,j}}{P_{re,j}}\right\rfloor\right)\cdot T_{M\to S\_retr,j} \quad (13)$$

where the first and the second sum denote the workload for the ordinary transmissions (M→S and vice versa), and the third and the fourth sum represent the retransmissions. Here $h(t) \le t$, $\forall t$, must be fulfilled in order to ensure the feasibility of traffic allocations also when additional RTCs are introduced in the network. In [29] a reduction of computational complexity is proposed by only considering discrete instances, i.e., only whenever a message deadline occurs. If both the utilization and the feasibility constraints are fulfilled, the timely treatment of all allocated RTCs can be guaranteed. The tests only have to be executed when a new RTC is requesting access to the network. Once a traffic allocation for a certain set of RTCs has passed both tests, deadline guarantees can be met and maintained.

B. Message Error Rate

Without loss of generality, we assume that all messages from all RTCs consist of n packets each. The length in bits of each packet is denoted L. For simplicity, we also assume a fixed bit error probability, P_b, in our numerical evaluation of the MER, although in our computer simulations we have used both a fixed and a randomly varying P_b. The packet error probability, P_e, can then be obtained from the bit error probability, according to:

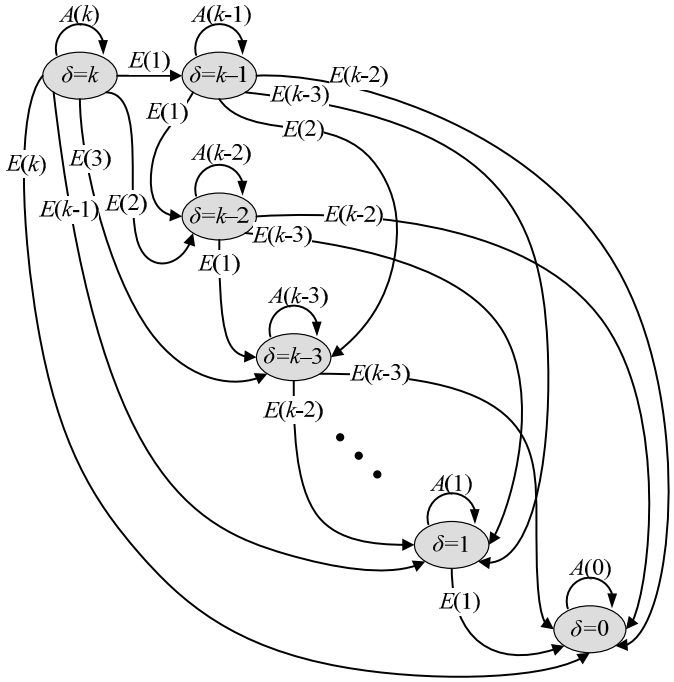

Figure 3. Transition diagram for the probability of having δ ReRTCs left.

$$P_e = 1 - \left(1 - P_b\right)^L . \tag{14}$$

Assume that we have k available ReRTCs, each capable of retransmitting exactly one packet. For a system without retransmissions, $k = 0$ and the MER is given by

$$\begin{aligned} P_{k=0} &= \sum_{\alpha=1}^{n} \binom{n}{\alpha} P_e^{\alpha} \left(1-P_e\right)^{n-\alpha} \\ &= \sum_{\alpha=1}^{n} \binom{n}{\alpha} P_{NoARQ}\left(n,\alpha\right). \end{aligned} \tag{15}$$

This provides an upper bound for the MER of our framework. A system that allows one retransmission for each erroneous packet, i.e., $k = nX$, where X is the total number of RTCs, has a MER of:

$$\begin{aligned} P_{k=nX} &= \sum_{\alpha=1}^{n} \binom{n}{\alpha} \sum_{\beta=1}^{\alpha} \binom{\alpha}{\beta} P_e^{\alpha} \left(1-P_e\right)^{n-\alpha} P_e^{\beta} \left(1-P_e\right)^{\alpha-\beta} \\ &= \sum_{\alpha=1}^{n} \binom{n}{\alpha} \sum_{\beta=1}^{\alpha} \binom{\alpha}{\beta} P_{ARQ=1}\left(n,\alpha,\beta\right) \end{aligned} \tag{16}$$

where α indicates the number of erroneous packets in the message after the ordinary transmission, and β those that are still erroneous after the retransmission. Note that the probability of message error in this case is the same for all RTCs, since all erroneous packets may be retransmitted once. Equations (15) and (16) constitute upper and lower bounds to the MER in our system, since we allow some, but not all packets to be retransmitted. As in the calculation of U, we assume the worst case of all RTCs starting their periods at the same time. A maximum of one retransmission per packet is allowed, using one of k available ReRTCs. Further, due to reasons of complexity, we assume that all ReRTCs are released at the end of the hyperperiod, i.e., that all ReRTC have a period equal to the hyperperiod, allowing us to consider only one hyperperiod. Assume first that $0 < k < n$. For the first RTC, $i = 1$, to be transmitted, there are exactly k retransmissions left, and the MER is:

$$\begin{aligned} P_{k,i=1} = & \underbrace{\sum_{\alpha=k+1}^{n} \binom{n}{\alpha} P_{NoARQ}\left(n,\alpha\right)}_{\substack{\text{packet error(s) and not enough} \\ \text{retransmissions available}}} + \\ & + \underbrace{\sum_{\alpha=1}^{k} \binom{n}{\alpha} \sum_{\beta=1}^{\alpha} \binom{\alpha}{\beta} P_{ARQ=1}\left(n,\alpha,\beta\right)}_{\substack{\text{packet error(s), enough retransmissions available,} \\ \text{but retransmission(s) is/are erroneous}}} . \end{aligned} \tag{17}$$

For all the following messages, where $i > 1$, the probability of message error is given by:

$$\begin{aligned} P_{k,i>1} = & \sum_{\alpha=k+1}^{n} \binom{n}{\alpha} P_{NoARQ}\left(n,\alpha\right) + \\ & + \sum_{\alpha=1}^{k} \sum_{\gamma=0}^{\alpha-1} \binom{n}{\alpha} P_{NoARQ}\left(n,\alpha\right) \mathbf{R}\left[1,\gamma+1\right]^{(i-1)} + \\ & + \sum_{\alpha=1}^{k} \sum_{\gamma=\alpha}^{k} \binom{n}{\alpha} \sum_{\beta=1}^{\alpha} \binom{\alpha}{\beta} P_{ARQ=1}\left(n,\alpha,\beta\right) \mathbf{R}\left[1,\gamma+1\right]^{(i-1)} \end{aligned} \tag{18}$$

Recall that retransmissions are granted only when there are enough available ReRTCs such that all erroneous packets in the message can be retransmitted. Consequently, the first and second term correspond to the case where RTC i experiences packet errors, but not enough retransmission channels are available, either because there are more than k packet errors or because previous RTCs have already used some of the ReRTCs. Note that the probability of message error now depends on i since the number of available retransmissions, k, is potentially reduced for each new RTC introduced. Consequently, $\mathbf{R}[1,\gamma+1]^{i-1}$ denotes the probability of having exactly γ available ReRTC left after transmission of RTC i. This probability can be described in a state transition diagram, Fig. 3, resulting in the probability transition matrix:

$$\mathbf{R} = \begin{bmatrix} E(k) & E(k-1) & \cdots & E(1) & A(k) \\ E(k-1) & \cdots & E(1) & A(k-1) & 0 \\ \vdots & \iddots & \iddots & 0 & 0 \\ E(1) & A(1) & 0 & \vdots & \vdots \\ A(0) & 0 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

The transition probabilities in Fig. 3 correspond to the indices in matrix **R** and are given by:

$$A\left(\delta\right) = \left(1-P_e\right)^n + \sum_{\varphi=\delta+1}^{n} \binom{n}{\varphi} P_e^{\varphi} \left(1-P_e\right)^{n-\varphi} \tag{20}$$

$$E(\delta)=\binom{n}{\delta}P_e^{\delta}\left(1-P_e\right)^{n-\delta}. \quad (21)$$

Here, $A(\delta)$ relates to the probability of remaining in the same state and thus not using any ReRTC, either because of no errors or because of too many errors, whereas $E(\delta)$ denotes the probability of using exactly δ ReRTCs. The total MER of our system is found by averaging over the message errors experienced by each of the individual RTCs.

Assume now that $k = n$. For the first message, there are exactly k retransmissions left. This is always enough as $k = n$ is the maximum number of packets that can be erroneous. Therefore the only possibility of message error is erroneous retransmissions, and the probability of message error is given by (16). For the following messages where $i > 1$, the probability of message error can be calculated as

$$P_{k,i>1}=\sum_{\alpha=1}^{n}\sum_{\gamma=0}^{\alpha-1}\binom{n}{\alpha}P_{NoARQ}(n,\alpha)\mathbf{R}[1,\gamma+1]^{(i-1)}+ \\ +\sum_{\alpha=1}^{n}\sum_{\gamma=\alpha}^{k}\binom{n}{\alpha}\sum_{\beta=1}^{\alpha}\binom{\alpha}{\beta}P_{ARQ=1}(n,\alpha,\beta)\mathbf{R}[1,\gamma+1]^{(i-1)} \quad (22)$$

Finally, assume that $n < k < nX$. For the first message, there are exactly k retransmissions left, which is always more than necessary, and the probability of message error is again given by (16). For the following messages, where $i > 1$, the probability of message error will be given by (22) since there can never be more than n packet errors per message.

VI. Simulation results

We have evaluated the performance of our framework and validated our analytical expressions by discrete-time, Monte-Carlo computer simulations implemented in Matlab. The goal was to study the obtained bandwidth utilization, the MER improvement, the utilization penalty suffered due to the introduction of retransmission channels, and possible jitter introduced into the system by our retransmission scheme. In the simulator we implemented both the MAC layer extension and the transport layer protocol, with and without retransmissions (still providing deadline guarantees) to get a suitable benchmark for comparison.

The simulator models a single-hop network with 50 nodes, one of them acting as master. The data rate was set to 54 Mb/s, according to the IEEE 802.11 standard, with a maximum packet length of 1000 bits and an ACK and a polling packet length of 100 bits. These packet lengths are somewhat shorter than what is typical for today's WLAN standard, but appropriate in an industrial context. Since ACK and poll packets are very short, we assume an error free feedback channel and thus neither packet types are retransmitted. The propagation delay is set to a constant 1 μs, corresponding to a distance of approximately 300 m assuming a wave propagation speed through air of $3\cdot10^8$ m/s.

For all simulations, the data traffic consists of RTCs randomly (assuming a uniform distribution) chosen from a specified set of traffic classes. The sender and receiver nodes are randomized (also here assuming a uniform distribution), with the master always being on one side of each traffic flow. The set of retransmission channels are varied between simulations, but as the length of a retransmission packet (L_{re}), the data rate (r), the retransmission period (P_{re}), and the number of retransmission channels (M) is known, the theoretical maximum network utilization by the retransmission channels (U_{re}) can be calculated as:

$$U_{re}(M)=\frac{\left(L_{re}/r\right)}{P_{re}}\cdot M \quad (23)$$

All transmitted data packets are subject to random noise represented as a certain bit error probability resulting in a specific packet error rate. A constant BER of 10^{-4} was used for verification of the analytical results, while for the rest of the simulations we assume a bursty error behaviour, modelled by a typical two-state Markov model (Gilbert-Elliot). The two states have a BER of 10^{-2} and 10^{-4} and the state changing probabilities are 0.5 and 0.99, respectively. The traffic classes out of which the data traffic is randomized are specified in Table I, while the two different simulated specifications of retransmission channels are given in Table II, together with their theoretical maximum utilization, U_{re}.

The network utilization was both simulated on the packet level and analytically calculated on the traffic flow level using (12). Fig. 4 depicts the utilization values for the two cases with and without retransmissions, derived both through analysis and through simulation. U is plotted as a function of the number of *requested* RTCs. The utilization value is the sum of *accepted* RTCs, excluding all retransmission channels since we want to study the influence of the retransmission channels on the regular traffic. Whether or not a channel is accepted, is decided based on the outcome of the real-time schedulability test implemented as a part of the transport layer functionality. It should be noted that the inclusion of retransmission channels reduces the maximum possible bandwidth utilization by normal real-time channels. However, the reduction is not only due to the usage of bandwidth by the retransmission channels, since in that case the reduction would correspond to the theoretical maximum

TABLE I. TRAFFIC SPECIFICATION

Ordinary traffic			
Traffic class	P_i(ms)	D_i(ms)	C_i(bits)
TC1	2	2	1000
TC2	4	4	2000
TC3	8	8	3000

TABLE II. RETRANSMISSION CHANNEL SPECIFICATION

Retransmissions			
ReRTC	P_{re}(ms)	D_{re}(ms)	C_{re}(bits)
1	2	0.8	1000
$N_{attempt}$=2, k=2 or 8			
$U_{re}(k=2)$=1.85%, $U_{re}(k=8)$=7.41%			

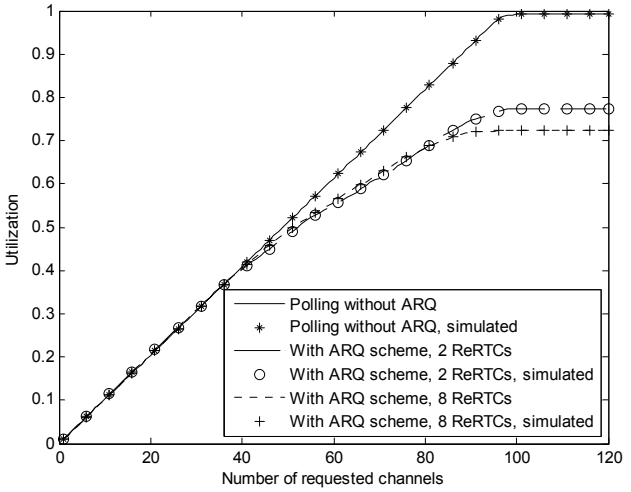

Figure 4. Utilization results derived through analysis and simulation

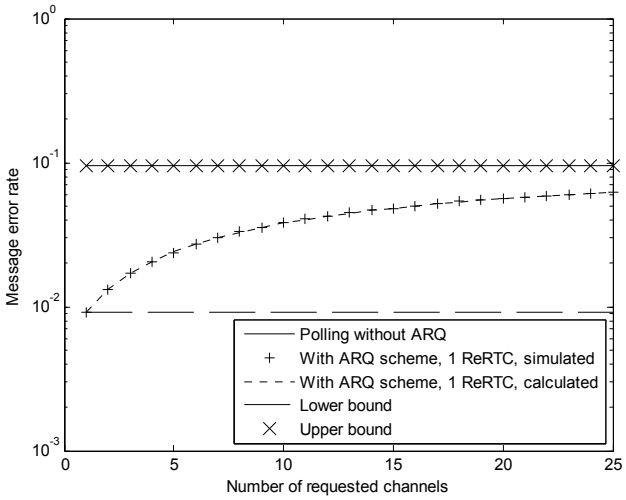

Figure 5. Analytical and simulated MER for a simple traffic case

bandwidth utilization of the retransmission channels, U_{re} as shown in Table II. The penalty observed both in the analytical and the simulated curve is instead mainly due to the fact that the ReRTCs introduced for the retransmissions have a deadline which is considerably shorter than their period, which makes it more difficult to find feasible schedules. Furthermore, also the deadline of regular RTCs is shortened (from D to D_{ord}) contributing to this decrease of schedulability. Intuitively, low rate traffic channels with long periods and long deadlines are easier to schedule in such a context. The resulting network utilization penalty was found to be approximately 23 percentage points when using two retransmission channels, while eight retransmission channels increase the penalty to 28 percentage points. The fact that the simulated and the analytical value correspond with each other verifies the correctness of our results.

The second parameter studied was the MER both when not using our retransmission scheme, and after the introduction of retransmissions. The simulated MER values are the result of a packet level Monte-Carlo simulation, while the analytical values are given by (17) and (22). Fig. 5 shows a simple traffic case with merely one traffic class. The period and the deadline were set to 1 ms, with a message length of 2 kb. Further we assumed $M = 1$ and $N_{attempt} = 1$, with the ReRTC's period set to 1 ms, its deadline at 0.2 ms, and a message length of 1 kB. The BER in this case was fixed to 10^{-4}. The theoretical upper bound on the MER is identical to the simulated case of no retransmissions, while both the simulated and the analytical MER value when using our framework are between the bounds. As the analytical and the simulated curves coincide, our results are validated.

Studying the MER for the traffic case specified in Table I and under the assumption of bursty error behaviour, we can see in Fig. 6 that, when using two retransmission channels, the MER can be improved from 10^{-1} down to almost 10^{-2}, depending on the number of RTCs in the system. Increasing the number of retransmission channels pushed the MER further down for the case of few RTCs, while the difference in improvement for a high number of RTCs is considerably smaller. There is a distinct difference in the level of improvement for a smaller amount of RTCs as compared to the saturated network. With fewer RTCs, fewer channels need to request retransmissions, which means that the competition for retransmission channels is low and more of the required retransmissions can be sent. However, this holds only true when the number of requested RTCs reaches a level where all ReRTCs can be exploited. Assuming a very small numbers of traffic flows in the network, increasing the number of ReRTCs would not improve the MER any further, as we cannot retransmit more packets than actually exist. Instead the deadline or the number of allowed retransmissions will become the limiting factor for further MER improvement.

When considering the utilization and MER curves together, a clear, and expected, trade-off becomes obvious. The introduction of retransmission channels leads to an improvement regarding message errors, but this gain in reliability has to be paid with a decreased grade of utilization. This penalty manifests itself noticeably at high network loads where the utilization penalty is highest, while the MER improvement is smallest. However, at medium network loads, the trade-off is less costly, which proves the usefulness of our approach. Additionally, it must not be forgotten that bandwidth not booked by hard real-time traffic and its retransmissions can always be used by best-effort traffic since no bandwidth is lost for retransmission opportunities not needed by the RTCs.

In order to see the effect of our retransmission approach on individual traffic classes, we studied the introduction of retransmission channels on different traffic classes as specified in Table I. The period of the retransmission channels in our simulations is equal to the shortest of the periods amongst the regular traffic channels. For the sake of readability, only the results for $M = 2$ are presented in Fig. 7. When studying the improvement for the different traffic classes, it can be seen that the overall improvement in MER

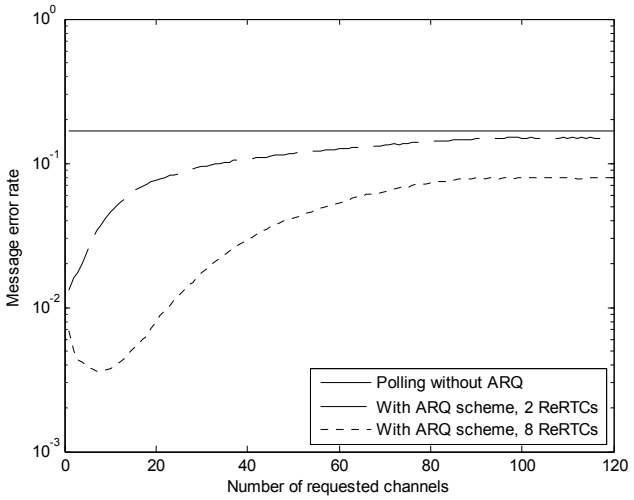

Figure 6. MER results derived by simulation

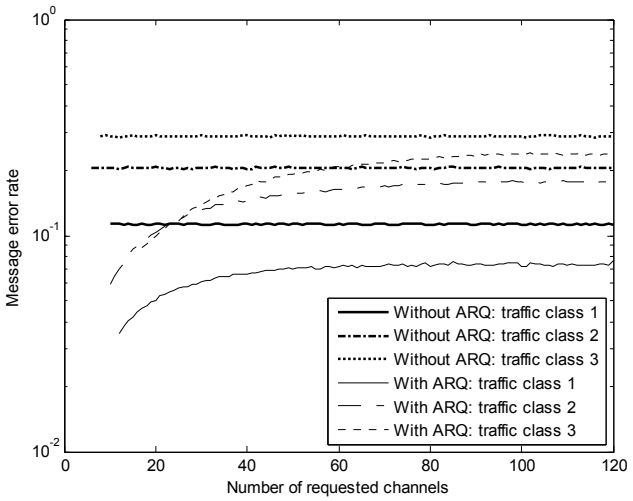

Figure 7. MER per traffic class (TC) derived by simulation

is highest for TC1 which is characterized by short messages, a short period and a short deadline. Both TC2 and TC3 are experiencing smaller improvements than TC1, but only for high numbers of requested RTCs, when the competition for retransmissions is increased. (For very small numbers of requested RTCs, too few message errors could be recorded to provide statistically reliable results.) Obviously, having only one type of retransmission channel does not favour all traffic classes equally well. This indicates room for improvement in the design of the details of our scheme, where the periods and the deadlines for the ReRTCs could be designed differently for different ReRTCs, relaxing our assumption of identical retransmission channels.

Many real-time applications are sensitive not only to deadline misses, but also to variations in delay. Since any retransmission scheme will influence the message delay, we studied the average message delay experienced in our simulation. The values have been calculated for the maximum number of 120 requested RTCs (Fig. 8 and 9) and the result shows the delay distribution in ten equal intervals. It can be seen that for the case without retransmissions the delay distribution has, as expected due to the nature of the traffic classes, a high number of messages with a quite short delay (< 2 ms), and none of the messages experiences a longer delay than about 5.5 ms. Introducing retransmissions shortens the ordinary deadline, leading to a decrease in delay for many messages. However, when the first transmission is erroneous, further time is added until the initiation of a retransmission, leading to an increase in delay, visible in the rightmost bin in Fig. 9. Our simulation shows that, as expected, the spreading in the delay distribution is higher, but also that the percentage of messages with short delay is higher when using retransmissions. As the introduction of retransmissions leads to a shortening of the deadline and thereby influences the schedulability in the network, the absolute number of messages in the histograms cannot be compared directly. They differ as the number of traffic channels admitted to the network differs in the two cases.

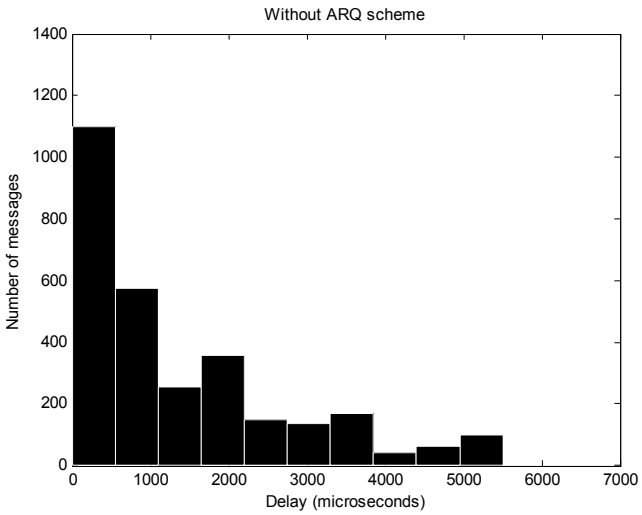

Figure 8. Message delay for the case without retransmissions

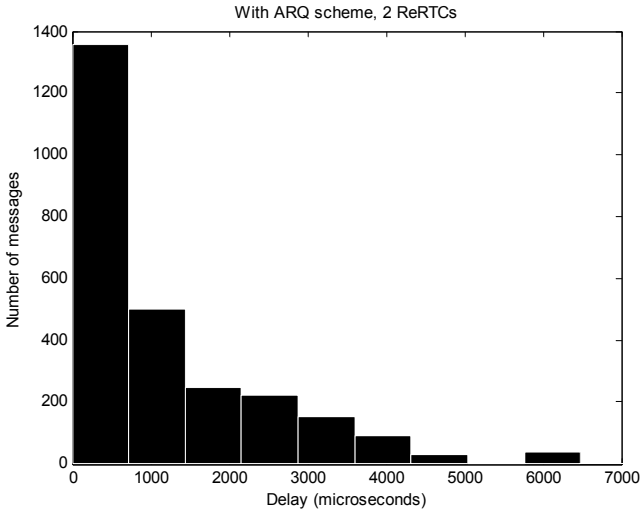

Figure 9. Message delay for the case with 2 ReRTCs

VII. CONCLUSION

We presented a framework suitable to fulfil requirements on absolute predictability and high reliability in a single-hop wireless network. Building on existing chipsets supporting IEEE 802.11, we extended the MAC layer to provide deterministic channel access, and designed a transport layer admission control and retransmission functionality. We improve the MER for hard real-time traffic substantially while keeping the utilization penalty at a reasonable level. Consequently our approach is a tractable solution for networks carrying real-time traffic in need of concurrent reliability and deadline guarantees such as, e.g., industrial monitoring and surveillance systems. A solution that enables the use of wireless technologies even in challenging radio environments with inherently high noise and interference levels, opens up for a new area of applications.

In contrast to many other solutions, we evaluate our approach both by analytical calculations and computer simulations, each method contributing with different insights. While analysis makes it possible to study the relationship between MER and e.g. message length, retransmission attempts per packet, and number of retransmissions channels, in the simulations we can study the effect of, e.g., different channel models, the interaction of different traffic classes, or the actual (not worst case) bandwidth utilization.

REFERENCES

[1] M. Jonsson and K. Kunert, "Towards reliable wireless industrial communication with real-time guarantees", *Trans. on Ind. Inf.*, 5(4), 2009, pp. 429-442.

[2] K. Kunert, E. Uhlemann, and M. Jonsson, "Predictable real-time communications with improved reliability for IEEE 802.15.4 based industrial networks", *Proc. IEEE Int. Works. Factory Comm. Syst.*, Nancy, France, May 2010.

[3] A. Mesquita, J. Hespanha, and G. Nair. "Redundant data transmission in control/estimation over wireless networks". *Proc. Amer. Contr. Conf.*, 2009, pp. 3378-3382.

[4] D.D. Demarch and L.B. Becker, "An integrated scheduling and retransmission proposal for firm real-time traffic in IEEE 802.11e", *Proc. Euromicro Works. on Real-Time Syst.*, 2007.

[5] S. Pejhan, M. Schwartz, and D. Anastassiou, "Error control using retransmission schemes in multicast transport protocols for real-time media," *IEEE/ACM Trans. on Networking*, 4(3), June 1996, pp. 413-427.

[6] X. Chen, H. Zhai, X. Tian, and Y. Fang, "Supporting QoS in IEEE 802.11e wireless LANs", IEEE *Trans. on Wireless Comm.*, 5(8), 2006, pp. 2217-2227.

[7] E. Uhlemann, P.-A. Wiberg, T.M. Aulin, and L.R. Rasmussen, "Deadline dependent coding - a framework for wireless real-time communication," *Proc. Int. Conf. Real-Time Comp. Syst. and Appl.*, 2000, pp. 135-142.

[8] E. Uhlemann and L.K. Rasmussen, "Incremental redundancy deadline dependent coding for efficient wireless real-time communications," *Proc. IEEE Conf. Emerging Techn. & Factory Automation*, 2005, pp. 417-424.

[9] M.M. Butt, "Provision of guaranteed QoS with hybrid automatic repeat request in interleave division multiple access systems", *IEEE Int. Conf. on Comm. Sys.*, 2006, pp. 1-5.

[10] G. Giancola, S. Falco, and M.G. Di Benedetto, "A novel approach to error protection in medium access control design," *Proc. Int. Works. on Mobile & Wirel. Comm. Netw.*, 2002.

[11] X. Fan, M. Jonsson, and J. Jonsson, "Guaranteed real-time communication in packet-switched networks with FCFS queuing", *Computer Netw.*, 53(3), Feb. 2009, pp. 400-417.

[12] M. Pandya and M. Malek, "Minimum achievable utilization for fault-tolerant processing of periodic tasks", *IEEE Trans. on Computers*, 47(10), Oct. 1998, pp. 1102-1112.

[13] A. Burns, R. Davis, and S. Punnekkat, "Feasibility analysis of fault-tolerant real-time task sets", *Proc. Euromicro Works. on Real-Time Syst.*, June 1996, pp. 29-33.

[14] K.W. Tindell, H. Hansson, and A.J. Wellings, "Analysing real-time communications: controller area network (CAN)", *Proc. Real-Time Syst. Symp.*, Dec. 1994, pp. 259-263.

[15] K. Tindell, A. Burns, and A.J. Wellings, "Calculating controller area network (CAN) message response times", *Control Eng. Practice,* 3(8), 1995, pp. 1163-1169.

[16] S. Punnekkat, H. Hansson, and C. Norström, "Response time analysis under errors for CAN", *Proc. Real-Time Techn. & Appl. Symp.*, 2000, pp. 258-265.

[17] H.A. Hansson, T. Nolte, C. Norström, and S. Punnekkat "Integrating reliability and timing analysis of CAN-based systems", *IEEE Trans. Ind. Electr.*, 49(6), 2002, pp. 1240-1250.

[18] R. Davis, A. Burns, R. Bril, and J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised", *Real-Time Syst.*, 35(3), Apr. 2007, pp. 239-272.

[19] I. Broster and A. Burns, "Timely use of the CAN protocol in critical hard real-time systems with faults", *Proc. Euromicro Works. on Real-Time Syst.*, June 2001, pp. 95-102.

[20] J. Ferreira, P. Pedreiras, L. Almeida, and J.A. Fonseca, "FTT CAN error confinement", *Proc. Int. Conf. on Fieldbus Syst. & their Appl.,* 2001, pp. 8-15.

[21] J. Ferreira, L. Almeida, A. Fonseca, P. Pedreiras, E. Martins, G. Rodriguez-Navas, J. Rigo, and J. Proenza, "Combining operational flexibility and dependability in FTT-CAN", *Trans. on Ind. Inf.,* 2(2), 2006.

[22] H. Aydin, "On Fault-Sensitive Feasibility Analysis of Real-Time Task Sets", *Proc. Int. Real-Time Sys. Symp.*, 2004.

[23] H. Beitollahi, S.G. Miremadi, and G. Deconinck, "Fault-tolerant earliest-deadline-first scheduling algorithm", *Proc. IEEE Int. Parallel & Distr. Processing Symp.*, 2007.

[24] C.L. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment", *J. of the ACM*, 20(1), Jan. 1973, pp. 46-61.

[25] M. Spuri, "Analysis of deadline scheduled real-time systems", *Tech. Rep. 2772*, INRIA, France, 1996.

[26] H. Hoang and M. Jonsson, "Switched real-time Ethernet in industrial applications - deadline partitioning", *Asia-Pacific Conf. on Comm.*, Sept. 2003, pp. 76-81.

[27] S.K. Baruah, A.K. Mok, and L.E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor", *Proc. Real-Time Systems Symp.*, Dec. 1990, pp. 182-190.

[28] 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", *Proc. Int. Real-Time Syst. Symp.*, 2(4), Nov. 1990, pp. 301-324.

[29] J.A. Stankovic, M. Spuri, K. Ramamritham, and G.C. Buttazzo, Deadline scheduling for real-time systems - EDF and related algorithms, Kluwer Acad. Publ., Boston, USA, 1998.